Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Fakultät für **Informatik**

Institut für Telematik
Cooperation & Management
Prof. Dr. Sebastian Abeck

Ein Dienstverzeichnis mit Semantikunterstützung für die Aus- und Weiterbildung

Diplomarbeit
von

Tilmann Kuhn

Verantwortlicher Betreuer:	Prof. Dr. Sebastian Abeck
Betreuender Mitarbeiter:	Dipl.-Inform. Karsten Krutz

Bearbeitungszeit: 01. Mai 2006 - 31. Oktober 2006

Universität Karlsruhe (TH) · Institut für Telematik · Geb. 20.20 · Zirkel 2 · 76131 Karlsruhe

Ehrenwörtliche Erklärung

Ich erkläre hiermit, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Karlsruhe, den 31. Oktober 2006

Tilmann Kuhn

Dank

Diese Diplomarbeit wurde in der Forschungsgruppe Cooperation & Management (C&M) am Institut für Telematik der Fakultät für Informatik an der Universität Karlsruhe verfasst.

Für die Ermöglichung dieser Arbeit bedanke ich mich besonders bei Herrn Professor Dr. Sebastian Abeck, der als Verantwortlicher der C&M-Leitung immer großen Wert auf eine gute Betreuung der *SeniorStudents* legt.

Ebenso danke ich Karsten Krutz für die gute Betreuung, seine Unterstützung bei der Literaturrecherche, viele richtungsgebende Hinweise und seine Geduld in den langen und spannenden Diskussionen in unserem Team.

Dank gilt aber auch den weiteren Teammitgliedern Christian Maier und Christophe Gnad für die gute Zusammenarbeit in angenehmer Arbeitsatmosphäre.

Nicht unerwähnt bleiben dürfen…

- … die weiteren Mitarbeiter von C&M, die mich bei der Diplomarbeit unterstützten, insbesondere Dr. Christian Mayerl und Christof Momm mit Informationen zu den Projekten ALFI und KIM.
- … Diana Glasstetter und Martin Schuhmacher, die mir mit konstruktiver Kritik und durch Korrekturlesen bei der Qualitätssicherung dieser Arbeit geholfen haben.

Mein herzlicher Dank gilt im Besonderen meiner Freundin Claudia Durand für ihre Geduld, wenn ich öfters mal etwas weniger Zeit hatte und am Wochenende arbeiten musste, und für ihren unermüdlichen Einsatz beim Korrekturlesen.

Sie alle haben zum erfolgreichen Abschluss dieser Arbeit beigetragen.

Inhaltsverzeichnis

1 EINLEITUNG

1.1 Einführung in das Themengebiet

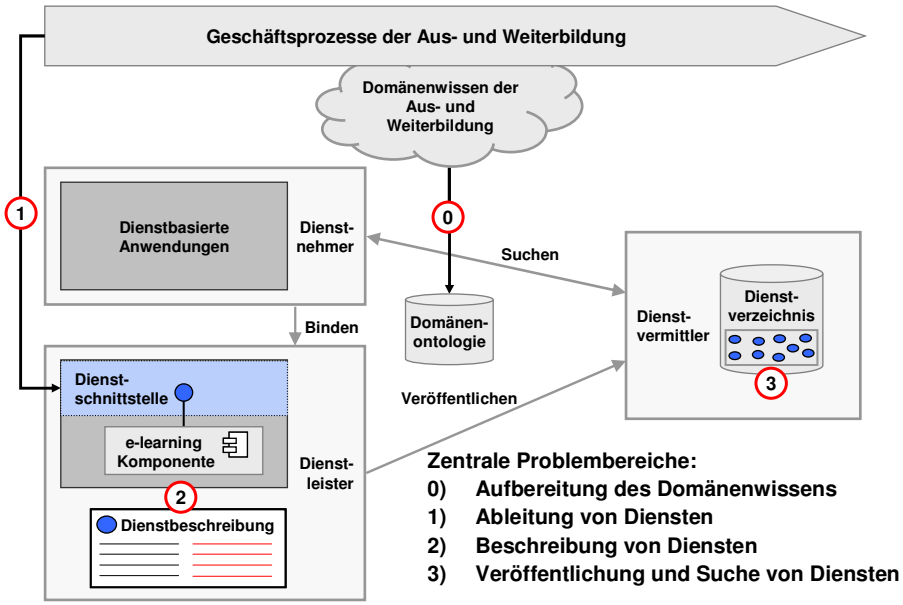
Ein Teilbereich innerhalb der Softwareentwicklung, die Erstellung verteilter Geschäftsanwendungen, befindet sich im Umbruch. Die De-jure- und De-facto-Standards der letzten Jahre rund um Webservices ermöglichen die zielgerichtete Verschaltung neuer und bereits vorhandener Softwarekomponenten zur Verwirklichung von rechnerunterstützten Geschäftsprozessen innerhalb von Unternehmen, aber auch über deren Grenzen hinweg [CT+04a]. Die Neuerungen sind jedoch nicht allein auf die technische Ebene beschränkt. Vielmehr sind sie eng verknüpft mit einem Umdenken bezüglich der einzusetzenden Entwicklungsprozesse und Architekturen der Systeme. Meistdiskutierter Vertreter der zuletzt genannten ist in diesem Zusammenhang die serviceorientierte Architektur (SOA) [AB+04].

Die Analyse und Entwicklung von SOA-Anwendungen und Lösungen für weitere unmittelbar mit ihnen verknüpfte Aspekte zählen zu den Kernkompetenzen der Forschungsgruppe Cooperation & Management (C&M) [C&M-P] am Institut für Telematik der Universität Karlsruhe (TH). Die Forschungsgruppe, die bereits ein eigenes SOA-Referenzmodell aufgestellt hat, beschäftigt sich darüber hinaus auch mit Fragestellungen, wie die Lehre durch IT-Dienste unterstützt werden kann. Dies untersucht sie bei ihrer Mitarbeit in der Arbeitsgruppe Lehrunterstützung der Fakultät für Informatik (ALFI) [URL-01], dem Projekt Karlsruher Integriertes Informationsmanagement (KIM) [URL-02] und in eigenem Interesse bei der Durchführung der Vorlesungen Informatik 1 und ISWA^{interactive}. Vor diesem Hintergrund entsteht die Dissertation mit dem Arbeitstitel “Dienstorientierte Rechnerunterstützung der Aus- und Weiterbildung” von Karsten Krutz in deren Kontext die vorliegende Diplomarbeit einzuordnen ist.

Im Rahmen der Dissertation wird thematisiert, mit welchen Methoden und Werkzeugen die Entwickler einer SOA-basierten Rechnerunterstützung der Aus- und Weiterbildung bei ihrer Tätigkeit unterstützt werden können. Serviceorientierte Anwendungen sind stark an den Prozessen und davon abgeleiteten Diensten des betrachteten Geschäftsbereichs ausgerichtet und benötigen zur Umsetzung eine geeignete Klassifizierung und Spezifikation dieser Dienste [AB+04]. Ein Kerngedanke dieser Anwendungen ist die lose Kopplung zwischen dem Dienstanbieter, der seine Dienste zur Nutzung anbietet, und den Dienstnutzern, die Dienste entsprechend ihren Interessen auswählen und in Anspruch nehmen können [AB+04]. Damit es überhaupt zu einem Dienstverhältnis kommt, muss der Dienstnehmer in die Lage versetzt werden für ihn geeignete Dienste aufzufinden. Zentraler Punkt einer serviceorientierten Anwendung zur Gewährleistung der Sichtbarkeit und Wiederverwendbarkeit der angebotenen Dienste ist ein Dienstverzeichnis [Cl06]. Ein weiterer Kerngedanke ist die Bereitstellung von Diensten mit hoher Geschäftsrelevanz durch die Komposition wieder verwendbarer Grunddienste entlang von Geschäftsprozessen [EW+06]. Auch bei einer solchen Verschaltung müssen die zur jeweiligen Situation passenden Grunddienste ausfindig gemacht werden. In beiden Fällen ist es im Interesse des Dienstanbieters, möglichst aussagekräftige Dienstsbeschreibungen zur Verfügung zu stellen. Besonders aussagekräftig sind dabei Dienstbeschreibungen, die auch die Semantik (Bedeutung) der jeweiligen Dienste mit einbeziehen, die sich aus dem Geschäftsbereich der Dienste, in diesem Fall dem Geschäftsbereich der Aus- und Weiterbildung, ableiten lässt [Bu04] [SV+03] [MP+04].Solche semantischen Dienstspezifikationen könnten in einem dafür geeigneten Dienstverzeichnis zur Suche und zum Abruf bereitgestellt werden. Durch die Berücksichtigung sowohl des Geschäftsbereichs als auch von Aspekten der serviceorientierten Anwendungsentwicklung bei der Erstellung eines solchen Dienstverzeichnisses ist es möglich, einen wichtigen Grundstein für eine zukünftige Rechnerunterstützung der Aus- und Weiterbildung zu legen.

1.2 Übergeordneter Rahmen

Mit einem Dienstverzeichnis allein (3 in Information 1) kann allerdings keine Rechnerunterstützung der Aus- und Weiterbildung erreicht werden. Zur SOA-basierten Unterstützung des Geschäftsbereichs sind einige weitere Schritte nötig. So gilt es unter anderem das Verzeichnis mit sinnvollen Diensten zu füllen. Aufgrund der Vielzahl der Fragestellungen in diesem Themengebiet werden parallel zwei weitere Diplomarbeiten durchgeführt, die weitere Aspekte der Rechnerunterstützung der Aus- und Weiterbildung betrachten, die teilweise eng mit dieser Arbeit und mit dem zu erstellenden Dienstverzeichnis verknüpft sind beziehungsweise aufeinander aufbauen.



Information 1: Dienstorientierte Rechnerunterstützung der Aus- und Weiterbildung

Um ein flexibles Zusammenspiel bereits vorhandener Geschäftsprozesse und IT-Systeme sowie der neuen Rechnerunterstützung zu erreichen, bietet sich so wohl die Analyse der Prozesse des Geschäftsbereichs insgesamt als auch eine darauf folgende Ableitung (1 in Information 1) und Spezifikation (2 in Information 1) der zu erbringenden Dienste an, so dass letztendlich eine Abbildung derselben auf alte und neue Systeme und eine anschließende Verschaltung zur Erfüllung der Geschäftsziele ermöglicht wird. Im Zuge dessen wurden bereits Arbeiten in den Bereichen Planung von Schulungen [Ba02], Lehrmaterialerstellung [Ru06], Suche und Verwaltung von Lehrmaterial [AG+04] [KE+03] [Ec02], Auslieferung von Lehrmaterial [AG+04], sowie Durchführung von Prüfungen durchgeführt, die einen Teil der Dienste der jeweiligen Bereiche identifizieren. Eine Ergänzung um fehlende Teile sowie eine übergreifende Anpassung der Beschreibung und Modellierung fand jedoch noch nicht statt, was einer Weiterverarbeitung im Rahmen einer Ingenieurstätigkeit entgegensteht. In der Diplomarbeit "Prozessorientierte Ableitung und semantische Beschreibung von Diensten der Aus- und Weiterbildung" von Christian Maier werden unter anderem die Probleme der Dienstableitung und anschließenden semantischen Dienstbeschreibung thematisiert. Da in unterschiedlichen Aus- und Weiterbildungsszenarien oftmals auf gleiche oder ähnliche Dienste und Abläufe zurückgegriffen wird, bietet es sich an, diese zu analysieren, sie zu beschreiben und die aufgefundenen Dienste letztendlich in einem Dienstverzeichnis (3 in Information 1) zum Abruf bereitzustellen. Ein solches Angebot an wieder verwendbaren Diensten würde die Erstellung und flexible Gestaltung einer Rechnerunterstützung fördern [Cl06].

Bei der Erstellung einheitlicher (Geschäftsbereichs-) Modelle spielt die Namensgebung eine wesentliche Rolle. Einerseits um über alle Modelle eine innere Konsistenz mit aussagekräftigen

Bezeichnungen zu erlangen, andererseits aber auch um Konsistenz mit externen Begriffswelten zu erzielen, zum Beispiel durch die Vermeidung von Homonymen. Es empfiehlt sich also zu prüfen, welche Benennungen es in externen Standards oder Projekten gibt, und ob diese für den eigenen Gebrauch möglicherweise unter Anpassungen wieder verwendet werden können. Im Zuge dessen soll auch eine Ausrichtung an den Vorgaben und Folgerungen aus der Erklärung von Bologna [EU99] zur Vereinheitlichung der Hochschulausbildung in der Europäischen Union erfolgen. Ein Ziel ist dabei die Entwicklung einer einheitlichen Terminologie für den betrachteten Fachbereich in Form einer Domänenontologie (0 in Information 1). Ein weiteres ist die Analyse der Geschäftsobjekte der Aus- und Weiterbildung und die Nutzung der daraus folgenden Erkenntnisse für die semantische Spezifikation der Dienste des Geschäftsbereichs. Diese Fragestellungen werden in der Diplomarbeit "Eine Ontologie zur Entwicklung von dienstorientierten Anwendungen in der Aus- und Weiterbildung" von Christophe Gnad aufgegriffen.

Die beiden Diplomarbeiten sind vom Themengebiet und der Zweckbindung stark mit der hier vorliegenden verknüpft und werden in enger Zusammenarbeit durchgeführt. Auf die genauen Aspekte der Zusammenarbeit, wird in den Kapiteln 1.5 (Beschreibung des Demonstrators) und 6 (Ontologiebasiertes dienstorientiertes Entwicklungsvorgehen) näher eingegangen.

1.3 In der Arbeit behandelte Fragestellungen

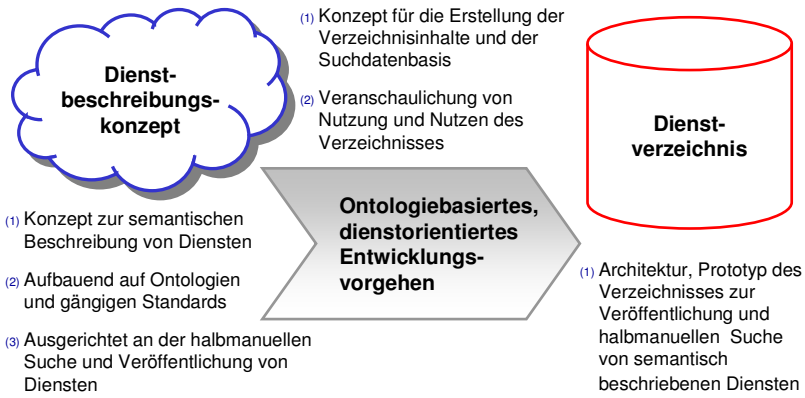
Um einen zur Wiederverwendung benötigten Dienst in einem Dienstverzeichnis zielgerichtet manuell oder halbmanuell suchen zu können, beispielsweise bei der Verschaltung von Grunddiensten zu erweiterten, den Geschäftsprozess unterstützenden Diensten, müssen die im Verzeichnis enthaltenen Dienste hinreichend genau spezifiziert sein. Der Dienstanbieter benötigt die Mittel, um eine für diesen Zweck angemessene und möglichst aussagekräftige formale Dienstbeschreibung zu liefern. Die etablierten Ansätze für die Dienstsuche sowie die Standards auf diesem Gebiet berücksichtigen diese Anforderung jedoch nur unzureichend [JM+03]. Die wichtigsten unter ihnen sind auf die syntaktische Ebene der Dienstbeschreibung beschränkt, was eine Einbeziehung des Wissens über den Geschäftsbereich und die Geschäftsobjekte, mit denen umgegangen wird, behindert. Einige neuere Entwicklungen auf diesem Gebiet ermöglichen die Erstellung von semantischen Dienstbeschreibungen. Diese sind jedoch nicht allgemein verbreitet und beschränken sich regelmäßig auf die reine Beschreibung der Dienste. Eine Unterstützung für diese Beschreibungen durch Dienstverzeichnisse und die Suche darin sowie die Kompatibilität mit gängigen Standards sind bislang schwierig [JM+03] [PK+02]. Bis auf wenige Beispiele zur Veranschaulichung des jeweiligen Ansatzes bieten die Entwickler der Neuerungen auch kaum Informationen und Richtlinien dafür, wie eine sinnvolle semantische Dienstbeschreibung aussieht oder gar wie sie im Rahmen einer Ingenieurstätigkeit (z.B. der Softwareentwicklung) erstellt werden kann.

Entsprechend der genannten Defizite werden in dieser Arbeit die Einbringung von Semantik in die Dienstspezifikation und die Nutzung derselben durch ein Dienstverzeichnis und bei der halbmanuellen Suche darin betrachtet. Dabei soll soweit wie möglich auf gängige Standards und bereits erarbeitete Methodiken für die semantische Dienstbeschreibung zurückgegriffen werden.

Es stellt sich zunächst die Frage, welche der bereits vorgeschlagenen Methodiken zur semantischen Dienstbeschreibung verwendet werden können und wie diese sinnvoll zu nutzen sind. Möglicherweise können Ansätze angepasst oder miteinander kombiniert werden. Es ist auch zu berücksichtigen, welche Anforderungen sich aus dem Ziel der Dienstbeschreibung und Dienstsuche an die semantische Spezifikation des Geschäftsbereichs, seiner Objekte und Dienste ergeben.

Darauf aufbauend ist klärungsbedürftig, wie die so gewonnene Dienstbeschreibung zur Veröffentlichung in ein Dienstverzeichnis eingebracht werden kann, so dass eine durch die semantischen Informationen verbesserte halbmanuelle Suche erzielt wird und ein Mehrwert gegenüber konventionellen Suchansätzen entsteht. Dieser soll als Beleg mittels einer prototypischen Umsetzung des zuvor entworfenen Dienstverzeichnisses aufgezeigt werden.

Schließlich kann die aus dem Dienstbeschreibungskonzept und der Verwirklichung des Dienstverzeichnisses gewonnene Erkenntnis über die Erstellung und Aufbereitung der Dienstbeschreibungen mit Fertigkeiten aus der Softwareentwicklung verknüpft werden. Das Interesse gilt hier einem möglichen Vorgehen bei der Entwicklung sowohl der Dienste, die im Verzeichnis veröffentlicht werden sollen, als auch der dazu nötigen Semantikbeschreibungen.



Information 2: Zielbestimmung

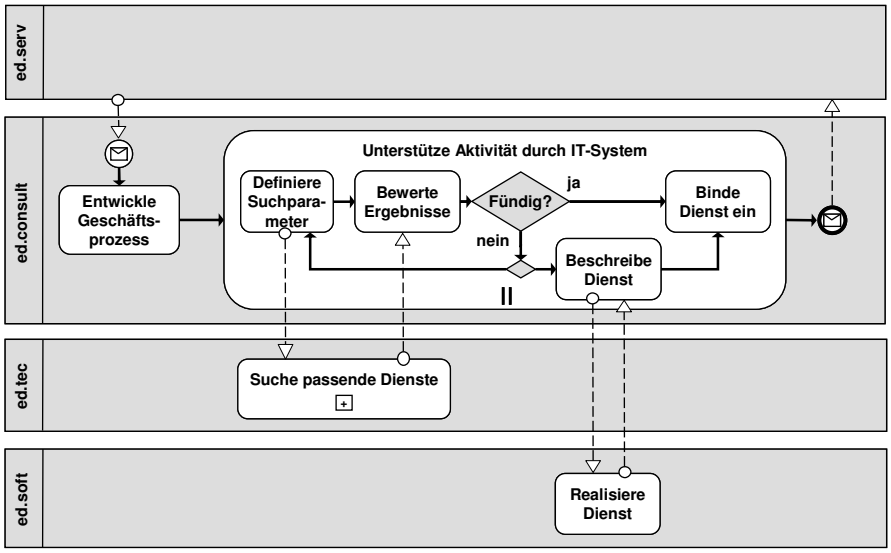
1.4 Formatierung und Schreibweise

Grundsätzlich ist diese Arbeit in deutscher Sprache verfasst und es wurde auch großen Wert darauf gelegt wo möglich entsprechende deutsche Bezeichnungen zu verwenden. In einigen Fällen, bei denen es kein geeignetes verständliches Äquivalent im Deutschen gibt, wurde auf die englische Schreibweise ausgewichen, die dann im Text in Kursivschrift hervorgehoben ist. Eigennamen von Gesellschaften und Produkten bilden eine Ausnahme dieser Regel. Englisch wurde auch in solchen Fällen gewählt, wo in Informationsabbildungen oder Text zu Prozessmodellen des betrachteten Geschäftsbereichs, zu einer Softwareentwicklung oder zu Dienstbeschreibungsinhalten erstellt wurden, da sich bei der Softwareentwicklung Englisch als Standardsprache etabliert hat. Für Informationen, die Diagramme in Business Process Modeling Notation (BPMN, Kapitel 2.1) zeigen, gibt es dabei eine weitere Besonderheit: Diagramme mit Bezug zum betrachteten Geschäftsbereich sind Grau auf Weiß formatiert, wohingegen Diagramme, die die Metaebene der Softwareentwicklung zum Gegenstand haben, umgekehrt Weiß auf Grau eingefärbt sind.

1.5 Beschreibung des Demonstrators

Ausgangspunkt für die Betrachtungen und Beispiele in dieser Arbeit ist der Geschäftsbereich der Aus- und Weiterbildung, der durch das repräsentative IT-Supported-Training-Szenario (IST-Szenario, Kapitel 2.3) greifbar gemacht wird.

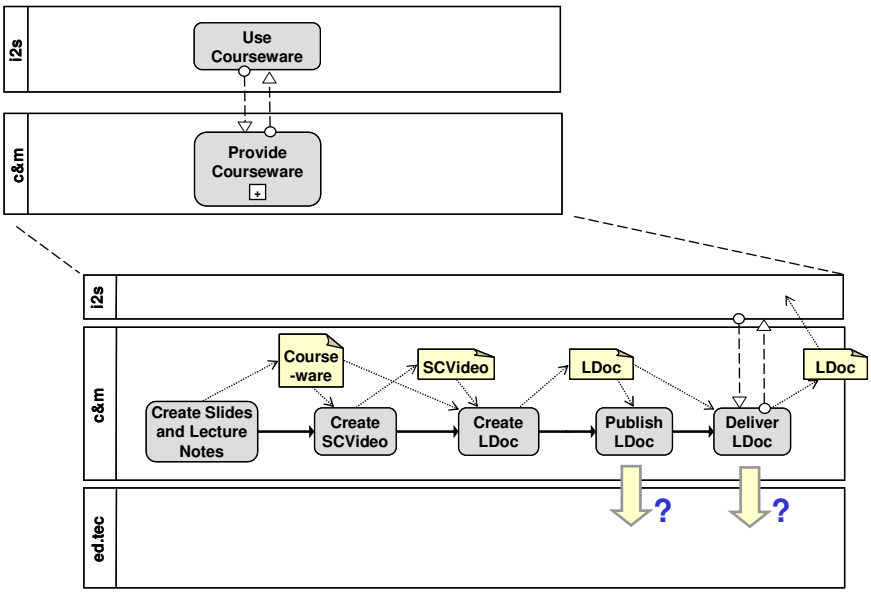
Das Schulungsunternehmen c&m plant seinen Erfolg mit neuen Arten von Schulungen, die geänderte Abläufe mit sich bringen, auszubauen und ist dabei auf die IT-Unterstützung seines Betreibers ed.serv angewiesen. Um flexibler und schneller auf die Anforderungen und Geschäftsprozesse seiner Kunden eingehen zu können, hat der IT-Dienstleister ed.serv in Beratung mit seinen Partnern ed.consult und ed.soft beschlossen, sein ed.tec-Softwaresystem ausgerichtet an einer serviceorientierten Architektur zu restrukturieren. Dieses Vorhaben begünstigt auch die Planung von ed.serv, vermehrt externe Systeme an die ed.tec-Umgebung durch ed.consult anbinden zu lassen und seine IT-Dienste nicht nur als Komplettlösung direkt an Schulungsunternehmen wie c&m zu verkaufen, sondern auch einzelne wieder verwendbare Dienste oder individuell zusammengestellte Dienstmodule im Markt für IT-Dienstleistungen zu platzieren.



Information 3: Nutzung der Dienstsuche beim Programmieren im Großen

Es soll erreicht werden, dass mit Änderungen beauftragte Softwareentwickler von ed.soft und ed.consult sowie externe Interessenten die bereits von ed.serv betriebenen Dienste auf einfache Weise anhand der Beschreibung ihrer Funktionalität in einem Verzeichnis auffinden können, um diese Informationen beim Programmieren im Großen nutzen zu können [EM+05]. Dies dient der IT-Unterstützung von Geschäftsprozessen und vereinfacht das Erweitern des vorhandenen Dienstangebots bei Bedarf [Cl06]. In Information 3 ist in einem BPMN-Diagramm (Kapitel 2.1) eine Möglichkeit skizziert, wie die Suche im Dienstverzeichnis zur Unterstützung bei der Umsetzung der Geschäftsprozesse eingesetzt werden kann.

Im Folgenden wird die Anwendung dieses Vorgehens an einem konkreten Beispiel vorgeführt und aufgezeigt, wie dabei die Dienstsuche eingesetzt werden kann. Information 4 zeigt einen Ausschnitt aus einem Geschäftsprozess, den c&m mit seinem Kunden i2s abwickelt (Kapitel 2.3.2).

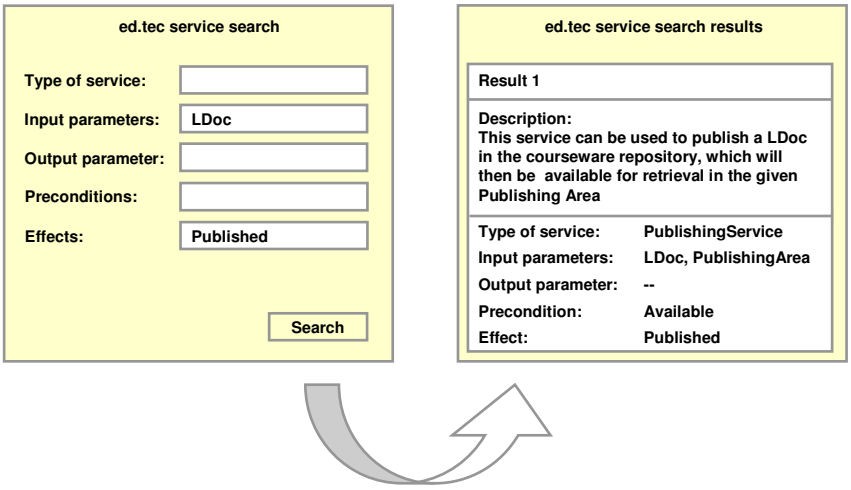


Information 4: Erstellung des Provide-Courseware-Unterprozesses

In diesem Geschäftsprozess kommt die Aktivität *Provide Courseware* vor (Information 4, oben), die durch ed.tec unterstützt werden soll. Bei dieser Aktivität versorgt c&m seinen

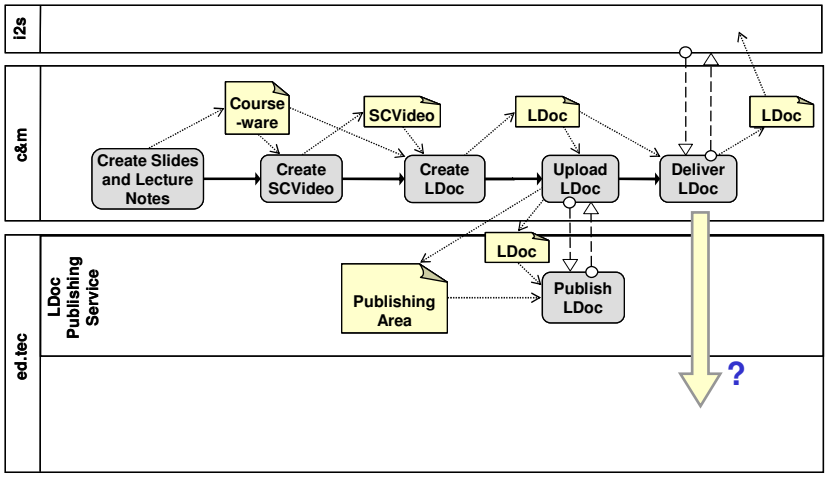
Kunden mit dem Lernmaterial zu seiner Schulung. Um eine IT-gestützte Umsetzung zu erreichen, wird der Prozess zunächst verfeinert. Unter anderem will c&m seinem Kunden i2s ein Living Document (LDoc) zur gebuchten Schulung zugänglich machen. Dies soll allerdings rechnergestützt über das Internet geschehen und nicht etwa per Post. c&m soll dabei lediglich die Aufgabe zukommen, das LDoc zur Verfügung zu stellen. Die Publizierung (*Publish LDoc*) und Auslieferung (*Deliver LDoc*) soll durch das ed.tec-System realisiert werden.

