February 2005

WS-Specification: Ein Spezifikationsrahmen zur Beschreibung von Web-Services auf Basis des UDDI-Standards

Sven Overhage
Universität Augsburg

Peter Thomas
Technische Universität Darmstadt

In: Ferstl, Otto K, u.a. (Hg) 2005. *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety*; 7. Internationale Tagung Wirtschaftsinformatik 2005. Heidelberg: Physica-Verlag

ISBN: 3-7908-1574-8

WS-Specification: Ein Spezifikationsrahmen zur Beschreibung von Web-Services auf Basis des UDDI-Standards

Sven Overhage
Universität Augsburg

Peter Thomas
Technische Universität Darmstadt

Zusammenfassung: In diesem Beitrag wird ein Spezifikationsrahmen für die Beschreibung von Web-Services entwickelt. Dieser stellt eine wichtige Voraussetzung zur Unterstützung der Auswahl und Kopplung von Web-Services dar. Die Entwicklung des Spezifikationsrahmens, der den Namen WS-Specification trägt, wird durch eine Analyse des UDDI-Standards motiviert, die eine Reihe von zu behebenden Schwächen offenbart. Der hierzu neu entwickelte Spezifikationsrahmen basiert auf dem Konzept des Software-Vertrags und wahrt Kompatibilität zum UDDI-Standard. Mit ihm lassen sich Informationen über folgende Merkmale eines Web-Service bereitstellen: Allgemeine und kommerzielle Informationen (White Pages), Klassifikationen (Yellow Pages), fachliche Funktionalität (Blue Pages), logische Architektur (Green Pages) und physische Qualität (Grey Pages).

Schlüsselworte: Web-Service, Spezifikation, UDDI, WS-Specification, Software-Vertrag

1 Einleitung

Mit ihrem vergleichsweise einfach anmutendem Entwicklungsparadigma, nach dem Anwendungen durch *Auswahl* eines geeigneten Bündels von Diensten aus einem Katalog und deren anschließende *Kopplung* zu entwickeln bzw. (im Falle einer Anwendungsintegration) miteinander zu verbinden sind, verheißt die Web-Service-Technologie einen wichtigen Fortschritt auf dem Weg zur komponentenorientierten Anwendungsentwicklung [Bett01, S. 302; KoLe04, S. 117] bzw. Anwendungsintegration [Minz+02, S. 7; Beute02, S. 27f.], die maßgeblich zur Bewältigung der bis heute fortbestehenden Software-Krise beitragen sollen [McIl68].

Um diese Vision Wirklichkeit werden zu lassen, ist es von einem softwaretechnischen Standpunkt aus betrachtet notwendig, den mit dem neuen Entwicklungspa-

radigma eingeführten Kompositionsprozess, d.h. die *Auswahl* und *Kopplung* von Web-Services methodisch zu unterstützen [Minz+02, S. 11]. Hierbei gilt es, eine Vielzahl neuer Herausforderungen zu lösen und vor allem Methoden bzw. Werkzeuge für die *Suche*, die Durchführung von *Kompatibilitätstests*, die *Adaptergenerierung*, die *Kopplung* von Web-Services sowie für die *Vorhersage* der (funktionalen und qualitativen) Eigenschaften von Anwendungen, die durch die Kopplung von Web-Services entstehen, bereitzustellen [Minz+02, S. 11; Crnk02, S. 132f.].

Zum kritischen Erfolgsfaktor für die Entwicklung und Anwendung der vorgenannten Methoden wird dabei die Bereitstellung geeigneter *Spezifikationen*, die die von einem Web-Service angebotenen Dienste und die bei ihrer Inanspruchnahme jeweils zu erfüllenden Bedingungen in angemessener Weise beschreiben [HoJu02, S. 34]. Ohne die Verfügbarkeit solcher Informationen könnte die Komposition von Web-Services erst im Anschluss an eine Analyse ihrer jeweiligen Eigenschaften erfolgen. Diese wird jedoch dadurch erschwert, dass Web-Services ihre Implementierung verbergen. Die somit zur Analyse einzusetzenden Methoden des Testens und des Reverse Engineering verursachen einen hohen Aufwand und sind meist nicht in der Lage, die benötigten Informationen in akkurater Form zu liefern. Damit würden durch ihren Einsatz die in Aussicht gestellten Vorteile der Web-Services beeinträchtigt bzw. zunichte gemacht [Weyu01, S. 503; Garl+95, S. 25].

Bei der Entwicklung der Web-Service-Technologie [Boot+03; Aust+04] wurde diesem Umstand mit der Einführung entsprechender Standards bereits von Beginn an Rechnung getragen. So wird durch die Basis-Technologie (den sog. Basis-Stack [KoLe04, S. 120]) nicht nur das Protokoll zur Durchführung von Prozedurfernaufrufen, das sog. Simple Object Access Protocol (SOAP [Mitr03]), definiert, sondern auch die *Beschreibung* von Web-Service-Schnittstellen sowie die *Katalogisierung* von Web-Services standardisiert. Hierzu wurden mit der Web Services Description Language (WSDL [Chin+04]) und dem Universal Description, Discovery, and Integration Standard (UDDI [UDDI02]) entsprechende Vorgaben geschaffen, die bei der Entwicklung von Web-Services zur Anwendung kommen.

Für die Auswahl und Kopplung von Web-Services kommt vor allem dem *UDDI-Standard*, dessen Aufgabe die Bereitstellung von adäquaten Informationen über die einzelnen Web-Services ist [Caul+01, S. 181f.], eine besondere Bedeutung zu. Um diese Aufgabe zu erfüllen, definiert der UDDI-Standard neben einer standardisierten Schnittstelle für den Zugriff auf Informationen vor allem einen *Spezifikationsrahmen*, der die bereitzustellenden Informationen über Web-Services sowie die zur Beschreibung einzusetzenden Notationen vorgibt. Dieser Spezifikationsrahmen befindet sich derzeit in der Version 3.0 und gilt als weitgehend stabil, da nach seiner Einführung in der ersten Version bislang nur jeweils geringfügige Änderungen vorgenommen wurden [Cera02, S. 158]. Im Rahmen dieses Beitrags wird die Eignung des UDDI-Standards zur Spezifikation von Web-Services im Detail untersucht. Dabei wird eine Reihe von Schwächen identifiziert, zu deren Beseitigung ein zu UDDI kompatibler Spezifikationsrahmen mit dem Namen *WS-Specification* entwickelt wird.

Der entwickelte Spezifikationsrahmen basiert auf dem zentralen Konzept des *Software-Vertrags* (Design by Contract [Meye92]) und baut auf Vorarbeiten zur Standardisierung der Spezifikation von Komponenten [Acke+02; Over04] auf, die zum Teil in Zusammenarbeit mit dem Arbeitskreis WI-KobAS der Gesellschaft für Informatik durchgeführt wurden. Für die Untersuchung des UDDI-Standards werden in Kap. 2 zunächst grundlegende Anforderungen betrachtet, die an einen Spezifikationsrahmen zur Beschreibung von Web-Services zu stellen sind. Daran anschließend erfolgt in Kap. 3 eine detaillierte Analyse des im UDDI-Standard enthaltenen Spezifikationsrahmens, wodurch die Entwicklung von WS-Specification in Kap. 4 motiviert wird. Mit einer Abgrenzung verwandter Arbeiten aus dem wissenschaftlichen Umfeld sowie einem Ausblick auf die Einbindung des erarbeiteten Ansatzes in die bestehende Web-Service-Technologie wird der Beitrag schließlich abgeschlossen.

