Kostenpflichtige Web-Services

Dietrich Boles, Markus Schmees
Universität Oldenburg

Zusammenfassung: Qualitativ hochwertiger Content wird im WWW in zunehmendem Maße nur noch gegen Bezahlung zur Verfügung gestellt. Hierfür sind in den letzten Jahren eine Menge von Technologien und Systemen entwickelt worden. Dahingegen sind in die Web-Service-Technologien bisher keine vollständigen Mechanismen integriert, um die Nutzung der Dienste bzw. Funktionen kostenpflichtig gestalten zu können. Vor diesem Hintergrund werden in diesem Artikel ein Konzept und eine Implementierung für kostenpflichtige Web-Services vorgestellt. Leistungserbringer können mit dem implementierten System Angebote für ihre Dienste erstellen und darauf basierende Zugriffslizenzen verkaufen. Eine rechtmäßige Nutzung ihrer Dienste wird durch das System sichergestellt.

Schlüsselworte: Web-Service, Dienstleistung, eCommerce, Geschäftsmodell

1 Einleitung

In den entwickelten Volkswirtschaften nimmt der Dienstleistungssektor einen ständig steigenden Anteil an der gesamten Wertschöpfung und Beschäftigung ein. Eine Erklärung für diese Entwicklung liefert die so genannte "Drei-Sektoren-Hypothese" von Clark und Fourastie (vgl. [ReMö95]). Diese Hypothese prognostiziert eine nicht umkehrbare, stetig voranschreitende Verschiebung der Beschäftigung von der Agrarproduktion (primärer Sektor) über die Industrieproduktion (sekundärer Sektor) hin zu der Dienstleistungsproduktion (tertiärer Sektor).

Auch im Bereich der Internet-Technologien wird dem Dienstleistungsbereich in den letzten Jahren eine erhöhte Aufmerksamkeit zuteil, bspw. im Umfeld des Application Service Providing (ASP), in letzter Zeit jedoch verstärkt im Umfeld von Web-Services. Web-Services stellen dabei eine auf allgemein akzeptierten Standards basierende Technologie dar, um digitale Dienstleistungen, die bestimmte Applikationen zur Verfügung stellen, beschreiben, veröffentlichen und aufrufen zu können.

Im WWW zeichnet sich in letzter Zeit verstärkt der Trend ab, qualitativ hochwertigen digitalen Web-Content nur noch gegen Bezahlung zugänglich zu machen[1]. Hierfür sind diverse Systeme, wie spezielle Online-Shops für digitale Produkte [Bole02], und Technologien, wie bspw. Digital-Rights-Management-Technologien (DRM) [Iann01], entwickelt worden. Dahingegen sind in die Web-Service-Technologien bisher keine Mechanismen integriert, um die darüber angebotenen Dienste kostenpflichtig gestalten zu können.

Mit genau diesem Aspekt setzt sich diese Arbeit auseinander. Nachdem in Abschnitt 2 einige Begriffsklärungen vorgenommen werden, stellt Abschnitt 3 Geschäftsmodelle für digitale Dienstleistungen im Allgemeinen und für Web-Services im Speziellen vor. In Abschnitt 4 wird der Kern eines auf diesen Geschäftsmodellen basierenden Softwaresystems für die Realisierung kostenpflichtiger Web-Services skizziert. Anbieter können mit dem System Angebote für ihre Dienste erstellen und darauf basierende Zugriffslizenzen verkaufen. Eine rechtmäßige Nutzung ihrer Dienste wird durch das System sichergestellt. Abschnitt 5 geht kurz auf den aktuellen Einsatzbereich des Systems ein. Abschnitt 6 enthält einen Vergleich mit anderen Ansätzen. Die Arbeit schließt in Abschnitt 7 mit einer Zusammenfassung und einer kurzen Auflistung komplexerer Eigenschaften des Systems, die in diesem Artikel aus Platzgründen nicht im Detail besprochen werden können.

2 Web-Services als digitale Dienstleistungen

In diesem Abschnitt wird der Zusammenhang zwischen traditionellen Dienstleistungen, digitalen Dienstleistungen und Web-Services verdeutlicht.

2.1 Dienstleistungen

Dienstleistungen lassen sich charakterisieren als spezielle Funktionen bzw. Prozesse, die von Leistungsnehmern initiiert und von Leistungserbringern unter Nutzung von Kapazitäten und Ressourcen ausgeführt werden. Dabei werden durch den Leistungsnehmer externe Produktionsfaktoren wie Information, Güter oder Menschen in den Prozess eingebracht und gegebenenfalls manipuliert. Unter Umständen bringt die Dienstleistung als Ergebnis ein neues Produkt hervor, das dem Leistungsnehmer ausgeliefert wird (vgl. bspw. [Cors01]).

Nach [Male97] lassen sich bezüglich der Erbringung einer Dienstleistung drei Phasen unterscheiden: Angebot, Verrichtung und Ergebnis. In der Angebotsphase stellt ein Leistungserbringer sein Dienstleistungspotential zur Verfügung und signalisiert dadurch die Bereitschaft, eine bestimmte Dienstleistung erbringen zu

[1] vgl. bspw. http://www.heise.de/newsticker/data/jk-09.04.02-001/ (Abruf am 2003-05-06)

können. In der Verrichtungsphase wird der eigentliche Dienstleistungsprozess ausgeführt. Die dritte Phase steht schließlich für das Ergebnis der Dienstleistung, das im Allgemeinen an Veränderungen eingebrachter Objekte oder in Form neu erstellter Produkte sichtbar wird.

Die wesentlichen Unterschiede zwischen Dienstleistungen und Gütern bestehen darin, dass Dienstleistungen Prozesse und keine Objekte sind. Sie werden auf Anfrage eines Leistungsnehmers hin jedes Mal neu erbracht und nicht auf Vorrat für einen anonymen Markt erzeugt und gelagert.

2.2 Digitale Dienstleistungen

Als "digitale Dienstleistungen" werden im Folgenden Dienstleistungen bezeichnet, bei denen der Leistungserstellungsprozess im Wesentlichen unter Zuhilfenahme elektronischer Medien erfolgt. Externe Faktoren werden mittels elektronischem Datenaustausch integriert, d.h. zwischen dem Erbringer und dem Empfänger der Dienstleistung kommt es zu keinem physischen Kontakt (vgl. auch die Definitionen in [Brun02]).