Für den Softwareentwickler von ed.consult, der mit der Umsetzung des Prozesses betraut ist, stellt sich nun die Frage, ob das ed.tec-System bereits die Dienste zur Publizierung und Auslieferung eines LDocs enthält und diese wiederverwendet werden können. Dazu wird eine im ed.tec-Dienstverzeichnis integrierte Suchfunktion verwendet, deren Aufruf in Information 5 dargestellt wird.



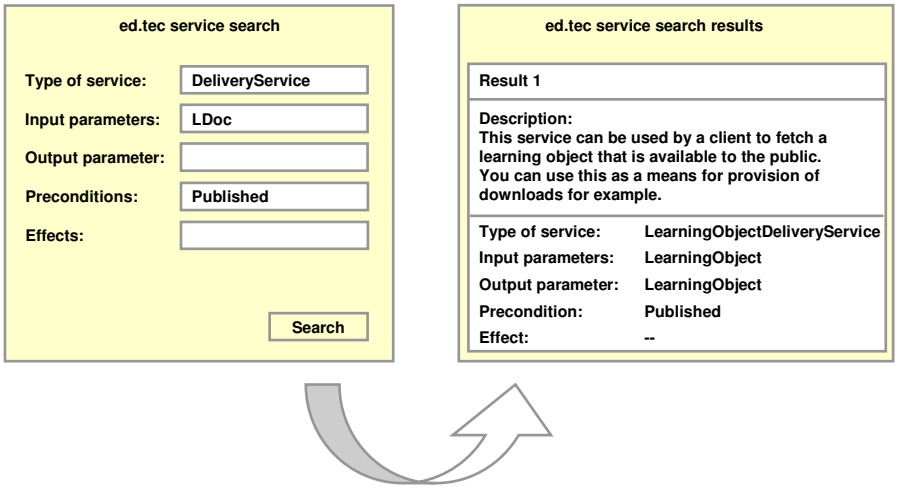
Information 5: Dienstsuche in ed.tec - Publizierungsdienst

Um die Suche einfach zu gestalten, dienen als Eingaben semantische Informationen über den zu suchenden Dienst. Im gezeigten Fall wird nach einem Dienst gesucht, der ein LDoc als Eingabe annimmt und als Nachbedingung *Published* erzielt. Gefunden wurde ein spezieller Publizierungsdienst, der ein LDoc in einer zu bestimmenden *PublishingArea* veröffentlicht (Kapitel 2.3.3). Da der gefundene Dienst der gesuchten Funktionalität entspricht, kann dieser manuell in den Prozess integriert (Information 6) und mit dafür geeigneten Werkzeugen bei der Umsetzung der Rechnerunterstützung eingebunden werden.



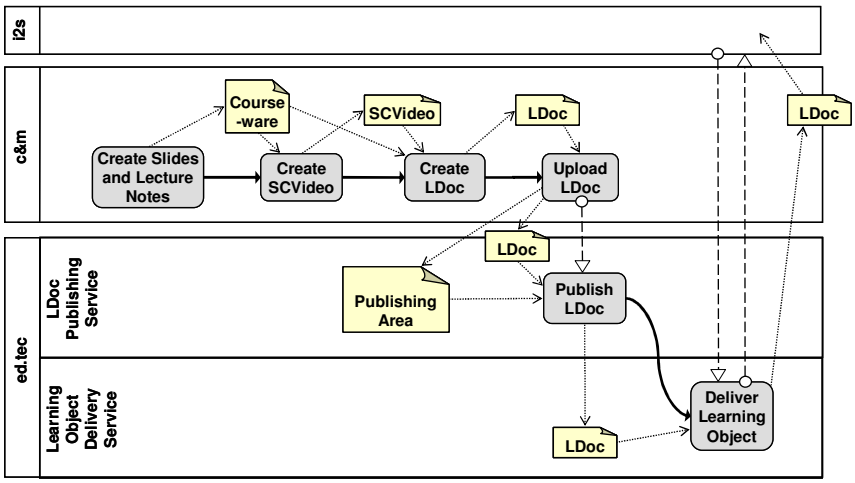
Information 6: Einbindung des Publizierungsdienstes

Jetzt kann mit dem nächsten Dienst fortgefahren werden. Für die Unterstützung des Prozesses wird nun ein Auslieferungsdienst für LDocs benötigt. Es wird also eine Suchanfrage für einen *DeliveryService* formuliert, der ein LDoc als Eingabe annimmt, das zuvor publiziert wurde (Information 7). Wie auch bei der ersten Anfrage bleiben einige der Eingabefelder der Suchmaske leer und werden nicht miteinbezogen. Wie man an den angedeuteten Suchergebnissen sieht, werden die gemachten Angaben nach ihrer Bedeutung interpretiert und nicht nur als reines Textsuchmuster verwendet.



Information 7: Dienstsuche in ed.tec - Auslieferungsdienst

In diesem Fall wurde kein spezieller Dienst für die Auslieferung von LDocs gefunden. Das Suchsystem hat jedoch anhand der Semantikinformationen erkannt, dass ein LDoc auch ein *LearningObject* ist und einen dazu passenden Auslieferungsdienst (Kapitel 2.3.3) gefunden. Bei genauerer Betrachtung der Dienstbeschreibung und Parameter kann man erkennen, dass dieser allgemeine Dienst den geforderten Ansprüchen genügt und abschließend in den Prozess eingebunden werden kann (Information 8).

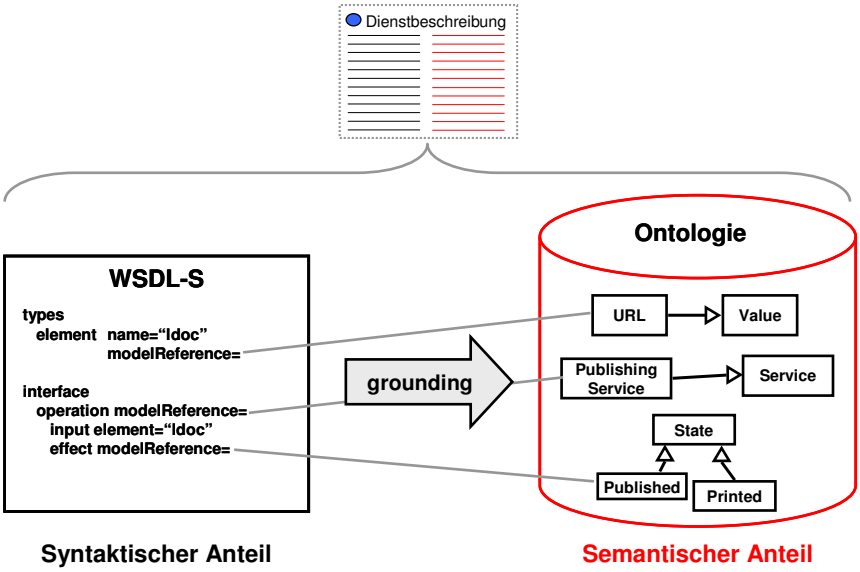


Information 8: Einbindung des Auslieferungsdienstes

Die soeben aufgezeigte manuelle Dienstverschaltung mit Suche stellt wesentliche Ansprüche an die Beschreibung der vorhandenen Dienste. Die Beschreibungen der Ein- und Ausgabe sowie der Vor- Nachbedingungen müssen dem Suchenden logisch erscheinen und die verwendeten

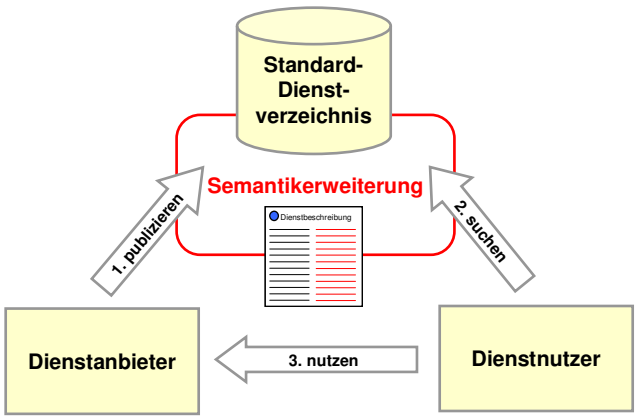
Wörter sollten einheitlich aus einer definierten und dokumentierten Taxonomie stammen, um Missverständnisse zu vermeiden.

ed.consult erhält den Auftrag für die Konzeption des Gesamtsystems der nötigen Dienstbeschreibung und des Dienstverzeichnisses sowie der Konzeption des gemeinsamen Entwicklungsvorgehens zusammen mit ed.serv und ed.soft. Um den Anforderung an die Dienstbeschreibung gerecht zu werden, rät ed.consult zur Verwendung von formalen semantischen Dienstspezifikationen auf der Basis von Ontologien (Kapitel 3.3.2). Bei diesen kann die zum Aufruf der Dienste benötigte syntaktische Schnittstelleninformation mit semantischen Konzepten verknüpft werden. Dadurch können der Diensttyp, Ein- und Ausgabeparameter und Vor- und Nachbedingungen eines Dienstaufrufs genauer spezifiziert werden (Information 9).



Information 9: Anteile der Dienstbeschreibung

Für die Nutzbarmachung der semantischen Dienstbeschreibung in einem Dienstverzeichnis gibt es bisher keine Standards. Das standardisierte Dienstverzeichnis für die Veröffentlichung von Webservices, das UDDI-Verzeichnis (Kapitel 2.2.4), wird den Anforderungen für eine semantische Suche nur sehr eingeschränkt gerecht [JM+03] [PK+02] [MP+04] [SH+03]. Um eine semantische Suche zu realisieren muss es ergänzt werden (Information 10). Ein Konzept für eine solche Ergänzung, passend zum eingesetzten semantischen Dienstbeschreibungsstil, sowie eine prototypische Umsetzung des Verzeichnisses soll durch ed.consult erarbeitet werden.



Information 10: Dienstverzeichnis mit Semantikerweiterung

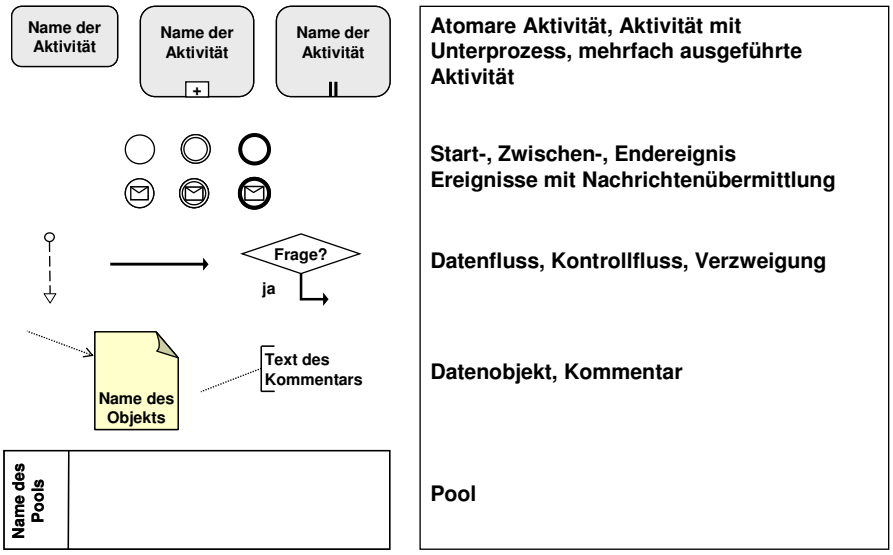
1.6 Gliederung der Arbeit

Der Aufbau der folgenden Ausarbeitung entspricht im Wesentlichen den verfolgten Fragestellungen. Nach dieser Einleitung wird im Kapitel 2 zunächst zur Lektüre der Folgekapitel benötigtes Hintergrundwissen zusammenfassend aufbereitet. Kapitel 3 setzt sich dann mit den gängigen Techniken und der aktuellen Literatur auseinander, die für die Lösung des gestellten Problems relevant erscheinen. Die Kapitel vier bis sechs enthalten die Erarbeitung des Dienstverzeichnisses. Zunächst wird die zu Grunde liegende Art der semantischen Dienstbeschreibung konzipiert (Kapitel 4) und dann untersucht und aufgezeigt wie diese mit einem Dienstverzeichnis verknüpft werden kann (Kapitel 5). Der Nachweis für die Praktikabilität der Beschreibungsmethode und die Tauglichkeit des Verzeichnisses wird anhand eines möglichen Entwicklungsvorgehens in Kapitel 6 erbracht. Kapitel 7 beinhaltet eine Zusammenfassung des Erreichten und gibt einen kurzen Ausblick, bevor Verzeichnisse und weitere Anhänge die Ausarbeitung abschließen.

2 GRUNDLAGEN

2.1 Geschäftsprozessmodellierung mit BPMN

Für die Modellierung von Geschäftsprozessen kommt in dieser Diplomarbeit die Business Process Modeling Notation (BPMN) [BPMI-BPMN1.0] zum Einsatz. BPMN bietet einen Baukasten an graphischen Objekten zur visuellen Darstellung beliebig komplexer Prozesse. In diesem Unterkapitel werden die BPMN-Elemente erläutert, die in den Diagrammen dieser Arbeit Verwendung finden. Eine Übersicht gibt Information 11:



Information 11: BPMN-Elemente

In Kapitel 1.5 sind einige Geschäftsprozesse aufgeführt. Als Beispiel für die BPM-Notation kann der Prozess "Nutzung der Dienstsuche beim Programmieren im Großen" aus Information 3 dienen, der die meisten der oben aufgeführten Modellierungselemente enthält. Ein Geschäftsprozess findet immer in einem Pool statt. Als Beschriftung für den Pool wird in der Regel die Organisation oder Person gewählt, die den Prozess ausführt. Im Beispiel sind dies ed.serv, ed.consult, ed.tec und ed.soft. Ein Prozess beginnt entweder mit einem Startereignis, wie im Pool von ed.consult einer Nachrichtenübermittlung, oder mit der (dann unbedingt eindeutigen) Aktivität, die keinen eingehenden Kontrollfluss besitzt, wie im Pool von ed.tec. Der Prozess verläuft entlang des Kontrollflusses, der Aktivitäten und Verzweigungen bis zu einem Endereignis oder zu einer Aktivität, die keinen ausgehenden Kontrollfluss mehr besitzt. Verschiedene Prozesse, die dann in unterschiedlichen Pools liegen, können durch Datenfluss miteinander kommunizieren. Der Beispielprozess zeigt noch einige erweiterte Möglichkeiten von BPMN. So können Aktivitäten Unterprozesse besitzen. Im Beispiel von Information 3 ist dies für die Aktivitäten "Unterstütze Aktivität durch IT-System" und "Suche passenden Dienst" der Fall. Bei letzterer ist der Unterprozess allerdings nicht ausnotiert, sondern nur dessen Existenz durch das Plus-Symbol innerhalb der Aktivität angezeigt. Die Aktivität mit dem ausnotierten Unterprozess besitzt zudem noch eine Markierung mit zwei parallelen Balken, was bedeutet, dass die Aktivität mehrfach ausgeführt wird. Im Beispiel wird dies für jede Aktivität stattfinden, die durch ein IT-System unterstützt werden soll. Eine weitere Besonderheit liegt im Pool von ed.serv vor: ed.serv nimmt am dargestellten Szenario ebenfalls mit einem Prozess teil, der aber für den aktuellen Modellierungsfokus nicht wichtig ist und daher ausgeblendet wurde. Bei diesem sogenannten abstrakten Prozess wird lediglich der Datenfluss über die Kante des Pools modelliert. Wie an anderer Stelle in Kapitel 1.5 zu sehen, können weitergereichte Informationen mittels Datenobjekten modelliert werden. Zusätzlich kann jedes Modellierungselement bei Bedarf mit Textkommentaren versehen werden.

2.2 Dienste und Webservices

2.2.1 Dienste

Der Begriff des Dienstes ist sehr weitreichend und schließt jegliche manuell erbrachten Dienstleistungen ein [Li04:20], jedoch auch Dienstleistungen, die unter Mitwirkung von IT-Systemen erbracht werden. Bei diesen spricht man auch von IT-Diensten. Über die genaue Bedeutung der Bezeichnungen Dienst und IT-Dienst gehen die Meinungen in der Fachwelt auseinander [Pr04]. Die C&M-eigene Definition von IT-Dienst erweitert die Definition von Dienstleistung des internationalen Standards ISO 8402, 1995-08 und lautet:

> Ein IT-Dienst ist ein immaterielles Produkt eines IT-Dienstleisters, das entweder auf IT basiert oder im Zusammenhang mit IT steht und dem Zweck dient, den Zustand des IT-Dienstnehmers unmittelbar zu verbessern. Der Zustand des Dienstnehmers bezieht sich dabei auf seinen Informationsstand, sein Wissen, seine wirtschaftliche Situation usw. Ein IT-Dienst zeichnet sich durch seine Funktionalität bzw. Leistung sowie durch eine Menge von Dienstmerkmalen aus.

Die Überlegungen in dieser Diplomarbeit befassen sich nicht mit Diensten im Allgemeinen; stattdessen wird ein eingeschränkter Betrachtungswinkel zu Grunde gelegt. Sie beziehen sich auf die spezielle Art von IT-Diensten, die durch Webservices realisiert sind. Zur vereinfachten Lesbarkeit wird in dieser Ausarbeitung das Wort Dienst als Synonym für Operation eines webservicebasierten IT-Dienstes verwendet (siehe auch Kapitel 2.2.3).

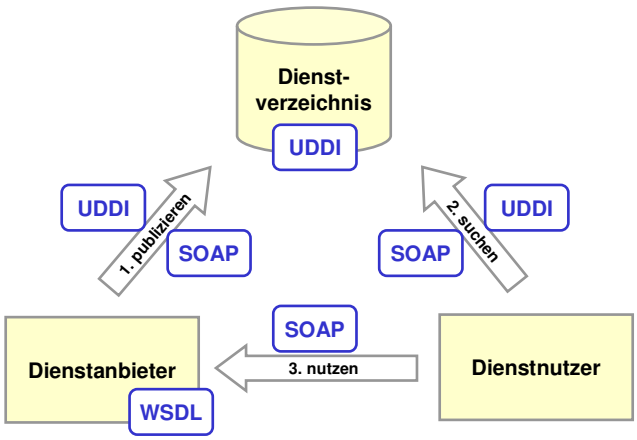
2.2.2 Webservices

Webservices sind eine Technologie für die Erstellung von Softwarekomponenten mit Schnittstellen, die mittels standardisierter Kommunikationsprotokolle über das Internet angesprochen werden können. Kommunikation und Datenformate basieren auf gängigen Standards, denen oftmals die Extensible Markup Language (XML) [W3C-XML] zu Grunde liegt. Ein Webservice, der auf einem Server betrieben wird, kann als IT-Dienst gemäß der obigen Definition angesehen werden und trägt damit zum Grundgedanken hinter Webservices bei, das Internet zu einer verteilten Ansammlung von Diensten auszubauen. Das World Wide Web Consortium (W3C) [URL-03], das maßgeblich bei der Entwicklung der Webservicestandards beteiligt ist, definiert den Begriff Webservice folgendermaßen:

> *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.* [HB04]

Um die Interoperabilität unterschiedlicher und verteilter Softwaresysteme über das Internet zu erreichen, dienen bei Webservices einerseits etablierte Kommunikationsprotokolle wie das Hypertext Transfer Protocol (HTTP) [IETF-HTTP1.1] und das Simple Mail Transfer Protocol (SMTP) [IETF-SMTP] andererseits auch neuere Standards. Darunter die Web Service Description Language (WSDL, Kapitel 2.2.3) zur Beschreibung der Webserviceschnittstelle, das Simple Object Access Protocol (SOAP) [W3C-SOAP1.2], das die Nachrichtenübermittlung bei der Inanspruchnahme der Webservices regelt und UDDI (Kapitel 2.2.4), das Dienstverzeichnisse zur Vermittlung von Webservices beschreibt. Information 12 zeigt eine Einordnung dieser Standards in das allgemein anerkannte Nutzungsmodell der Webservices, nach dem ein Dienstanbieter seine Webservices in einem Dienstverzeichnis publizieren kann.

Dienstnutzer können solche veröffentlichten Webservices und deren Anbieter suchen und anschließend die Dienstfunktionalität direkt beim Anbieter in Anspruch nehmen.



Information 12: Webservicemodell mit verwendeten Standards nach [JM+03]

2.2.3 WSDL

Zur Spezifikation von Webserviceschnittstellen kommt die Web Service Description Language (WSDL) zum Einsatz, die durch das W3C standardisiert wurde. Obwohl mittlerweile eine Standardisierung der Version 2.0 vorliegt, wird von gängigen Webservice-Implementierungen oftmals die Version 1.1 [W3C-WSDL1.1] verwendet, die auch in dieser Arbeit als Grundlage dient. In WSDL-Dateien, denen die XML-Syntax zu Grunde liegt, werden zu jedem Webservice (*service*) die Kommunikationsendpunkte (*ports*) und die davon unterstützten Operationen (*operations*) definiert. Diese sind wiederum mit ein- und ausgehenden Nachrichten (*input*, *output*) verknüpft. Die Struktur der beim Dienstaufruf übertragenen Nachrichten wird durch XML-Schema-Datentypen (XSD) [W3C-XS-DT] festgelegt. Anhand der WSDL-Dateien kann ein Dienstnutzer die syntaktischen Kommunikations- und Schnittstelleninformationen und Zugriffsadressen in Erfahrung bringen, die er benötigt, um einzelne Operationen des Webservice aufzurufen. Gerade eine solche Operation ist es, die in der weiteren Ausarbeitung als Dienst bezeichnet wird.

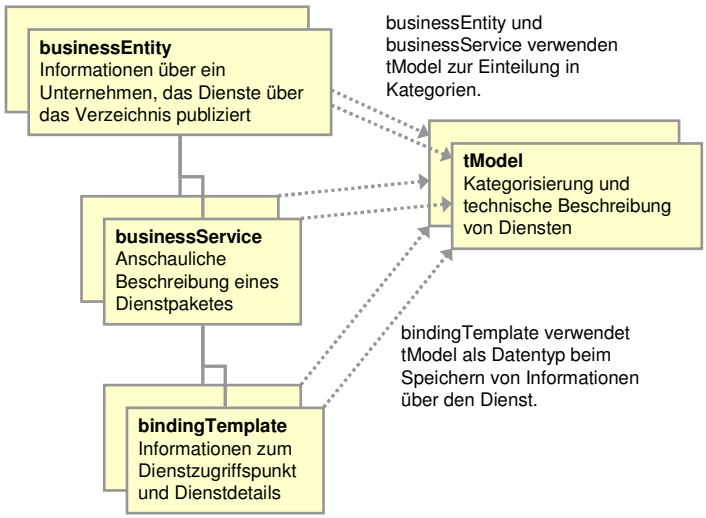
2.2.4 UDDI

Die Abkürzung UDDI steht für Universal Description, Discovery and Integration of Web Services. Dahinter verbirgt sich ein standardisiertes Dienstverzeichnis für die Veröffentlichung und Suche von Webservices. UDDI, das ursprünglich als Projekt der Unternehmen Ariba, IBM und Microsoft erarbeitet wurde, wurde 2002 an die Organization for the Advancement of Structured Information Standards (OASIS) [URL-04] übergeben, standardisiert und liegt mittlerweile in der Version 3.0.2 [OASIS-UDDI3.0] vor. In zahlreichen Implementierungen des Standards und in der Wirtschaft wird allerdings regelmäßig die Version 2 eingesetzt, die auch in dieser Arbeit Verwendung findet.

Ein UDDI-Verzeichnis kommt einem Katalog gleich, in den Informationen über Webservices und über die Unternehmen, die diese anbieten, eingetragen werden können, um sie später bei einer Suche auffinden zu können. Dazu werden die Datenstrukturen [OASIS-UDDI2.0-DS], die Zugriffsschnittstellen [OASIS-UDDI2.0-API], die Betriebsrichtlinien [OASIS-UDDI2.0-OS] und der Datenaustausch [OASIS-UDDI2.0-RS] in Bezug auf UDDI-Verzeichnisse definiert.

Datenstrukturen

Die Informationen, die in UDDI gespeichert werden, lassen sich im Wesentlichen in die unten beschriebenen Bereiche *businessEntity*, *businessService*, *bindingTemplate* und *tModel* einteilen. Gemäß dem Standard werden die Daten in XML repräsentiert und ausgetauscht, wofür auch XML-Schemata [URL-05] gegeben sind.



Information 13: UDDI-Datenstruktur nach [OASIS-UDDI2.0-DS]

Die ***businessEntity*** ist ein Container für Informationen über ein Unternehmen oder einen Unternehmensteil, der Dienste im Verzeichnis publiziert hat. Unter anderem können Name, Beschreibung, Kontaktdaten, Identifikationsschlüssel und eine Kategorisierung von *businessEntity*s gespeichert werden. Die *businessEntity* enthält alle vom Unternehmen veröffentlichten Dienste als *businessService*s und die Kategorisierung wird mittels *tModel*s umgesetzt.

Die ***businessService***-Informationseinheit fasst mehrere logisch verwandte Dienste zusammen und kann daher verwendet werden, um ein Paket von Diensten zu kategorisieren und zu beschreiben. Das Dienstpaket wird durch eine Menge von *bindingTemplate*s angegeben. Die Kategorisierung verwendet *tModel*s.

Das ***bindingTemplate*** enthält Informationen über einen einzelnen Webservice. Diese sind eine Beschreibung, der Dienstzugriffspunkt, sowie Referenzen auf *tModel*s, die den Dienst genauer spezifizieren.

tModel ist die Kurzform für *taxonomyModel*. Ein *tModel* ist eine Datentypdefinition, die bei ihrer Referenzierung mit Werten gefüllt werden kann. Dies geschieht im Sinne von Name-Wert-Paaren. Dabei wird auf ein *tModel* durch seine Identifikation Bezug genommen und das festzulegende Paar von Name und Wert zusammen mit der Referenz, der so genannten *keyedReference*, angegeben. *tModel*s sind Metadatensätze, die aus *bindingTemplate*s referenziert werden, um die technischen Details der Dienste zu spezifizieren. *tModel*s können auch verwendet werden, um *businessEntity*s, *businessService*s, *bindingTemplate*s und andere *tModel*s zu identifizieren und zu kategorisieren. Im Einzelnen definiert sich ein *tModel* durch seine Identifikation, seinen Namen, eine Beschreibung und eine Referenz auf externe Dokumentation.

Zugriffsschnittstellen

Der UDDI-Standard definiert Schnittstellen, die von Programmierern genutzt werden können, um Operationen auf dem UDDI-Verzeichnis aufzurufen. Zur Nachrichtenübermittlung kommen dabei SOAP-Aufrufe [W3C-SOAP1.2] mit den in der Spezifikation angegebenen XML-Nachrichten zum Einsatz. Die verfügbaren Operationen lassen sich in zwei Bereiche einteilen.

Abfragefunktionen wie beispielsweise `find_Service`, `find_binding` oder auch `get_serviceDetail` dienen der Suche nach Verzeichniseinträgen und dem Abruf von Inhalten der Verzeichniseinträge.

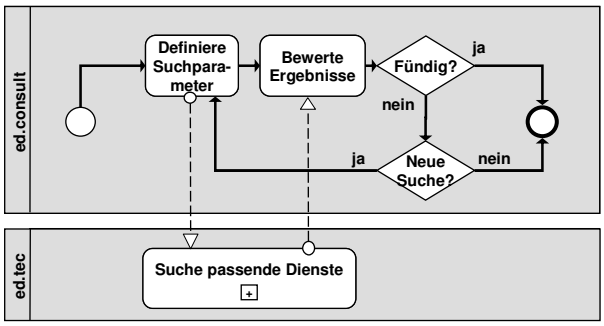
Publizierungsfunktionen wie zum Beispiel `save_binding` oder `delete_service` werden verwendet, um Verzeichniseinträge zu erstellen, zu modifizieren oder zu entfernen.

2.2.5 Halbmanuelle Dienstsuche

Ziel des zu entwickelnden Dienstverzeichnisses ist die Unterstützung bei der halbmanuellen (halbautomatischen) Dienstsuche. Sie wird von einer Person mit Unterstützung durch ein IT-System durchgeführt und lässt sich daher in zwei Richtungen abgrenzen, die hier jedoch nicht weiter betrachtet werden sollen: Die manuelle Dienstsuche, bei der eine Person die Einträge einer Liste aller verfügbaren Dienste durchsieht und manuell mit den gestellten Anforderungen vergleicht, und die automatische Dienstsuche, bei der ein Softwaresystem selbstständig Anforderungen an einen zu suchenden Dienst ermittelt, die Suche durchführt und mit der Entscheidung für genau einen ausgewählten Dienst endet. Letztere ist ebenfalls Gegenstand aktueller Forschungsarbeiten, noch nicht ausgereift und nicht in jedem Szenario praktikabel [SH+03].