2 Grundlegende Anforderungen

Spezifikationen bilden die methodische Basis zur Unterstützung der Auswahl- und Kopplung von Web-Services [HoJu02, S. 34]. Obwohl der Spezifikation damit eine zentrale Bedeutung für die Entwicklung mit Web-Services (bzw. generell für die komponentenorientierte Entwicklung) zukommt [Spee+01, S. 689], gibt es bislang kaum wissenschaftliche Vorarbeiten, die als Maßstab zur Bewertung des im UDDI-Standard enthaltenen Spezifikationsrahmens herangezogen werden können [Appe+01, S. 768f.]. Dennoch lassen sich anhand der Aufgaben, die mit den bereitgestellten Informationen im Rahmen des Kompositionsprozesses zu unterstützen sind (vgl. Kap. 1), einige Anforderungen ableiten. So sollte ein Spezifikationsrahmen, der zur Beschreibung von Web-Services eingesetzt wird, zumindest folgende *grundlegende Eigenschaften* aufweisen:

- Der Spezifikationsrahmen muss die für die Auswahl und Komposition relevanten Eigenschaften von Web-Services *vollständig* beschreiben. Insbesondere muss es möglich sein, die bei der Auswahl und Kopplung zu bewältigenden Aufgaben nur mit den zur Verfügung gestellten Spezifikationen auszuführen.
- Die Vorgaben des Spezifikationsrahmens sind *normativ*, d.h. er schreibt sowohl den Inhalt (was zu spezifizieren ist) als auch die einzusetzenden Notationen (womit zu spezifizieren ist) verbindlich vor. Damit gewährleistet er die Einheitlichkeit und Vergleichbarkeit von Spezifikationen, die Voraussetzungen für deren Verwendbarkeit (etwa durch Entwicklungswerkzeuge) sind.
- Die gemäß den Vorgaben des Rahmens erstellten Spezifikationen sind sowohl *von Entwicklern als auch von Werkzeugen auswertbar*. Für die Werkzeugunterstützung sowie die Automatisierung der Auswahl und Kopplung von Web-Services ist die Bereitstellung formaler Beschreibungen unerlässlich. Dennoch

müssen diese für Entwickler verständlich bleiben, da eine Werkzeugunterstützung (derzeit) nicht für alle Aufgaben des Entwicklungsprozesses gegeben ist.

- Der Spezifikationsrahmen besitzt eine *modulare Struktur*, so dass neue Spezifikationen abwärtskompatibel hinzugefügt werden können. Dadurch bleibt gewährleistet, dass ältere Werkzeuge mit neueren Versionen des Spezifikationsrahmens umgehen und weiter verwendet werden können.
- Zur Spezifikation werden *etablierte* und *industriefähige Notationen* verwendet. Obwohl damit ggf. auf den Einsatz formaler Notationen verzichtet und gegenüber dem Stand der Wissenschaft ein suboptimales Ergebnis erzielt wird, sichert gerade dies die Akzeptanz in der Anwendungsentwicklung der Praxis.

3 Der UDDI-Standard

Ausgehend von den zuvor aufgestellten grundlegenden Anforderungen wird der Spezifikationsrahmen des UDDI-Standards im Folgenden näher untersucht. Hierzu wird vorab ein einführender Überblick gegeben und anschließend auf die Erfüllung der in Kap. 2 aufgestellten grundlegenden Anforderungen eingegangen. Die dabei aufgedeckten Schwächen des Spezifikationsrahmens werden bei der Entwicklung von WS-Specification in Kap. 4 erneut aufgegriffen.

Mit dem UDDI-Standard [UDDI02] werden zwei verschiedene, jedoch nicht präzise voneinander abgegrenzte Spezifikationsrahmen eingeführt, die bei der Analyse voneinander getrennt zu betrachten sind [Caul+01, S. 188, 190]: ein Spezifikationsrahmen für die Beschreibung von *Unternehmen*, der die Basis für die Entwicklung von Unternehmensverzeichnissen bildet, sowie ein Spezifikationsrahmen für die Beschreibung von *Web-Services*, der zum Aufbau von Web-Service-Katalogen verwendet wird. Web-Services werden gemäß dem UDDI-Standard üblicherweise von einem Unternehmen angeboten. Wegen der Möglichkeit, zusätzlich auch auf Web-Services, die von Fremdunternehmen angeboten werden, zu verweisen, stehen beide Spezifikationsrahmen jedoch in einem m:n Verhältnis zueinander. Somit kann ein Unternehmen mehrere Web-Services anbieten und von mehreren Unternehmen auf einen im Katalog publizierten Web-Service verwiesen werden.

Der Spezifikationsrahmen zur Beschreibung von Unternehmen findet sich im Datenmodell des UDDI-Standards, das im XML-Format vorliegt, unter dem konzeptionellen Bezeichner `<businessEntity>` [Caul+01, S. 188-190; Cera02, S. 162-165]. Mit ihm lassen sich Merkmale von Unternehmen beschreiben, wobei zwischen den Aspekten „Allgemeine Informationen“ und „Klassifikationen“ unterschieden wird (vgl. Abb. 1). Zu den allgemeinen Informationen, die im UDDI-Jargon als „White Pages“ bezeichnet werden, gehört der Name des Unternehmens, die Kontaktdaten (Ansprechpartner, Adresse, Kommunikationsdaten etc.), eine Beschreibung, eindeutige Kennzeichner wie etwa die Umsatzsteuernummer (die

sich unter dem Bezeichner <identifierBag> angeben lassen [Caul+01, S. 189]) sowie eine global eindeutige Kennzeichnung (Unique ID).

```
<businessEntity businessKey="32E8F1C4-3BC9-470C-A7C4-702D1BF6EB35">
  <name> Oversoft Software </name>
  <description>
    Oversoft Software is specialized in enabling
    XML Web services and component-based application development.
  </description>
  <businessServices>
    <businessService serviceKey="74cebe59-4adb-4919-9f52-8cebf6ca4c28">
      <name> Oversoft GeneralLedger </name>
      <description>
        Manage your general ledger over the Internet.
      </description>
      <bindingTemplates>
        <bindingTemplate bindingKey="f5296cc1-8498-4b09-8e84-7ce9a73d112b">
          <description> GeneralLedger WebService Binding </description>
          <accessPoint URLType="http">
            http://www.generalledger.oversoft.biz/generalledger.asmx
          </accessPoint>
          <tModelInstanceDetails>
            <tModelInstanceInfo
              tModelKey="uuid:4e45aa2c-1876-4d03-8264-12983789935d"/>
          </tModelInstanceDetails>
        </bindingTemplate>
      </bindingTemplates>
      <categoryBag>
        <keyedReference
          tModelKey="70a80f61-77bc-4821-a5e2-2a406acc35dd"
          keyName="Internet Business services, n.e.c." keyValue="7399"/>
      </categoryBag>
    </businessService>
      ...
  </businessServices>
  <categoryBag>
    <keyedReference
      tModelKey="70a80f61-77bc-4821-a5e2-2a406acc35dd"
      keyName="Internet Business services, n.e.c." keyValue="7399"/>
  </categoryBag>
</businessEntity>

<tModel tModelKey="uuid:4e45aa2c-1876-4d03-8264-12983789935d">
  <name> General Ledger Interface </name>
  <overviewDoc>
    <overviewURL>
      http://www.generalledger.oversoft.biz/generalledger.asmx?WSDL
    </overviewURL>
  </overviewDoc>
</tModel>
```

Abbildung 1: Beispielspezifikation mit Unternehmen, Web-Service und WSDL-Verweis

Demgegenüber beinhalten die sog. „Yellow Pages“ eine Menge von Klassifikationen, mit denen sich Unternehmen näher einordnen (kategorisieren) lassen. Hierzu werden verschiedene standardisierte Taxonomien zur Verfügung gestellt, mit denen der Geschäftsbereich (UN/SPSC bzw. NAICS) oder der Sitz bzw. Handelsraum (ISO 3166) eines Unternehmens spezifiziert werden kann [Cera02, S. 164]. Die einzelnen Klassifikationen finden sich im Datenmodell des UDDI-Standards unter dem konzeptionellen Bezeichner <categoryBag> [Caul+01, S. 189]. Schließlich erlaubt der zur Beschreibung von Unternehmen vorgesehene Spezifi-

kationsrahmen auch die Angabe eines bzw. mehrerer angebotener Web-Services, deren technische Merkmale in den sog. „Green Pages“ zu spezifizieren sind. Diese Spezifikationen sind im UDDI-Datenmodell unter dem konzeptionellen Bezeichner `<businessServices>` (vgl. Abb. 1) zusammengefasst [Caul+01, S. 190f.; Cera02, S. 165f.]. Sie sind gemäß einem Spezifikationsrahmen zur Beschreibung von Web-Services anzufertigen, der mit dem Ziel entwickelt wurde, die automatisierte Auswahl und Kopplung von Web-Services durch die Bereitstellung adäquater Informationen zu unterstützen [Caul+01, S. 181f.].