Aus softwaretechnischer Sicht lassen sich digitale Dienstleistungen als Menge von Prozeduren bzw. Funktionen charakterisieren. Über Parameter werden digitale Objekte durch den Leistungsnehmer in den Prozess eingebracht, Ergebnisse werden in Form von digitalen Objekten als Funktionsrückgabewerte zurückgeliefert. Bei der Ausführung einer Funktion werden Ressourcen (CPU-Zeit, Speicher, Bandbreite, u.U. auch menschliche Ressourcen, ...) in Anspruch genommen. Die Ausführung kann sowohl Manipulationen an den übergebenen Objekten als auch interne Zustandsänderungen sowie weitere Seiteneffekte bewirken.

2.3 Web-Services

Web-Services sind Softwarekomponenten, die die Fähigkeit besitzen, Software-Funktionen im Internet bereitzustellen und auszuführen. Im Grunde handelt es sich beim Web-Service-Konzept um eine neue Methode des *Distributed Computing*, bei der durch Verwendung von Standard-Internettechnologien eine Plattform- und Programmiersprachenunabhängigkeit erreicht wird [KuWö02].

Ein Web-Service besteht aus zwei Teilen: der eigentlichen Implementierung sowie einer Beschreibung der angebotenen Funktionen. Als Beschreibungssprache wird im Allgemeinen die *Web-Service Description Language* (WSDL) eingesetzt. Die Beschreibungen werden entweder direkt durch den Anbieter des Web-Service oder indirekt über ein Verzeichnis veröffentlicht und damit anderen zugänglich und recherchierbar gemacht. Als Standard für derartige Verzeichnisse hat sich inzwischen die *Universal Description, Discovery and Integration* (UDDI) etabliert. Der Zugriff auf Web-Services sowie der Datenaustausch mit der aufrufenden An-

wendung erfolgt über das *Simple Object Access Protocol* (SOAP). WSDL, UDDI und SOAP basieren auf XML.

Mit den vorgestellten Konzepten stellen Web-Services damit eine moderne Technologie dar, mit deren Hilfe digitale Dienstleistungen beschrieben, veröffentlicht, erkundet, aufgerufen und ausgeführt werden können.

3 Geschäftsmodelle für Web-Services

Ein Geschäftsmodell legt insbesondere fest, auf welche Art und Weise ein Unternehmen Umsatz und Gewinn erzielen will. Zentrale Faktoren sind hierbei die Erlösformen und Zahlungsaspekte. Auf diese beiden Faktoren wird in diesem Abschnitt in Bezug auf digitale Dienstleistungen im Allgemeinen und in Bezug auf Web-Services im Speziellen eingegangen.

3.1 Erlösformen

Die Erlösform legt die Finanzierung einer Geschäftstätigkeit fest. Sie beinhaltet vor allem, von wem tatsächliche Einnahmen stammen. Hierbei lassen sich direkte von indirekten Erlösformen unterscheiden. Bei direkten Erlösformen stammen die Einnahmen von den Kunden selbst, bei indirekten Erlösformen von Dritten (z.B. Subventionierung oder Werbung). Im Folgenden werden aus Platzgründen nur direkte Erlösformen betrachtet. Eine detaillierte Diskussion von indirekten Erlösformen für Web-Content und -Services findet sich in [Bole02] bzw. [Schm02].

Zur Nutzung digitaler Dienstleistungen können Kunden Nutzungsrechte an digitalen Dienstleistungen erwerben, so genannte "Lizenzen". In der Literatur hat sich daher auch der Begriff "Lizenzmodelle" etabliert [Webe00]:

- **Zeitlizenzen** spezifizieren eine bestimmte Zeitspanne, in denen der Kunde beliebig oft eine bestimmte Menge an digitalen Dienstleistungen in Anspruch nehmen kann. Umfasst die Menge alle digitalen Dienstleistungen eines Anbieters, wird häufig auch der Begriff "Subskription" verwendet.
- Bei **Mengenlizenzen** bezahlt der Kunde für eine bestimmte Anzahl an Nutzungen einer oder mehrerer digitaler Dienstleistungen. Ist die Anzahl gleich Eins, spricht man auch vom Lizenzmodell "pay-per-use".
- **Intensitätslizenzen** bilden eine Mischform, bei der nach Anzahl der Nutzungen pro festgelegter Zeiteinheit abgerechnet wird.

Alle aufgeführten Erlösformen sind nicht nur für digitale Dienstleistungen, sondern auch für den Erwerb digitaler Produkte bzw. den Erwerb von Rechten an digitalen Produkten geeignet. Digitale Dienstleistungen unterscheiden sich jedoch in

zwei Eigenschaften von digitalen Produkten, die Auswirkungen auf den Erwerb von Lizenzen haben können:

- Die Ausführung digitaler Dienstleistungen kann abhängig sein von den durch den Leistungsnehmer übergebenen Objekten (sowohl Anzahl als auch Werte). Das kann bedingen, dass die tatsächlich anfallenden Kosten erst unmittelbar nach Aufruf der entsprechenden Funktion berechnet werden können. Wird zum Beispiel eine Funktion als digitale Dienstleistung angeboten, die eine Menge von als Parameter übergebener Objekte in eine Datenbank beim Anbieter abspeichert, so können sich die Kosten an der zur Laufzeit übergebenen Anzahl an Objekten orientieren. Diesbezüglich haben wir so genannte **Parameterlizenzen** eingeführt. Hierbei erwirbt ein Kunde eine bestimmte Menge von Bytes, die dann beim Aufruf derartiger Funktionen bezüglich der tatsächlich übergebenen Objekte verrechnet werden.
- Digitale Dienstleistungen sind keine festen Objekte, sondern zeitabhängige Prozesse, die jedes Mal individuell ausgeführt werden. Die Kosten können dabei abhängig sein von der tatsächlichen Inanspruchnahme bestimmter Ressourcen (CPU-Zeit, Speicher, Bandbreite, ...), d.h. die tatsächlich anfallenden Kosten stehen in diesem Fall erst nach Beendigung der Ausführung der digitalen Dienstleistung fest. Für diesen Fall verwenden wir so genannte **Ressourcenlizenzen**, bei denen ein Kunde bspw. eine Lizenz mit einer bestimmten Menge an CPU-Zeit oder Speicher erwerben muss, die vor der Rückgabe von Ergebnissen der verrichteten digitalen Dienstleistung verrechnet werden.