Ablauf

Die halbmanuelle Dienstsuche wurde erstmals in Kapitel 1.5 anschaulich wiedergegeben und läuft wie in Information 14 dargestellt ab.



Information 14: Der Ablauf der halbmanuellen Dienstsuche

(1) Der Suchende, hier ein Mitarbeiter von ed.consult, gibt in einer Suchmaske (Information 15) einige Parameter als Suchbegriffe ein, die er aus den Anforderungen an den zu suchenden Dienst ableitet.

(2) Das Suchsystem, hier das Dienstverzeichnis der ed.tec-Software, führt die Suche durch und präsentiert dem Suchenden die zu den Suchparametern passenden Ergebnisse.

(3) Der Suchende betrachtet die Ergebnisliste (Information 15), bewertet die Ergebnisse und fällt darauf aufbauend seine Entscheidungen. Zunächst muss festgestellt werden, ob sich unter den Ergebnissen mindestens ein Dienst befindet, der den Anforderungen gerecht wird. Ist dies der Fall, so muss noch unter den passenden Diensten gewählt werden.

(4) Ist dies jedoch nicht der Fall, so besteht die Möglichkeit, dass eventuell mit einer neuen Suche und anderen Suchparametern ein passender Dienst gefunden werden kann.

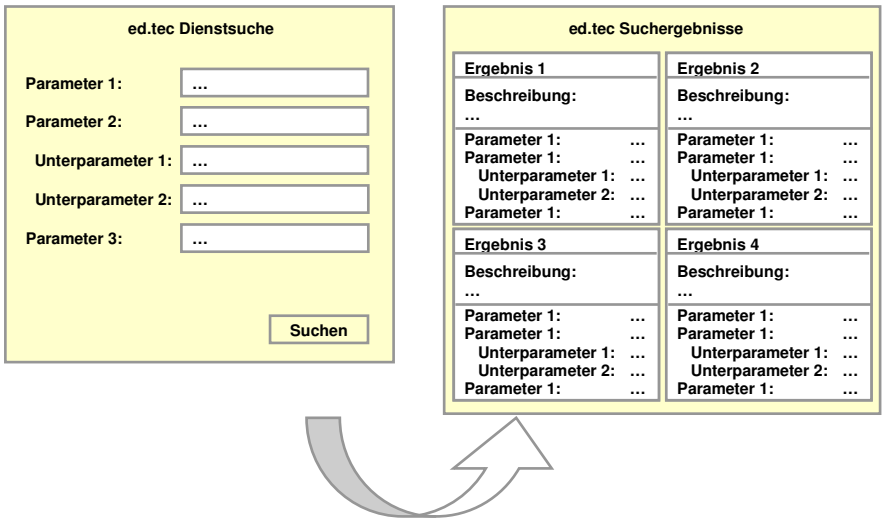
Bewertung der Effizienz

Die Bewertung klassischer *Information-Retrieval*-Systeme kann anhand von *Recall* und *Precision*, Nutzeraufwand, Antwortzeit, Ergebnispräsentation und Abdeckung vorgenommen werden [SM87:172-178]. Für die halbmanuelle semantische Dienstsuche werden hier der Nutzeraufwand und die Antwortzeit betrachtet. Die Ergebnispräsentation spielt zwar ebenfalls eine wichtige Rolle, wirkt sich aber direkt auf den Nutzeraufwand aus, und wird daher nicht gesondert behandelt.

Wenn bei einer Suche kein passender Dienst gefunden wurde, kann dies unterschiedliche Gründe haben. Entweder gibt es keinen passenden Dienst oder er wurde aufgrund der angegebenen Suchparameter von der Suchlogik nicht gefunden. Unabhängig davon kann der Suchende sich entscheiden, es nochmals mit neuen Parametern zu versuchen, so dass Parametrisierung, Suche, und Bewertung erneut durchgeführt werden. Suchwiederholungen wirken daher als Multiplikator für den Nutzeraufwand und die Antwortzeit und sollten vermieden werden.

Nutzeraufwand

Der Aufwand der halbmanuellen Dienstsuche bestimmt sich durch mehrere Größen, die den einzelnen Schritten im Ablauf der Suche zugeordnet werden können. Leider sind die unterschiedlichen Werte jedoch nicht voneinander unabhängig. So kann beispielsweise durch eine Mehrinvestition bei der Formulierung der Suchparameter der später folgende Bewertungsaufwand verringert oder eine Suchwiederholung vermieden werden. Dies hat zur Folge, dass sich keine allgemeingültige mathematische Formel für den Gesamtaufwand angeben lässt, allenfalls eine auf empirischen Untersuchungen basierende Schätzung.



Information 15: Masken bei der halbmanuellen Dienstsuche

Unter **Parametrisierungsaufwand** wird der (meist zeitliche) Aufwand zur Formulierung der Suchparameter verstanden. Er hängt einerseits von der Anzahl der erforderlichen Suchparameter ab, andererseits aber auch von deren Komplexität und wie leicht die geforderten Informationen zu beschaffen sind.

Der **Bewertungsaufwand** bezeichnet die Zeit, die vom Benutzer zu investieren ist, um die Suchergebnisse durchzusehen und mit den gestellten Anforderungen zu vergleichen. Sie hängt wesentlich von der Anzahl der Listeneinträge sowie von der Komplexität und Eignung der Ergebnispräsentation für einen Vergleich mit den an den Dienst gestellten Anforderungen ab.

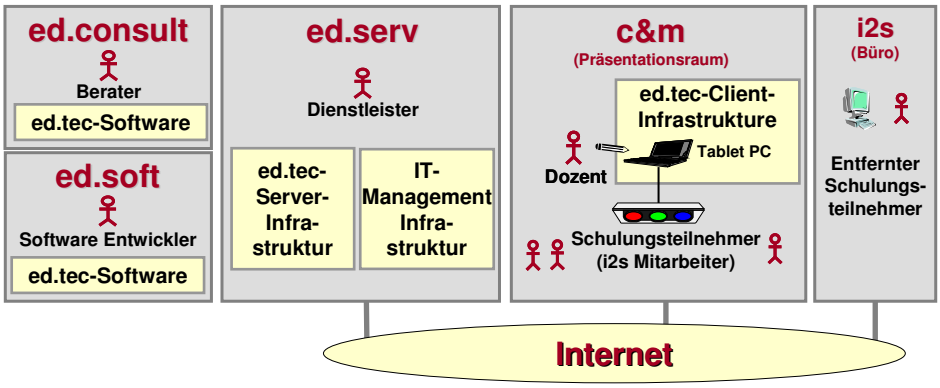
Antwortzeit

Die Antwortzeit ist bei der halbmanuellen Dienstsuche die Zeit, die das Suchsystem benötigt, um die Suchanfrage auszuwerten, die passenden Dienste zu finden und eine Ergebnisliste zusammenzustellen.

2.3 Zugrunde liegendes Szenario

Alle Beispiele dieser Diplomarbeit sind am Szenario der IT-gestützten Schulung (*IT-Supported Training*, IST) ausgerichtet, das den betrachteten Geschäftsbereich verdeutlicht.

2.3.1 Übersicht



Information 16: Beteiligte Partner im Szenario "IT-Supported Training"

(1) Das Schulungsunternehmen cooperation & more (c&m) bietet Kurse über Informationstechnik an. Dabei setzt es verstärkt IT-Unterstützung ein, um seinen Kunden eine qualitativ hochwertige Dienstleistung zu bieten. c&m entwickelt sein Kursangebot häufig weiter und setzt dabei auf innovative Konzepte.

(2) intelligent internet solutions (i2s), ein IT-Unternehmen mit Kompetenzen im Bereich Webapplikationen, ist ein Kunde von c&m. Die Angestellten von i2s besuchen Kurse von c&m, um sich über neue Webtechnologien zu informieren.

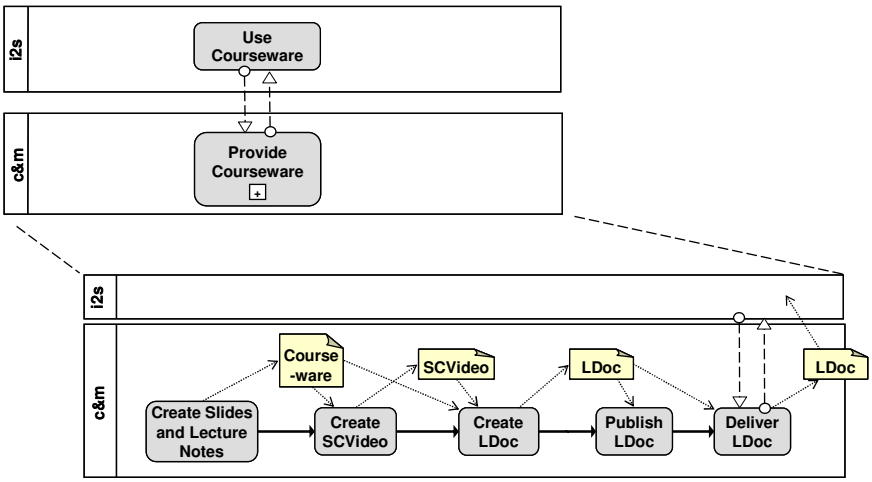
(3) Der IT-Dienstleister educational.services (ed.serv) ist Spezialist für IT-gestützte Schulungen, der seinen Kunden hochwertige Dienste bereitstellt, die er mit seinen Partnern entwickelt. Einer seiner Kunden ist c&m. ed.serv betreibt die ed.tec-Umgebung, die c&m nutzt, um seine Geschäftsprozesse abzuwickeln.

(4) ed.soft ist ein Softwarespezialist für E-Learning- und Management-Software. Im vorgestellten Szenario entwickelt ed.soft neue, an die Bedürfnisse von ed.serv angepasste ed.tec-Komponenten in Form von Webservices.

(5) ed.consult ist ein Unternehmen für IT-Beratung. Eine Kernkompetenz von ed.consult ist die Konzeption von dienstorientierten Lösungen für die Aus- und Weiterbildung. Eine weitere ist die Integration von Standard-Software-Komponenten in eine bestehende E-Learning-Umgebung.

2.3.2 Der Beispielprozess "Provide Courseware"

Alle Beispiele und Veranschaulichungen in dieser Arbeit sind aus dem Kontext der Aus- und Weiterbildung gegriffen. Zur weiteren Eingrenzung werden im Wesentlichen nur Ausschnitte eines Geschäftsprozesses von c&m und seinen Partnern betrachtet. Es handelt sich dabei um den bereits in Kapitel 1.5 angesprochenen Geschäftsprozess *Provide Courseware*, den c&m durchführt, um seinen Schulungsteilnehmern Schulungsmaterial zur Nutzung anzubieten (Information 17).



Information 17: Der Beispielprozess "Provide Courseware"

Der Prozess besteht aus mehreren Aktivitäten, wovon hier nur die für die Beispiele relevanten aufgezeichnet sind. Der Ablauf insgesamt sieht wie folgt aus: Zunächst werden ein Schulungsskript und die zugehörigen Präsentationsfolien erstellt, hier unter dem Begriff *Courseware* zusammengefasst (*Create Slides and Lecture Notes*). Anschließend wird der Vortrag gehalten und dabei aufgezeichnet, so dass ein Bildschirmvideo (*screen capture video*, *SCVideo*) entsteht (*Create SCVideo*). Zusammen mit den Folien und dem Skript wird das Video zu einem *Living Document* (LDoc) verarbeitet (*Create LDoc*). Dabei handelt es sich um ein innovatives interaktives Lernmaterial, das die Vorteile von textbasierten Skripten und Videoaufzeichnungen zum Lerninhalt vereint. Im nächsten Schritt wird die Existenz und Verfügbarkeit des neuen LDoc publiziert, etwa indem es in einen Katalog mit den angebotenen Schulungsmaterialien aufgenommen wird (*Publish LDoc*). Auf Nachfrage des Kunden i2s wird diesem schließlich das LDoc ausgeliefert (*Deliver LDoc*).

2.3.3 Beispieldienste

Im Folgenden werden zwei Dienste betrachtet, die zur Durchführung einzelner Aktivitäten des soeben vorgestellten Geschäftsprozesses bemüht werden können. Beide wurden bereits in Kapitel 1.5 angesprochen. Die beiden Dienste dienen in dieser Arbeit lediglich als Beispiele für die Veranschaulichung der vorgestellten Konzepte und Lösungen. Die Publizierung und Auslieferung von Lernobjekten ist bereits für sich ein komplexes Thema, das hier nicht im Detail betrachtet wird [VW05a].

Der LDoc-Publizierungsdienst

Kurzbeschreibung:	Ein Living Document (LDoc) zu einer Schulung wird auf den Schulungsmaterialserver (courseware server) übertragen, so dass es von Schulungsteilnehmern aufgefunden und heruntergeladen werden kann.
Vorbedingungen:	Das fertige LDoc steht zur Verfügung. Im Schulungsmaterialserver existiert bereits ein Downloadbereich (PublishingArea) für Material zu dieser Schulung.
Eingaben:	Aktueller Ablageort des LDoc (URL), Identifikation des gewünschten Downloadbereichs
Nachbedingungen:	Das LDoc wird in der Materialliste des Downloadbereichs aufgeführt und ist als Download für die Schulungsteilnehmer verfügbar. Das LDoc kann über die Suchfunktion des Schulungsmaterialservers aufgefunden werden.
Ausgaben:	
Ablauf:	(1) LDoc auf den Server übertragen (2) Metainformationen des Dokuments und Downloadlink in die Materialliste übernehmen (3) Metainformationen des Dokuments für die Suche indizieren

Information 18: Der LDoc-Publizierungsdienst (LDocPublishingService)

ed.serv betreibt für jedes Schulungsangebot, das von c&m angeboten wird, einen Schulungsmaterialserver (*courseware server*), mit dessen Hilfe c&m seinen Kunden Lernmaterial (so genannte Lernobjekte) für seine Schulungen verfügbar macht. Für die LDocs kann c&m den LDoc-Publizierungsdienst (*LDocPublishingService*) nutzen um dies zu erreichen. Bei der Ausführung des Dienstes wird das LDoc auf den Schulungsmaterialserver übertragen und dort in eine Sammlung von Schulungsmaterial, auch Downloadbereich (*PublishingArea*) genannt, zu einer bestimmten Schulung aufgenommen. Danach werden den Schulungsteilnehmern, die über das Internet auf den Materialserver zugreifen, auf der WWW-Seite mit einer Liste aller Materialien in diesem Downloadbereich auch das neu hinzugefügte LDoc und seine Metainformationen, wie zum Beispiel Autor oder Publikationsdatum, angezeigt. Auch mit der Suchfunktion des Materialservers können die Schulungsteilnehmer das LDoc finden.

Der Auslieferungsdienst für Lernobjekte

Kurzbeschreibung:	Ein Lernobjekt zu einer Schulung, das auf dem Schulungsmaterialserver (courseware server) veröffentlicht wurde, soll für die Auslieferung an einen Schulungsteilnehmer (z.B. zum Download) abgerufen werden.
Vorbedingungen:	Das Lernobjekt wurde zuvor auf dem Schulungsmaterialserver veröffentlicht.
Eingaben:	Identifikation des Lernobjekts
Nachbedingungen:	Das Lernobjekt wurde an den Aufrufenden übertragen
Ausgaben:	Das Lernobjekt
Ablauf:	(1) Das Lernobjekt anhand der Identifikation lokalisieren (2) Das Lernobjekt an Aufrufenden übertragen

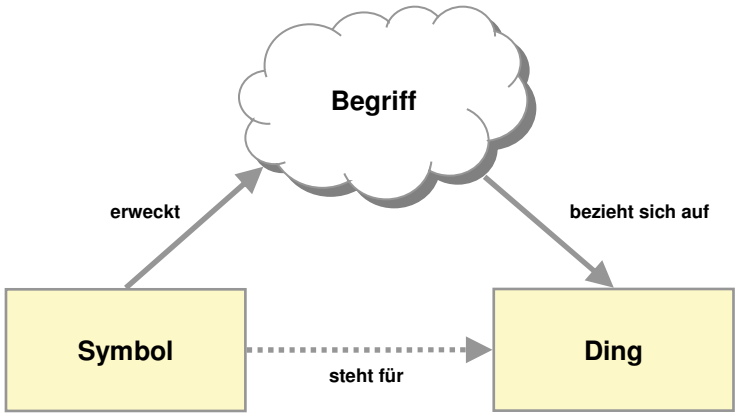
Information 19: Der Auslieferungsdienst für Lernobjekte (LearningObjectDeliveryService)

Unabhängig davon, wie die Schulungsteilnehmer ein Lernobjekt auf dem Materialserver aufgefunden hat, kann er es über einen Link herunterladen. Dazu wird im Hintergrund der Auslieferungsdienst für Lernobjekte (*LearningObjectDeliveryService*) tätig. Seine Aufgabe ist es, ein Lernobjekt, wie zum Beispiel ein LDoc, das auf dem Server veröffentlicht wurde, an die aufrufende Instanz auszuliefern. Das gewünschte Lernobjekt kann dabei anhand der beim Aufruf mitgelieferten Identifikationsinformation lokalisiert werden und wird anschließend als Ausgabeparameter des Dienstes übertragen.

2.4 Explizite Begriffsbildung und Semantik

People can't share knowledge if they don't speak a common language.
Thomas H. Davenport [DP98:98]

Die Kommunikation zwischen Personen beruht stark auf einem gemeinsamen Begriffsverständnis der beteiligten Partner. In der Sprachwissenschaft wird diese Abhängigkeit durch das semiotische Dreieck (u.a. Maedche et al. [MS+01], Ogden und Richards [OR49]) illustriert.



Information 20: Das semiotische Dreieck [MS+01]

Es verdeutlicht die Beziehungen zwischen Symbolen (z.B. dem geschriebenen Wort), den zugehörigen Begriffen und den tatsächlich in der Welt existierenden Dingen. Dabei ist der "Begriff" als das in der Vorstellung des Menschen existierende Bild eines Dings zu verstehen. Bei der Kommunikation werden zwischen den Teilnehmern lediglich Symbole ausgetauscht, wodurch sich Schwierigkeiten ergeben können. So erwecken Symbole je nach Teilnehmer und dessen Hintergrund und Wissen unterschiedliche Begriffe. Auch die Dinge selbst können durch Begriffe nicht immer vollständig erfasst werden, da das menschliche Gehirn nur mit abstrakten Abbildungen der Wirklichkeit arbeitet. Die gestrichelt angedeutete Verbindung zwischen Symbol und Ding ist keine direkte Beziehung zwischen den beiden, sondern ergibt sich nur implizit über die anderen beiden Kanten des Dreiecks [OR49].

Dieses Thema ist auch ein wesentlicher Kernpunkt bei der gemeinsamen Nutzung und Übertragung von Daten in der Informatik. Man spricht hier bei übertragenen Symbolen von Nachrichten, die einer gewissen Syntax genügen. Das eigentliche Ziel der Kommunikation ist aber nicht eine reine Nachrichtenübermittlung, da der Empfänger in diesem Fall nicht weiß, was mit den empfangenen Nachrichten anzufangen ist. Während bei der menschlichen Wahrnehmung die Interpretation der Symbole quasi unterbewusst stattfindet, benötigen Maschinen eine passende Interpretationsvorschrift, um Nachrichten sinnvoll verarbeiten zu können. Diese ordnet den übertragenen syntaktischen Einheiten ihre Bedeutung, die auch Semantik genannt wird, zu und ermöglicht so eine Abbildung der Nachrichten auf Informationen [Ab05b:104-105]. Die Information entspricht also dem "Begriff" aus Information 20 und die Interpretation anhand der Vorschrift der skizzierten "erweckt"-Relation.

Offensichtlich lässt sich Davenports Aussage (siehe oben) nicht nur allein auf die Sprache an sich beziehen. Sie kann vielmehr auch dahingehend verstanden werden, dass ebenso wie die gemeinsame Sprache auf syntaktischer Ebene für erfolgreichen Wissensaustausch eine möglichst weitgehend übereinstimmende Interpretation des Gesprochenen in die Begriffswelt erforderlich ist. Jeder am Austausch beteiligte Partner muss ableiten können, welche Semantik sich hinter den verwendeten Bezeichnungen verbirgt, um erfolgreich zu kommunizieren.

Die angesprochene Problematik der Mehrdeutigkeit der "erweckt"-Relation behindert folglich die Kommunikation zwischen Menschen bzw. Maschinen untereinander und zwischen Mensch

und Maschine. Sie kann dadurch eingedämmt werden, dass die Interpretation der Symbole zu Begriffen, also der Übergang von syntaktischer Einheit zur Semantik, explizit und möglichst eindeutig vorgegeben wird. Um dies zu erreichen, müssen allerdings die verwendeten Begriffe definiert werden. Es kann vertreten werden, dass sich in den verwendeten Begriffsdefinitionen das Wissen über den betrachteten Ausschnitt der Wirklichkeit widerspiegelt, wenn auch die Frage, was unter Wissen überhaupt zu verstehen ist, weder vollständig geklärt noch unumstritten ist [DP98:5]. Zum Austausch von Wissen muss die Begriffswelt selbst und ihre Verknüpfung zu den Symbolen wiederum in der syntaktischen Ebene dargestellt werden. Zur Erstellung einer derartigen Wissensbasis gibt es einige Ansätze unterschiedlicher Mächtigkeit, die als semantische Modelle oder auch Wissensrepräsentationen bezeichnet werden. Darunter gibt es unter anderen Taxonomien, Thesauri, *Topic Maps*, semantische Wissensnetze und Ontologien [St02a] [UM+03] [Ec02].

2.5 Ontologien als Wissensrepräsentation

Ontologien im Sinne der Informatik sind eine Weiterentwicklung der semantischen Wissensnetze. Bevor unten ihre Anwendungsmöglichkeiten geschildert werden, soll nun zunächst der Begriff der Ontologie an sich und seine Ausprägung in der Informatik erläutert werden.

2.5.1 Der Begriff der Ontologie

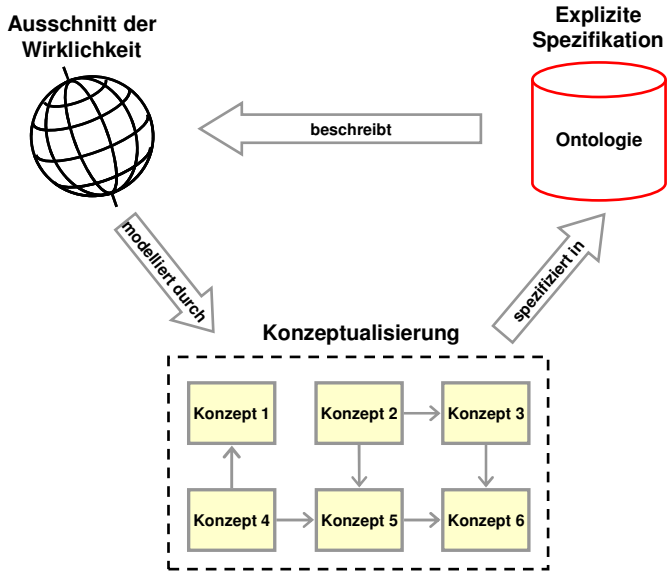
Die Bezeichnung Ontologie stammt aus dem Griechischen und wird seit dem 19. Jahrhundert für die Benennung einer von Aristoteles geprägten Disziplin der Philosophie verwendet, die sich mit der Existenz, dem Aufbau und der Systematik des Existierenden auseinandersetzt. Die von Aristoteles ursprünglich gewählte Bezeichnung war “Kategorien” [St02:8]. Der Begriff der Ontologie wurde im Informatikfachgebiet der künstlichen Intelligenz erstmals im Jahre 1991 von Neches et al. in ihrer Ausarbeitung zu Technologien für die gemeinsame Nutzung von Wissen eingesetzt [NF+91:38] [Gó99:33]. Inzwischen gibt es jedoch viele, teilweise widersprüchliche Definitionen für den Begriff der Ontologie [Gó99:34] [La05:25]. Eine der meistzitierten Definition im Sinne der Informatik [Gó99:33] [La05:27], die laut deren Autor selbst im Einklang mit der früheren philosophischen Bedeutung des Begriffs steht, stammt von Thomas Gruber [Gr93:199] aus dem Jahr 1993:

An ontology is an explicit specification of a conceptualization.

Diese Definition setzt sich ihrerseits aus zwei Teilen, der expliziten Spezifikation und der Konzeptualisierung, zusammen, die teils von Gruber weiter erläutert werden und teils andernorts definiert sind.

Laut Gruber baut jedes Wissensmanagementsystem auf einer Konzeptualisierung auf. Man versteht darunter Konzepte, Objekte und sonstige existierende Entitäten und die Relationen, die diese in Verbindung setzen. Sie stellt ein abstraktes Modell des im Blinkwinkel stehenden Ausschnitts der Wirklichkeit dar [SB+98:185] [La05:28].

Explizite Spezifikation heißt im Wesentlichen, dass alle in der Konzeptualisierung benutzten Konzepte und ihrer Benutzung auferlegte Einschränkungen ausdrücklich definiert sein müssen [SB+98:185]. Wie diese Definitionen auszusehen haben, ist nicht genauer festgelegt, meist werden dafür jedoch Prädikatenlogiken und so genannte Beschreibungslogiken (*description logics*, DL) eingesetzt [CJ+99:21,23] [BH+04]. Darüber hinaus sind sich die meisten Autoren einig, dass die Spezifikation formal, also lesbar und verarbeitbar für Maschinen, erfolgen soll.



Information 21: Ontologiedefinition nach Gruber [La05:28]

Ein weiterer Kerngedanke der Ontologien heute, um den Grubers Definition nachträglich erweitert worden ist, wurde aus ihren Anwendungen motiviert: Um in ihrem Zweck, Wissen über einen bestimmten Anwendungsbereich zu repräsentieren und zu vereinheitlichen, erfolgreich zu sein, müssen ihre Inhalte von allen Akteuren des Bereichs akzeptiert werden. Nur so kann eine gemeinsame Begriffsbildung unter allen Beteiligten erreicht werden. Man findet in der englischsprachigen Literatur daher oft die Bezeichnung *"shared conceptualization"* [Fe01:8] [Gó99:33].

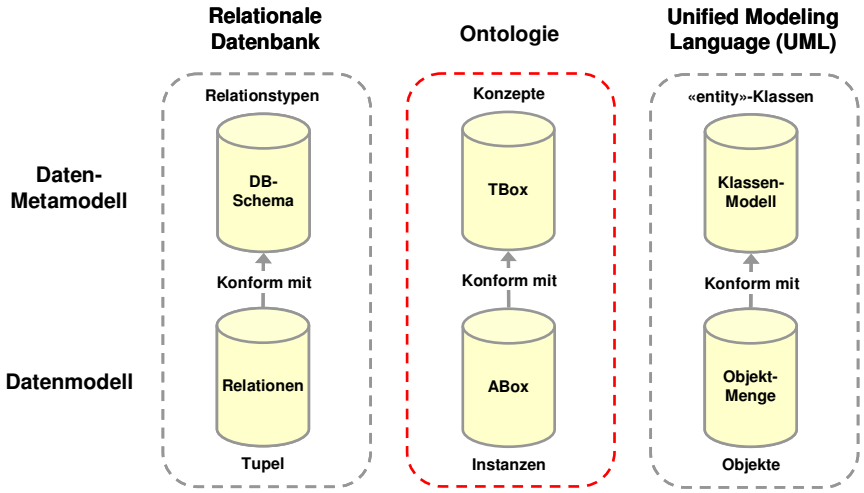
2.5.2 Ontologie in der Informatik

Die oben angeführte Definition ist sehr allgemein gehalten, daher soll nun ein genaueres Bild davon gegeben werden, was unter einer Ontologie in der Informatik verstanden wird. Würde man den philosophischen Begriff der Ontologie zu Grunde legen, so würde man implizit Ontologien verwenden oder erstellen, sobald man sich mit der Systematik der Entitäten eines Geschäftsbereichs befasst, auch wenn dabei von Ontologien nicht explizit die Rede wäre [NF+91:40] [La05:32]. Im Sinne der Informatik ist jedoch eine echte Spezifikation der Ontologieinhalte notwendig, da diese durch Rechnersysteme verarbeitet werden sollen.

Laut [CJ+99:20-21] gibt es aus der künstlichen Intelligenz heraus im Wesentlichen zwei Sichtweisen für Ontologien: zum einen die Ontologie als konkreten Wortschatz einschließlich einer Begriffswelt (*vocabulary*), die die Konzepte eines betrachteten Bereichs sprachunabhängig wiedergibt, zum anderen als Mittel zur Formalisierung von Wissensinhalten (*content theory*), die von KI-Systemen verarbeitet werden können, z.B. für logische Schlussfolgerungen oder durch Problemlösungsstrategien. Unabhängig davon welche der beiden Sichtweisen man verfolgt, liegt die Signifikanz von Ontologien für die Informatik darin, dass sie sich hervorragend dazu eignen, Wissen über den Betrachtungsbereich strukturiert aufzubereiten und damit den Austausch von Wissen zu ermöglichen und die Wiederverwendbarkeit von Wissen zu erhöhen.