Wie der Spezifikationsrahmen zur Beschreibung von Unternehmen unterscheidet auch dieser zwischen verschiedenen Aspekten. So sind zunächst einige allgemeine Informationen über den jeweils angebotenen Web-Service bereitzustellen, die korrekterweise als „White Pages“ des Web-Service zu bezeichnen wären. Zu diesen gehört seine Bezeichnung, eine informelle Beschreibung sowie eine global eindeutige Kennzeichnung (Unique ID) [Caul+01, S. 190f.]. Ferner unterstützt der Spezifikationsrahmen die Klassifikation des Anwendungsbereichs, zu dem der jeweils angebotene Web-Service gehört. Hierzu können unter dem konzeptionellen Bezeichner `<categoryBag>` entsprechende Eintragungen vorgenommen werden, die de facto die „Yellow Pages“ eines Web-Service darstellen [Caul+01, S. 190f.].

Neben den beiden vorgenannten Aspekten wird vor allem die Angabe technischer Informationen unterstützt, die für Entwickler bzw. deren Werkzeuge von Interesse sind und die Kopplung von Web-Services unterstützen sollen. Diese Spezifikationen werden in Übereinstimmung mit dem UDDI-Standard als „Green Pages“ bezeichnet und sind im UDDI-Datenmodell unter dem konzeptionellen Bezeichner `<bindingTemplates>` zu finden [Caul+01, S. 191f.; Cera02, S. 166]. Sie liefern Details über die Web-Service-Schnittstellen wie bspw. deren Ort und Syntax. Zur Spezifikation der Schnittstellensyntax wird dabei üblicherweise ein Verweis auf ein WSDL-Dokument angegeben, der vom UDDI-Standard jedoch *nicht* zwingend vorgeschrieben wird [Cera02, S. 167]. Für den Verweis auf ein WSDL-Dokument ist vielmehr ein spezieller Zusatz, ein sog. `<tModel>`, zu spezifizieren (vgl. Abb. 1) [Cera02, S. 167]. Damit ergibt sich für den Spezifikationsrahmen zur Beschreibung von Web-Services die in Abb. 2 dargestellte Struktur, die sich aus dem UDDI-Standard mit einiger Interpretation ableiten lässt.

Analysiert man den Spezifikationsrahmen zur Beschreibung von Web-Services im Hinblick auf die in Kap. 2 aufgestellten grundlegenden Anforderungen, so offenbart sich eine Reihe von Schwächen. Zwar erfüllt er zwei der Anforderungen, da er einerseits die Erstellung von Spezifikationen begünstigt, die sowohl *vom Entwickler als auch von Werkzeugen auswertbar* sind, und andererseits zur Spezifikation von Web-Services solche Notationen einsetzt, die in der industriellen Anwendungsentwicklung breite Unterstützung erfahren. Dies gilt insbesondere für die Spezifikation der Web-Service-Schnittstellen, die unter Verwendung des WSDL Standards erfolgen kann. Jedoch ist der Web-Service-Spezifikationsrahmen des UDDI-Standards nicht als *normativ* einzustufen, da er zur Spezifikation der Schnittstellen von Web-Services weder die Verwendung von WSDL noch irgend-

einer anderen Methode vorschreibt [Cera02, S. 167]. Somit bleibt dem Anbieter eines Web-Service die Entscheidung überlassen, welche Notation er zur Beschreibung der Schnittstellen einsetzt (falls er sie überhaupt beschreibt). Um die Homogenität der Spezifikationen in einem Web-Service-Katalog zu garantieren, sollte daher zumindest die Verwendung von WSDL verbindlich vorgeschrieben werden.

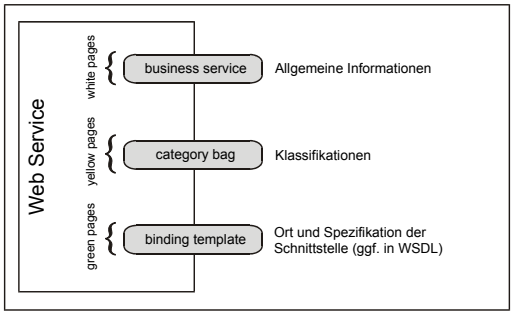

Abbildung 2: Struktur des UDDI-Spezifikationsrahmens für Web-Services

Auch im Hinblick auf die geforderte *Vollständigkeit* weist der Spezifikationsrahmen Mängel auf, so dass es fraglich scheint, ob sich die Auswahl und Kopplung von Web-Services durch die Bereitstellung von Informationen gemäß dem UDDI-Standard hinreichend unterstützen lässt. Dies wird durch die Erarbeitung zusätzlicher Standards im Rahmen der Weiterentwicklung der Web-Service-Technologie bestätigt, mit denen z.B. Informationen zur Sicherheit (WS-Security [Atki+02]) sowie über Transaktionen (WS-Transaction [Cabr+02]) spezifiziert werden können. Dies lässt vermuten, dass die durch den UDDI-Standard vorgesehene Beschreibung der Schnittstellensyntax alleine *nicht* ausreicht. Zudem fehlen dem Spezifikationsrahmen Vorgaben zur Beschreibung der Nutzungsbedingungen, der Funktionalität sowie der Qualität der angebotenen Web-Services, die für deren Auswahl und Kopplung wesentliche Informationen darstellen [Vith+03, S. 652f.].

Schließlich wird auch das Kriterium der *Modularität* durch den UDDI-Standard nicht optimal erfüllt. Hierzu trägt zum einen die Vermischung der beiden Spezifikationsrahmen zur Beschreibung von Unternehmen und Web-Services bei. So beziehen sich die Informationen, die im Standard als „White Pages“ und „Yellow Pages“ bezeichnet werden, auf die Unternehmen, während sich lediglich die Angaben, die unter der Bezeichnung „Green Pages“ zusammengefasst werden, auf den Web-Service beziehen [Caul+01, S. 187; KoLe04, S. 122]. Völlig unberücksichtigt bleibt jedoch, dass auch zur Beschreibung von Web-Services Informationen zu spezifizieren sind, die sich unter den Begriffen „White Pages“ und „Yellow Pages“ subsumieren lassen [Cera02, S. 18]. Zum anderen ist die getrennte Betrachtung der verschiedenen Aspekte nicht bis in das für die Spezifikation ausschlaggebende Datenmodell durchgehalten worden (vgl. Abb. 1). Dort werden die einzelnen Aspekte nicht als eigenständige Informationseinheiten sichtbar und bleiben zudem meist hinter kryptischen Bezeichnern verborgen.

4 WS-Specification

Aufgrund der identifizierten Schwächen ist der UDDI-Standard in seiner aktuellen Version nicht geeignet, die Auswahl und Kopplung von Web-Services in der beabsichtigten Weise zu unterstützen. Im Folgenden wird daher ein Spezifikationsrahmen für die Beschreibung von Web-Services entwickelt, der den eingangs aufgestellten grundlegenden Anforderungen genügt und die benötigte Unterstützung gewährleisten soll. Er trägt den Namen *WS-Specification* (Web Services Specification) und basiert auf dem zuvor analysierten Spezifikationsrahmen des UDDI-Standards, zu dem er (abwärts-) kompatibel bleibt. Hierdurch wird es möglich, die in einem auf dem UDDI-Standard basierenden Verzeichnis unter dem Bezeichner `<businessService>` abgelegten Spezifikationen transparent durch solche zu ersetzen, die gemäß den Vorgaben von WS-Specification erstellt wurden.