3.2 Bezahlung und Abrechnung

Wichtige Aspekte der Bezahlung der Inanspruchnahme digitaler Dienstleistungen eines Anbieters sind der Zahlungszeitpunkt und das Zahlungsverfahren.

In Abhängigkeit davon, ob die Zahlung vor oder nach der Leistungserfüllung durch den Anbieter erfolgt, können Debit- und Kredit-Zahlungen unterschieden werden. Debitzahlungen gelten dabei als anbieterfreundlich, da für den Händler die Bezahlung sichergestellt ist. Im Gegensatz dazu sind Kreditzahlungen kundenfreundlich, da die Bezahlung erst nach Erhalt der Leistung erfolgen muss. Einen Mittelweg bildet die Einschaltung eines Trust-Centers, das sowohl dem Anbieter als auch dem Käufer die Erfüllung des Kaufvertrages garantiert.

Verfahren und Systeme für die Bezahlung im Internet sind in den letzten Jahren zu hunderten entwickelt worden (vgl. [Webe00]). Zur Zeit dominieren Zahlungen per Kreditkarte. Wegen der relativ hohen Transaktionskosten ist die Zahlung per Kreditkarte jedoch nur im Bereich ab ca. 5 Dollar bzw. Euro geeignet. Die Kosten für einzelne digitale Dienstleistungen werden allerdings in der Regel unter diesem Betrag liegen. Für derart kleine Beträge würde sich bspw. elektronisches Geld eig-

nen. Entsprechende Systeme (z.B. eCash und CyberCoin) konnten sich jedoch (bisher) nicht durchsetzen.

Etabliert haben sich im Mikropaymentbereich Verfahren, bei denen nicht jede einzelne Nutzung direkt bezahlt wird, sondern der Anbieter interne Kundenkonten führt, auf denen anfallende Kosten registriert werden. Nach bestimmten Zeiträumen oder beim Überschreiten eines bestimmten Betrages wird dann per Kreditkarte oder Rechnung das Geld eingefordert. Interne Konten sind dabei sowohl für Debit- als auch für Kreditzahlungen geeignet. Bei Debitzahlungen muss das interne Konto zunächst aufgeladen werden. Bei Kreditzahlungen kann dem internen Konto seitens des Anbieters ein bestimmter Kreditbetrag eingeräumt werden. Interne Konten können natürlich auch bei einem Trust-Center anstelle beim Anbieter selbst geführt werden.

4 Konzeption und Implementierung kostenpflichtiger Web-Services

Im Rahmen eines Forschungsprojektes haben wir ein Softwaresystem entwickelt, das das Anbieten von Web-Services in Form kostenpflichtiger digitaler Dienstleistungen unterstützt [Schm02]. Das System, das in Java implementiert wurde, besteht aus drei Komponenten:

- einer Spezifikationskomponente zum Erstellen und Verwalten von Angeboten durch Anbieter,
- einer Geschäftskomponente zum Erwerb von Lizenzen durch Kunden sowie
- einer Einsatzkomponente zur Implementierung der kostenpflichtigen Funktionen sowie zur Kontrolle ihrer rechtmäßigen Nutzung durch fremde Applikationen.

Abbildung 1 skizziert das Zusammenspiel und die Funktionalität dieser Komponenten, die im Folgenden genauer beschrieben werden.

4.1 Spezifikation von Angeboten

Soll ein Web-Service bzw. eine einzelne Funktion eines Web-Service in Form einer digitalen Dienstleistung kostenpflichtig angeboten werden, sind neben seiner Adresse und seiner Signatur, die in einer entsprechenden WSDL-Beschreibung angegeben werden, Angaben zum zugrunde liegenden Geschäftsmodell notwendig. Das heißt, vom Leistungserbringer oder von einem separaten Anbieter müssen konkrete Angebote spezifiziert werden, die Kunden (Personen oder auch Applikationen) einsehen und verstehen können müssen.