Eine Ontologie ist, wie auch aus Information 21 deutlich wird, in der Regel an einem bestimmten Bereich oder einer Aufgabe ausgerichtet, die im Blickwinkel der Anwender liegen [CJ+99:20,23] [La05:28]. Einen solchen eingeschränkten Bereich der Realität bezeichnet man auch als Miniwelt. Die Ontologie stellt ein mögliches Modell dieser Miniwelt dar, das von einer oder mehreren Personen entwickelt und formell spezifiziert wurde. Folglich gibt es nicht eine einzige, beste oder genaueste Ontologie für einen Bereich, sondern beliebig viele unterschiedliche je nach Grenzziehung der Miniwelt und den Absichten und Vorstellungen der Entwickler [CJ+99:23]. Vergleichbar ist eine Ontologie etwa mit einem Datenbanksystem. Die

Semantik der Daten ist jedoch nicht in einem proprietären Datenbankschema oder dem Quellcode einer Anwendung versteckt, sondern explizit definiert [La05:29].



(1) TBox: Terminologie-Anteil (terminological components)
(2) ABox: Aussagen-Anteil (assertional components)

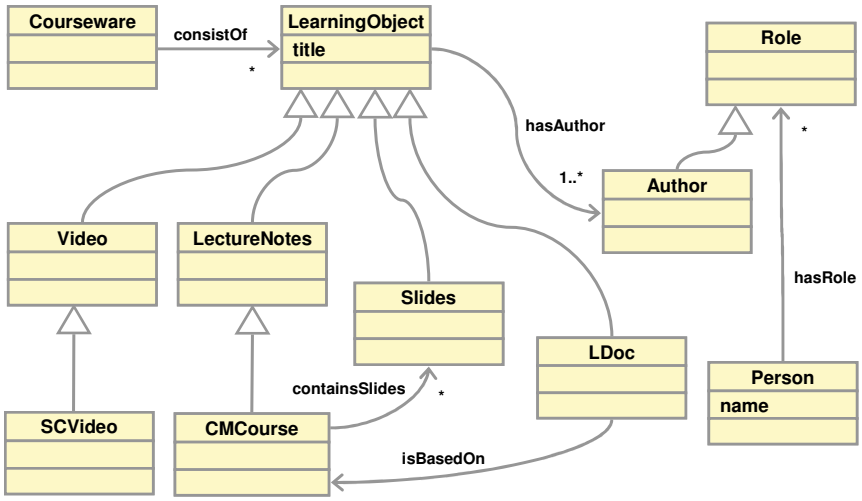
Information 22: Gegenüberstellung relationale Datenbank, Ontologie und UML

Auch wenn die Geschichte der Ontologien in der Informatik relativ jung ist und man auf andere Bezeichnungen stößt als beispielsweise bei relationalen Datenbanken (RDB) oder bei der objektorientierten Softwareentwicklung, lassen sich wie Information 22 zeigt im Aufbau zahlreiche Parallelen zu den bisherigen Ansätzen erkennen. Das Modell der Miniwelt kann auch bei Ontologien in die beiden Teile Datenmodell und Daten-Metamodell aufgespaltet werden [UM+03:7] [La05:26], wobei die Enge der Koppelung jedoch von der jeweiligen Ontologieausprägung abhängt. Auch die Inhalte und Art und Weise der Spezifikation in den beiden Teilen unterscheiden sich von Ontologie zu Ontologie. In [CJ+99:22] sind einige Gemeinsamkeiten ausformuliert, die verbreitet Anerkennung finden. Für die folgenden Ausführungen in diesem Kapitel liegt die Orientierung lediglich auf solchen Ausschnitten, die später benötigt werden.

TBox

Das Daten-Metamodell wird auch TBox genannt und enthält den Terminologieanteil der Ontologie. Dieser besteht aus der spezifizierten Konzeptualisierung der Miniwelt. Die darin enthaltenen Konzepte (Klassen, *concepts, classes*) repräsentieren gemeinsam das Vokabular der Miniwelt und legen die Eigenschaften (*properties*, *attributes*), Relationen (*relations*, *associations*) und logischen Zusammenhänge der so genannten Instanzen (*instances*, *individuals*, *objects*) aus der ABox fest [CJ+99:20] [UM+03:6]. Die Namensgebung dieser Begriffe ist in der Literatur nicht einheitlich und zum Teil an der jeweils verwendeten Technologie angelehnt, daher wurden hier Synonyme bzw. englische Bezeichnungen in Klammern mit angegeben. Für Konzepte wird oft, und so auch in dieser Arbeit, das Wort Klassen verwendet, wenn es um eine konkrete Umsetzung geht [HK+04:14].

Ein Beispiel für einen Ausschnitt aus der TBox einer Ontologie bietet Information 23. Hierbei wurde als graphische Notation das UML-Klassendiagramm (Kapitel 2.5.3) gewählt.

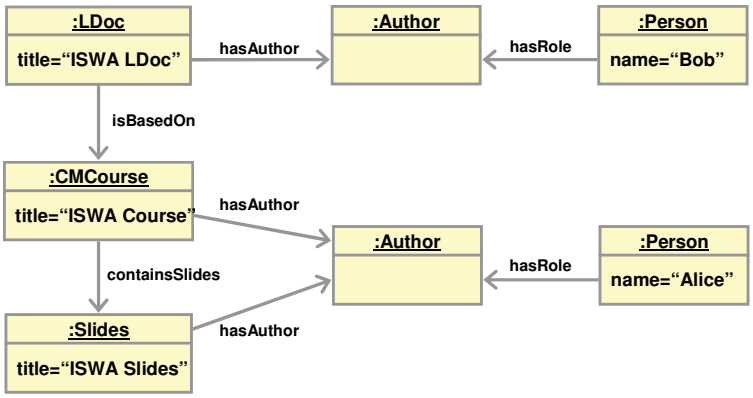


Information 23: Ausschnitt aus der TBox einer Ontologie

In dieser Ontologie sind verschiedene Konzepte spezifiziert. Teilweise sind sie in Unterklassenbeziehungen hierarchisch angeordnet wie hier unter anderem *LearningObject* und seine Derivate *Video*, *SCVideo*, *LDoc*, *Slides*, *LectureNotes* und *CMCourse*. Für manche Klassen sind Eigenschaften angegeben. So hat ein *LearningObject* einen Titel und eine Person einen Namen. Darüber hinaus existieren Relationen zwischen einzelnen Klassen: Einer Person kann beispielsweise eine Rolle zugeordnet werden und ein *LDoc* basiert auf einem *CMCourse*. *Courseware* setzt sich aus *LearningObjects* zusammen.

ABox

Im ABox genannten Datenmodell können die tatsächlich in der Miniwelt existierenden Repräsentanten modelliert werden. Dabei werden verschiedene Instanzen der in der TBox definierten Klassen erstellt, untereinander in Relation gesetzt und die Werte ihrer Eigenschaften festgelegt. Bei der Definition der Klassenzugehörigkeit der Instanzen kommt die bei Ontologien eingesetzte logische Sichtweise auf die Spezifikation zum Tragen: Bei der so genannten *open-world semantics* kann auf Grund der gegebenen Eigenschaften und Relationen einer Instanz implizit auf die Zugehörigkeit dieser Instanz zu einer oder mehreren Klassen geschlossen werden. Das heißt, dass Informationen, die in der Spezifikation nicht auftauchen, als unbekannt aber möglicherweise existent angenommen werden. Im Beispiel aus Information 24 ließe sich auf Grund der Tatsache, dass die Instanz rechts oben sowohl einen Namen als auch eine Rolle hat, folgern, dass es sich um eine Instanz der Klasse *Person* handeln muss, auch wenn dies nicht wie hier ausdrücklich angegeben wäre. Im Gegensatz dazu kommt bei Programmiersprachen in der Regel die *closed-world semantics* zum Einsatz. Bei dieser ist die Klassenzugehörigkeit explizit anzugeben woraus sich die Einschränkungen auf die Eigenschaften und Relationen der zugehörigen Instanzen ableiten lassen. Hier gilt: Was nicht spezifiziert ist, existiert auch nicht. Im UML-Objektdiagramm (Kapitel 2.5.3) der Information 24 ist für die Instanz rechts oben angegeben, dass sie der Klasse *Person* angehört. Mit Hilfe des Wissens über diese Klasse folgt dann, dass sie einen Namen und eine Rolle haben kann, nicht jedoch einen Autor, da sie keiner der Klassen *LDoc*, *CMCourse* oder *Slides* angehört [La05:33] [HK+04:20,69].



Information 24: Ausschnitt aus der ABox einer Ontologie

Das Diagramm veranschaulicht eine mögliche konforme Instantiierung des TBox-Ausschnitts aus Information 23 als UML-Objektdiagramm (Kapitel 2.5.3). Es zeigt ein Szenario, in dem ein LDoc mit dem Titel "ISWA LDoc" existiert, das von Bob auf der Grundlage eines *CMCourse* erstellt wurde, der seinerseits durch die Autorin Alice bearbeitet wurde.

2.5.3 Ontologieeinsatz

Die Einsatzmöglichkeiten von Ontologien sind zahlreich. Ontologien können in den verschiedensten Bereichen zur Verbesserungen beitragen. Ontologien zeigen durch ihre Eigenschaft der Begriffsstandardisierung überall dort Relevanz, wo natürliche Sprache interpretiert werden muss [CJ+99:23]. Ein solcher Anwendungsfall wären Transaktionen zwischen Geschäftspartnern. Hier ist beispielsweise ein gemeinsames Verständnis über Inhalte von Bestellungen oder Dienstleistungen erforderlich. Auch beim Wissensmanagement, das für Unternehmen eine wichtige Rolle spielt, können Ontologien eingesetzt werden, zum Beispiel zur Vernetzung von Wissensquellen oder zur Personalisierung beim Wissenstransfer [MS+01].

Ontologien bieten die Möglichkeit die Beziehungen in einer Anwendungsdomäne explizit und objektorientiert zu modellieren und zu spezifizieren. Dieses Vorgehen hat das Potential einen methodischen Wechsel in der Softwareentwicklung herbeizuführen: Ontologien sollen ermöglichen, das Wissen über Eigenschaften und Zusammenhänge eines Geschäftsbereichs, das vormals bei jeder Entwicklung wieder neu analysiert und versteckt in den Quellcode von Programmen eingearbeitet wurde, losgelöst von diesem zu formalisieren, um es schließlich über Anwendungsgrenzen hinweg erfolgreich wiederverwenden zu können [NF+91:37-38] [MS+01] [La05:26, 29] [CJ+99:23].

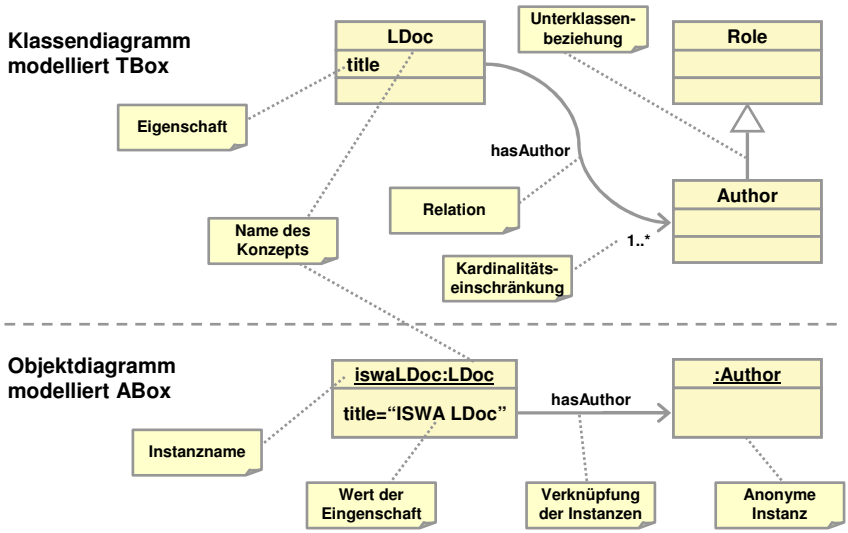
Eines der wichtigsten Anwendungsgebiete von Ontologien im Gebiet der Künstlichen Intelligenz ist bei der Verwirklichung von wissensbasierten Problemlösungsmechanismen. Aus den auf *description logic* basierenden Ontologieinhalten können mit Hilfe von logischen Kalkülen Folgerungen gezogen werden. Diese können zur Lösung von Aufgabenstellung oder zur Erweiterung der Wissensbasis genutzt werden [CJ+99:24] [AF+05]. Softwaresysteme, die solche Rückschlüsse erwirken, werden Inferenzmaschinen genannt. Bei der Verwendung mit Ontologien ist die englische Bezeichnung *reasoner* gebräuchlich.

2.5.4 Ontologierepräsentation

Bei der Frage, welche (möglichst standardisierte) Darstellung gewählt werden sollte, muss selbstverständlich die Zielgruppe und die hinter der Darstellung stehende Absicht in den Mittelpunkt gerückt werden. Für Ontologien gibt es unterschiedliche Möglichkeiten der Darstellung, wobei hier die in dieser Arbeit zur Verwendung kommenden Ansätze erläutert werden sollen. Die (nicht standardisierte) UML-Darstellung von Ontologien dient im Wesentlichen als Visualisierung von kleinen Ontologieausschnitten zur Anschauung, während die OWL im Gegensatz dazu eine ausführliche formale Niederschrift der Ontologie zur Verarbeitung durch Rechnersysteme ist.

UML

Die Darstellung von Ontologien und Ontologieausschnitten mit Hilfe der Unified Modeling Language (UML) [OMG-UML2.0-SS] [OMG-UML2.0-IS] wird schon seit geraumer Zeit diskutiert und befürwortet [CP99] [BV+04]. Auch in dieser Arbeit wird die UML zur Visualisierung von Ontologieinhalten herangezogen. Dabei kommt jedoch nur ein kleiner Ausschnitt aus den Klassen- und Objektdiagrammen der UML zum Einsatz (Information 25).



Information 25: UML als Ontologierepräsentation

Ontologiekonzepte werden als UML-Klassen modelliert, ihre Eigenschaften als UML-Attribute und ihre Relationen als UML-Assoziationen, die mit Namen und falls gewünscht mit einer Einschränkung auf die Kardinalität versehen werden. Da Ontologiekonzepte keine Operationen aufweisen, bleibt der Operationenabschnitt der UML-Klassen immer leer.

Ontologieinstanzen werden als UML-Objekte modelliert und Instanzverknüpfungen gemäß der Konzeptrelationen als einfache Objektbeziehungen. Wenn der Instanzname keine Rolle spielt kann er weggelassen werden was zu einem anonymen UML-Objekt führt. Objektattribute können bei Instanzen mit Werten versehen werden.

OWL

Die Web Ontology Language (OWL) [W3C-OWL-O] [AH04] ist eine auf das Resource Description Framework (RDF) und RDF Schema [W3C-RDF-P] [W3C-RDF-S1.0] [Mc04] aufbauende XML-Sprache, die sich dazu eignet Ontologieinhalte standardisiert zu formalisieren und auszutauschen. OWL gliedert sich in drei jeweils aufwärtskompatible Untersprachen unterschiedlicher Mächtigkeit auf, zwischen denen man wählen kann. Die mächtigste Variante, OWL-*Full*, bietet volle Kompatibilität zu RDF. Eine OWL-*Full*-Ontologie ist allerdings nicht entscheidbar im Sinne der Berechenbarkeitstheorie und damit nur schlecht für die Abfrage durch *reasoner* geeignet. Bei OWL-DL *(description logic*) wurden Einschränkungen vorgenommen, so dass die Ontologien der Sprache durch DL-*Reasoner* erfasst und abgefragt werden können. OWL-*Lite* schließlich führt weitere Einschränkungen ein, die das Verständnis und die Erstellung von Werkzeugen erleichtern sollen, bietet aber schwächere Ausdruckskraft. So können beispielsweise keine disjunkten Klassen oder Kardinalitäten explizit angegeben werden.

3 STAND DER TECHNIK

3.1 Semantische Webservices

Für die semantische Beschreibung von Webservices gibt es noch keinen Standard. Das W3C hat allerdings mit der Semantic Annotations for WSDL Working Group [URL-06] ein Gremium eingerichtet, das einen solchen entwerfen soll. Ziel ist es dabei, den Status der W3C-*Recommendation* bis März 2007 zu erreichen. Wesentliche Arbeitsgrundlage für diesen Vorstoß ist die von IBM und der University of Georgia Research Foundation Inc. bereitgestellte WSDL-S-Einreichung [AF+05a] [URL-07]. Die Arbeitsgruppe hat jetzt bereits seit einigen Monaten ihre Arbeit aufgenommen und ausgehend von der WSDL-S-Einreichung ein erstes Arbeitsdokument erstellt [FL06].

Der Vollständigkeit halber sei hier noch erwähnt, dass es neben den oben genannten noch zwei weitere W3C-Einreichungen zur Verknüpfung von Semantikbeschreibungen mit Webservices gibt, die jedoch auf Grund der gewünschten Anlehnung an gängige Standards hier nicht weiter betrachtet werden. Dies sind zum einen die Web Service Modeling Ontology [URL-09] [URL-10] und zum anderen das Semantic Web Services Framework [URL-11] [URL-12].

Ein neuer Vorstoß zur semantischen Dienstbeschreibung, der einige interessante Aspekte enthält und einige Defizite der oben genannten Verfahren adressiert, ist die Diane Service Description (DSD) [Kl04] von Michael Klein. Die DSD wurde unter anderem an der Universität Karlsruhe (TH) im Rahmen des Projektes Dienste in Ad-hoc-Netzen (DIANE) [URL-13] entwickelt, und soll hier ergänzend zu den beiden Standardkandidaten kurz vorgestellt werden.

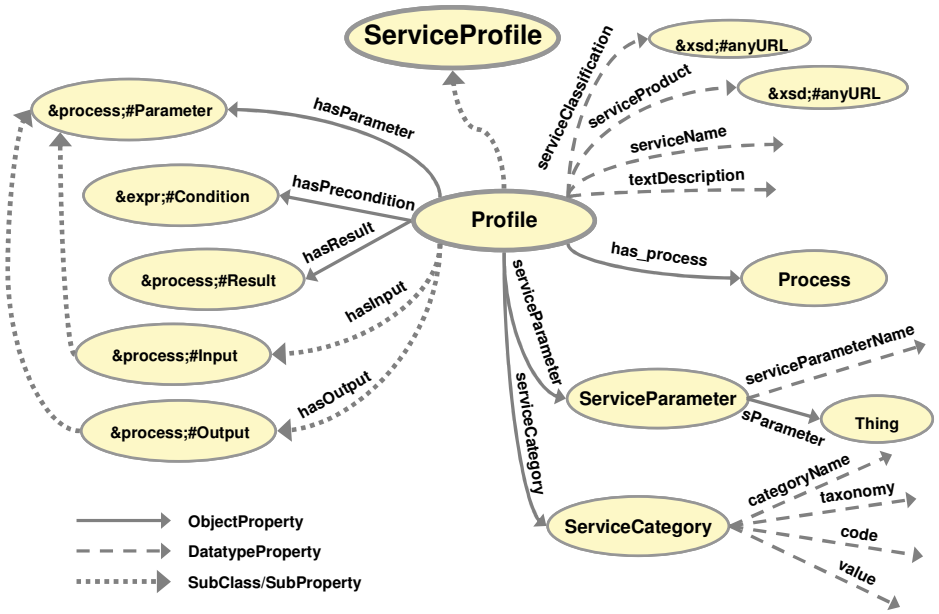
3.1.1 OWL-S

OWL-S [MB+04] [URL-14] steht für Web Ontology Language for Services. Sie wurde vormals unter dem Namen DAML-S [AB+02] basierend auf dem OWL-Vorgänger DAML+OIL entwickelt. Es ist einer der ältesten Ansätze für die semantische Beschreibung von Webservices. Das von der OWL-S-Koalition erklärte Ziel ist es, eine Beschreibungssprache für Webservices zur Verfügung zu stellen:

> OWL-S ist eine auf OWL basierende Webservice-Ontologie und bietet eine Grundmenge von Auszeichnungsprachenkonstrukten zur Beschreibung der Eigenschaften und Fähigkeiten von Webservices in einer eindeutigen und durch Computer interpretierbaren Form. [URL-08]

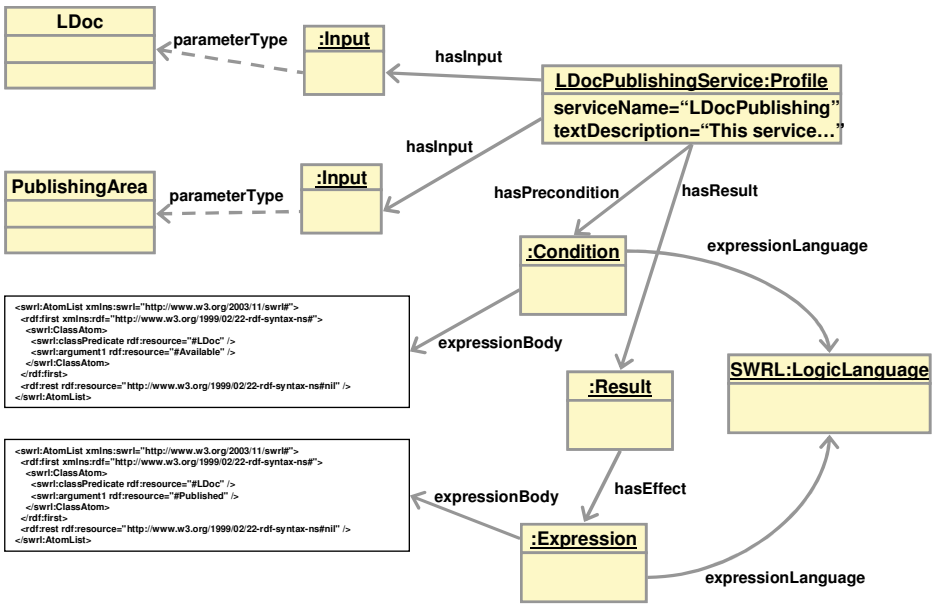
OWL-S ist eine in OWL gegebene Ober-Ontologie, die man nutzen und erweitern kann um etliche Aspekte von Webservices zu beschreiben. Dazu zählen Informationen für die Dienstsuche, -ausführung, -komposition und Interoperabilität. Die Möglichkeiten, die OWL-S bietet, sind sehr weitreichend und überlappen mit zahlreichen parallel zu OWL-S entstandenen Standards. So bietet OWL-S zum Beispiel ein Beschreibungsmodell für Prozesse bestehend aus mehreren Diensten und konkurriert, auch wenn das von den OWL-S-Autoren abgestritten wird [URL-15], zumindest teilweise mit der Business Process Execution Language for Web Services (BPEL) [OASIS-BPEL1.1].

Die Dienstbeschreibung ist bei OWL-S in drei Bereiche unterteilt. Das *ServiceProfile* gibt Auskunft darüber, was ein Dienst leistet; das *ServiceModel* darüber, wie er aufgebaut ist und abgearbeitet wird und das *ServiceGrounding* legt fest, wie auf ihn zugegriffen werden kann. Für die semantische Dienstsuche ist nur das *ServiceProfile* relevant und wird daher im Folgenden näher betrachtet.



Information 26: OWL-S: ServiceProfile [MB+04]

Information 26 zeigt einen Ausschnitt der Konzepte (auf Grund der Originaltreue hier oval dargestellt) und Relationen des *Profile* (welches seinerseits ein *ServiceProfile* ist), wie sie in [MB+04] definiert sind. Für alle Dienste können grundlegende Angaben wie Name, Beschreibung etc. gemacht werden (oben rechts). OWL-S ermöglicht aber auch für jeden Dienst die Angabe von Ein- und Ausgabeparametern sowie von Vor- und Nachbedingungen (*input*, *output*, *preconditions*, *effects*; IOPEs), dargestellt im linken Bereich. Mittels *ServiceCategory* kann eine Verknüpfung zu Standardisierungskatalogen oder Taxonomien hergestellt werden und mittels *ServiceParameter* können eigene Ergänzungsangaben zum Dienst erfasst werden (unten rechts). Zu guter Letzt kann noch jeder Dienst mit einem OWL-S-*Process* verknüpft werden. Bereits dieser kleine Ausschnitt lässt vermuten, dass sich hinter OWL-S große Komplexität verbirgt, was allein die Größe der OWL-S-*Profile*-Ontologie bestätigt. Inklusive ihrer *Imports* enthält sie 78 Klassen und 84 Relationen und Eigenschaften.



Information 27: OWL-S: Instantiierter Beispieldienst

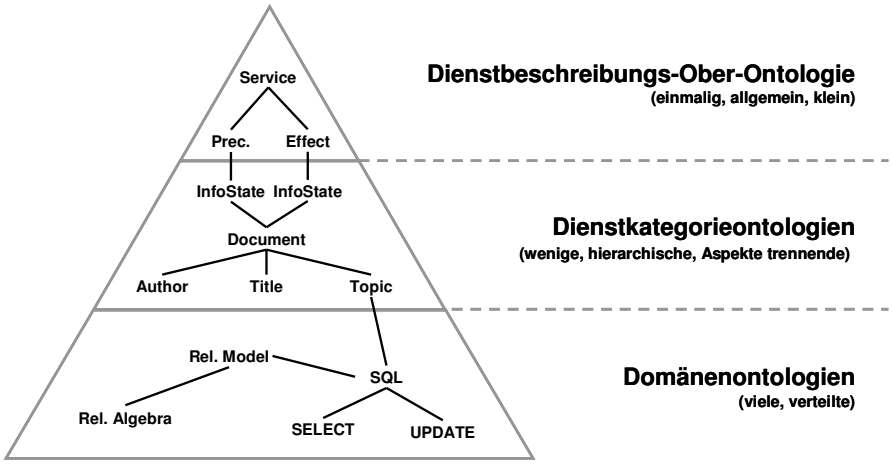
Information 27 zeigt einen kleinen Ausschnitt der OWL-S-Möglichkeiten an einer beispielhaften und auf das notwendigste beschnittenen Instantiierung der ontologischen Konzepte des *LDocPublishingService* aus Information 5. Zum Zwecke der Veranschaulichung ist Information 27 nur teilweise UML- und OWL-S-konform.

Die Spezifizierung der Typen von Eingabeparametern wird indirekt über Instanzen spezieller Parameterklassen (hier *Input*) erreicht. Bei Vor- und Nachbedingungen wird ähnlich vorgegangen. Hier kann über eine *Expression*-Instanz (bzw. deren Unterklasse *Condition*) wie angedeutet ein Ausdruck einer Logiksprache eingebunden werden. Dieser wird mit der Relation *expressionBody* festgelegt, wohingegen durch *expressionLanguage* noch referenziert werden muss, um welche Logiksprache es sich handelt. Im gezeigten Beispiel ist das die *Semantic Web Rules Language* (SWRL) [HP+04]. Bei den Nachbedingungen erhöht sich die Indirektion der Verknüpfung mit dem *ServiceProfile* nochmals durch den *Result*-Container, der noch weitere Informationen zum Ergebnis einer Dienstausführung beinhalten kann.

Obwohl bei der eben betrachteten Instantiierung eines Beispieldienstes nur ein geringer Teil der Möglichkeiten von OWL-S sichtbar wurde, – *ServiceModel*, *ServiceGrounding* und einige Teile des *ServiceProfile* blieben unberücksichtigt – entstand bereits eine verschachtelte Menge an Instanzen in der Ontologie. Mit der Komplexität der Dienstspezifikationen in der Ontologie steigt jedoch auch der Aufwand für Entwickler und Nutzer. Da die Anfragesprache bei der automatischen Suche die gleichen Möglichkeiten wie die Dienstspezifikation selbst aufweisen muss, erschwert sich sowohl die Erstellung der Spezifikation selbst als auch die Formulierung von Suchanfragen und deren Vergleich mit den angebotenen Diensten, [Kl06:56-58]. Es ist anzunehmen, dass sich dieser Mehraufwand auf beiden Seiten für die Verwirklichung einer halbmanuellen Dienstsuche nicht amortisiert.