Der in WS-Specification enthaltene Spezifikationsrahmen basiert auf den Konzepten des Dienst- und Kompositionsvertrags, die aus dem etablierten Prinzip des *Software-Vertrags* (Design by Contract) [Meye92] abgeleitet wurden. Dieses Prinzip findet derzeit vor allem in der objektorientierten Programmierung Anwendung, wo es zur Spezifikation sog. *Dienstverträge* verwendet wird. Ein Dienstvertrag beschreibt die Vorbedingungen, die von einem Dienstnehmer zur *Laufzeit* erfüllt werden müssen, damit dieser einen Software-Dienst in Anspruch nehmen kann., sowie die Nachbedingungen, die vom Dienstgeber unter der Voraussetzung garantiert werden, dass der Dienst mit erfüllten Vorbedingungen in Anspruch genommen wurde [Meye92, S. 42]. Das Prinzip des Software-Vertrags findet auch in der komponentenorientierten Anwendungsentwicklung Verwendung, wo es zur Spezifikation sog *Kompositionsverträge* genutzt wird. Mit diesen lassen sich die Dienste, die eine Komponente an ihren Schnittstellen für einen Aufruf anbietet, sowie die zur *Entwicklungszeit* (also bei der Kopplung) zu berücksichtigenden Bedingungen spezifizieren [ChDa01, S. 17f.; Over04, S. 7]. Ein solcher Kompositionsvertrag lässt sich auf verschiedenen Vertragsebenen spezifizieren, wobei die Schnittstellten (bzw. die dort angebotenen Dienste) aus jeweils unterschiedlichen Perspektiven beschrieben werden [Beug+99, S. 39; Acke+02, S. 4; Over04, S. 9].

Der in WS-Specification enthaltene Spezifikationsrahmen greift das Konzept des Kompositionsvertrags auf und verwendet es für die Beschreibung von Web-Services. Gleichzeitig wird versucht, eine möglichst *vollständige* Beschreibung zu erreichen, indem verschiedene Vertragsebenen berücksichtigt werden, die teilweise von einem Standardisierungsprojekt des GI-Arbeitskreises WI-KobAS identifiziert wurden [Acke+02, S. 4]. Diese Vertragsebenen lassen sich aus einem *Klassifikationsschema* ableiten (vgl. Abb. 3), das zur vollständigen Beschreibung eines Web-Service in der Horizontalen die während des Entwicklungsprozesses auftretenden verschiedenen Abstraktionsebenen und in der Vertikalen die zur vollständigen Beschreibung in zahlreichen Methodiken [Olle+91, S. 12f.; CoDa94, S. 21; Schi97, S. 51f.; Sche98, S. 37; DSWi99, S. 43] verwendeten Modellierungs- bzw. Spezifikationsperspektiven unterscheidet. Dabei ist zu beachten, dass sich einem

Kompositionsvertrag gemäß dem Klassifikationsschema auch Dienstverträge als spezielle Ebene zuordnen lassen.

Jede der unterschiedenen Abstraktionsebenen fügt einem Kompositionsvertrag Spezifikationen mit unterschiedlichem Inhalt hinzu: durch die *fachliche* Abstraktionsebene wird die *Funktionalität* eines Web-Service beschrieben. Diese lässt sich mit domänenspezifischen (Fach-) Begriffen spezifizieren, die bspw. als Lexikon [DSWi99, S. 571] oder konzeptuelles Modell dargestellt werden können [DSWi99, S. 558]. Auf der *logischen* Abstraktionsebene wird die *systemtechnische Architektur* der Schnittstellen des Web-Service beschrieben. Dies geschieht durch die Angabe von Signaturlisten, Dienstverträgen und Interaktionsprotokollen. Durch die *physische* Abstraktionsebene wird schließlich die *Qualität* beschrieben, mit der die angebotenen Dienste durch einen Web-Service realisiert werden. Diese lässt sich mit Qualitätsattributen spezifizieren, die aus dem Qualitätsmodell des ISO 9126 Standards [ISO01; ISO03] abgeleitet werden können.

	Funktionalität / Terminologie (fachlich)	**Architektur / Schnittstellenform (logisch)**	**Implementierung / Qualität (physisch)**
Statische Sicht (Struktur)	Informationsobjekte (Dinge)	Typdeklarationen, Eigenschaften (Attribute)	Verwendbarkeit, Wartbarkeit, Portabilität
Operationale Sicht (Wirkungen)	Funktionen (Geschehnisse)	Methoden, Ausnahmen, Ereignisse, Zusicherungen	Funktionalität
Dynamische Sicht (Interaktionen)	Prozesse (Abläufe)	Interaktionsprotokolle	Zuverlässigkeit, Effizienz

Abbildung 3: Klassifikationsschema zur Ableitung von Vertragsebenen

Durch die orthogonal dazu unterschiedenen Spezifikationsperspektiven wird jede Abstraktionsebene weiter differenziert. Dabei fokussiert die *statische Perspektive* (Datensicht) auf die *Struktur* eines Softwareartefakts. Die *operationale Perspektive* (Funktionssicht) beschreibt die *Wirkungen* der Operationen, die auf einem Softwareartefakt ausgeführt werden können. Die *dynamische Perspektive* (Prozesssicht) spezifiziert schließlich den *Ablauf*, also die Interaktionen, an denen ein Softwareartefakt mitwirken kann.

Die mithilfe dieser Einteilung ermittelten neun Vertragsebenen bilden den Kern des Spezifikationsrahmens von WS-Specification. Er wird durch zwei weitere Vertragsebenen vervollständigt, die die Angabe allgemeiner und kommerzieller Informationen sowie die Klassifikation von Web-Services ermöglichen (vgl. Abb. 4). Dabei wird die Struktur des im UDDI-Standard enthaltenen Spezifikationsrahmens um „Blue Pages“ zur Spezifikation der Funktionalität sowie um „Grey Pages“ zur Beschreibung der Qualität eines Web-Service erweitert. Auf den elf Ebenen werden durch WS-Specification darüber hinaus verschiedene, jeweils auf die Web-Service-Technologie angepasste Notationen zur Spezifikation vorgeschrieben. Diese werden im Folgenden anhand eines Spezifikationsbeispiels näher vorgestellt.

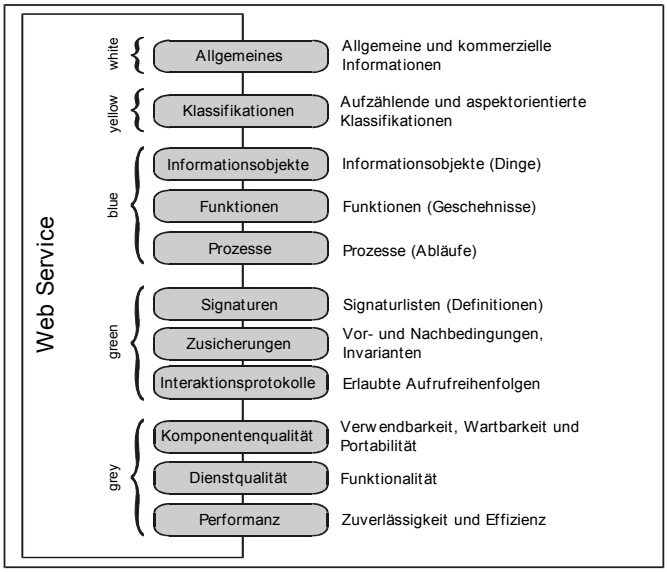

Abbildung 4: Struktur des mit WS-Specification eingeführten Spezifikationsrahmens

4.1 White Pages: Allgemeine und kommerzielle Informationen

Wie bereits erwähnt wurde, unterstützt der UDDI-Standard die Angabe *allgemeiner Informationen* über Web-Services. WS-Specification gestattet darüber hinaus auch die Spezifikation der *Nutzungsbedingungen*, die für die Auswahl und Kopplung eine wichtige Rolle spielen. So wird einerseits die Angabe verschiedener Preismodelle ermöglicht, wobei jedes Preismodell aus einem Preis, den akzeptierten Bezahlverfahren und dem dafür geleisteten Lieferumfang (d.h. einer Liste der Artefakte, auf die zugegriffen werden kann) besteht. Andererseits wird die Angabe der bei der Nutzung geltenden rechtlichen Vereinbarungen (Lizenzvereinbarungen, Garantiebestimmungen etc.) unterstützt. Die dabei zu machenden Angaben sind vorwiegend beschreibender Natur und werden somit größtenteils umgangssprachlich erfasst. Um die Auswertbarkeit dieser Informationen durch Werkzeuge zu gewährleisten, gibt WS-Specification jedoch eine Reihe von Taxonomien mit vorgefertigten Alternativen vor, mit denen entweder direkt zu spezifizieren ist bzw. Informationen zu klassifizieren sind (vgl. Abb. 5).