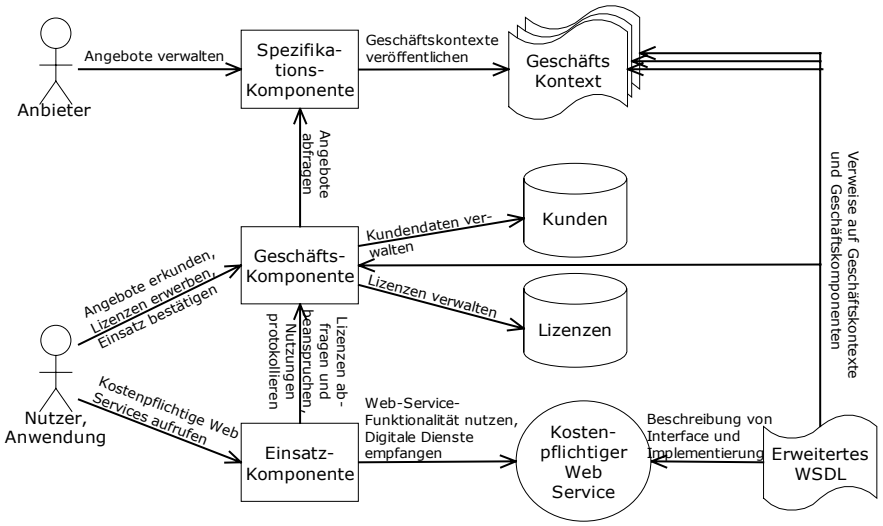


Abbildung 1: Komponenten und deren Zusammenspiel

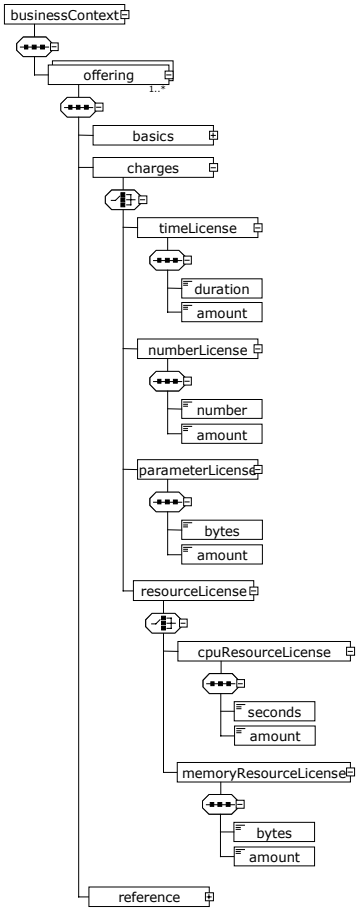
Zur Spezifikation derartiger Geschäftskontexte haben wir ein XML-Schema entwickelt, von dem eine vereinfachte Version im Folgenden erläutert und in Abbildung 2 skizziert wird. Ein Geschäftskontext (*businessContext*) setzt sich dabei aus ein oder mehreren Angeboten (*offering*) zusammen. Jedes Angebot besitzt einen eindeutigen Identifikator. Es besteht aus einem allgemeinen Teil (*basics*), einem Teil, der speziell den Kosten gewidmet ist (*charges*) sowie einem dritten Teil, der die Web-Services und deren Funktionen auflistet, für die dieses Angebot gilt (*reference*).

Erforderliches Element des *basic*-Teils ist die Angabe eines Zeitraumes, in dem das Angebot gültig ist (*validityInterval*). Als optionale Elemente können u.a. Informationen zum Anbieter (*provider*) und zur Dienstleistung selbst (*serviceInfo*) angegeben werden.

Im *charges*-Teil werden die Erlösformen und die Bezahlung festgelegt. Kosten werden dabei immer in Bezug auf eine Währung (*currency*) angegeben. Zeitlizenzen (*timeLicense*) sind bspw. durch Angabe der Kosten (*amount*) für ein bestimmtes Zeitintervall (*duration*) spezifizierbar, für Mengenlizenzen (*numberLicense*) sind die Kosten (*amount*) pro Nutzungen (*number*) anzugeben. Parameterlizenzen (*parameterLicense*) erfordern die Festlegung eines Betrages (*amount*) für eine bestimmte Menge an Parametern in Bytes (*bytes*). Bei Ressourcenlizenzen (*resourceLicense*) spezifiziert der Anbieter die Kosten (*amount*) für eine bestimmte Menge an Ressourcen. In Abbildung 2 sind als Beispiele für konkrete Ressourcenlizenzen CPU- (*cpuResourceLicense*) und Speicherplatz-Ressourcenlizenzen (*memoryResourceLicense*) eingezeichnet. Ein Kunde kann nun eine Lizenz mit ei-

ner gewünschten Anzahl an Ressourcen erwerben (bspw. eine Lizenz mit 1000 CPU-Sekunden) und diese dann nach und nach aufbrauchen.

Ein Angebot bezieht sich auf eine, mehrere oder alle Funktionen eines oder auch mehrerer Web-Services. Deren Aufzählung ist im reference-Teil notwendig.




```
<?xml version="1.0" encoding="UTF-8"?>
<businessContext ...>
 <offering ID="Square-001">
  <basics>
   <validityInterval>
    <begin date="2003-01-01T00:00:00"/>
    <end date="2003-12-31T23:59:59"/>
   </validityInterval>
  </basics>
  <charges>
   <numberLicense>
    <amount currency="EURO"> 0.20 </amount>
    <number> 1 </number>
   </numberLicence>
  </charges>
  <reference>
   <webService>
    <serviceName> SquareService </serviceName>
    <serviceLocation>
     http://service.provider.com/SquareService
    </serviceLocation>
    <operations>
     <individualOperation>
      square
     </individualOperation>
    </operations>
   </webService>
  </reference>
 </offering>
</businessContext>
```

Abbildung 2: Schema-Spezifikation eines Geschäftskontextes

Im rechten Teil von Abbildung 2 wird als Beispiel die Funktion *square* eines Web-Service *SquareService* in Form einer Mengenlizenz für 0.20 Euro pro Nutzung angeboten. Das Angebot ist gültig zwischen dem 01.01.2003 und dem 31.12.2003. Auf der Basis dieses Angebots kann nun ein Kunde eine entsprechende Lizenz mit einer gewünschten Anzahl *n* an Nutzungen für *n*amount* EURO erwerben.

4.2 Veröffentlichung von Angeboten

Geschäftskontexte werden in einem separaten Dokument spezifiziert und nicht direkt in die WSDL-Spezifikation eines Web-Service integriert. Dadurch ist es zum einen möglich, Angebote zu erstellen, die sich auf mehrere Web-Services unter Umständen mehrerer Leistungserbringer beziehen (Bündelung). Zum anderen sind dadurch auch Geschäftsszenarien denkbar, bei denen mehrere Anbieter die Dienstleistungen eines Leistungserbringers zu unterschiedlichen Konditionen anbieten können.

Um sowohl auf Dokumente, die Angebote zu einem bestimmten Web-Service enthalten, als auch auf Geschäftsumgebungen zu verweisen, in denen entsprechende Lizenzen erworben werden können, wird die WSDL-Spezifikation jedoch geringfügig erweitert. Sämtliche neuen Informationen werden zu einem *business-Information*-Element zusammengefasst und dieses auf der Ebene der *definitions* eingefügt. Es enthält Verweise auf Geschäftskomponenten (*businessComponent*), bei denen Lizenzen auf Basis zugehöriger Geschäftskontexte (*contextAt*) erworben werden können.

4.3 Spezifikationskomponente

Eine Spezifikationskomponente stellt einem Anbieter kostenpflichtiger Web-Services Funktionen zum Erstellen neuer Angebote, zur Verwaltung bereits zuvor erstellter Angebote und zum Veröffentlichen von Angeboten zur Verfügung. Hierfür wurde ein Java-Servlet entwickelt, über das ein Zugriff auf die Spezifikationskomponente über Standard-WWW-Browser möglich ist. Für Geschäftskomponenten wird eine Web-Service-Schnittstelle bereitgestellt, um gültige Angebote abfragen zu können.

4.4 Geschäftskomponente

Kunden können über Geschäftskomponenten Angebote zu kostenpflichtigen Web-Services einsehen und auf deren Basis konkrete Lizenzen erwerben. Dies kann via WWW-Browser über ein entsprechendes Java-Servlet geschehen, und zwar ähnlich wie beim Erwerb von Produkten in Online-Shops. Für einen automatisierten Lizenzerwerb durch Applikationen wurde weiterhin eine Web-Service-Schnittstelle entwickelt.