3.1.2 Erstellung einer Dienstbeschreibung mit DAML-S

In ihrer Publikation zur Dienstbeschreibung mit dem OWL-S-Vorgänger DAML-S identifizieren Klein und König-Ries [KK03] im Wesentlichen drei verschiedene Arten von Ontologien, die dabei zum Einsatz kommen. Sie referenzieren sich gegenseitig, können aber bei der Umsetzung dennoch leicht getrennt gehalten werden. Dies sind zunächst die wiederverwendbaren Teile Dienstbeschreibungs-Ober-Ontologie (*service upper ontology*) und Domänenontologien der betrachteten Geschäftsbereiche. Ergänzt werden diese beiden durch die Dienstkategorieontologien, in der die Informationen über die verfügbaren Dienste zusammengestellt werden.



Information 28: Geschichtete Dienstontologie nach [KK03]

Die Dienstbeschreibungs-Ober-Ontologie ist ein Art Metamodell für die semantische Dienstbeschreibung und dient in dieser Eigenschaft als Schablone, aber auch als Erbgrundlage zur Erstellung eigener dienstbeschreibender Konzepte und Instanzen. Die Domänenontologien

finden bei der Beschreibung der Ein- und Ausgabeparameter sowie der Vorbedingungen und Effekte der Dienstaufrufe Verwendung. In den Dienstkategorienontologien werden die zu beschreibenden Dienste klassifiziert. Zusätzlich enthält sie alle Konzepte, die zur Beschreibung der Vorbedingungen und Effekte notwendig sind, aber nicht einer Domänenontologie zugeordnet werden können, da sie nur in Verbindung mit dem zu beschreibenden Dienst sinnvoll sind. Leider gibt [KK03] so gut wie keine Auskunft über die sinnvolle Verwendung der Ein- und Ausgabeparameter-Spezifikationen von DAML-S.

Für die Erstellung der semantischen Dienstbeschreibung schlagen Klein und König-Ries einen vierstufigen iterativen Prozess vor, der später in abgewandelter Form für unsere Zwecke herangezogen werden kann.

(1) Übernahme einer Ober-Ontologie (z.B. DAML-S)

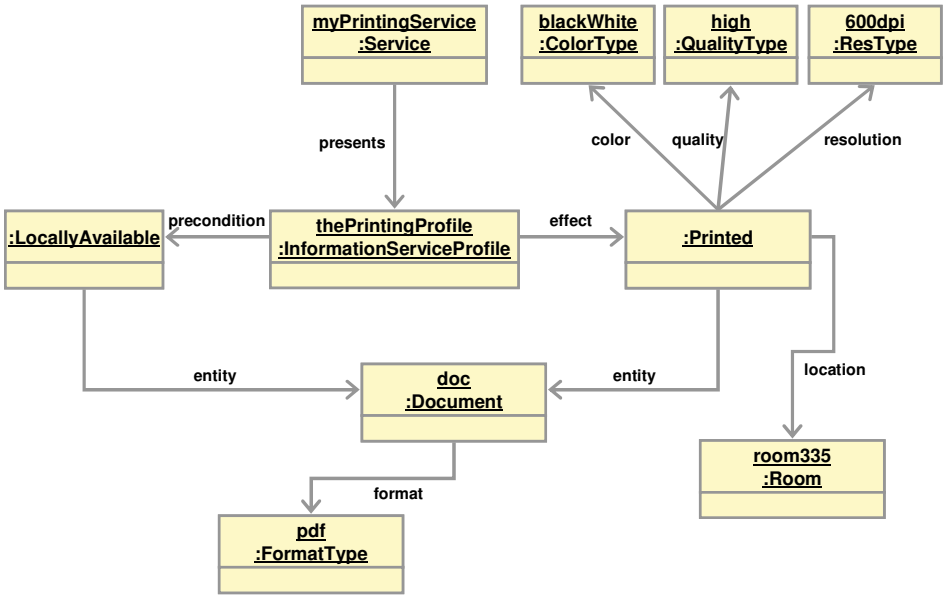
(2) Definition der Dienstkategorien
  (1) Kategorisierung von Dienstprofilen
  (2) (Definition von Vorbedingungen und Effekten)

(3) Einfügen von Domänenontologien
  (1) Festlegung von Eingaben und Rückgaben
  (2) (Verfeinerung von Vorbedingungen und Effekten)

(4) Instantiierung der Dienstbeschreibung
  (1) Umsetzen der abstrakten Dienstbeschreibung in eine konkrete

Information 29: DAML-S-Entwicklung nach [KK03]

In Schritt (1) wird lediglich die einheitlich über alle Dienstbeschreibungen verwendete Ober-Ontologie ausgewählt, in ihrem Fall ist dies der OWL-S-Vorgänger DAML-S. Der wichtigste und auch schwierigste Teil ist Schritt (2). Hier werden die Dienste in Kategorien und eine flache Hierarchie steigender Allgemeinheit eingeteilt. Zusätzlich werden weitere Konzepthierarchien für die unterschiedlichen Aspekte der Dienstbeschreibung erstellt. Im Beispiel von [KK03] (Information 28) wären dies die Zustände (*InfoState*) und die Informationen über das Dokument (*Document*, *Author*, *Title*, *Topic*). Meiner Meinung nach wäre es jedoch besser, diese einer Domänenontologie zu entnehmen, wenn eine passende zur Verfügung stünde. Solche Domänenontologien werden bei [KK03] erst im Schritt (3) eingebunden und mit der bereits vorhandenen Dienstbeschreibung verknüpft, wobei immer davon ausgegangen wird, dass die nötigen Domänenontologien bereits vorliegen. Eine parallele Entwicklung derselben wird nicht betrachtet. Abschließend (4) wird eine Instanz der Dienstbeschreibung erstellt. Bei Klein und König-Ries wird diese durch die Erzeugung von Instanzen der Konzepte innerhalb der Ontologie realisiert. An ihrem Beispieldienst zum Drucken von Dokumenten sieht dies wie in Information 30 dargestellt aus.

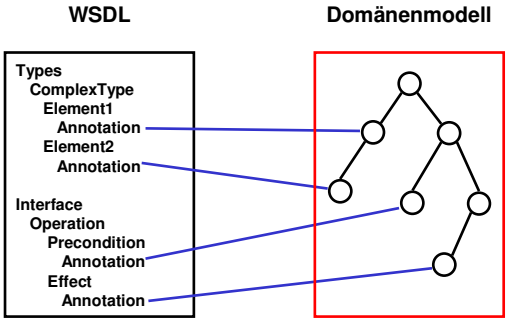


Information 30: Instantiierter Druckdienst nach [KK03]

Der beschriebene Dienst kann lokal verfügbare PDF-Dokumente in schwarzweiß mit hoher Qualität und 600dpi drucken, wobei die Ausgabe in Raum 335 stattfindet. Bei der Dienstspezifikation verzichten Klein und König-Ries vollständig auf die Angabe von Ein- und Ausgabeparametern. Dies wird dadurch begründet, dass ein Dienst wie der vorgestellte das zu druckende Dokument normalerweise nicht als Ein- oder Ausgabeparameter verwendet, sondern lediglich Kontrollinformationen wie den Ablageort des Dokuments oder die Uhrzeit, zu der der Druck abgeschlossen sein wird, übermittelt werden. Nach meiner Einschätzung würde durch diese unintuitive Vorgehensweise jedoch die halbmanuelle Dienstsuche erschwert, da für die Formulierung von Ein- und Ausgaben in Suchanfragen ein komplexes Gebilde von Zuständen und zugeordneten Objekten aus der Domäne erstellt werden müsste.

3.1.3 WSDL-S

Bei der Web Service Description Language – Semantics (WSDL-S) [AF+05a] [URL-17], die im Rahmen des METEOR-S-Projektes [URL-16] entwickelt wurde, handelt es sich um ein Vorgehen ähnlicher Zielrichtung aber anderer Vorgehensweise. Auch hier wird das Problem angegangen, dass der klassischen Dienstbeschreibung und Vermittlung mittels WSDL und UDDI die semantische Aussagekraft fehlt, um für eine automatisierte Dienstsuche ausreichend zu sein [SV+03]. Gleichzeitig wird aber großen Wert auf Rückwärtskompatibilität der Beschreibungsverfahren gelegt und daher kein neues Beschreibungsformat eingeführt. Bei WSDL-S werden die gängigen WSDL-Beschreibungen dahingehend erweitert, dass verschiedene darin enthaltene Konstrukte durch Referenzen auf Konzepte einer Ontologie mit semantischen Informationen verknüpft werden können.



Information 31: WSDL-S: Verknüpfung von WSDL-Elementen mit Semantik [AF+05a]

Auch bei WSDL-S werden die IOPEs mit Hilfe einer Ontologie genauer beschrieben. Für die Ein- und Ausgabeparameter können die zugehörigen Ontologiekonzepte aus den XML-Schema-Definitionen heraus direkt referenziert werden. Vor- und Nachbedingungen können durch Erweiterungselemente semantisch annotiert werden, die im *Operation*-Element von WSDL eingefügt werden, das seinerseits mit einem Konzept aus der Ontologie verbunden werden kann. Vorgesehen ist jeweils eine reine Verknüpfung der WSDL-Konstrukte mit Konzepten der Ontologie. Als Instantiierung des Dienstes kann dann das WSDL-S Dokument angesehen werden. Am Beispiel in gekürztem WSDL 1.1 betrachtet sieht dies so aus:

```
<definitions name="ldocPublishingService" targetNamespace="http://cm-tm.uka.de/services/ldocPublishingService"
             xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
             xmlns:wssem="http://www.ibm.com/xmlns/WebServices/WSSemantics"
             xmlns:cmOnt="http://cm-tm.uka.de/cmEducational.owl">
  <types>
    <xs:schema targetNamespace="http://cm-tm.uka.de/services/ldocPublishingService"
               xmlns="http://cm-tm.uka.de/services/ldocPublishingService">
      <xs:element name="ldocPublishingRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="ldoc" type="xs:base64binary" wssem:modelReference="cmOnt#LDoc" />
            <xs:element name="publishingArea" type="xs:string" wssem:modelReference="cmOnt#PublishingArea" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="ldocPublishingResponse" type="xs:string" />
    </xs:schema>
  </types>
  ...
  <portType name="ldocPublishingPortType">
    <operation name="publishLDoc"
          wssem:modelReference="cmOnt#LDocPublishingService">
      <input message="tns:ldocPublishingRequest" />
      <output message="tns:ldocPublishingResponse" />
      <wssem:precondition name="LDocAvailable" modelReference="cmOnt#Available"/>
      <wssem:effect name="LDocPublished" modelReference="cmOnt#Published"/>
    </operation>
  </portType>
</definitions>
```

Information 32: WSDL-S: Instantiierter Beispieldienst

Um eine Verbindung zwischen Inhalten der Ontologie und den WSDL-Elementen herzustellen, werden das Attribut *modelReference* und ein XML-Namensraum verwendet, der die Erweiterungen kennzeichnet. Die beiden Eingabeparameter *ldoc* und *publishingArea* sind in ihrer Schemadefinition mit den Klassen *LDoc* und *PublishingArea* aus der Ontologie verknüpft und in ein übergeordnetes Element zusammengefasst, das der weiter unten definierten Operation als Eingabe dient. Die Operation selbst ist als *LDocPublishingService* markiert und durch WSDL-S-Erweiterungselemente mit der Vorbedingung *Available* und der Nachbedingung *Published* versehen.

Die Instantiierung eines Webservice mit WSDL-S ist aufgrund der einfach zu handhabenden XML-Attribute übersichtlich und unkompliziert. Die semantische Dienstbeschreibung verteilt sich dann allerdings auf zwei Dokumente, das WSDL-Dokument und die referenzierte Ontologie. Bei ausgelagerten XML-Schema-Definitionen erhöht sich die Anzahl der Dokumente entsprechend weiter. Ein konzeptionelles Modell existiert bei WSDL-S nicht. Es gibt im Gegensatz zu OWL-S außer den zu annotierenden WSDL-Elementen keinen Anhaltspunkt und keine Vorschrift darüber, welche Konzepte in der Ontologie sinnvollerweise benötigt werden oder wie diese strukturiert werden können. Dadurch erhöht sich zwar die Freiheit bei der Erstellung der Dienstspezifikation, darunter leidet allerdings die Vergleichbarkeit von Diensten und Suchanfragen. Da es sich bei WSDL-S um eine WSDL-Erweiterung handelt, ist auch ersichtlich, dass WSDL-S zunächst auf die reine Beschreibung von Webservices beschränkt ist und sich nicht allgemein für beliebige IT-Dienste verwenden lässt, was bei OWL-S prinzipiell möglich wäre.

3.1.4 Diane Service Description

Laut Michael Klein sind die Dienstbeschreibungsverfahren von OWL-S und WSDL-S für eine vollautomatische Dienstsuche in dynamischen Umgebungen aufgrund einiger Nachteile nicht tauglich. In seiner Dissertation "Automatisierung dienstorientierten Rechnens durch semantische Dienstbeschreibung" [Kl06] formuliert er Anforderungen, die an eine semantische Dienstbeschreibung für diesen Einsatzzweck zu stellen sind. Er stellt die Diane Service Description (DSD) [Kl04] vor und zeigt auf, inwieweit sie den gestellten Anforderungen gerecht wird.

Ontologie mit speziellen Möglichkeiten

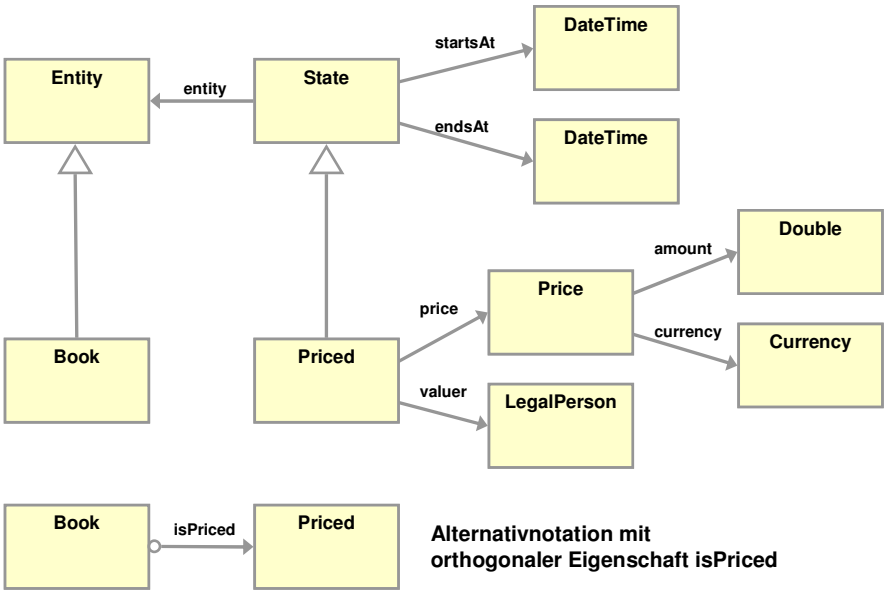
Die DSD baut nicht auf den bisherigen Vorschlägen der semantischen Dienstbeschreibung auf, sondern ist eine Neuentwicklung. Es wurde auch die eingesetzte Ontologiesprache samt graphischer Notation neu entworfen, um dabei bereits den Bedürfnissen der semantischen Dienstbeschreibung gerecht zu werden. Die DSDs sind damit laut Autor vermutlich nicht eins zu eins in andere Ontologiesprachen wie OWL übertragbar [Kl06:121]. Die DSD-Ontologie verwirklicht unter anderem die nun folgenden Konzepte. Zum Verständnis: Bei der DSD heißen die Relationen zwischen Konzepten "Eigenschaften"; eine Unterscheidung zwischen Eigenschaften und Relationen ist nicht gegeben.

Unterscheidung zwischen wertbestimmten und Entitätsklassen

In der DSD-Ontologie wird zwischen zwei Typen von Klassen unterschieden [Kl06:130; Kl04:28]. Bei wertbestimmten Klassen bestimmen sich die Instanzen aus den gültigen Werten ihrer Eigenschaften. Das bedeutet, dass zwei Instanzen bei gleichen Werten der Eigenschaften identisch sind und bei unterschiedlichen Eigenschaften stets unterschiedliche Instanzen beschrieben werden. Ein klassisches Beispiel für eine wertbestimmte Klasse ist der Preis. Er ist mit den Eigenschaften Währung und Wert eindeutig bestimmt und zu jeder gültigen Kombination aus den beiden gibt es genau eine namenlose Preisinstanz. Entitätsklassen beschreiben die Individuen der Miniwelt und ihre Instanzen erhalten global eindeutige Namen. Bei den Entitätsklassen führt nicht jede mögliche Wertekombination ihrer Eigenschaften zu einer sinnvollen Instanz. Änderungen an der Belegung ihrer Eigenschaften führen auch nicht zwangsläufig zu einer neuen Entität. Beispiele für Entitätsklassen sind Person und Universität.

Gesonderte Behandlung extrinsischer Eigenschaften

Die sogenannten extrinsischen Eigenschaften [Kl06:127] von Instanzen der Ontologie spiegeln deren für die Existenz der Instanz nicht zwangsläufig nötigen Beziehungen zu anderen Entitäten wider. Dabei handelt es sich auch oftmals um Eigenschaften, die sich im Laufe der Zeit ändern, wie beispielsweise der Preis, der Besitzer oder der Aufenthaltsort einer Entität. Intrinsische Eigenschaften sind im Gegensatz dazu konstituierend für die zugehörige Entität, wie beispielsweise der Titel eines Buches. Extrinsische Eigenschaften werden in der DSD-Ontologie über zwischengeschaltete Instanzen von Zustandsklassen (Klassen, die von einer gemeinsamen Oberklasse *State* erben), die sogenannten orthogonalen Eigenschaften modelliert. Information 33 zeigt dies am Beispiel der extrinsischen Eigenschaft *Priced* in der UML-ähnlichen, ausführlichen graphischen DSD-Notation oben und der Kurzschreibweise als "orthogonale Eigenschaft" unten:

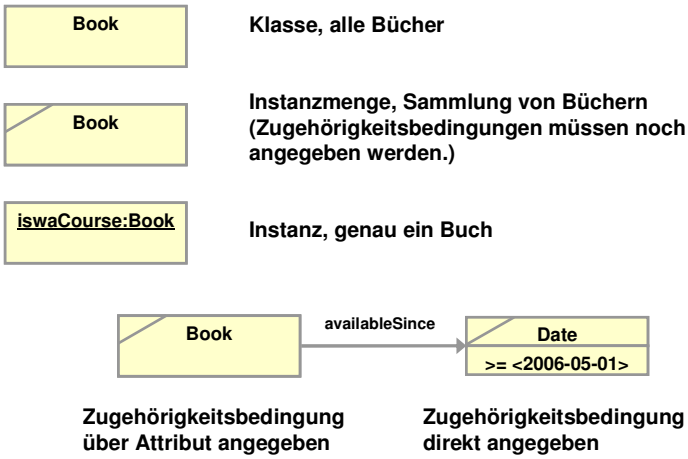


Information 33: DSD: Extrinsische Eigenschaft Priced

In diesem Beispiel kann ein *Book* dank seiner Oberklassenbeziehung zu *Entity* mit der Zustandsklasse *Priced* verknüpft werden, die ihrerseits Informationen zum eigentlichen Preis und Verantwortlichen verfügbar macht. Die zeitliche Komponente der Verknüpfung wird durch die Oberklasse *State* für alle Zustände einheitlich modelliert. Um diese Möglichkeiten auszuschöpfen, müssen die Klassen der DSD-Domänenontologie von Konzepten einer *Top-Level*-Ontologie erben, die unter anderem die hier dargestellte Klasse *Entity* enthält.

Spezifikation von Instanzmengen

Im Gegensatz zu anderen Ontologien können bei der DSD-Ontologie nicht nur ganze Klassen und einzelne Instanzen, sondern auch Mengen von Instanzen, die bestimmten Randbedingungen genügen, adressiert werden [Kl06:144] [Kl04:33]. Diese Möglichkeit wird bei der Spezifikation von Diensten häufig verwendet, um Ein-, Ausgaben oder Wirkung des Dienstes genauer einzugrenzen. In der graphischen DSD-Notation werden Instanzmengen durch einen schrägen Strich in der linken oberen Ecke gekennzeichnet (Information 34).



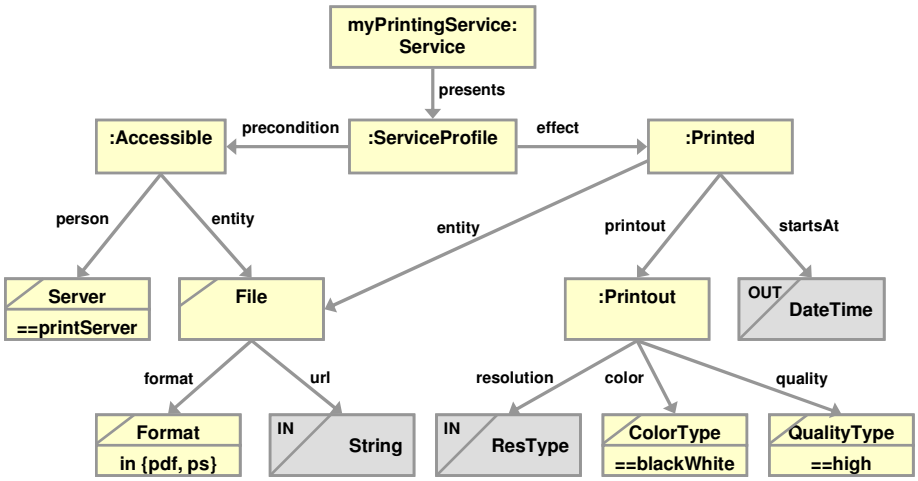
Information 34: DSD: Instanzmengen

Für Instanzmengen kann angegeben werden, welche Instanzen tatsächlich zur Menge gehören. Dies kann durch Vergleichsoperatoren direkt erfolgen, wie im Beispiel bei der *Date*-Menge rechts unten, die alle Daten ab dem ersten Mai 2006 beschreibt, oder auch indirekt über die

Eigenschaften, wie bei der *Book*-Menge links davon, die eben jene Bücher beschreibt, deren Veröffentlichungsdatum in der *Date*-Menge liegt.

Ein DSD-Beispiel

Der bereits aus Information 30 bekannte Druckdienst könnte in der graphischen Notation der DSD wie folgt aussehen:



Information 35: DSD-Beispiel

Auch die DSD verwendet ein *ServiceProfile* zur Beschreibung der Wirkung eines Dienstes. Vor- und Nachbedingung der Dienstausführung werden im Beispiel mit Hilfe der *State*-Unterklassen *Accessible* und *Printed*, die hier ebenso wie das *ServiceProfile* anonym instantiiert sind, ausgedrückt. In der Vorbedingung wird verlangt, dass nur *File*-Instanzen ausgedruckt werden können, die erstens über einen Uniform Resource Locator (URL) zugreifbar durch den Druckserver sind und zweitens im PDF- oder PS-Format vorliegen. Als Ergebnis des Dienstes ist angegeben, dass das *File* aus der Vorbedingung in den Zustand *Printed* überführt wird. Über die Verbindung zu einer anonymen *Printout*-Instanz wird weiter angegeben, dass ein Schwarzweißdruck mit hoher Qualität erfolgt. Die DSD-Ontologie bietet zusätzlich zu den oben genannten Erweiterungen auch die Möglichkeit Ein- und Ausgabeparameter des Dienstes als Variablen direkt anzugeben. Diese sind in der graphischen Notation grau unterlegt und haben über einem Schrägstrich links oben einen Vermerk, ob sie eingehende oder ausgehende (*IN* / *OUT*) Variablen sind. Es gibt zwei eingehende Variablen, der URL der zu druckenden Datei und die gewünschte Druckauflösung. Ausgabe des Dienstes ist der Fertigstellungszeitpunkt des Drucks. Wie aus Information 35 weiterhin ersichtlich ist, sind Eigenschaften und Notation der DSD-Ontologie an der semantischen Dienstbeschreibung ausgerichtet. Gezeigt werden in einer leicht verständlichen Schreibweise die Instanzen des *Service*, *ServiceProfile* und der Zustände, die wiederum durch Instanzmengen und Ein- und Ausgaben eingegrenzt werden.

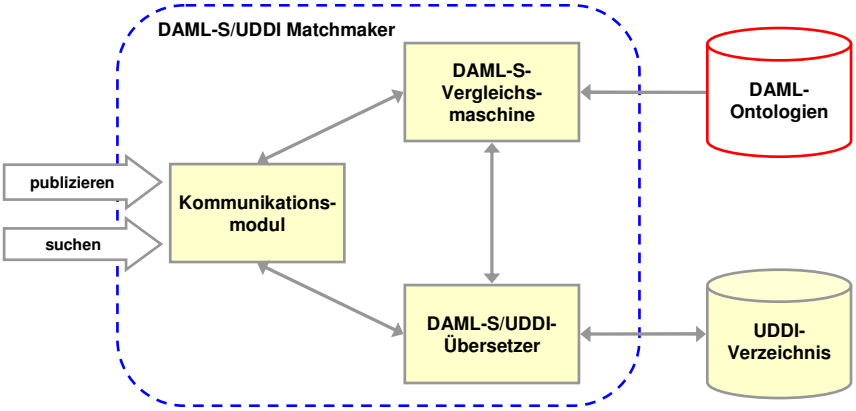
3.2 Semantische Dienstsuche

Jedes semantische Dienstbeschreibungsverfahren dient letzten Endes dem gemeinsamen Verständnis über die Schnittstelle und Funktion des beschriebenen Dienstes sowie der Suche nach Diensten. Daher wurden in der Regel parallel zu den einzelnen Vorstößen der semantischen Dienstbeschreibung jeweils auch die Möglichkeiten zur Dienstsuche betrachtet.

3.2.1 DAML-S/UDDI Matchmaker

Paolucci et al. beschreiben in [PK+02] ein Verfahren zur semantischen Dienstsuche, das auf einer Kombination von UDDI und DAML-S-Dienstbeschreibungen basiert. Dabei werden die Dienste zunächst wie vorgesehen in DAML-S beschrieben. Anschließend wird die

Dienstbeschreibung in einer UDDI-Datenstruktur repliziert und im UDDI-Verzeichnis hinterlegt. Dazu werden alle Daten des DAML-S *Profile* in entsprechende UDDI-Strukturen überführt. In allen Fällen, in denen kein entsprechendes Konstrukt existiert (zum Beispiel für die IOPEs), wird mittels eines *tModel*s ein entsprechender Datentyp in UDDI ergänzt. In einer DAML-S-Vergleichsmaschine wird die DAML-S-Beschreibung zusammen mit einer Referenz auf die UDDI-Repräsentation gespeichert, damit diese für eine spätere Suche zur Verfügung steht (Information 36).



Information 36: DAML-S/UDDI Matchmaker nach [PK+02]

Eine semantische Suche wird durch die Vergleichsmaschine auf ihrer Datenbasis durchgeführt. Alternativ kann wie bei konventionell veröffentlichten Webservices das UDDI-Verzeichnis direkt zur Suche verwendet werden. Dank des Datenreplikaton und der zusätzlichen *tModel*s kann auch im UDDI (mit einigen Einschränkungen) die semantische Information durchsucht werden. Dabei ist lediglich eine schlüsselwortbasierte Suche möglich. Konzeptuelle Zusammenhänge und Vererbungshierarchien können nicht berücksichtigt werden.

3.2.2 METEOR-S und Lumina

Im METEOR-S Projekt [PO+04] [URL-16], in dem WSDL-S entstanden ist, wurde auch ein Verfahren zur Dienstsuche basierend auf WSDL-S und UDDI entwickelt und durch das Eclipse-*Plug-In* Lumina [URL-18] implementiert [Li05]. Zur Veröffentlichung der Dienste werden die WSDL-Anteile der Beschreibung wie von OASIS empfohlen [CJ04] in das UDDI-Datenformat übertragen. Zusätzlich werden die im WSDL-S erweiterten Verknüpfungen mit Ontologiekonzepten mittels eigens dafür definierter *tModel*s in die UDDI-Daten im *bindingTemplate* oder *businessService* abgebildet [SV+03]. Für die in Lumina implementierte Dienstsuche bedeutet dies, dass nach Diensten anhand ihrer Kategorisierung und ihrer Ein- und Ausgabeparameter gesucht werden kann. Wie bei allen auf UDDI aufsetzenden Verfahren ist die Suche auch hier schlüsselwortbasiert und daher nicht in der Lage, Ergebnisse durch logische Schlussfolgerung anhand der Ontologie aufzufinden. Eine Erweiterung dahingehend, dass durch heuristische Vor- und Nachverarbeitung von UDDI-Anfragen die semantischen Suchmöglichkeiten ausgebaut werden, ist angedacht, aber soweit ersichtlich noch nicht umgesetzt [RM+05].