4.2 Yellow Pages: Klassifikationen

Wie der Spezifikationsrahmen des UDDI-Standards unterstützt auch WS-Specification die Angabe von *Klassifikationen*, mit denen der *Anwendungsbereich* eines Web-Service beschrieben werden kann. Jedoch stellt WS-Specification hier-

für eine spezialisierte Taxonomie (das sog. Standard Application Domain Classification System) zur Verfügung, die eine Systematik anwendungsspezifischer (vertikaler) und generischer (horizontaler) Domänen enthält. Somit muss zur Spezifikation des Anwendungsbereichs nicht auf die Taxonomien zurückgegriffen werden, die zur Klassifikation von Geschäftsfeldern bestimmt sind (wie dies beim UDDI-Standard der Fall ist). Darüber hinaus unterstützt WS-Specification die Klassifikation der Web-Service-Technologie, mit der der zu spezifizierende Web-Service realisiert wurde. Dies ist grundsätzlich notwendig, da die der Web-Service-Technologie zugrunde liegenden Standards ständig weiter entwickelt werden und deren Kompatibilität dabei nicht zwangsläufig sichergestellt ist.

```
<businessService serviceKey="74cebe59-4adb-4919-9f52-8cbbf6ca4c28">
    ...
  <distribution>
    <distributionForm>
      <name> unlimited use (flat rate) monthly payment </name>
      <price currencyKey="811464A4-823F-4a87-9F85-5B69443705B1" name="usd">
        9.90
      </price>
      <acceptedPayments>
        <payment key="3c27116-5493-4931-9411-dd2218e84e11" name="debit advice"/>
        ...
      </acceptedPayments>
      <scopeOfSupply>
        Oversoft GL .NET Client, Oversoft GL SDK ...
      </scopeOfSupply>
    </distributionForm>
    ...
  </distribution>
  <termsOfUse>
    <agreement key="35f7f0e5-1e0c-4535-8a6a-3c3546f65340" name="license agreement">
      Oversoft End User License Agreement ...
    </agreement>
    ...
  </termsOfUse>
    ...
</businessService>
```

Abbildung 5: Beispielspezifikation gemäß der White Pages von WS-Specification

4.3 Blue Pages: Spezifikation der fachlichen Funktionalität

Die Möglichkeit, die *fachliche Funktionalität* von Web-Services mittels eines Begriffssystems zu beschreiben, das in Form einer *Ontologie* [Hess02, S. 477] spezifiziert werden kann, stellt eine Erweiterung des UDDI-Standards dar. Hierfür bietet WS-Specification im Rahmen der „Blue Pages“ die Möglichkeit, die fachliche Semantik, die der Implementierung des Web-Service zugrunde liegt, in Form von Fachbegriffen zunächst zu definieren und diese anschließend miteinander in Beziehung zu setzen. Die so erreichte Spezifikation der Funktionalität eines Web-Service ist insbesondere zur Unterstützung der Auswahl von Web-Services von Bedeutung [Vith+03, S. 652f.]. Dabei unterscheidet WS-Specification grundsätzlich zwischen drei verschiedenen Begriffsklassen, die in Anlehnung an die Refe-

renzmodellierung als *Informationsobjekte*, *Funktionen* und *Prozesse* bezeichnet werden. Informationsobjekte beschreiben fachliche Informationen, die mit einem Web-Service ausgetauscht werden. Funktionen beschreiben die Funktionalität der bereitgestellten Dienste. Prozesse beschreiben das fachliche Protokoll, gemäß dem die Dienste eines Web-Service zu einem komplexen Ablauf verbunden werden können. Daneben stellt WS-Specification vordefinierte Beziehungsarten zur Verfügung, mit denen die Begriffe einer Klasse jeweils zueinander in Beziehung gesetzt werden können: *Abstraktionen* (Identität, Spezialisierung etc.) drücken dabei ein gewisses Maß an Gleichheit zwischen verschiedenen Begriffen aus, während *Kompositionen* (Teil-Ganze-Beziehungen, Reihenfolgen etc.) verwendet werden, um Begriffe zu zusammengesetzten Strukturen zu verbinden.

```
<businessService serviceKey="74cebe59-4adb-4919-9f52-8cbbf6ca4c28">
    ...
  <functionality>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description rdf:about="IGeneralLedger::Balance">
        <Type> InformationObject </Type>
        <Term> Balance </Term>
        <Definition>
          A BALANCE yields a comparison of ASSETS and LIABILITIES of a company
          at a special date (CUTOFF DATE) based on a LEGAL REGULATION.
        </Definition>
      </rdf:Description>
      <rdf:Description rdf:about="IGeneralLedger::CreateBalance()">
        <Type> Function </Type>
        <Term> Create Balance </Term>
        <Definition> ... </Definition>
      </rdf:Description>
      ...
      <rdf:Description rdf:about="IGeneralLedger">
        <Type> Function </Type>
        <Term> Create Annual Account </Term>
        <Definition> ... </Definition>
        <Composition>
          Create Balance, Create Profit and Loss Account, Close Accounts
        </Composition>
      </rdf:Description>
      <rdf:Description rdf:about="IGeneralLedger">
        <Type> Process </Type>
        <Term> Annual Accounting </Term>
        <Definition> ... </Definition>
        <Sequence>
          Close Accounts, Create Profit and Loss Account, Create Balance
        </Sequence>
      </rdf:Description>
    </rdf:RDF>
    ...
  </functionality>
    ...
</businessService>
```

Abbildung 6: Beispielspezifikation gemäß der Blue Pages von WS-Specification

Durch die Vorgabe der Begriffsklassen sowie der möglichen Beziehungsarten zwischen diesen wird sichergestellt, dass bei der Spezifikation der Funktionalität eines Web-Services jeweils eine einfache Ontologie mit *gleichartigen Bestandteilen* entsteht, die mit den Methoden des *Semantic Web* (insbesondere dem Resource

Description Framework [LaSw99; BrGu03]) gespeichert und ausgewertet werden kann. Dadurch wird es möglich, die spezifizierte Funktionalität eines Web-Service auch einer Auswertung in semantischen Suchverfahren und automatisierten Kompatibilitätstests zugänglich zu machen. Bei der Entwicklung von WS-Specification wurden hierzu die oben genannten Begriffsklassen und Beziehungsarten als *RDF Schema* [BrGu03] modelliert, so dass sie zur Spezifikation der Begriffsysteme mit RDF verwendet werden können. Die Spezifikation der Funktionalität besteht somit aus zusammengehörigen *RDF-Ausdrücken*, die direkt in das Datenmodell übernommen und in entsprechenden, auf dem Spezifikationsrahmen von WS-Specification basierenden Verzeichnissen abgelegt werden (vgl. Abb. 6).

Die Spezifikation der fachlichen Funktionalität dient jedoch nicht nur den bereits genannten Zwecken. Sie kann auch zur näheren Beschreibung der systemtechnischen Architektur der Schnittstellen eines Web-Service verwendet werden. Unter Verwendung der spezifizierten Fachbegriffe lässt sich festhalten, welches Element der Web-Service-Schnittstelle eine bestimmte Funktionalität implementiert: Informationsobjekte können insbesondere den Typdeklarationen, Attributen sowie Parametern und Rückgabewerten zugeordnet werden. Funktionen lassen sich den Diensten bzw. Ereignissen zuordnen, und Prozesse korrespondieren mit systemtechnischen Aufrufreihenfolgen. Damit kann die Spezifikation der Funktionalität auch zur Erkennung syntaktischer Heterogenitäten beitragen, bspw. wenn Schnittstellenelemente zu finden sind, die die gleiche Funktionalität aufweisen aber unterschiedliche Signaturen (z.B. Namen, Parameterreihenfolgen etc.) besitzen [ZaWi95]. Deshalb wird die Angabe von (Implementierungs-) Beziehungen zwischen Fachbegriffen und Schnittstellenelementen explizit unterstützt.