Eine Geschäftskomponente verwaltet erworbene Lizenzen sowie Kundendaten. Für die Bezahlung und Abrechnung ist die Geschäftskomponente mit der Funktionalität ausgestattet, interne Kundenkonten führen zu können. Anfallende Kosten werden hierüber verrechnet. Die Komponente ist jedoch erweiterbar gestaltet und kann relativ einfach an konkrete Zahlungsverfahren bzw. -systeme angepasst werden.

4.5 Einsatzkomponente

Der Aufruf einer kostenpflichtigen Web-Service-Funktion wird durch eine Einsatzkomponente gekapselt und überwacht. Ruft eine Anwendung eine solche Funktion auf, kontaktiert die Einsatzkomponente eine zugehörige Geschäftskomponente, die überprüft, ob für den Kunden eine entsprechende gültige Lizenz existiert, und die (optional) vom Kunden eine Autorisierung der Lizenzbeanspruchung einholt, bspw. durch Eingabe einer PIN. Existiert eine gültige Lizenz und fällt die Autorisierung positiv aus, werden der Geschäftskomponente die Inanspruchnahme der Lizenz gemeldet und die Ausführung der Funktion gestartet. Im anderen Fall werden Exceptions an die aufrufende Anwendung zurückgegeben. Ist keine gültige Lizenz vorhanden, enthält die Exception neben einer entsprechenden Fehlermeldung Verweise auf Geschäftskomponenten, bei denen eine Lizenz erworben werden kann. Für die Überprüfung ist es erforderlich, jeder kostenpflichtigen Funktion als Parameter ein Kundenobjekt zu übergeben, über das der Kunde identifiziert werden kann.

Setzt die Ausführung einer Funktion eine Ressourcenlizenz voraus, wird zunächst die Funktion vollständig ausgeführt. Dabei werden die in Anspruch genommen Ressourcen intern vermerkt. Existiert eine ausreichende Ressourcenlizenz, wird deren Inanspruchnahme der Geschäftsumgebung gemeldet und anschließend wird das Funktionsergebnis der Anwendung zurückgeliefert. Ist keine ausreichende Ressourcenlizenz vorhanden, wird das Ergebnis zurückgehalten und stattdessen wie oben beschrieben eine Exception geworfen. Für diesen Fall sollte der Web-Service eine zusätzliche Funktion bereitstellen, um nach Erwerb einer ausreichenden Ressourcenlizenz, das gespeicherte Ergebnis abfragen zu können, ohne erneut die komplette Funktion ausführen zu müssen. Der komplette Web-Service kann dabei für diesen Kunden solange gesperrt werden, bis er die entsprechende Ressourcenlizenz erworben hat. Unter Umständen sind in die Web-Service-Implementierung Funktionen zu integrieren, die nach einer gewissen Zeit, in der der Kunde der Bezahlung nicht nachgekommen ist, während der Ausführung der Funktion hervorgerufene Zustandsänderungen wieder rückgängig machen.

Es kann natürlich vorkommen, dass ein Kunde K eine kostenpflichtige Funktion F1 eines Anbieters A1 in Anspruch nimmt, bei deren Abarbeitung wiederum eine weitere kostenpflichtige Funktion F2 eines Anbieters A2 aufgerufen wird. In diesem Fall ist eine Implementierung der Funktion F1 auf zweierlei Art und Weise möglich. Zum einen kann das Kundenobjekt an die Funktion F2 weitergeleitet werden, d.h. der Kunde K muss entsprechende Lizenzen beim Anbieter A2 besitzen oder erwerben. Zum anderen kann jedoch auch Anbieter A1 durch Instantiierung und Übergabe eines entsprechenden Kundenobjektes als Kunde gegenüber Anbieter A2 auftreten und bspw. im Voraus die Dienste von A2 subskribieren oder entsprechende Mengenlizenzen für die Funktion F2 erwerben. Die ihm hierbei entstehenden Kosten hat er dann bereits bei der Spezifikation eines Angebotes für F1 zu berücksichtigen.

4.6 Beispiel

Als kostenpflichtige Funktion eines Web-Service soll die bereits in Abschnitt 4.1 skizzierte Funktion *square* zu den dort spezifizierten Konditionen angeboten werden. Die Funktion berechnet und liefert das Quadrat eines ihr als Parameter übergebenen *int*-Wertes.

Für die Implementierung der Funktion muss von der Klasse *Function*, die von der Einsatzkomponente bereitgestellt wird, eine Klasse (hier *SquareFunction*) abgeleitet und darin eine Methode *calculate* überschrieben werden. Diese führt die eigentliche Berechnung durch.

```
public class SquareFunction extends Function {
  ...
  public void calculate() {
    ...
    int value = ((Integer)this.param[0]).intValue();
    this.ergebnis = new Integer(value * value);
    ...
  }
}
```

Die Klasse *Function* implementiert (und vererbt) eine Methode *execute*. Diese Methode überprüft zunächst durch Kommunikation mit einer Geschäftskomponente die Rechtmäßigkeit eines Aufrufs. Ist bspw. keine gültige Lizenz vorhanden, wirft sie eine *LicenseException* (diese Klasse sowie weitere Fehlerklassen werden ebenfalls von der Einsatzkomponente zur Verfügung gestellt). Existiert eine gültige Lizenz wird intern die dynamisch gebundene Methode *calculate* aufgerufen.

Für andere Lizenztypen als die in diesem Beispiel benutzen Mengenlizenzen stellt die Einsatzkomponente weitere von der Klasse *Function* abgeleitete spezialisierte Klassen zur Verfügung.

Weiterhin muss eine Funktion *square* implementiert und als Web-Service veröffentlicht werden.

```
public int square(Customer customer, int value)
                  throws LicenseException {
  ...
  SquareFunction f =
    new SquareFunction(businessComponent,
```

```
                        "SquareService", "http://...",
                        "square");
  Object[] parameter = new Object[1];
  parameter[0] = new Integer(value);
  f.execute(customer, parameter);
  int result = ((Integer)f.getResult()).intValue();
  return result;
}
```

Als Parameter bekommt die Funktion Informationen zu dem Kunden (*Customer*) sowie den zu quadrierenden Wert übergeben. Innerhalb der Funktion wird zunächst ein Objekt der Klasse *SquareFunction* erzeugt. Danach wird die oben beschriebene Methode *execute* des Objektes aufgerufen. Zum Schluss wird das Ergebnis abgefragt und zurückgeliefert.