3.3 Ontologie

3.3.1 Entwicklung

Die Entwicklung von Ontologien besitzt eine gewisse Ähnlichkeit zur Entwicklung von Software. Ontologie-Entwicklungsprozesse lassen sich wie auch gängige Software-Entwicklungsprozesse in entsprechende, teilweise iterative Unterschritte aufgliedern wie beispielsweise von Uschold und King [UK95] beschrieben:

(1) Zweck bestimmen

(2) Ontologie erstellen
  (1) Ontologie erfassen
  (2) Ontologie programmieren
  (3) Existierende Ontologien integrieren

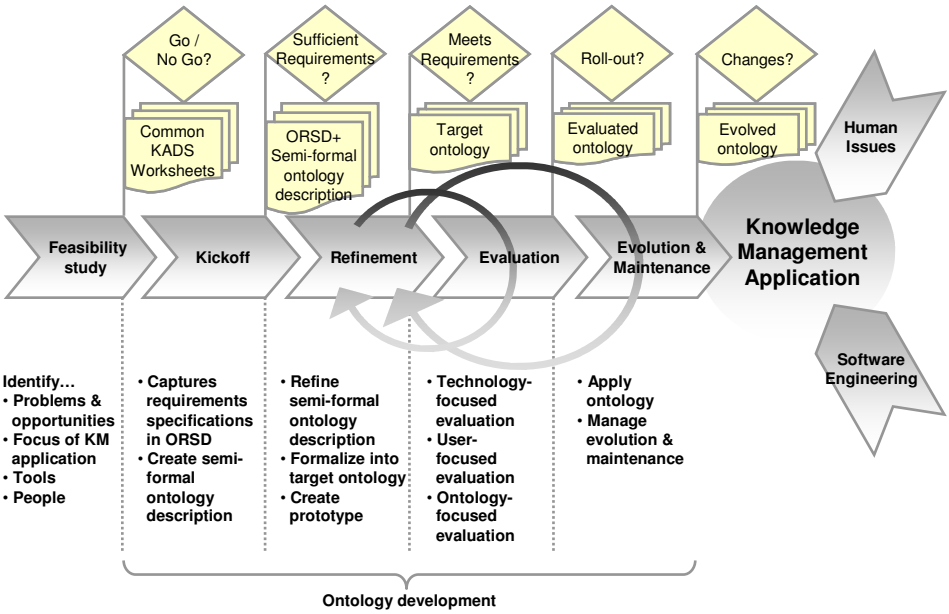
(3) Bewerten

(4) Dokumentieren

Information 37: Ontologieentwicklung nach Uschold und King [UK95]

Damit einher geht auch eine breite Vielfalt an tatsächlich angewandten Ontologie-Entwicklungsprozessen. Wie bei der Softwareentwicklung gibt es kein einheitliches Vorgehen, sondern viele unterschiedliche Vorschläge für einzusetzende Prozessschritte oder Techniken [UK95] [FG97] [Pr03] [SB+98]. Die Ontologieentwicklung wird sogar teilweise als Kunsthandwerk angesehen [St02:16] [FG02:130] und die meisten Ontologieautoren verwenden ein für das jeweilige Projekt angepasstes Entwicklungsverfahren. [FG02] bietet einen Vergleich einiger vorgeschlagener Entwicklungsmethoden anhand des IEEE Standard for Developing Software Life Cycle Processes (1074-1995) [IEEE-DSLCP-95] und kommt zu dem Ergebnis, dass alle Verfahren eine Existenzberechtigung besitzen und für ein Projekt das jeweils passende Verfahren oder eine Kombination verschiedener Verfahren einzusetzen ist. Die von den Autoren selbst entwickelte Methode Methontology sei aber im Vergleich zu den meisten anderen am besten ausgereift. Eine weitere Ähnlichkeit zur Softwareentwicklung wird auch bei der Verwendung von Mustern deutlich: In [CT+04] werden wiederkehrende in Ontologien verwendete Wissensmuster identifiziert ähnlich zu den in der Softwareentwicklung bereits gebräuchlichen Entwurfsmustern der Gang of Four [GH+96].

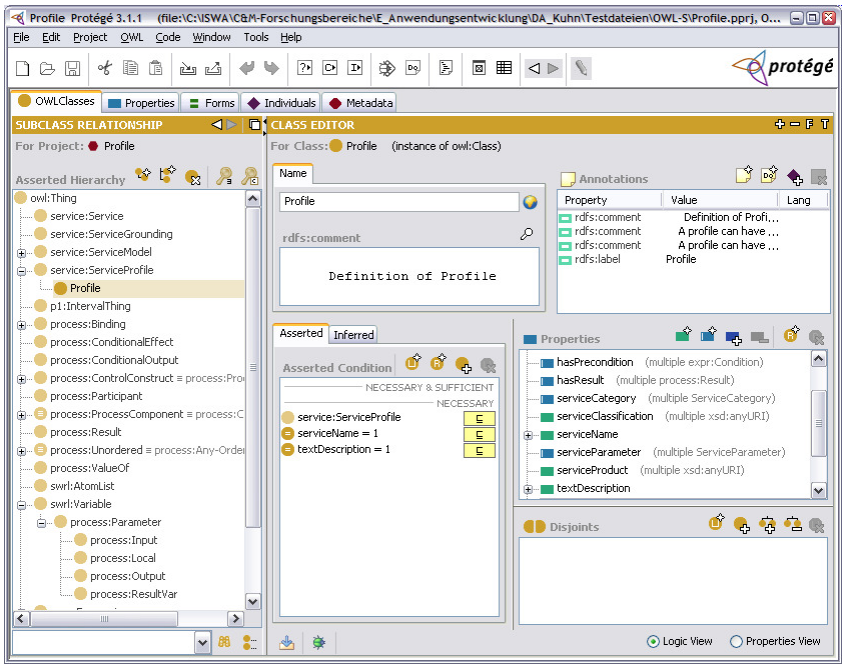
Ein gut ausgearbeitetes Verfahren zur Ontologieentwicklung bieten York Sure, Steffen Staab und Rudi Studer mit der On-To-Knowledge Methodology (OTKM) [SS+04a] [SS02] an, das während des On-To-Knowledge-Projektes [URL-19] im Programm der Information Society Technologies [URL-20] der europäischen Kommission entwickelt wurde. Darin wird die Ontologieentwicklung als Ergänzung zur Softwareentwicklung bei der Erstellung von *Knowledge Management Applications* (Anwendungen zum Wissensmanagement in Unternehmen) betrachtet. Der vorgestellte, teilweise iterative Prozess erweitert die von Uschold und King [UK95] eingeführte Methode. Er hat vier Phasen, die jeweils anhand ihrer Unterschritte, den zu stellenden Fragen und den zu erzielenden Ergebnissen erläutert werden. Einen Überblick der Ontologieentwicklung, die in [SS+04a] Knowledge Meta Process genannt wird, bietet Information 38.



Information 38: Der Knowledge Meta Process nach [SS+04a]

Die erste eigentliche Phase der Ontologieentwicklung heißt *Kickoff*. Darin werden in einem Ontology Requirements Specification Document (ORSD) ausreichend Anforderungen an die zu erstellende Ontologie erfasst, um mit der Entwicklung fortfahren zu können. Dazu zählen unter anderem die Wissensdomäne, Wissensquellen, Zweck und Anwendung der Ontologie. Anschließend wird eine erste semiformale Ontologiebeschreibung verfasst. Dies kann zum Beispiel in Form von *mind maps* geschehen. Der Focus liegt dabei auf dem terminologischen Anteil der Ontologie, etwa einer Taxonomie. In der Phase *Refinement* wird die Ergebnisontologie erstellt. Dazu wird unter Zuhilfenahme der Wissensquellen und anhand verschiedener Vorgehensmuster die halbformale Beschreibung verfeinert und schließlich formalisiert. Parallel dazu kann ein erster Prototyp der Anwendung erstellt werden, die die Ontologie nutzen wird. In einer weiteren Phase wird die erhaltene Ontologie anhand unterschiedlicher Kriterien und Betrachtungsweisen bewertet. Werden bei der Bewertung Mängel festgestellt, so wird ein weiterer Iterationszyklus mit *Refinement* angestrengt. Ansonsten kann die Ontologie im Softwaresystem zum produktiven Einsatz freigegeben und verwendet werden. Bei Änderungen in der Anwendungsdomäne oder der Anforderungen an die Anwendung kann es erforderlich werden, die Ontologie anzupassen. Dies ist in Information 38 durch den zweiten Iterationspfeil angedeutet. Das von Sure, Staab und Studer vorgestellte Verfahren des Knowledge Meta Process ist allgemein formuliert und bietet so die Möglichkeit, in angepasster Form bei der Erstellung der Domänenontologie für die Dienstbeschreibung Verwendung zu finden. Auf Grund des zeitlichen Rahmens der involvierten Diplomarbeiten und der Erfahrung der Beteiligten wird eine Beschneidung des Umfangs jedoch unumgänglich sein.

Zur Unterstützung der Ontologieentwicklung gibt es mittlerweile einige unterschiedliche Werkzeuge. Das Spektrum reicht vom schlichten Ontologieeditor bis hin zu speziell am zugehörigen Prozess ausgerichteten Entwicklungsumgebungen [Mi04]. Für die Beispielontologien und Ontologieausschnitte dieser Arbeit kommt der Ontologieeditor Protégé [GM+03] [URL-21] zum Einsatz. Er ist angenehm zu bedienen und bietet durch die Erweiterbarkeit mit *plug-ins*, die bereits zahlreich zur Verfügung stehen, Anpassbarkeit an individuelle Bedürfnisse. So werden zum Beispiel unterschiedliche Arten der Ontologierepräsentation (OWL, Baumansicht, graphisch) unterstützt.



Information 39: OWL-S-Profile in Protege 3.1.1

Information 39 zeigt einen *screen shot* des Protégé-Werkzeugs der Version 3.1.1. Die geöffnete Ontologie ist die OWL-S-Ober-Ontologie. Auf der linken Seite ist eine Baumansicht aller verfügbaren Klassen zu sehen, in der momentan die Klasse *Profile* ausgewählt ist. In der rechten Hälfte werden dann die Eigenschaften, Relationen und prädikatenlogischen Einschränkungen zur Auswahl angezeigt. Die selektierte Klasse besitzt zum Beispiel die Relation *hasPrecondition*.

3.3.2 Abfrage und Reasoning mit DIG

Wie bereits in Kapitel 2.5.3 erwähnt, können mit einem Ontologie-*Reasoner* zu einer gegebenen Ontologie deren Inhalte und logische Folgerungen daraus berechnet und abgefragt werden. Der erste Versuch eine einheitliche Schnittstelle zwischen Anwendungssystemen und solchen *reasoners* zu schaffen, findet sich im DIG Description Logic Interface [BM+03]. Dabei steht DIG für die Description Logic Implementation Group [URL-27], die sich um die Standardisierung dieser Schnittstelle bemüht. Die aktuell verfügbare Version DIG 1.1 [Be03] wird von einigen Ontologie-*Reasoner*n implementiert. Dazu zählen unter anderem RacerPro [URL-26] und Pellet [URL-29]. Im Gegensatz zur Version DIG 2.0 [URL-28], die sich gerade in Entwicklung befindet, hat die Version 1.1, die durch die *reasoner* angeboten wird, noch starke Einschränkungen bei der Formulierung von Anfragen (Genaueres hierzu in Kapitel 5.3.2 bei der Implementierung des Prototyps). DIG-Anfragen werden in einem dafür vorgesehenen XML-Dialekt verfasst und per HTTP an den Server gestellt. Die HTTP-Antwort enthält dann das Ergebnis der Anfrage.

4 DIENSTBESCHREIBUNGSKONZEPT

In diesem Kapitel wird untersucht, wie eine sinnvolle semantische Dienstbeschreibung aussieht, die eine gute Unterstützung bei der halbmanuellen Dienstsuche verspricht, so dass eine höhere Sucheffizienz als bei konventionellen Suchansätzen erzielt werden kann.

4.1 Anforderungen

Aus der verfügbaren Literatur und der in den Kapiteln 1.5, 2.2.5 und 6.7 vorgestellten Nutzung der halbmanuellen Dienstsuche beim Programmieren im Großen ergeben sich einige Anforderungen an die semantische Dienstbeschreibung. Durch die Einschränkung auf die halbmanuelle Dienstsuche fallen einige, in der der Literatur für automatische Dienstsuche formulierte, Anforderungen weg oder können abgeschwächt werden. Neu hinzukommen im Wesentlichen die Verwendung von eingängigen Bezeichnungen bei der Dienstbeschreibung und die Angabe der Ein- und Ausgabeparameter aus einer logischen Sichtweise heraus. Wünschenswert ist vor allem, dass man bei der halbmanuellen Dienstsuche mit möglichst wenigen Suchwiederholungen Erfolg hat, um den Gesamtaufwand der Suche zu minimieren.

4.1.1 Für die halbmanuelle Suche

Eingängige Bezeichnungen

Die Wahl der Bezeichnungen bei der Dienstbeschreibung und den von den Diensten bearbeiteten Geschäftsobjekten hat starken Einfluss auf die Effizienz der Dienstsuche. Es ist wichtig, dass dem Suchenden die Auswahl und Angabe der passenden Suchparameter einfach gemacht wird. Das heißt, es sollte leicht möglich sein, Suchparameter anzugeben, die zu einem erfolgreichen Suchergebnis führen. Zur Verdeutlichung ein Beispiel: Bei der Suche nach dem LDoc-Publizierungsdienst (Information 5, Kapitel 2.3.3) wurde als Eingabeparameter LDoc angegeben. Man kann jedoch nicht ohne weiteres davon ausgehen, dass der Dienst auch gefunden worden wäre, wenn stattdessen nach der Eingabe *Living Document* gesucht worden wäre. Doch woher soll der Suchende wissen, wie er die Bezeichnungen zu wählen hat? Es gibt Möglichkeiten, dieses Problem zu entschärfen. Als Mindestvoraussetzung sollten die verwendeten Bezeichnungen dienstübergreifend spezifiziert und konsistent verwendet werden, was beispielsweise durch ein gemeinsames Glossar, eine Taxonomie oder eine Ontologie geschehen kann. Dazu können die Bezeichnungen an einen allgemein akzeptierten Standard angelehnt werden, der das betreffende Gebiet abdeckt. Standardkonformität alleine hätte im oben geschilderten Fall des LDoc allerdings nicht zum Erfolg geführt, da das LDoc eine Eigenentwicklung der Forschungsgruppe C&M und daher in keinem branchenüblichen Standard vertreten ist.

Logische Ein- und Ausgabeparameter

Oftmals unterscheiden sich bei Diensten die vom Suchenden erwarteten und die tatsächlichen Ein- und Ausgabeparameter. Beim LDoc-Publizierungsdienst (Information 5, Kapitel 2.3.3) lässt sich aufgrund des BPMN-Diagramms und von Vermutungen annehmen, dass er ein LDoc als Eingabe annimmt, das er anschließend auf dem Server speichert. Tatsächlich benötigt er jedoch als Eingabe eine URL von der das LDoc bezogen werden kann. Solche Abweichungen in den Ein- und Ausgabeparametern des Dienstes und den Erwartungen kommen häufig vor (z.B. bei [KK03:149]) und können sich stark auf die Qualität der Dienstsuche auswirken. Der Suchende will (und kann) in der Regel nicht alle Eventualitäten für mögliche Feinheiten bei der Parameterwahl berücksichtigen, während er die Suchbegriffe formuliert, da er als Geschäftsanalytiker auf einem höheren Abstraktionsniveau arbeitet als die mit der Umsetzung der Prozesse betrauten Personen. Im schlechtesten Fall wird ein vorhandener und eigentlich passender Dienst nicht gefunden. Zur Optimierung der Dienstbeschreibung für die halbmanuelle Dienstsuche empfiehlt es sich daher, scheinbar logische, d.h. für den Suchenden zu erwartende, Dienstparameter mit anzugeben und dadurch ein Versagen der Suche zu verringern. Da die gängige Fachliteratur immer auf die automatisierte Verschaltung von Diensten fokussiert und

nur die dafür notwendigen tatsächlichen Dienstparameter betrachtet werden, treten solche Schwierigkeiten, die sich aus dem Abstraktionsniveau bei der Prozessmodellierung ergeben, dort nicht auf.

4.1.2 Aus der Literatur

Universelles konzeptionelles Modell

Die semantische Dienstbeschreibungssprache sollte ein konzeptionelles Modell beinhalten, durch das vorgegeben wird, wie die Semantik der Dienste erfasst und formalisiert werden soll. Das Modell definiert die grundlegenden Begriffe der Dienstbeschreibung, ist aber unabhängig von der tatsächlichen Anwendung. Es werden ausschließlich für die Dienstbeschreibung universell einsetzbare Konzepte durch das Modell spezifiziert [Kl06:40]. Das universelle konzeptionelle Modell ist nur ein grundlegender Teil der semantischen Dienstbeschreibung. Es wird durch weitere semantische Konzepte zu einem dienst- und domänenspezifischen Konzeptmodell erweitert.

Tatsächliche Ein- und Ausgabeparameter

Unter den tatsächlichen Ein- und Ausgabeparametern werden die Parameter und Datentypen verstanden, die beim Dienstaufruf Verwendung finden. Auch wenn zur Verbesserung der Dienstsuche die logischen Ein- und Ausgabeparameter spezifiziert werden, sollte man keinesfalls auf die Angabe der tatsächlichen Parameter verzichten, da diese für den eigentlichen Dienstaufruf benötigt werden. Die halbmanuelle Dienstsuche wird während des Programmierens im Großen verwendet und die tatsächlichen Übergabeparameter werden für die Feinauswahl, wenn mehrere ähnliche Dienste gefunden wurden, und für Verschaltung und Aufruf der Dienste benötigt. Für die semantische Dienstbeschreibung bei der halbmanuellen Suche kann der Abstraktionsgrad, mit dem die tatsächlichen Parameter beschrieben werden, jedoch im Vergleich zum Notwendigen bei der automatischen Suche angehoben werden. So könnten beispielsweise Einschränkungen der Menge zulässiger Übergabewerte innerhalb eines Datentyps in einer textuellen Beschreibung untergebracht werden anstatt in der Ontologie.

Eindeutigkeit

Für die automatische Dienstsuche wird hundertprozentige Eindeutigkeit für die Dienstbeschreibung angebotener und mindestens eine deterministische Beschreibung gesuchter Dienste gefordert sowie die Eindeutigkeit verwendeten Begriffe der Domäne [Kl06:40-42] [KK+05a:3]. Während letzterem auch bei der Dienstbeschreibung für die halbmanuelle Suche eine hohe Wichtigkeit obliegt (siehe oben unter “Eingängige Bezeichnungen”), kann die Dienstbeschreibung für angebotene Dienste weniger streng gehandhabt werden, da die Eindeutigkeit nicht vollständig durch explizite Semantik erreicht werden muss. Für die Suche soll die Eindeutigkeit der Anfrage nicht per se festgelegt sein, sondern durch den Suchenden oder das Suchverfahren in geeigneten Grenzen festgelegt werden können.

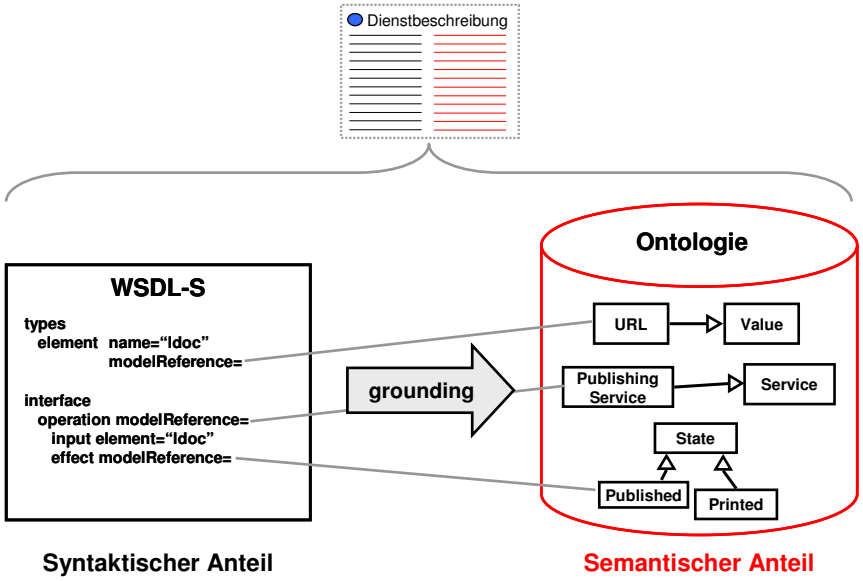
Grounding

Die Verankerung eines beschriebenen Dienstes mit einer konkreten Implementierung, die den Dienst erbringt wird, auch als *grounding* bezeichnet. Sie muss vorhanden sein, um den letztendlich gewählten Dienst nutzen zu können und ist daher notwendiger Bestandteil jeder Dienstbeschreibung [AB+02:354].

4.2 Grundkonzept

Zur semantischen Dienstbeschreibung im Rahmen dieser Arbeit soll einerseits der WSDL-S-Ansatz zur Anwendung kommen, da dieser im Wesentlichen dem entspricht, was gerade auf dem Weg ist, ein offizieller Standard zu werden. Andererseits stellt WSDL-S wie in Kapitel 3.1.3 gesehen lediglich Mittel zur Verfügung, um in WSDL die syntaktischen Einheiten einer Dienstinstanz zu beschreiben und diesen Elemente aus einer Ontologie zuzuordnen. WSDL-S

bietet also gerade die Möglichkeiten, die für ein *grounding* der in der Ontologie beschriebenen Dienste sorgen. Ein für die Dienstbeschreibung gefordertes semantisches Konzeptmodell fehlt jedoch und muss ergänzt werden, soll die Dienstspezifikation innerhalb der Ontologie einer allgemeinen Regelmäßigkeit gehorchen. Beim Einsatz von WSDL-S teilt sich die Dienstspezifikation folglich in zwei Teile auf: den syntaktischen Anteil, der wie bei klassischen Webservices auch in WSDL abgelegt wird, und den semantischen Anteil, der als Ontologie vorliegt, die über das WSDL-S-*grounding* aus dem WSDL-Dokument heraus referenziert wird (Information 40).



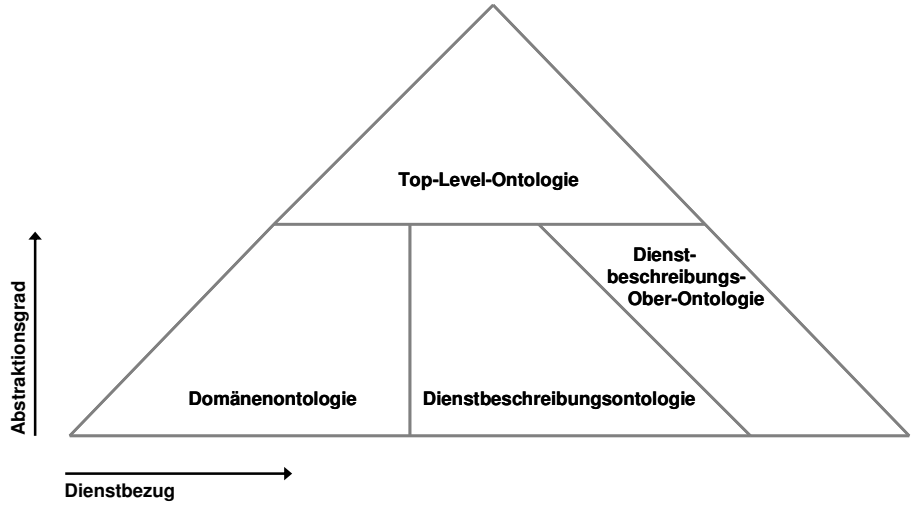
Information 40: Syntaktischer und semantischer Anteil der Dienstbeschreibung

Der Aufbau des syntaktischen Anteils ist anhand der WSDL-Spezifikation vorgegeben und wird hier, abgesehen von einigen Beispielen, nicht weiter ausgearbeitet. Für den semantischen Anteil gibt es jedoch keine expliziten Vorgaben. Es existiert also vor allem beim Entwurf der Ontologie ein großer gestalterischer Freiraum, wobei es lohnenswert ist, bei der Konzeption der Ontologiestrukturen einen Blick auf den OWL-S-Ansatz (Kapitel 3.1.1), seine Anwendung (Kapitel 3.1.2) und den neueren Ansatz der DSD (Kapitel 3.1.4) zu werfen. Dabei sollte berücksichtigt werden, dass ein dem gewünschten Ziel entsprechendes Verhältnis der Ausdrucksmächtigkeit der Dienstbeschreibung und ihrer Verarbeitbarkeit durch Menschen erreicht wird. [Kl06:12].

Der OWL-S-Ansatz verwendet die standardisierte Ontologiesprache OWL und bietet sehr viele Möglichkeiten, zielt jedoch speziell auf die automatisierte Dienstsuche ab und ist für den Einsatz bei der halbmanuellen Dienstsuche unnötig detailreich und undurchsichtig. Besonders die Erstellung der Vor- und Nachbedingungen für die Dienstausführung ist sehr komplex und unübersichtlich. Wünschenswert, auch aus dem Blickpunkt des WSDL-S-*grounding*, wäre eine einfache und menschenlesbare Methode zur Angabe dieser Bedingungen direkt als ontologische Konzepte anstatt in einer komplexen Regelsprache.

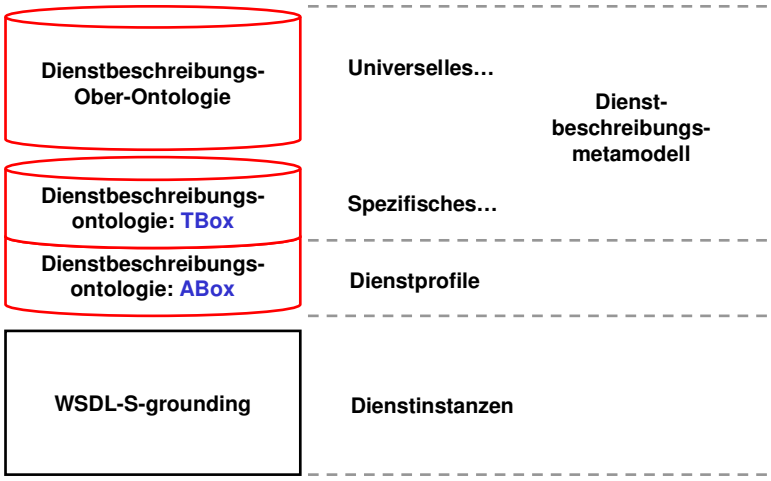
Eine solche findet sich im Ansatz der DSD, bei dessen Entwurf auch einige weitere Probleme des älteren OWL-S berücksichtigt und Defizite behoben wurden [Kl06:81]. Das Konzeptmodell ist klarer strukturiert und die fertige Dienstbeschreibung übersichtlicher für den Betrachter, so dass für einige der folgenden Konzepte die DSD zugrunde gelegt werden kann. Dennoch zielt auch die DSD auf die automatische Dienstsuche ab und kann an manchen Stellen vereinfacht werden. Die DSD-Ontologie wurde speziell für die Dienstbeschreibung entworfen, es liegt aber keine Abbildung in die OWL vor, die für die Ontologieerstellung zum Einsatz kommen soll. Die speziellen Eigenheiten der DSD-Ontologiesprache (z.B. Instanzmengen) sind in OWL leider nicht verfügbar.

Aus der DSD lässt sich eine Vierteilung der bei der Dienstbeschreibung beteiligten Ontologien motivieren (Information 41). An oberster Stelle steht dabei eine generische *Top-Level*-Ontologie, die unabhängig von den anderen Ontologieteilen und damit universell verwendbar ist. Am rechten Rand ist die Dienstbeschreibungs-Ober-Ontologie notiert, die ein für den Einsatzzweck der Dienstbeschreibung geeignetes Metamodell liefern sollte. Am linken Rand befindet sich die (oder eventuell mehrere) Domänenontologie(n), die die Geschäftsobjekte des betrachteten Bereichs beisteuert. Dazwischen ist die eigentliche Dienstbeschreibungsontologie zu entwickeln, die den Vorgaben der Ober-Ontologie entspricht und die Konzepte aus der Domänenontologie einbindet. Mit der Bezeichnung Dienstbeschreibungsontologie soll diese Dienstbeschreibungsontologie im engeren Sinne gemeint sein. Die Gesamtheit der vier Ontologieteile bei der Dienstbeschreibung kann auch als Dienstbeschreibungsontologie im weiteren Sinne bezeichnet werden.



Information 41: Die Dienstbeschreibungsontologie im weiteren Sinne

Wie bereits in Kapitel 3.1 (Stand der Technik) deutlich wurde, ist die semantische Spezifikation der IOPEs eines Webservices in WSDL-S und in der Ontologie möglich. Diese Tatsache lässt sich zur Erfüllung der Anforderungen der Spezifikation von sowohl logischen als auch tatsächlichen Ein- und Ausgabeparametern ausnutzen. Es liegt nahe, die logischen Parameter in der Dienstbeschreibungsontologie unterzubringen, wohingegen die Semantikvermerke in WSDL-S direkt mit dem Schemadatentypen der übertragenen Daten verknüpft sind und damit eher den technischen Charakter der tatsächlichen Ein- und Ausgabeparameter aufweisen.