4.4 Green Pages: Spezifikation der logischen Architektur

Durch die „Green Pages“ wird die *logische Architektur* der Web-Service-Schnittstellen beschrieben, wobei wiederum zwischen statischen, operationalen und dynamischen Bestandteilen unterschieden wird. Die hier abgelegten Informationen sind für den korrekten Aufruf eines Web-Service von Bedeutung und unterstützen somit vor allem den Kopplungsprozess. Zur Spezifikation des statischen sowie eines Teils der operationalen Architektur, der Methoden und Ereignisse, wird auf die Web Service Description Language (WSDL [Chin+04]) zurückgegriffen, mit der sich die Schnittstelle eines Web-Service in Form von Signaturlisten spezifizieren lässt (vgl. Abb. 7). Hierdurch lassen sich die von einem Web-Service vorgenommenen *Typ- bzw. Konstantendeklarationen*, *Attribute*, *Dienste* (Methoden), *Ausnahmen* und *Ereignisse* spezifizieren. Im Gegensatz zum UDDI-Standard schreibt WS-Specification jedoch die Verwendung von WSDL *verbindlich* vor und erfasst die entsprechenden Spezifikationen *direkt* im Datenmodell.

Da die Spezifikation von Signaturlisten für den korrekten Aufruf eines Web-Service zwar notwendig, jedoch im Allgemeinen nicht hinreichend ist, unterstützt

WS-Specification die Angabe zusätzlicher Informationen über die Architektur der Schnittstellen. Hierzu gehören die beim Aufruf einer Methode jeweils zu erfüllenden *Vor-* sowie die nach dem Aufruf zugesicherten *Nachbedingungen*, d.h. die *Dienstverträge* der an der Schnittstelle angebotenen Methoden [Meye92]. Diese werden gegenwärtig unter Verwendung der OCL [OMG97] als formale Notation spezifiziert (vgl. Abb. 7), da spezielle, auf die Web-Service-Technologie zugeschnittene Notationen (z.B. WSMF [FeBu02]) noch in der Entwicklung sind.

```
<businessService serviceKey="74cebe59-4adb-4919-9f52-8cbbf6ca4c28">
    ...
  <assertions>
    <specification key="E9D7C918-C8BA-42f0-8559-B1C5B2DB18FE" name="ocl">
      <formalStatement>
        context IGeneralLedger::Login() pre: self.LoginCredentials.length > 0
      </formalStatement>
      <description>
        The login credentials must not be empty.
      </description>
    </specification>
    ...
  </assertions>
  <interfaceDefinitions>
    <specification key="CF53E356-FF7E-4538-AEEE-7B98867F1A9F" name="wsdl1.2">
      <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
        <types>
          ...
      </definitions>
    </specification>
    ...
  </interfaceDefinitions>
    ...
</businessService>
```

Abbildung 7: Beispielspezifikation gemäß der Green Pages von WS-Specification

Darüber hinaus unterstützt WS-Specification die Spezifikation der Interaktionsprotokolle, d.h. der erlaubten *Reihenfolgen,* gemäß denen die an den Schnittstellen angebotenen Dienste aufgerufen werden können. Dienste können im Allgemeinen nicht in beliebigen, sondern nur in wohlgeordneten Abfolgen aufgerufen werden, die auf der Basis spezifizierter Signaturlisten jedoch nur schwer zu bestimmen sind. Somit stellt die Spezifikation der erlaubten Aufrufreihenfolgen eine wesentliche Voraussetzung für den effizienten Einsatz von Kopplungsmethoden (wie BPEL [Andr+03] oder WSCL [Kava+04]) dar. Prinzipiell lassen sich Aufrufreihenfolgen durch die Angabe der zwischen den einzelnen Diensten existierenden *Abhängigkeiten* sowie durch eine Aufzählung sämtlicher erlaubter *Aufrufsequenzen* spezifizieren [Tolk03, S. 131]. Dabei zeichnet sich der erstgenannte Ansatz vor allem durch seine Effizienz aus, birgt jedoch gleichzeitig erhebliche Risiken, da durch nicht spezifizierte Abhängigkeiten unzulässige Aufrufreihenfolgen zugelassen werden. Hingegen führt das (versehentliche) Auslassen einer erlaubten Aufrufsequenz prinzipiell nicht zu entsprechenden Nachteilen, weswegen diese Methode im Rahmen von WS-Specification Anwendung findet.

Hierzu wird eine formale (XML) Notation verwendet, mit der sich „Regular Types“ spezifizieren lassen [Nier95, S. 108-110]. Diese basieren auf endlichen *regulären Automaten* und besitzen den Vorteil einer effizienten Auswertbarkeit (z.B. für Kompatibilitätstests). Sie weisen jedoch auch eine Reihe von Nachteilen auf, da sowohl ihre *Aussagekraft* als auch die mit ihnen erzielbare *Präzision* eingeschränkt ist: so beschreiben Regular Types prinzipiell nur eine Obermenge der erlaubten Aufrufsequenzen und enthalten dabei ggf. auch unerlaubte Aufrufreihenfolgen [Nier95, S. 109]. Daher wird z. Zt. an der Integration anderer Formalismen gearbeitet, die diese Mängel beheben (z.B. Petri-Netze oder temporale Logiken).

4.5 Grey Pages: Spezifikation der physischen Qualität

Neben den „Blue Pages“ zur Spezifikation der Funktionalität wird dem UDDI-Standard durch WS-Specification mit den „Grey Pages“ ein weiterer Aspekt hinzugefügt, der die Spezifikation der Qualität ermöglicht, mit der ein Web-Service seine Dienste erbringt. Diese Informationen sind sowohl für die Auswahl als auch die Kopplung von Web-Services von Bedeutung. Basierend auf dem standardisierten ISO 9126 Qualitätsmodell [ISO01] unterscheidet WS-Specification dabei zwischen statischen, operationalen und dynamischen Qualitätsmerkmalen: zu den statischen Qualitätsmerkmalen zählen die *Anwendbarkeit* und *Wartbarkeit*, zu den operationalen Qualitätsmerkmalen gehört die *Funktionalität* und zu den dynamischen Qualitätsmerkmalen zählen die *Zuverlässigkeit* sowie die *Effizienz*.

Für jedes der Qualitätsmerkmale wurden bei der Entwicklung von WS-Specification verschiedene messbare Kennzahlen, die aus der ISO 9126 Norm [ISO03] übernommen wurden, als charakteristische *Attribute* in Form einer Bibliothek vorgegeben. So lässt sich die Anwendbarkeit bspw. durch die mittlere *Einarbeitungszeit* (Time-to-Use) und die Wartbarkeit durch den mitgelieferten *Dokumentationsumfang* quantifizieren. Die Funktionalität lässt sich anhand verschiedener Merkmale beurteilen, wobei besonders die Sicherheit sowie die Transaktionalität im Mittelpunkt stehen, da diese durch Rückgriff auf entsprechende Standards der Web-Service-Technologie (WS-Security [Atki+02] und WS-Transaction [Cabr+02]) spezifiziert werden können. Weitere Aspekte (wie z.B. Persistenz) lassen sich der Forschung zur aspektorientierten Programmierung entnehmen.