Die zugehörige WSDL-Beschreibung hat folgende Gestalt:

```
<?xml version="1.0" encoding="UTF-8">
<wsdl:definitions ...>
  <wsdl:types>
    ...
    <complexType name="tns1:Customer">
      ...
    </complexType>
    <complexType name="tns1:LicenseExc">
      ...
    </complexType>
    ...
  </wsdl:types>
  <wsdl:message name="squareRequest">
    <wsdl:part name="customer" type="tns1:Customer"/>
    <wsdl:part name="value" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="squareResponse">
```

```
    <wsdl:part name="return" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="LicenseException">
  ...
  </wsdl:message>
  <wsdl:portType name="SquareService">
    <wsdl:operation name="square"
                    parameterOrder="customer value">
      <wsdl:input message="intf:squareRequest"
                  name="squareRequest"/>
      <wsdl:output message="intf:squareResponse"
                   name="squareResponse"/>
      <wsdl:fault message="intf:LicenseExc"
                  name="LicenseExc"/>
    </wsdl:operation>
  </wsdl:portType>
  ...
  <wsdl:service name="SquareService">
  ...
  </wsdl:service>
  <bcx:businessInformation>
    <bcx:businessComp> http://... </bcx:businessComp>
    <bcx:contextAt>
      http://.../offering1.bcx
    </bcx:contextAt>
  </bcx:businessInformation>
</wsdl:definitions>
```

Der Aufruf der kostenpflichtigen Web-Service-Funktion *square* mittels JAX-RPC[2] wird im Folgenden angedeutet:

```
try {
  ...
  Customer customer = ...
  int value = ...
  SquareServiceIF service =
    new SquareServiceImpl().getSquareServiceIF();
  int result = service.square(customer, value);
  ...
} catch (LicenseExc exc) {
  // Lizenz erwerben und u.U. erneut aufrufen
} ...
```

5 Aktuelles Einsatzgebiet

Eingesetzt wird das beschriebene System aktuell in einem Forschungsprojekt, in dem ein verteiltes Lernmanagementsystem entwickelt wird. Lernmanagementsysteme (LMS) sind spezielle Softwaresysteme für die Organisation und Betreuung webbasierten Lernens. Sie bieten insbesondere Funktionalitäten zur Präsentation von Lehrinhalten (Folien, multimediale Anwendungen, ...), zur Kommunikation (Diskussionsforen, Chats, ...), zur Durchführung von Tests (Generierung, Korrektur, ...), zur Veranstaltungsorganisation (Kalender, Aufgaben, Nutzerverwaltung, ...) und zur Evaluation von Lehrveranstaltungen (Statistiken, Umfragen, Noten, ...) an [Bau+02].

Aktuell existieren weit über 100 LMS[3]. Darunter finden sich sowohl Open-Source-LMS, wie ILIAS[4], als auch kommerzielle LMS, wie Blackboard[5] oder WebCT[6], für deren Lizensierung eine mittelgroße Universität jährliche Kosten von 50.000 Euro und mehr veranschlagen muss.

[2] http://java.sun.com/xml/jaxrpc/ (Abruf am 2003-05-06)
[3] siehe bspw. http://iol3.uibk.ac.at/virtuallearning/directory/214/listeGesamt (Abruf am 2003-05-06)
[4] http://www.ilias.uni-koeln.de/ios/index.html (Abruf am 2003-05-06)
[5] http://www.blackboard.com/ (Abruf am 2003-05-06)
[6] http://www.webct.com/ (Abruf am 2003-05-06)

Unsere Erfahrungen beim Einsatz von LMS in der Hochschullehre haben dabei Folgendes gezeigt:

- Viele Veranstalter nutzen nur eine kleine Teilmenge der vom LMS bereitgestellten Funktionalitäten (so dass die hohen Kosten nicht gerechtfertigt sind).
- Einzelne LMS-Komponenten decken nicht immer alle Anforderungen ab, die ein Lehrender für seine Veranstaltung braucht (aber die LMS sind in der Regel nicht einfach anpassbar).

Diese Probleme versuchen wir durch die Entwicklung eines verteilten, komponentenbasierten, konfigurierbaren LMS zu beseitigen (siehe Abbildung 3).

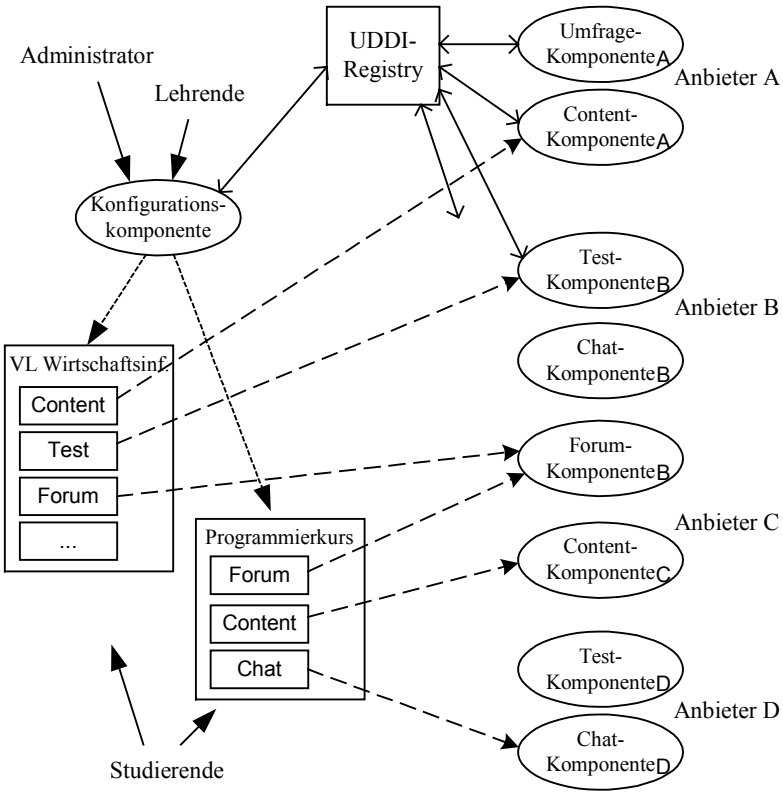


Abbildung 3: Ein verteiltes, komponentenbasiertes, konfigurierbares LMS

Hierbei können verschiedene Anbieter Komponenten implementieren und auf ihren Servern bereitstellen, die einzelne LMS-Funktionalitäten kapseln. Beispiele für derartige Komponenten sind Content-Management-Komponenten, Kalenderkomponenten, Diskussionsforenkomponenten oder auch Test-Komponenten. Eine von uns entwickelte Konfigurationskomponente ermöglicht es Administratoren

oder auch Lehrenden selbst, aus diesen bereitstehenden Komponenten einen webbasierten Kurs so zusammenzustellen, wie sie es für ihre spezifischen Belange benötigen.