Information 42: Abstraktionsebenen der semantischen Dienstbeschreibung

Entsprechend den Ausführungen der vergangenen Absätze ergibt sich eine Aufteilung der dienstbeschreibenden Elemente in drei unterschiedliche Abstraktionsebenen. Auf der obersten Ebene befindet sich das Dienstbeschreibungsmetamodell, das die konzeptionellen Grundlagen zur Definition von Dienstprofilen vorhält. Dieses kann nochmals in das universelle konzeptionelle Modell der Dienstbeschreibungs-Ober-Ontologie und die TBox der Dienstbeschreibungsontologie, die bereits Spezifika der zu beschreibenden Dienste und deren Domäne enthält, aufgegliedert werden. Darunter befinden sich die Dienstprofile. Diese werden als Ontologieinstanzen in der ABox der Dienstbeschreibungsontologie realisiert. Auf der untersten Ebene befinden sich die tatsächlichen Dienstinstanzen. Diese werden durch WSDL-S-Dokumente spezifiziert und mit den Dienstprofilen aus der Ontologie verknüpft. Die einzelnen Abstraktionsebenen sind so gewählt, dass dazwischen jeweils Wiederverwendung stattfinden kann: Verschiedene Dienstinstanzen können ein und dasselbe Dienstprofil realisieren und verschiedene Dienstprofile können auf dieselben Konzepte aus dem Dienstbeschreibungsmetamodell zurückgreifen.

In den folgenden Unterkapiteln wird dieses Beschreibungskonzept weiter ausgearbeitet und die einzelnen Ontologieteile sowie die Verknüpfung der Dienstbeschreibungsontologie mit den WSDL-Konstrukten erläutert.

4.3 Die Top-Level-Ontologie

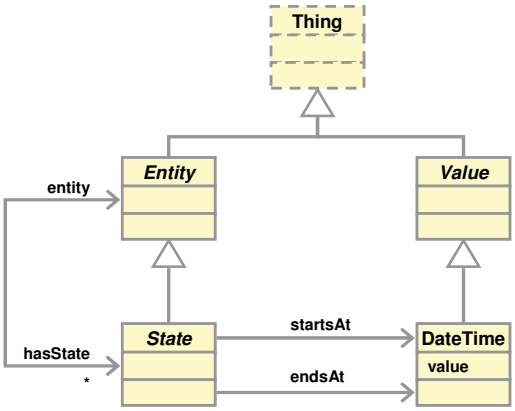
Als *Top-Level*-Ontologie wird ein Ontologieteil bezeichnet, der im Vererbungsbaum der Ontologie zwischen die Urklasse (*Thing* bei OWL und in den folgenden Folien) und die tatsächlich verwendeten Klassen, also ganz oben im Vererbungsbaum, eingefügt wird. Dies wird im Ontologie-Überblicksbild durch die Platzierung in der oberen Spitze des Dreiecks symbolisiert.

Information 43: Die Top-Level-Ontologie

In der *Top-Level*-Ontologie werden fundamentale Konzepte festgehalten, die allgemeine Verwendung finden können und nicht speziell an der Dienstbeschreibung ausgerichtet sind. Sie dient gleichermaßen als Erbgrundlage für Klassen der drei weiteren Ontologieteile wie auch als nicht erzwingbare Entwurfsrichtlinie für die abgeleiteten Klassen. Beispiele für *Top-Level*-Ontologien gibt es beinahe so viele wie Ontologien selbst, einige werden in [FH97] vorgestellt.

4.3.1 Klassen

Der Inhalt der hier verwendeten *Top-Level*-Ontologie ist aus den Ontologien *top* und *category* der DSD motiviert [Kl06:161,165-166]. Sie ermöglicht es bestimmte Klassen durch Vererbung als Entitäten (*Entity*) auszuweisen, die mit Zuständen (*State*) versehen werden können. Der Inhalt der *Top-Level*-Ontologie ist in Information 44 dargestellt. Klassen sind bei den folgenden Informationsgrafiken immer dann gestrichelte gezeichnet, wenn sie in einem anderen Teil der Ontologie definiert als dem momentan dargestellten.



Information 44: Klassen der Top-Level-Ontologie

Value

Im Gegensatz zu den unten erläuterten Entitäten stehen sogenannte wertbestimmte Klassen wie die Angabe eines Zeitpunktes oder ein Preis, die durch die Werte ihrer Eigenschaften vollständig definiert sind [Kl06:130]. Wertbestimmte Klassen werden als Unterklassen von *Value* modelliert.

DateTime

Die Klasse *DateTime* ist eine solche wertbestimmte Klasse. Sie repräsentiert einen über die Eigenschaft *value* definierten Zeitpunkt.

Entity

Eine Entität (*Entity*) entspricht einem Konzept aus der Miniwelt mit einer komplexen Identität wie beispielsweise eine Person, eine Universität oder eine Information. Dies drückt sich dadurch aus, dass ihren Instanzen mittels der *hasState*-Relation eine beliebige Menge an Zuständen zugeordnet werden können, die die Instanzen genauer charakterisieren. Die Klasse ist abstrakt modelliert, da sie erst durch Unterklassenbildung eine aussagekräftige Semantik erhält. [Kl06:166] bietet eine mögliche Diversifizierung in weitere Unterklassen an, die bei der weiteren Ontologieentwicklung hilfreich sein kann. Diese ist allerdings weder für die Dienstbeschreibung noch für Domänenontologien zwangsläufig nötig und wird im Zuge der Übersichtlichkeit und des größeren Gestaltungsfreiraums in der Domänenontologie für diesen Entwurf ausgespart.

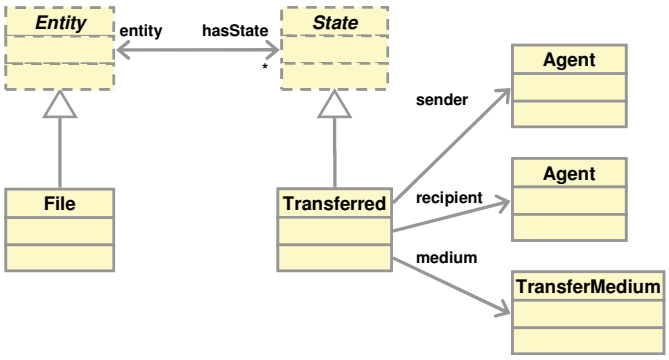
State

Die abstrakte Klasse *State* beziehungsweise ihre Unterklassen werden verwendet um mögliche Zustände von Entitäten zu modellieren. Ein Zustand bezieht sich dabei immer mindestens auf eine Entität (*entity*-Relation). Zusätzlich kann er durch die beiden Relationen *startsAt* und *endsAt* mit Informationen darüber versehen werden, in welchem Zeitraum er gültig sein soll. Auch für die Zustandsklassen bietet [Kl06:166] eine weitergehende Diversifizierung einhergehend mit den unterschiedlichen Unterklassen von *Entity* an, die bei der Entwicklung der Dienstbeschreibungsontologie in Betracht gezogen werden kann, aber nicht muss. Zu beachten ist weiterhin, dass ein Zustand selbst wieder eine Entität ist und damit selbst durch weitere Zustände modifiziert werden kann.

4.3.2 Entwurfsrichtlinie

Die Klassen *Entity* und *State* dienen zur Modellierung der extrinsischen Eigenschaften von Konzepten der Miniwelt wie in Kapitel 3.1.4 beschrieben. Da in OWL die Angabe von orthogonalen Eigenschaften nicht wie in der DSD direkt möglich ist, wurde die Verknüpfung von der *Entity* zur zugehörigen *State*-Instanz durch eine gewöhnliche Relation modelliert, die

bei Bedarf auch beerbt werden kann. Information 45 zeigt in einem Ausschnitt einer Dienstbeschreibungsontologie die Anwendung von Zuständen am Beispiel einer Datei (*File*), die über ein beliebiges Medium von einem Sender zu einem Empfänger (*recipient*) übertragen wird. Dazu wird eine Klasse *Transferred* eingeführt, die über ihre Oberklassenbeziehung mit *File* in Verbindung gebracht werden kann und noch weitere Informationen zur Übertragung als Relationen besitzt.

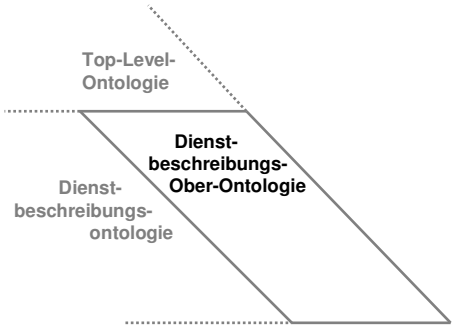


Information 45: Extrinsische Eigenschaft Transferred

Von dieser Modellierungstechnik soll in den Dienstbeschreibungs- und Domänenontologien Gebrauch gemacht werden. Sie kann später verwendet werden, um die Spezifikation von Vor- und Nachbedingungen in der Dienstbeschreibung zu ermöglichen.

4.4 Die Dienstbeschreibungs-Ober-Ontologie

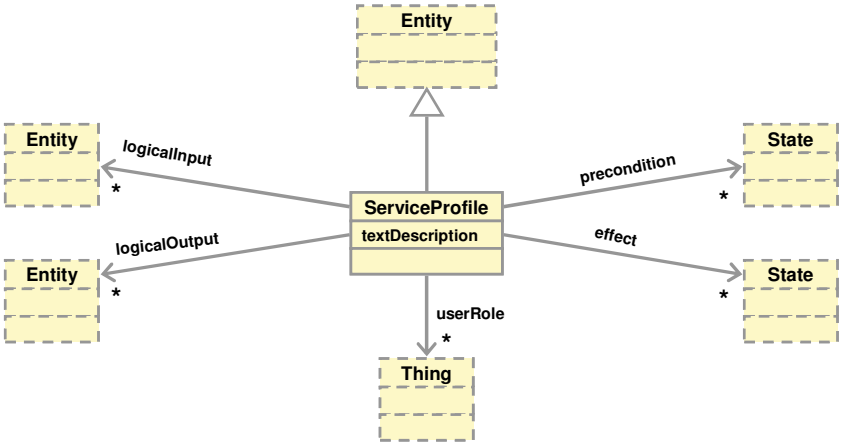
Die Dienstbeschreibungs-Ober-Ontologie gibt an, welche Klassen, Eigenschaften und Relationen für die Dienstbeschreibung verwendet werden sollen und stellt zusammen mit dem TBox-Anteil der Dienstbeschreibungsontologie ein Daten-Metamodell für die dienstbeschreibenden Instanzen innerhalb der Dienstbeschreibungsontologie dar.



Information 46: Die Dienstbeschreibungs-Ober-Ontologie

Für die Ausgestaltung der Ober-Ontologie können wie auch beim Entwurf der DSD nur wenige Anleihen bei OWL-S gemacht werden, da der Schlichtheit ein hoher Stellenwert zugemessen wird, um eine einfache und übersichtliche Struktur zu erhalten. Dies hat zwei Vorteile: zum einen ist die Beschreibung von Diensten durch Unterklassenbildung und Instantiierung wesentlich einfacher, zum anderen fällt dadurch auch der Aufwand bei der Suche geringer aus, beispielsweise bei der Formulierung der Suchanfrage. Die hier Vorgestellte Ober-Ontologie enthält inklusive aller *Imports* 5 Klassen und 10 Relationen und Eigenschaften. Dies ist ca. 10% der Anzahl von OWL-S und damit eine signifikante Reduzierung der Komplexität. Da mit der Dienstbeschreibung hier der Zweck verfolgt wird, die Dienste anhand ihrer Funktionalität durch eine Suche auffindbar zu machen, liegt der Fokus auf Informationen, die bei OWL-S dem *ServieProfile* zugeordnet sind. Durch die in Information 47 vorgestellte Ober-Ontologie wurde

wie auch bei OWL-S die Möglichkeit gegeben mittels einer *ServiceProfile*-Klasse IOPEs von Diensten zu spezifizieren. Konsistent mit OWL-S, WSDL-S und DSD beschreibt eine *ServiceProfile* Instanz später nur die IOPEs einer einzelnen Operation eines Webservices. Die zusätzlichen Indirektionsstufen wurden jedoch entfernt. Des Weiteren ist an den Namen der Relationen (*logicalInput*, *logicalOutput*) zu erkennen, dass hier die logischen Ein- und Ausgaben aus Sicht der Prozessmodellierung spezifiziert werden sollen, wodurch die Anforderung "Logische Ein- und Ausgabeparameter" adressiert wird. Die Wertebereiche der Relationen wurden sehr generell gehalten. Für Ein- und Ausgaben sind *Entity*-Unterklassen und für Vor- und Nachbedingungen *State*-Unterklassen vorgesehen.



Information 47: Klassen der Dienstbeschreibungs-Ober-Ontologie

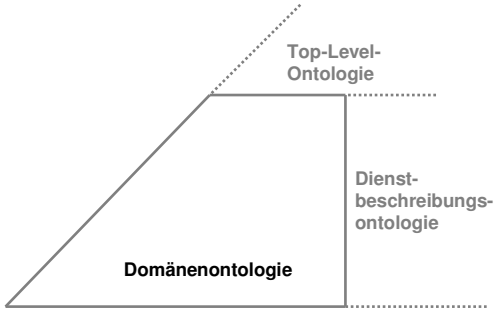
Über die zusätzliche Relation *userRole* lässt sich bei jedem Dienst angeben, für welche Benutzerrollen er gedacht ist. Damit ist eine zusätzliche Klassifizierung der Dienste nach ihrer Zielgruppe möglich. So kann zum Beispiel eine Suche nach Diensten für Autoren, Vortragende oder Übungsleiter eingeschränkt werden. Auch eine textuelle Beschreibung zur Dienstfunktionalität ist vorgesehen. Sie wurde als Eigenschaft der *ServiceProfile*-Klasse modelliert. Ein Dienst kann außer den funktionalen auch nichtfunktionale Eigenschaften besitzen. Eine erste wurde durch die Relation *userRole* modelliert. Des Weiteren können sich mehrere Dienstprofile die gleichen oder ähnliche funktionalen Eigenschaften teilen. Diesem Umstand wird durch die Möglichkeit Rechnung getragen, dass das *ServiceProfile* durch Unterklassenbildung in der Dienstbeschreibungsontologie mit strikteren Einschränkungen auf die Relationen versehen werden kann. Durch eine solche Diversifizierung von *ServiceProfile* können die Dienstprofilinstanzen in Kategorien eingeteilt werden, nach denen später gezielt gesucht werden kann.

Unberücksichtigt im hier vorgestellten Entwurf bleibt die Beschreibung weiterer nichtfunktionaler Diensteigenschaften. Diese könnten in einer Weiterentwicklung der *ServiceProfile*-Klasse ergänzt werden. Vorläufig kann zur Annotierung dieser Merkmale auch die *textDescription*-Eigenschaft verwendet werden.

Die *Top-Level*-Ontologie und die Dienstbeschreibungs-Ober-Ontologie geben die Richtlinien für die Beschreibungsanteile in der Ontologie, sind aber völlig domänenunspezifisch und bilden somit das geforderte universelle konzeptionelle Modell der semantischen Dienstbeschreibung.

4.5 Die Domänenontologie

Obwohl hier immer von **der** Domänenontologie die Rede ist, sei angemerkt dass es sich dabei keinesfalls um eine einzelne Ontologie handeln muss. Es ist durchaus möglich, dass mehrere sich ergänzende oder aufeinander aufbauende Domänenontologien zum Einsatz kommen.



Information 48: Die Domänenontologie

Die Domänenontologie dient dazu, das Wissen der Domäne für die Dienstbeschreibung geeignet aufzubereiten. Wichtig sind dafür im Wesentlichen die Klassen, die für die Spezifikation der Ein- und Ausgabeparameter erforderlich sind.

4.5.1 Anforderungen

Hier soll betrachtet werden, welche Anforderungen aus Sicht der semantischen Dienstbeschreibung an die Domänenontologie zu stellen sind.

Breite Domänenabdeckung bezüglich der Geschäftsobjekte

Wichtig aus Sicht der semantischen Dienstbeschreibung ist, dass die Domänenontologie alle Klassendefinitionen der verwendeten Geschäftsobjekte enthält. Das bedeutet für die Ontologie, dass sie eine ausreichende Abdeckung der Domäne in der Breite erreichen sollte. Hier sind vor allem die Anzahl und das Abstraktionsniveau der verfügbaren Konzepte von Interesse. Die Ontologie muss nicht "dick" sein. Das heißt, es ist wünschenswert eine Taxonomie möglichst aller Begriffe des Geschäftsbereichs zu haben. Über die Hierarchiebildung der Taxonomie hinausgehende, komplexe logische Verknüpfungen in der Ontologie schaden der Dienstbeschreibung an sich nicht [Sa06:28], sind jedoch unter Umständen der Übersichtlichkeit abträglich.

Allgemein anerkannte Bezeichnungen

Diese Anforderung lässt sich aus der Anforderung "Eingängige Bezeichnungen" an die semantische Dienstbeschreibung ableiten. Es sollte sichergestellt sein, dass durch die in der Domänenontologie aufgeführten Begriffe eine Homogenisierung des Sprachgebrauchs der Personen, die an den Geschäftsprozessen in der Domäne beteiligt sind, erzielt wird. Diese Homogenisierung lässt sich dann auch auf die Prozessmodellierung und Dienstspezifikation ausweiten. Beteiligte, Analytiker und Ingenieure, die an einer SOA-basierten Rechnerunterstützung des Geschäftsbereichs arbeiten, müssen eine Übereinkunft bezüglich der zu verwenden Bezeichnungen erwirken, die in diesem Fall durch die Ontologie bzw. deren Inhalt manifestiert werden soll. Eine Vereinheitlichung der Begriffswelt kann durch die Ausrichtung der Domänenontologie an gängigen Standards versucht werden, ist jedoch im Bereich der Aus- und Weiterbildung kritisch, da die meisten Standards nur Teile der für die Dienstbeschreibung benötigten Begriffe abdecken und gegenseitig nicht widerspruchsfrei sind. Auch im universitären Bereich weichen Auffassungen und Geschäftsprozesse von Fakultät zu Fakultät und auch bereits auf noch niedrigerer Ebene voneinander ab, so dass eine Homogenisierung hier zwar lohnenswert, aber schwer durchzusetzen ist.

Erweiterbarkeit

Ein Grund für die Einführung serviceorientierter Softwareunterstützung ist die Möglichkeit der flexiblen Anpassbarkeit an neue Bedürfnisse. Findet ein Wandel im Geschäftsbereich statt, müssen mitunter Prozessmodelle und Dienste samt Spezifikation angepasst oder neu hinzugefügt werden. Dabei muss auch die Domänenontologie entsprechend um neue Konzepte angereichert werden. Um die Weiterentwicklung der Geschäftsanwendung nicht unnötig aufzublähen, sollte der Inhalt der Domänenontologie und dessen Struktur so aufbereitet sein, dass Erweiterungen einfach und unkompliziert durchzuführen sind und keine Nebenwirkungen auf bereits bestehende Dienstspezifikationen auftreten. Es ist beispielsweise möglich, dass der Wartungsaufwand der Ontologie wie auch bei der objektorientierten Softwareentwicklung mit steigender Komplexität in der Vererbungshierarchie und den Assoziationen ebenfalls zunimmt [PU+03].

Anwendung der Top-Level-Ontologie

Für die spätere Verwendung bei der Dienstbeschreibung sollte die Domänenontologie am Oberklassenschema und der Entwurfsvorgabe der *Top-Level*-Ontologie ausgerichtet sein.

4.5.2 Entwicklung

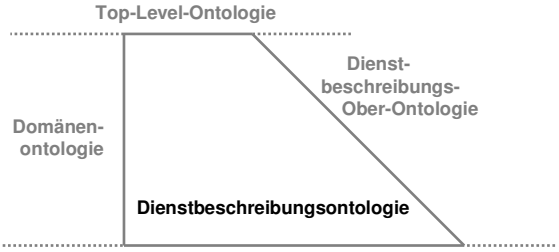
Die Entwicklung der Domänenontologie kann prinzipiell nach einem beliebigen Verfahren durchgeführt werden, das im Ergebnis die oben formulierten Anforderungen aus Sicht der Dienstbeschreibung und -suche berücksichtigt. Es kann also ein etabliertes Entwicklungsverfahren wie die On-To-Knowledge Methodology bzw. der darin enthaltene Knowledge Meta Process [SS+04a] Verwendung finden, in den die Anforderungen der Ontologie als Randbedingungen einfließen. Im Falle des Knowledge Meta Process würden diese in der *Kickoff*-Phase erfasst und im Dokument für die Ontologie-Anforderungsspezifikation (Ontology Requirements Spezification Document, ORSD) verarbeitet werden [SS02:38].

Optimal wäre, wenn die Domänenontologie bereits zu Beginn der Geschäftsprozessmodellierung zur Verfügung stünde, was aus zeitlichen Gründen nicht immer der Fall sein kann. Sollte die Entwicklung der Domänenontologie und die Prozessmodellierung und semantische Dienstspezifikation parallel ablaufen, so ergeben sich einige weitere Randbedingungen an das Vorgehen auf beiden Seiten, die zusätzlichen Abstimmungs- und Kommunikationsaufwand erzwingen. Einerseits sollen in die Domänenontologie genau die Inhalte eingefügt werden, die zur semantischen Dienstbeschreibung benötigt werden, andererseits werden aber auch die Konzepte und die Begriffshomogenisierung, die durch die Ontologie vorangetrieben wird, Einfluss auf die Nomenklatur in den Prozessmodellen und Dienstbeschreibungen haben. Eine Abstimmung zwischen den Verantwortlichen ist daher unumgänglich und beim Knowledge Meta Process als iterative Wiederholung der Schritte *Refinement* und *Evaluation* ohnehin vorgesehen [SS02:56] (Information 38). Um zeitliche Fehlinvestitionen zu vermeiden, kann die Abstimmung aber auch schon zu Beginn zwischen *Kickoff* und *Refinement* stattfinden (Kapitel 3.3).

Die Anforderung Erweiterbarkeit bezieht sich zwar primär auf den Inhalt und Struktur der Ontologie, spielt aber auch mit in die Ontologieentwicklung hinein. Der eingesetzte Entwicklungsprozess sollte, für den Fall, dass die Ontologie ergänzt oder gar umstrukturiert werden muss, ein Änderungsmanagement vorsehen. Auch dies ist beim Knowledge Meta Process vorgesehen, und kann durch eigene Anpassungen ausgestaltet werden. In Information 38 ist dies durch die iterative Wiederholung der Schritte *Refinement*, *Evaluation* und *Evolution & Maintenance* skizziert [SS02:66].

4.6 Die Dienstbeschreibungsontologie

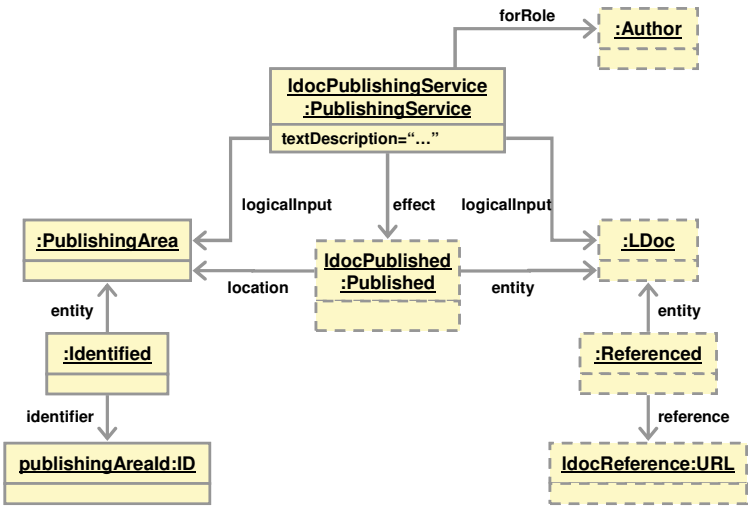
Die Dienstbeschreibungsontologie schließt die Lücke zwischen der Dienstbeschreibungs-Ober-Ontologie und der Domänenontologie und macht sich dabei die Vorgaben der *Top-Level*-Ontologie zu Nutze.



Information 49: Die Dienstbeschreibungsontologie

Da durch die Dienstbeschreibungsontologie letztendlich die existierenden Dienste spezifiziert werden, ist für diese die Anforderung der Eindeutigkeit besonders ausschlaggebend. Die Dienstfunktionalität sollte wenn möglich immer eindeutig beschrieben sein. Es wird jedoch in der Regel ein Kompromiss angestrebt werden müssen zwischen dem erzielten Grad der Eindeutigkeit, der für die Spezifikation investierten Zeit und der Aufteilung der Spezifikation auf die explizite Semantik in der Ontologie und ihre textuelle Ergänzung.

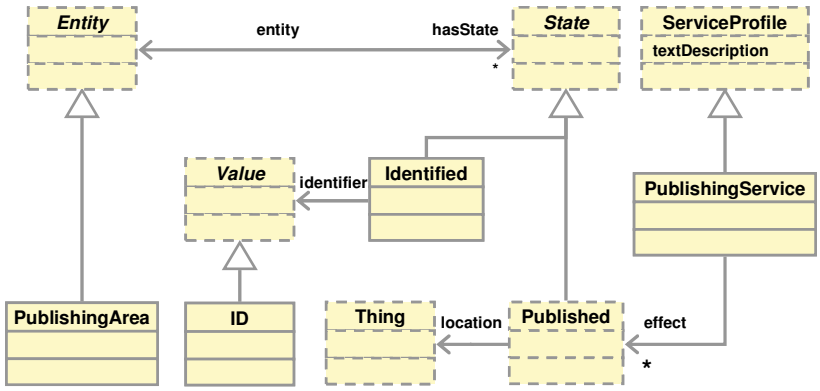
Die Dienstbeschreibungsontologie kann nochmals in zwei Teile untergliedert werden. Der TBox-Anteil enthält die benötigten Zustandsklassen und der ABox-Anteil die instantiierten Dienstbeschreibungen. Während die Klassen der logischen Ein- und Ausgabeparameter der Dienstbeschreibung in aller Regel direkt aus der Domänenontologie referenziert werden können, müssen die fehlenden Zustandsklassen erst angelegt und gegebenenfalls um weitere benötigte Klassen ergänzt werden, etwa solche Klassen, die technische Details der Dienste betreffen, die in der Domänenontologie fehl am Platze wären. Information 50 zeigt eine Instantiierung des Beispieldienstes zur Publizierung eines LDocs (Kapitel 2.3.3).



Information 50: Instantiierter LDoc-Publizierungsdienst

Es existieren jeweils Instanzen der einzelnen Klasen in der Ontologie, die entsprechend der Relationen der Klassen miteinander verknüpft sind. Die beiden Vorbedingungen, dass sowohl LDoc als auch *PublishingArea* vorhanden sein müssen, wurden nicht explizit modelliert. Durch die Verwendung von LDoc und *PublishingArea* als Eingabeparameter ist dies bereits

ausgedrückt. Die Klassen der gestrichelten Objekte entstammen der Ober- oder der Domänenontologie. Bevor die hier abgebildete Instantiierung vollzogen werden kann, müssen jedoch die von *State* erbende Klasse *Identified* sowie die Klassen *PublishingArea* und *ID* in der Dienstbeschreibungsontologie angelegt werden, da diese nicht in der Domänenontologie vorhanden sind (Information 51). Zur Kategorisierung des Dienstprofils wird darüber hinaus *PublishingService* als Unterklasse von *ServiceProfile* erstellt und ihre *effect*-Relation auf die *State*-Unterklasse *Published* eingeschränkt.