Von herausragender Bedeutung ist jedoch die Spezifikation der dynamischen Qualität, die auch als Quality-of-Service bezeichnet wird und in Form sog. *Service-Level Agreements* (SLAs [KnEg03, S. 29f.]) angegeben wird. Durch Service-Level wird die Spezifikation verschiedener Qualitätsniveaus unterstützt, so dass diese bspw. als jeweils voneinander unabhängige Vertriebsformen (vgl. Kap. 4.1) zu unterschiedlichen Preisen angeboten werden können. Für deren Spezifikation stellt WS-Specification dabei wiederum verschiedene Qualitätsattribute (*Verfügbarkeit*, *Durchsatz*, *Antwortzeit* etc.) zur Verfügung, die aus der ISO 9126 Norm [ISO03] abgeleitet wurden (vgl. Abb. 8).

```
<businessService serviceKey="74cebe59-4adb-4919-9f52-8cbbf6ca4c28">
    ...
  <quality>
    <functionality>
      <security>
        <specification key="f7827dd6-4bdd-4c79-a2a4-259c35480b2e" name="ws-security1.0">
          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
            <SignedInfo>
              <SignatureMethod Algorithm=" http://www.w3.org/2000/09/xmldsig#sha1"/>
              <DigestValue>LyLsF0Pi4wPU...</DigestValue>
            </SignedInfo>
          </Signature>
        </specification>
      </security>
      ...
    </functionality>
    <reliability>
      <serviceLevel name="High Quality">
        <availability metric="days/year" measurement="greaterThan"> 364.9 </availability>
        ...
      </serviceLevel>
      <serviceLevel name="Medium Quality">
        <availability metric="days/year" measurement="greaterThan"> 360 </availability>
        ...
      </serviceLevel>
    </reliability>
    <efficiency>
      <serviceLevel name="High Quality">
        <responseTime metric="milliseconds" measurement="lowerThan"> 20 </responseTime>
        <throughput metric="megabytes/second" measurement="greaterThan"> 1.5 </throughput>
        ...
      </serviceLevel>
      <serviceLevel name="Medium Quality">
        <responseTime metric="milliseconds" measurement="lowerThan"> 50 </responseTime>
        <throughput metric="megabytes/second" measurement="greaterThan"> 0.5 </throughput>
        ...
      </serviceLevel>
    </efficiency>
  </quality>
    ...
</businessService>
```

Abbildung 8: Beispielspezifikation gemäß der Grey Pages von WS-Specification

Wie Fachbegriffe lassen sich auch Qualitätsattribute zur näheren Charakterisierung von Elementen der systemtechnischen Architektur (speziell der an den Schnittstellen angebotenen Dienste) verwenden. Mit ihnen lässt sich beschreiben, welches systemtechnische Element (welcher Dienst) gewisse Qualitätsmerkmale aufweist. Die Angabe solcher Beziehungen zwischen Qualitätsattributen und Architekturelementen wird von WS-Specification unterstützt, so dass diese ebenfalls den entsprechenden Schnittstellenelementen zugeordnet werden können.

5 Verwandte Arbeiten

Für die Entwicklung von WS-Specification wurden Vorarbeiten aus der Komponentenorientierung genutzt, wobei vor allem diejenigen zu Software-Verträgen und Vertragsebenen [Beug+99; ChDa01; Acke+02; Over04] zu nennen sind. Somit werden bewährte Erkenntnisse auf die Beschreibung von Web-Services übertra-

gen, wodurch eine Parallelentwicklung vermieden wird. Zwar lassen sich einige der hier vorgeschlagenen Spezifikationen prinzipiell auch mit *Policies* beschreiben, die die Elemente einer Web-Service-Schnittstelle mit Zusicherungen (sog. Assertions) annotieren [KoLe04, S. 122f.]. Jedoch sind Policies schon wegen ihrer einfachen Struktur nicht in der Lage, die vorgeschlagenen Spezifikationsmechanismen (Ontologien, SLAs) zu ersetzen. Gleichzeitig umfassen die in diesem Beitrag dargelegten Konzepte eine Reihe von Ansätzen, die sich der Lösung spezieller Fragestellungen widmen, etwa der Spezifikation von fachlicher Semantik [MoAb03], Dienstverträgen [FeBu02] oder der Abhängigkeiten zwischen Diensten [Tolk03]. Mit ihrem Anspruch, einen Ersatz für den UDDI-Standard zu entwickeln, steht die Arbeit dabei in Zusammenhang zu anderen Ansätzen wie DAML-S (DARPA Agent Markup Language for Web Services [Anko+01]). Im Gegensatz zu diesen Ansätzen wahrt WS-Specification jedoch die Kompatibilität zum UDDI-Standard und positioniert sich damit als Weiterentwicklung.

6 Schlussbetrachtung

Im Rahmen dieses Beitrags wurden UDDI-Erweiterungen vorgeschlagen und mit WS-Specification zu einem eigenständigen Ansatz zusammengefasst. Ein besonderes Augenmerk wurde dabei auf die Erfüllung der eingangs genannten grundlegenden Anforderungen gelegt, so dass eine möglichst gute Unterstützung für die Auswahl und Kopplung von Web-Services gegeben ist. Durch das aus der Komponentenorientierung übernommene Konzept der Dienst- und Kompositionsverträge sowie die systematische Herleitung der einzelnen Vertragsebenen wurde einerseits versucht, das Kriterium der *Vollständigkeit* zu erfüllen. Andererseits verfügt WS-Specification dank dieser Vorgehensweise auch über einen *modularen Aufbau*, so dass weitere Spezifikationen mit geringem Aufwand in den Spezifikationsrahmen eingefügt werden können. Zu erwähnen bleibt jedoch, dass der modulare Aufbau wegen der angestrebten Kompatibilität zu UDDI nicht bis in das Datenmodell durchgehalten werden konnte. Zudem sind die inhaltlichen und notationstechnischen Vorgaben von WS-Specification verbindlich, so dass der Spezifikationsrahmen als *normativ* zu bezeichnen ist. Durch die Möglichkeit der Kommentierung und den Einsatz von Notationen, die im Rahmen der Web-Service-Technologie Verwendung finden, erfüllt WS-Specification auch die Kriterien der *Industriefähigkeit* sowie der *Verständlichkeit* für Entwickler und Werkzeuge.

Derzeit wird an einer Implementierung gearbeitet, die die Umsetzbarkeit demonstrieren soll. Während sich die Anwendung einzelner Spezifikationsteile bei der Auswahl und Kopplung von Web-Services aus den zitierten Arbeiten der Komponentenorientierung ergibt, ist die Einbringung von WS-Specification in die Web-Service-Technologie noch ungeklärt. Es wird jedoch angestrebt, den entwickelten Spezifikationsrahmen dem zuständigen Standardisierungsgremium vorzuschlagen.

Literatur

[Acke+02] Ackermann, J.; Brinkop, F.; Conrad, S.; Fettke, P.; Frick, A.; Glistau, E.; Jaekel, H.; Kotlar, O.; Loos, P.; Mrech, H.; Ortner, E.; Overhage, S.; Raape, U.; Sahm, S.; Schmietendorf, A.; Teschke, T.; Turowski, K.: Standardized Specification of Business Components, German Society of Computer Science, 2002.

[Andr+03] Andrews, T.; Curbera, F.; Dholakia, H.; Goland, Y.; Klein, J.; Leymann, F.; Liu, K.; Roller, D.; Smith, D.; Thatte, S.; Trickovic, I.; Weerawarana, S.: Business Process Execution Language for Web Services Version 1.1. Specification, IBM Corporation, 2003.

[Anko+01] Ankolekar, A.; Burstein, M.; Hobbs, J.; Lassila, O.; Martin, D.; McIlraith, S.; Narayanan, S.; Paolucci, M.; Payne, T.; Sycara, K.; Zeng, H.: DAML-S: Semantic Markup for Web Services. International Semantic Web Working Symposium (SWWS), 2001.

[Appe+01] Apperly, H.; Booch, G.; Councill, W. T.; Griss, M.; Heineman, G. T.; Jacobson, I.; Latchem, S.; McGibbon, B.; Norris, D.; Poulin, J.: The Near-Term Future of Component-Based Software Engineering. In: Councill, W. T.; Heineman, G. T. (Hrsg.): Component-Based Software Engineering: Putting the Pieces Together. Addison-Wesley: Upper Saddle River, NJ, 2001, S. 753-774.

[Atki+02] Atkinson, B.; Della-Libera, G.; Hada, S.; Hondo, M.; Hallam-Baker, P.; Klein, J.; LaMacchia, B.; Leach, P.; Manferdelli, J.; Maruyama, H.; Nadalin, A.; Nagaratnam, N.; Prafullchandra, H.; Shewchuk, J.; Simon, D.: Web Services Security (WS-Security) Version 1.0. Specification, 2002.

[Aust+04] Austin, D.; Barbir, A.; Ferris, C.; Garg, S.: Web Service Architecture Requirements. W3C Working Draft, World Wide Web Consortium, 2004.