Die Realisierung der Komponenten erfolgt durch den Einsatz der Web-Service-Technologien. Dabei können die Komponentenanbieter das in Abschnitt 4 beschriebene System nutzen, um ihre Dienste kostenpflichtig anzubieten. Bspw. könnte ein Anbieter eines Chats Zeitlizenzen einsetzen: Für jede Minute, die ein Nutzer im Chat aktiv ist, fallen Kosten von 0,01 Euro an. Oder eine Content-Management-Komponente rechnet über Parameterlizenzen ab: Pro Megabyte upgeloadeter Dateien ist 1,00 Euro zu bezahlen. Diskussionsforen könnten bspw. Mengenlizenzen nutzen (Lesen eines Diskussionsbeitrags: 0,01 Euro, Schreiben eines Beitrags: 0,02 Euro) oder sie könnten auf einer Subskription basieren (uneingeschränktes Nutzen des Forums durch einen Kurs für ein Semester für 100,00 Euro).

Bezahlen wird im Allgemeinen die Hochschule bzw. Institution, die die konfigurierten Kurse einsetzt. Es ist aber auch möglich, die Kosten (zumindest teilweise) auf die Studierenden zu verlagern.

6 Vergleich mit anderen Ansätzen

Es existieren bisher nur wenige alternative Ansätze, elektronische Dienstleistungen bzw. Web-Services kostenpflichtig zu gestalten. So ermöglicht bspw. das Zahlungsverfahren *click&buy* der Firma *Firstgate*[7] Anbietern, beim Zugriff auf URLs Gebühren zu verlangen. Hinter den URLs können sich dabei sowohl digitale Produkte (PDF-Dokumente, mp3-Dateien, ...) als auch Dienstleistungen (Suche in Datenbanken, Telefonauskunft, ...) verbergen. Ein weiteres Beispiel ist *XrML*, eine XML-basierte Sprache für die Spezifikation von Rechten bei der Nutzung digitaler Ressourcen[8]. Mit Hilfe von XrML können Anbieter beschreiben, wer unter welchen Bedingungen welche Nutzungsrechte beim Zugriff auf digitale Objekte bzw. Services hat. Zu den Bedingungen zählen dabei auch Geschäftsbedingungen, insbesondere Kosten.

Anders als in dem hier vorgestellten Ansatz werden in diesen Ansätzen digitale Objekte und digitale Dienstleistungen jedoch gleich behandelt. Es wird nicht berücksichtigt, dass digitale Dienstleistungen keine festen Objekte sondern Prozesse sind, die bei jedem Zugriff neu und individuell ausgeführt werden. Die existierenden Ansätze ermöglichen nicht die Berechnung von Kosten in Abhängigkeit der tatsächlich verbrauchten Ressourcen bzw. der tatsächlich übergebenen Parameter

[7] http://www.firstgate.de (Abruf am 2003-05-06)
[8] http://www.xrml.org (Abruf am 2003-05-06)

und sind im Gegensatz zum vorgestellten Ansatz gar nicht bzw. wenig homogen in die Web-Service-Technologien integriert.

Damit kostenpflichtige Web-Services vor unbefugten Zugriffen geschützt sind, verwaltet in unserem Ansatz eine Geschäftskomponente Informationen zu Konsumenten und weist ihnen insbesondere entsprechende Rechte zu, sobald Zahlungsmodalitäten geklärt sind.

Einen ähnlichen Ansatz verfolgt die *Organization for the Advancement of Structured Information Standards* (OASIS). Dort entwickelt das Provisioning Services Technical Committee[9] (PSTC) die *Service Provisioning Markup Language* (SPML) [Karp02]. Dabei handelt es sich um ein XML-basiertes Framework, das Zugriffsrechte von Nutzern oder Applikationen, bezogen auf Dienste oder andere Ressourcen, an zentraler Stelle verwaltet und Zugriffe darauf ermöglicht. Die SPML baut dazu auf drei Säulen auf, nämlich dem *Active Digital Profile*[10] (ADPr) zur Beschreibung von Ressourcen und Diensten für den geschäftlichen Ablauf, dem *eXtensible Resource Provisioning Management*[11] (XRPM) zum Zusammenbringen von Bereitstellungs- und Identifikationsdiensten sowie der *Information Technology Markup Language*[12] (ITML) zur Integration von Teilnehmern in Geschäftsprozesse [Coh+03]. Es stehen allerdings bislang keine Realisierungen bzw. Implementierungen zu diesem Ansatz zur Verfügung.

Das *Web Services Policy Framework* (WS-Policy) von IBM[13] stellt einen Ansatz dar, über den sich Fähigkeiten von Web-Services und Anforderungen an ihre Einsatzumgebung sowie an potentielle Nutzer beschreiben lassen [Box+02a]. Dabei ist es unwesentlich, ob die Anforderungen von der Umsetzung, z.B. zu verwendende Transportprotokolle oder Authentifikationsschemata, oder von der Anwendung, z.B. Zugriffsbeschränkungen oder Dienstgüte, abhängen. Es ist möglich, solche Policies analog zum hier vorgestellten Ansatz in separaten und wieder verwendbaren Dokumenten, so genannten *Attachments*, zu verwalten [Box+02b]. Auf diese kann dann aus einer WSDL-Beschreibung oder einem Eintrag im UDDI-Verzeichnis verwiesen werden. Jedoch existiert mit dem Policy-Konzept nur eine Grammatik zur Beschreibung spezieller Bedürfnisse von Web-Services und keine Umgebung, die deren Einhaltung bzw. bestimmte zu erfüllende Voraussetzungen prüfen kann.

Die *Web Services Interoperability Organization*[14] (WS-I) verfolgt das Ziel, insbesondere die Zusammenarbeit von Web-Services untereinander zu stärken, ihre An-

[9] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=provision (Abruf am 2003-05-06)
[10] http://www.adpr-spec.com (Abruf am 2003-05-06)
[11] http://xml.coverpages.org/xrpm.html (Abruf am 2003-05-06)
[12] http://xml.coverpages.org/itml.html (Abruf am 2003-05-06)
[13] http://www.ibm.com (Abruf am 2003-05-06)
[14] http://www.ws-i.org (Abruf am 2003-05-06)

nahme durch Dritte zu fördern sowie ihre Entwicklung und ihren Einsatz zu beschleunigen [Ehn+03]. Sie gibt Richtlinien zur Anwendung von Spezifikationen im Bereich Implementierung und Interoperabilität vor und entwickelt eine Testumgebung[15], mit der Web-Services auf Einhaltung dieser Richtlinien überprüft werden können. Produzenten von Web-Services können dann mit der Einhaltung dieser Richtlinien bei Entwicklung und Angebot werben, der Middleware-Bereich kann anbieten, zu diesen Richtlinien konforme Web-Services zu unterstützen, und Kunden, die Interoperabilität voraussetzen, können entsprechend konforme Web-Services nutzen. Diese Testumgebung befindet sich derzeit noch im Entwicklungsstadium.

7 Zusammenfassung und Anmerkungen

In diesem Artikel wurden ein Konzept und eine Implementierung vorgestellt, um die Ausführung von Web-Services kostenpflichtig gestalten zu können. Mit Hilfe der zur Verfügung gestellten Komponenten bzw. Klassen ist es auf der einen Seite für Leistungserbringer und Anbieter sehr einfach möglich, Web-Services so zu implementieren und anzubieten, dass damit Einnahmen erzielt werden können. Auf der anderen Seite können Anwendungsentwickler die kostenpflichtigen Web-Service-Funktionen ohne große Umstellungen gegenüber kostenfreien Funktionen in ihren Applikationen aufrufen und nutzen. Nutzer können schließlich mit Hilfe der Geschäftskomponente ähnlich wie in Online-Shops Lizenzen erwerben und somit letztendlich die Dienste in Anspruch nehmen.

Die hier vorgestellten Ergebnisse spiegeln jedoch aus Platzgründen nur den Kern unserer bisherigen Arbeiten wieder. Im vollständigen System ist die Angebotsspezifikation deutlich komplexer, als dies in Abschnitt 4.1 dargestellt wurde. So sind Kombinationen der aufgeführten Lizenzmodelle genauso möglich wie bspw. Rabatte, Sonderangebote oder kundenabhängige Angebote. Da finanzielle Transaktionen über das Internet ein hochsensibles Thema darstellen, kümmern wir uns aktuell um die Integration von Sicherheitsaspekten in unser Konzept.

Literaturverzeichnis

[Bau+02] Baumgartner, P., Häfele, H., Maier-Häfele, K.: E-Learning Praxishandbuch. Auswahl von Lernplattformen: Marktübersicht - Funktionen - Fachbegriffe. StudienVerlag Innsbruck, 2002.

[15] http://www.ws-i.org/Testing/Tools/2003/03/WSI_Test_Java_0.91_bin.zip (Abruf am 2003-05-06)

[Bole02] Boles, D.: Integration von Konzepten und Technologien des Electronic Commerce in digitale Bibliotheken. dissertation.de, 2002, http://www.dissertation.de/PDF/-db639.pdf, Abruf am 2003-05-06.

[Box+02a] Box, D., Curbera, F., Langworthy, D., et al: Web Services Policy Framework (WS-Policy), Draft vom 18.12.2002, http://www.ibm.com/developerworks/library/ws-polfram/, Abruf am 2003-05-06.

[Box+02b] Box, D., Curbera, F., Maruyama, H., et al: Web Services Policy Attachment (WS-PolicyAttachment), Draft vom 18.12.2002, http://www.ibm.com/developerworks/-library/ws-polatt/, Abruf am 2003-05-06.

[Brun02] Bruhn, M.: E-Services - eine Einführung in die theoretischen und praktischen Probleme. In: Bruhn, M., Stauss, B. (Hrsg.), Electronic Services - Dienstleistungsmanagement Jahrbuch 2002, Gabler, Wiesbaden, 2002, S. 3-41.

[Coh+03] Cohen, D., Sodhi, G., Lockhart, H., et al: OASIS Service Provisioning Markup Language (SPML), Draft Committee Specification, Version 0.08 vom 18.03.2003, http://www.oasis-open.org/committees/download.php/1772/draft-pstc-spml-core-09.doc, Abruf am 2003-05-06.

[Cors01] Corsten, H.: Dienstleistungsmanagement. Oldenbourg, 2001.

[Ehn+03] Ehnebuske, D., Ferris, C., Glover, T., et al: WS-I Overview, Whitepaper vom 15.01.2003, http://www.ws-i.org/docs/20030115.wsi.introduction.pdf, Abruf am 2003-05-06.

[Iann01] Iannella, R.: Digital Rights Management (DRM) Architectures. D-Lib Magazine 7(6), Juni 2001. http://www.dlib.org/dlib/june01/iannella/06iannella.html, Abruf am 2003-05-06.

[Karp02] Karpinski, R.: The Missing Link In Web Services Security: Provisioning, Artikel von InternetWeek.com, 29.03.2002, http://www.internetwk.com/newslead02/-lead032902.htm, Abruf am 2003-05-06.

[KuWö02] Kuschke, M., Wölfel, L.: Web Services kompakt. Spektrum Akademischer Verlag, 2002.

[Male97] Maleri, R.: Grundlagen der Dienstleistungsproduktion. Springer, 1997.

[ReMö95] Reichwald, R., Möslein, K.: Wertschöpfung und Produktivität von Dienstleistungen? - Innovationsstrategien für die Standortsicherung. In: Bullinger, H.-J., Dienstleistung der Zukunft - Märkte, Unternehmen und Infrastrukturen im Wandel, Gabler, Wiesbaden 1995, S. 324-376.

[Schm02] Schmees, M.: Kostenpflichtige Web-Services: Integration von Zahlungs- und Abrechnungsmechanismen in Web-Services. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, September 2002.

[Webe00] Weber, R.: Accounting and Payment Concepts for Fee-based Scientific Digital Libraries. Dissertation, TU München, 2000.