[Bett01] Bettag, U.: Web-Services. Informatik Spektrum 24 (5), 2001: S. 302-304.

[Beug+99] Beugnard, A.; Jézéquel, J.-M.; Plouzeau, N.; Watkins, D.: Making Components Contract Aware. IEEE Computer 32 (7), 1999: S. 38-45.

[Beute02] Beutenmüller, U.: Web Services: Top oder Flop? Information Management & Consulting 17 (3), 2002: S. 26-30.

[Boot+03] Booth, D.; Haas, H.; McCabe, F.; Newcomer, E.; Champion, M.; Ferris, C.; Orchard, D.: Web Services Architecture. W3C Working Draft, World Wide Web Consortium, 2003.

[BrGu03] Brickley, D.; Guha, R. V.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft, World Wide Web Consortium, 2003.

[Cabr+02] Cabrera, F.; Copeland, G.; Cox, B.; Freund, T.; Klein, J.; Storey, T.; Thatte, S.: Web Services Transaction (WS-Transaction) Version 1.0, Specification, 2002.

[Caul+01] Cauldwell, P.; Chawla, R.; Chopra, V.; Damschen, G.; Dix, C.; Hong, T.; Norton, F.; Ogbuji, U.; Olander, G.; Richman, M. A.; Saunders, K.; Zaev, Z.: Professional XML Web Services. Wrox Press: Birmingham, 2001.

[Cera02] Cerami, E.: Web Services Essentials. O'Reilly: Sebastopol, CA, 2002.

[ChDa01] Cheesman, J.; Daniels, J.: UML Components: A Simple Process for Specifying Component-Based Software. Addison-Wesley: Upper Saddle River, NJ, 2001.

[Chin+04] Chinnici, R.; Gudgin, M.; Moreau, J. J.; Weerawarana, S.: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft, World Wide Web Consortium, 2004.

[CoDa94] Cook, S.; Daniels, J.: Designing Object Systems: Object-Oriented Modelling with Syntropy. Prentice Hall: Englewood Cliffs, NJ, 1994.

[Crnk02] Crnkovic, I.: Component-Based Software Engineering - New Challenges in Software Development. Software Focus 2 (4), 2002: S. 127-133.

[DSWi99] D'Souza, D. F.; Wills, A. C.: Objects, Components, and Frameworks with UML: The Catalysis Approach. Addison-Wesley: Upper Saddle River, NJ, 1999.

[FeBu02] Fensel, D.; Bussler, C.: The Web Service Modeling Framework WSMF. Technical Report, Vrije Universiteit Amsterdam, 2002.

[Garl+95] Garlan, D.; Allen, R.; Ockerbloom, J.: Architectural Mismatch: Why Reuse is So Hard. IEEE Software 12 (6), 1995: S. 17-26.

[Hess02] Hesse, W.: Ontologie(n). Informatik Spektrum 26 (6), 2002: S. 477-480.

[HoJu02] Hoidn, H.-P.; Jungclaus, R.: Web Services aus Sicht der Unternehmens-Architektur. Information Management & Consulting 17 (3), 2002: S. 31-35.

[ISO01] ISO/IEC: Software Engineering - Product Quality - Quality Model. Nr. ISO/IEC Standard 9126-1, International Organization for Standardization, 2001.

[ISO03] ISO/IEC: Software Engineering - Product Quality - External Metrics. Nr. ISO/IEC Standard 9126-2, International Organization for Standardization, 2003.

[Kava+04] Kavantzas, N.; Burdett, D.; Ritzinger, G.: Web Services Choreography Description Language Version 1.0. W3C Working Draft, World Wide Web Consortium, 2004.

[KnEg03] Knoblauch, S.; Egardt, V.: Service Level Management als Grundlage erfolgreicher IT-Outsourcing-Kundenbeziehungen. Information Management & Consulting 18 (SH), 2003: S. 28-31.

[KoLe04] Kossmann, D.; Leymann, F.: Web Services. Informatik Spektrum 26 (2), 2004: S. 117-128.

[LaSw99] Lassila, O.; Swick, R. R.: Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, World Wide Web Consortium, 1999.

[McIl68] McIlroy, M. D.: Mass Produced Software Components. In: Naur, P.; Randell, B. (Hrsg.): Software Engineering: Report on a Conference by the NATO Science Committee, Brussels, 1968. S. 138-150.

[Meye92] Meyer, B.: Applying "Design by Contract". IEEE Computer 25 (10), 1992: S. 40-51.

[Minz+02] Minz, R.; Datel, A.; Wenzky, H.: Web Services - nur eine Schimäre? Information Management & Consulting 17 (3), 2002: S. 6-12.

[Mitr03] Mitra, N.: SOAP Version 1.2 Part 0: Primer. W3C Recommendation, World Wide Web Consortium, 2003.

[MoAb03] Montebello, M.; Abela, C.: DAML Enabled Web Services and Agents in the Semantic Web. In: Chaudri, A. B.; Jeckle, M.; Rahm, E.; Unland, R. (Hrsg.): Web, Web-Services, and Database Systems. Springer, Berlin, Heidelberg: Lecture Notes in Computer Science Vol. 2593, 2003, S. 46-58.

[Nier95] Nierstrasz, O.: Regular Types for Active Objects. In: Nierstrasz, O.; Tsichritzis, D. (Hrsg.): Object-Oriented Software Composition. Prentice Hall: Upper Saddle River, NJ, 1995, S. 99-121.

[Olle+91] Olle, T. W.; Hagelstein, J.; MacDonald, I. G.; Rolland, C.: Information Systems Methodologies: A Framework for Understanding. Addison-Wesley: Wokingham, 1991.

[OMG97] OMG: Object Constraint Language, Version 1.1. Specification, Object Management Group, 1997.

[Over04] Overhage, S.: Zur Spezifikation von Komponenten der Informationssystementwicklung mit Softwareverträgen. In: Rebstock, M. (Hrsg.): Modellierung betrieblicher Informationssyteme - MobIS 2004, Proceedings zur Tagung, 10. März 2004, Essen, Germany, 2004. Köllen: Lecture Notes in Informatics Vol. P-41, S. 3-20.

[Sche98] Scheer, A.-W.: ARIS: Vom Geschäftsprozess zum Anwendungssystem. 3. Ed., Springer: Berlin, Heidelberg, 1998.

[Schi97] Schienmann, B.: Objektorientierter Fachentwurf: Ein terminologiebasierter Ansatz für die Konstruktion von Anwendungssystemen. Teubner: Stuttgart, Leipzig, 1997.

[Spee+01] Speed, J.; Councill, W. T.; Heineman, G. T.: Component-Based Software Engineering as a Unique Engineering Discipline. In: Councill, W. T.; Heineman, G. T. (Hrsg.): Component-Based Software Engineering: Putting the Pieces Together. Addison-Wesley: Upper Saddle River, NJ, 2001, S. 673-691.

[Tolk03] Tolksdorf, R.: A Dependency Markup Language for Web Services. In: Chaudri, A. B.; Jeckle, M.; Rahm, E.; Unland, R. (Hrsg.): Web, Web-Services, and Database Systems. Springer, Berlin, Heidelberg: Lecture Notes in Computer Science Vol. 2593, 2003, S. 129-140.

[UDDI02] UDDI Organization: UDDI Open Draft Specification. Public Draft, Universal Description, Discovery, and Integration Standards Organization, 2002.

[Vith+03] Vitharana, P.; Zahedi, F.; Jain, H.: Knowledge-Based Repository Scheme for Storing and Retrieving Business Components: A Theoretical Design and an Empirical Analysis. IEEE Transactions on Software Engineering 29 (7), 2003: S. 649-664.

[Weyu01] Weyuker, E. J.: The Trouble with Testing Components. In: Councill, W. T.; Heineman, G. T. (Hrsg.): Component-Based Software Engineering: Putting the Pieces Together. Addison-Wesley: Upper Saddle River, NJ, 2001, S. 499-512.

[ZaWi95] Zaremski, A. M.; Wing, J. M.: Signature Matching: A Tool for Using Software Libraries. ACM Transactions on Software Engineering and Methodology 4 (2), 1995: S. 146-170.