Information 51: Erforderliche Klassen für den LDoc-Publizierungsdienst

Bemerkenswert an der Instantiierung (aus Information 50) ist, dass die Service-Instanz nicht nur die logischen Eingabeparameter *PublishingArea* und LDoc auszeichnet. Die tatsächlichen Eingabeparameter ID und URL sind ebenso vorhanden und können aus dem WSDL-S-Dokument referenziert werden. Damit werden die Anforderungen der semantischen Spezifikation dieser beiden Sichtweisen auf Parametern Rechnung getragen. Die syntaktische Spezifikation der tatsächlichen Parameter erfolgt später durch das *grounding*. Auch die wichtigsten Vor- und Nachbedingungen sind spezifiziert und eine weitere Ausdehnung der semantischen Beschreibung des Dienstes damit aus Sicht der halbmanuellen Dienstsuche nicht erforderlich. Was die Anforderung der Eindeutigkeit betrifft ist festzuhalten, dass die vorgestellte Beschreibungsmethode die wichtigsten funktionalen Aspekte des beschriebenen Dienstes formalsemantisch abbildet, so dass ein zielgerichtetes Auffinden der beschriebenen Dienste ermöglicht wird (Kapitel 6.7.2). Dies wird erreicht, ohne ein unübersichtliches Gesamtbild des beschriebenen Dienstes zu liefern. Es gibt aber durchaus ergänzende Informationen, die im Zuge dessen nicht in der Ontologie wiedergegeben wurden und daher in die Beschreibung in Textform einfließen sollten. Das sind in diesem Szenario zum Beispiel die weiteren Nachbedingungen, dass das publizierte LDoc nun von Kursteilnehmern gefunden und heruntergeladen werden kann. Denkbar wären aber auch weitere Einschränkungen der Parameter, zum Beispiel wenn für die vom Server zu publizierenden LDocs eine maximale Größe eingehalten werden muss.

4.7 Grounding mit WSDL-S

Bei der tatsächlichen Implementierung des bereits innerhalb der Ontologie beschriebenen Dienstes wird ein WSDL-S-*grounding* angelegt, das eine Verknüpfung zu den dienstbeschreibenden Instanzen herstellt. Wenn nötig werden als syntaktische Spezifikation für die tatsächlichen Ein- und Ausgaben entweder im WSDL-S- oder einem externen Dokument XML-Schema-Datentypen angelegt. Zusätzlich werden im WSDL-S-Dokument Elementdefinitionen erstellt, die mit der Ontologie verknüpft werden. Sie verweisen allerdings im Gegensatz zum von der WSDL-S-Koalition vorgeschlagenen WSDL-S-Einsatz nicht auf die zugeordneten Konzepte der Datentypen verweisen, sondern auf die Instanzen aus der semantischen Dienstbeschreibung. Diese Abweichung ergibt sich daraus, dass bei den für WSDL-S gegebenen Beispielen die Ontologieinhalte nicht weiter ausgearbeitet wurden, was hier jedoch wie auch bei OWL-S der Fall ist. Das gleiche Vorgehen gilt auch für Vor- und

Nachbedingungen. Diese werden wie in WSDL-S vorgesehen verknüpft, aber ebenfalls mit Instanzen anstelle von Konzepten. Die Referenzen werden unter dem *Operation*-Element eingefügt. Die Operationen aus dem WSDL-S selbst werden mit den Instanzen der *ServiceProfile*-Klasse aus der Ontologie verbunden. Wenn man die aktuelle WSDL-S-Weiterentwicklung [FL06] der Semantic Annotations for Web Services Description Language Working Group zu Grunde legt, ergibt sich für den Beispieldienst der LDoc-Publizierung das zur Übersichtlichkeit etwas gekürzte WSDL 1.1 Dokument aus Information 52.

```
<definitions name="ldocPublishingService" targetNamespace="http://cm-tm.uka.de/services/ldocPublishingService"
             xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
             xmlns:wssem="http://www.w3.org/2002/ws/sawsdl/spec/sawsdl#">
  <types>
    <xs:schema targetNamespace="http://cm-tm.uka.de/services/ldocPublishingService">
      <xs:element name="ldocPublishingRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="ldocRef" type="xs:uriReference“
              wssem:modelReference="http://cm-tm.uka.de/ontologies/serviceProfile.owl#ldocReference" />
            <xs:element name="paId" type="xs:CDATA“
              wssem:modelReference="http://cm-tm.uka.de/ontologies/serviceProfile.owl#publishingAreaId" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </types>
  ...
  <portType name="ldocPublishingPortType">
    <operation name="publishLDoc"
          wssem:modelReference="http://cm-tm.uka.de/ontologies/serviceProfile.owl#ldocPublishingService">
      <input message=“tns:ldocPublishingRequest" />
      <wssem:effect name="ldocPublished"
        modelReference=" http://cm-tm.uka.de/ontologies/serviceProfile.owl #ldocPublished"/>
    </operation>
  </portType>
</definitions>
```

Information 52: Grounding des LDoc-Publizierungsdienstes

5 DIENSTVERZEICHNIS

In diesem Kapitel wird ein Verzeichnis für Webservices konzipiert und dessen Realisierung beschrieben, das auf die in Kapitel 3.3.2 erarbeitete semantische Dienstbeschreibung aufbaut. Auch für das Dienstverzeichnis und die einzusetzende Suchmethode ergeben sich einige Anforderungen aus dem in Kapitel 1.5 vorgestellten Vorgehen bei der Suche nach passenden Diensten und den generellen Ansprüchen, die man an ein Dienstverzeichnis stellt.

5.1 Anforderungen

5.1.1 Generelle Anforderungen

CRUD-Operationen

Ein Verzeichnis sollte Standard-Ein- und Ausgabeoperationen anbieten, insbesondere solche zum Erstellen, Auslesen, Modifizieren und Löschen von Einträgen (*create, read, update, delete*, CRUD).

Semantische Suche

Das Dienstverzeichnis sollte die Möglichkeit bieten, die Informationen aus der semantischen Dienstbeschreibung bei der Suche miteinzubeziehen. Da der wesentliche Einsatz der semantischen Suche bei der Programmierung im Großen liegt, sollte zumindest nach Dienstkategorien, den Benutzerrollen, Vor- und Nachbedingungen sowie den logischen Ein- und Ausgabeparametern gesucht werden können. Dabei sollte auch die Vererbungshierarchie der semantischen Konzepte berücksichtigt werden. Eine Suche nach den jeweiligen Dienstanbietern wäre ebenfalls wünschenswert.

Abwärtskompatibilität

Im Bereich der semantischen Dienstbeschreibung gibt es bisher weder ein standardisiertes Vorgehen für die Beschreibung selbst noch eines für die Veröffentlichung semantisch beschriebener Dienste. Wohl aber gibt es Standards für die Veröffentlichung konventionell beschriebener Dienste. Die Nutzer solcher Standards sollten nicht ausgeschlossen werden. Solange keine Einigung auf ein standardisiertes Dienstverzeichnis erfolgt ist, ist es daher zweckmäßig, für externe Dienstnutzer auch die konventionellen Verzeichnisfunktionen anzubieten.

5.1.2 Anforderungen an die Suche

An die Suche werden im Detail noch die folgenden erweiterten Forderungen gestellt.

Einfache Nutzung der semantischen Informationen

Die Dienstsuche ist der zentrale Punkt, an dem Nutzen aus der Mehrarbeit bei der semantischen Dienstbeschreibung gezogen werden kann. Da die halbmanuelle Suche im Fokus liegt, sollte die Verwendung der Semantik durch den Bediener bei der Spezifikation von Suchanfragen möglichst einfach und transparent gehalten werden, um so den Parametrisierungsaufwand (Kapitel 2.2.5) zu minimieren.

Passende Unschärfe

Bei der Durchführung der Dienstsuche auf Basis einer gegebenen Suchanfrage gilt es, einen Kompromiss bezüglich der Passgenauigkeit der präsentierten Ergebnisse zur Anfrage zu finden. Dies kann durch eine passende Wahl der Unschärfe beim Vergleich der Parameter erreicht werden. Ist die Suchgenauigkeit zu niedrig, so wird der Suchende mit unpassenden Ergebnissen überschüttet, im anderen Fall wird die Auswahlgrenze zu eng gezogen und die Auslassung möglicherweise passender Dienste in Kauf genommen. Das für die Unschärfe gewählte

Verfahren beeinflusst wesentlich die Anzahl der Suchwiederholungen und damit den Gesamtaufwand bei der Suche (Kapitel 2.2.5).

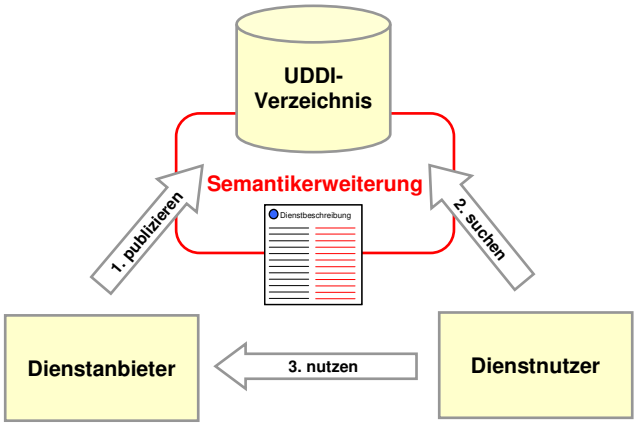
Sortierung nach Relevanz

Unabhängig von der eingesetzten Unschärfe bei der Suche sollten die gefundenen Resultate bewertet und nach ihrer Relevanz bezüglich der Suchanfrage sortiert dargestellt werden. Als Relevanz bezeichnet man einen Wert, der wiedergibt, wie genau ein Dienst in der Ergebnisliste mit den Parametern der Suchanfrage übereinstimmt [SM87]. Ziel dahinter ist es, dem Suchenden die am besten passenden Ergebnisse als erstes anzubieten, damit der Bewertungsaufwand (Kapitel 2.2.5) möglichst gering ausfällt.

5.2 Konzept des Dienstverzeichnisses

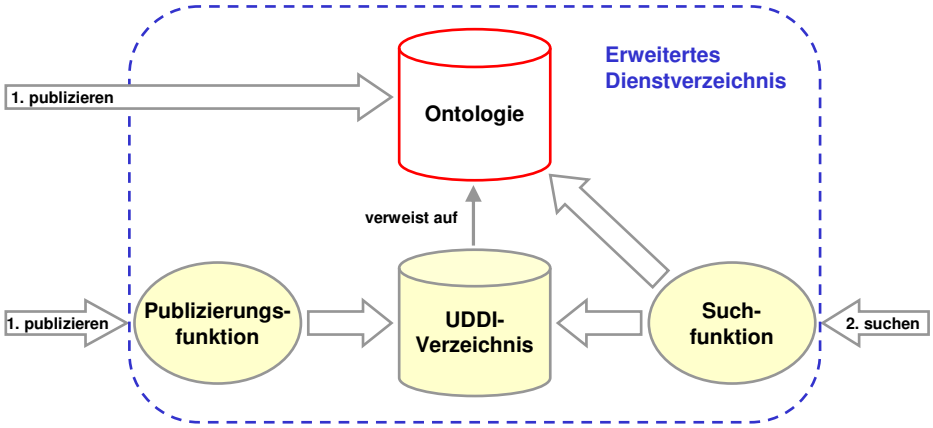
5.2.1 Architektur

Die Architektur des Dienstverzeichnisses mit Semantikunterstützung soll aus den oben genannten generellen Anforderungen motiviert werden. Zur Erfüllung der Abwärtskompatibilität bieten sich prinzipiell zwei Möglichkeiten an: Ein Standard-Dienstverzeichnis kann um die neue geforderte Funktionalität erweitert werden. Alternativ können die Standardschnittstellen an einem neuartigen Dienstverzeichnis zur Verfügung gestellt werden. Das letztere Vorgehen scheint besser geeignet, um auf die speziellen Anforderungen der Integration der semantischen Informationen in das Verzeichnis eingehen zu können, wäre aber mit erhöhtem Aufwand verbunden, da ein komplett neues Verzeichnis zu entwerfen und zu implementieren wäre, das obendrein noch an die alten Schnittstellen adaptiert werden muss. Eine sinnvolle Wiederverwendung bereits etablierter und qualitätsgesicherter Software würde nicht ausgeschöpft. Eine solche Investition wäre zudem riskant in einer Zeit, zu der die Standardisierung der semantischen Dienstbeschreibung noch in der Schwebe liegt. In Anbetracht dessen wird hier die erste Möglichkeit gewählt. Das Standard-Dienstverzeichnis, das UDDI-Verzeichnis, soll mit einer Semantikerweiterung versehen werden, die den geforderten Anforderungen genügt (Information 53).



Information 53: UDDI-Verzeichnis mit Semantikerweiterung

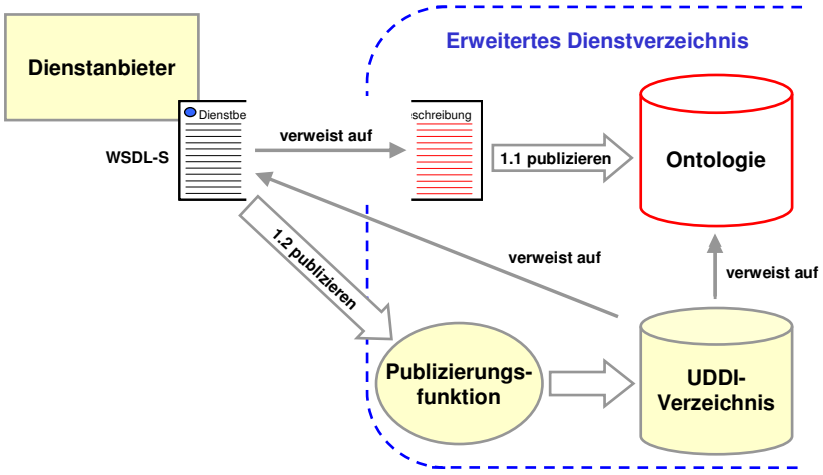
Die Semantikerweiterung betrifft dabei lediglich das Publizieren und Suchen von Diensten im Verzeichnis. Die Interaktion zwischen Dienstanbieter und Dienstnutzer bleibt völlig unverändert im Vergleich zur normalen Verwendung von Webservices. Zentrale Rolle bei der Publizierung und der Suche der Webservices spielt die in Kapitel 3.3.2 vorgestellte semantische Dienstbeschreibung.



Information 54: Komponenten des Dienstverzeichnisses

Wie aus Kapitel 3.3.2 hervorgeht, wird ein Teil der Dienstbeschreibung als semantische Konzepte in einer oder mehreren Ontologien abgelegt. Für das erweiterte Dienstverzeichnis spielt es keine Rolle, wie diese Ontologien verwaltet werden, solange einheitlich auf diese zugegriffen werden kann. Es kann offen gelassen werden, ob die Ontologie letztendlich als Datei, in einem Versionierungssystem oder in einem speziellen Ontologieverwaltungssystem, wie von [FF99] und [DW+01] befürwortet, platziert ist. Ebenso wenig relevant ist, ob die Ontologie lokal oder verteilt im Internet vorliegt. Da die semantische Suche ohne die Ontologie nicht funktionieren kann, ist sie im Vorgestellten Entwurf (Information 54) dennoch innerhalb des erweiterten Dienstverzeichnisses angesiedelt. Von einer Duplizierung der semantischen Dienstbeschreibung im erweiterten Dienstverzeichnisses, beispielsweise im UDDI wie beim DAML-S/UDDI Matchmaker (Kapitel 3.2.1), wird Abstand genommen. Um die Semantikinformationen mit dem UDDI-Verzeichnis zu verknüpfen und eine semantische Suche auf den eingepflegten Diensten zu ermöglichen, wird das UDDI-Verzeichnis durch spezielle Funktionen zur Publizierung von semantisch beschriebenen Webservices und zur Suche nach denselben ergänzt.

5.2.2 Publizierung von Webservices



Information 55: Publizierung von Webservices

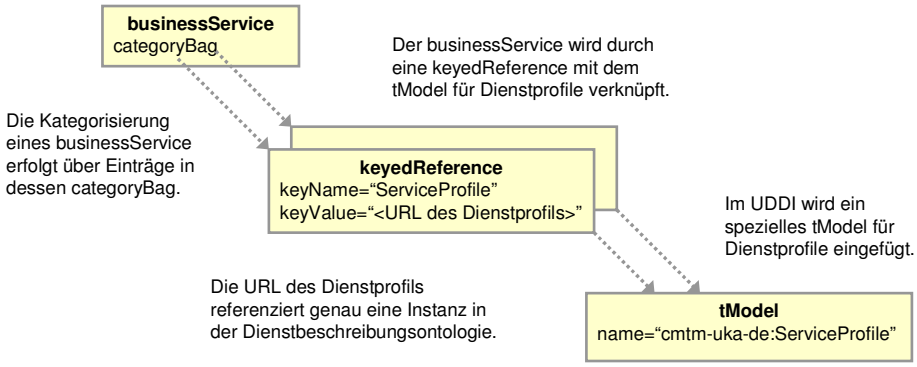
Mit der extern angesiedelten Ontologie verläuft die Publizierung der Webservices in zwei Schritten. Der Semantikanteil der Dienstbeschreibung, der die Dienstprofile spezifiziert, wird direkt in die Dienstbeschreibungsontologie aufgenommen. Dies kann bereits vor oder während der Dienstentwicklung geschehen, muss aber vor der Veröffentlichung einer Dienstinstanz, die

das Profil realisiert, geschehen, damit diese Instanz durch die semantische Dienstsuche gefunden werden kann. Die Publizierung der Webservice-Instanz geschieht über eine Publizierungsfunktion, die die WSDL-Informationen aus dem WSDL-S-Dokument in einem allgemein anerkannten Verfahren (z.B. [CJ02a] oder [CJ04]) in das UDDI-Verzeichnis überträgt und zusätzlich die Verweise auf die Konzepte des Dienstprofils aufrecht erhält (Information 55).

Im Vergleich zu den in Kapitel 3.2 beschriebenen Verfahren ergeben sich damit einige Unterschiede. Die semantischen Informationen werden ausschließlich in der Ontologie gehalten, die syntaktischen Informationen im WSDL-S-Dokument und die Geschäftsdaten werden ausschließlich im UDDI-Verzeichnis abgespeichert.

Verknüpfung zwischen UDDI und Ontologie

Was die Verknüpfung zwischen UDDI-Verzeichnis und Ontologie anbelangt, wird eine ähnliche Strategie wie bei METEOR-S und Lumina gewählt, die Verweise werden jedoch auf ein Minimum reduziert. Da bei der Dienstbeschreibung in der Ontologie nicht wie bei METEOR-S einzelne losgelöste Konzepte referenziert werden, sondern ein zusammenhängendes Dienstprofil, genügt es, dieses mit den UDDI-Einträgen der zugehörigen Dienstinstanzen zu assoziieren. Dazu wird von der in Kapitel 2.2.4 angedeuteten Kategorisierung des UDDI-*businessService* Gebrauch gemacht.



Information 56: Verknüpfung zwischen Ontologie und UDDI

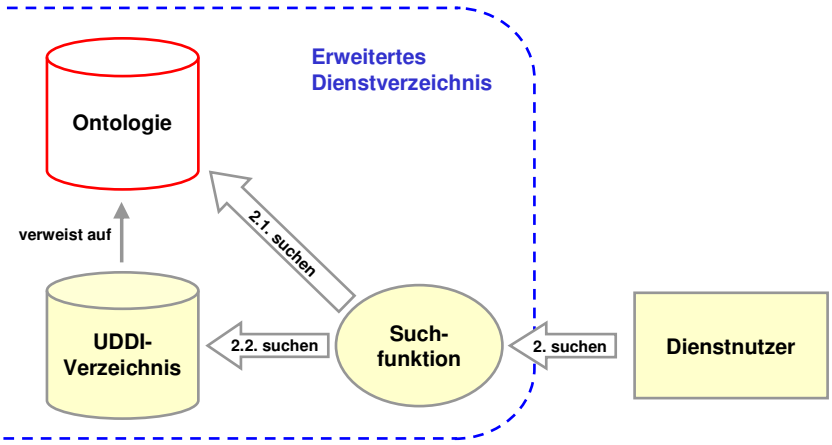
Im UDDI-Verzeichnis muss ein spezielles *tModel* für diese Kategorisierung mit dem willkürlich gewählten Namen "cmtm-uka-de:ServiceProfile" vorhanden sein oder angelegt werden. Die Zuordnung eines Dienstprofils zu einem *businessService* geschieht dann folgendermaßen: Jeder *businessService* besitzt zu seiner Kategorisierung einen so genannten *categoryBag*. In diesen wird eine *keyedReference* eingefügt, die auf das ServiceProfile-*tModel* verweist. Als Name für die Referenz wird wiederum willkürlich "ServiceProfile" gewählt und als Wert die URL, die das Dienstprofil eindeutig kennzeichnet. Auf diese Weise können einem *businessService* alle durch den zugehörigen Webservice realisierten Dienstprofile zugeordnet werden.

Abwärtskompatibilität und CRUDs

Die Publizierungsfunktion stellt lediglich eine einfache Möglichkeit der standardkonformen Veröffentlichung von Webservices im Verzeichnis dar. Die Schnittstellen des UDDI-Verzeichnisses selbst werden dadurch nicht verschattet, sie werden auch von außen Verwendung finden. Zum einen müssen sie für die restlichen CRUD-Operationen verwendet werden, da die Publizierungsfunktion nur für die Veröffentlichung von Webservices ausgelegt ist. Änderungen, Löschungen und die Veröffentlichung von Geschäftsinformationen erfolgen direkt über die UDDI-Schnittstellen. Zum anderen gewähren sie zusammen mit der gängigen Abbildung der WSDL-Konstrukte auf UDDI-Inhalte die geforderte Abwärtskompatibilität des Gesamtsystems.

5.2.3 Semantische Dienstsuche

Die weitere zum gewöhnlichen UDDI-Verzeichnis vorgenommene Erweiterung ist eine Funktion für die semantische Dienstsuche. Sie nutzt die Inhalte der Ontologie und des UDDI-Verzeichnisses, um Ergebnislisten für gestellte Suchanfragen zu generieren (Information 57). Die Suche wird fast vollständig auf der Dienstbeschreibungsontologie ausgeführt, da eine Berücksichtigung der Vererbungshierarchie nur dort möglich ist und das UDDI-Verzeichnis per Konzeption keine Semantikinformationen enthält. Lediglich die Instanzen für gefundene Dienstprofile und die Unternehmen, die diese veröffentlicht haben, werden anschließend im UDDI-Verzeichnis gesucht.



Information 57: Dienstsuche

Spezifikation der Suchparameter

Parametername	Zulässige Oberklasse	Beispiel
Unternehmen	---	ed.serv
Dienstkategorie	ServiceProfile	PublishingService
Benutzerrolle	Thing	Author, Learner, Provider
Vorbedingung	State	Available, Published
Nachbedingung	State	Published, Printed, Transferred
Eingabe	Entity	LDoc, LectureNotes
Ausgabe	Entity	LearningObject, Slides

Information 58: Zulässige Suchparameter

Vorgesehen als Suchparameter sind nach den generellen Anforderungen die Dienstkategorien, die Benutzerrollen, Vor- und Nachbedingungen und die logischen Ein- und Ausgabeparameter der Dienste. Dazu kommt noch das im UDDI zwangläufig vorhandene dienstpublizierende Unternehmen. Alle Parameter bis auf letzteren sind als Namen von Klassen aus der Dienstbeschreibungsontologie anzugeben. Dabei kann für die Dienstkategorie maximal eine Klasse ausgewählt werden. Für alle weiteren kann eine Menge von mehreren Namen angegeben werden, die bei der Suche mit einer UND-Verknüpfung berücksichtigt werden. Für alle angegebenen Parameter sollte beachtet werden, dass nur Angaben von Unterklassen der jeweils

zulässigen Klassen der Wertebereiche in der Ontologie zu sinnvollen Ergebnissen führen. Welche dies sind, leitet sich aus der Dienstbeschreibungs-Ober-Ontologie ab. Eine Zusammenfassung mit Beispielen für gültige Eingaben bietet Information 58.

Suchalgorithmus

Bei der Suche kommt ein iterativer Algorithmus zum Tragen, der in mehreren Intervallen mit steigender Unschärfe operiert (Information 59). In jedem Intervall werden maximal zwei Suchanfragen ausgeführt. Zunächst werden aus der Ontologie die Dienstprofile ermittelt, die den Suchparametern entsprechend der aktuellen Unschärfestufe entsprechen. Konnten solche Profile gefunden werden, können die Dienstinstanzen, die diese Profile implementieren, durch eine Suche anhand der vorgenommenen Kategorisierung im UDDI-Verzeichnis ermittelt werden. Wurden in einem Durchlauf keine passenden Dienste gefunden, so kann die Unschärfe erhöht und mit der Iteration fortgefahren werden. Alternativ bricht die Suche ab und meldet eine leere Ergebnismenge, wenn die verfügbaren Unschärfestufen ausgeschöpft sind.

(1) Solange Ergebnismenge leer
  (1) Suche nach passenden Dienstprofilen in der Ontologie
  (2) Für jedes gefundene Dienstprofil:
    (1) Suche nach zugehörigen Dienstinstanzen im UDDI
  (3) Erhöhung der Unschärfe oder Abbruch der Suche

Information 59: Ein iterativer Suchalgorithmus

Für die Variation der Unschärfe kann der Einfluss der Suchparameter in Stufen modifiziert werden. Die ausgewählten Variationen sind in Information 60 zusammengefasst. Anfänglich wird exakt nach den Angaben in den Suchparametern gesucht. Im zweiten Schritt kommen konforme Dienstkategorien, Benutzerrollen und Bedingungen hinzu. Dies bedeutet, dass für die Dienstkategorie, Benutzerrolle und Nachbedingung auch Unterklassen und für die Vorbedingung auch Oberklassen als passend angesehen werden. Im dritten Schritt wird dasselbe Konzept für Parameter übernommen. Unterklassen der Ausgabeparameter und Oberklassen der Eingabeparameter sind ebenfalls zulässig. Als nächstes bleibt die UND-Verknüpfung der Benutzerrollen, Parameter- und Bedingungsmengen unberücksichtigt. Zuletzt bleibt noch die Möglichkeit manche Angaben komplett zu ignorieren. Dabei bleiben die Benutzerrolle, die Dienstkategorie, und die Vor- und Nachbedingungen unberücksichtigt und zwar in dieser Reihenfolge. Es werden also nur noch die angegebenen Ein- und Ausgabeparameter zur Einschränkung der Suchergebnisse herangezogen.

(1) Alle Angaben exakt erforderlich, UND-Verknüpfung bei Listen
(2) Zusätzlich Konformität der Dienstkategorie, Benutzerrolle und Bedingungen
(3) Zusätzlich kontravariante Vererbung der Parameter
(4) Alternativ ODER-Verknüpfung bei Listen
(5) Benutzerrolle unberücksichtigt
(6) Dienstkategorie unberücksichtigt
(7) Bedingungen unberücksichtigt

Information 60: Unschärfestufen bei der Dienstsuche

Ermittlung der Relevanz

Für die Ermittlung der Relevanz wird ein einfaches Vektormodell aus dem klassischen *Information Retrieval* eingesetzt [BR99:27]. Dabei werden Suchergebnis und Suchanfrage jeweils als ein Vektor von Gewichten angesehen. Die Dimension der Vektoren bestimmt sich aus der Anzahl der Klassen in der Ontologie multipliziert mit der Anzahl der bei der Suche berücksichtigten Parameter (Dienstkategorie, Benutzerrolle, Vorbedingung, …), da jede Klasse in jedem Parameter auftauchen kann (Im gegebenen Konzept wird mit sechs Parametern gesucht). Jedes Gewicht im Vektor gibt die Wichtigkeit der Klasse für einen Parameter im betrachteten Suchergebnis beziehungsweise in der Suchanfrage an. Es werden folgende Gewichtungen vorgenommen:

1. Alle Klassen in den Suchergebnissen werden pro Parameter jeweils mit 1 gewichtet. Das bedeutet beispielsweise, dass das Gewicht immer gleich ausfällt unabhängig davon, ob ein Dienst nur ein einzelnes *LDoc* als Eingabe benötigt oder mehrere. Tritt jedoch *LDoc* sowohl als Eingabe als auch als Ausgabe oder in einem sonstigen Parameter auf, werden beide Fälle getrennt bewertet.
2. Alle im Suchparameter angegebenen Klassen werden ebenfalls pro Parameter jeweils mit 1 gewichtet. Es ist also nicht möglich, durch Mehrfachnennung einer Klasse in einem Parameter die Gewichtung zu erhöhen. Dennoch wirken sich wiederholte Nennungen derselben Klasse in verschiedenen Parametern aus.
3. Tritt eine Klasse in einem Parameter nicht auf, so wird sie (mit der Ausnahme 4.) für diesen mit 0 gewichtet. Daraus folgt, dass die Vektoren dünn besetzt sind. Das heißt, dass sie auch bei großen Ontologien nur wenige Einträge enthalten, die ungleich 0 sind.
4. In Fällen, in denen konforme Vererbung zu einem Suchergebnis führt, werden die konformen Klassen für den jeweiligen Parameter der Suchanfrage mit 0,5 gewichtet. Dies führt dazu, dass konforme Übereinstimmungen weniger zur Relevanz beitragen als direkte Treffer, jedoch mehr als nicht vorhandene Klassen.

Aus den Punkten 1. bis 4. folgt für den aktuellen Vorschlag ebenso, dass die unterschiedlichen Parameter gleich gewichtet werden. Ein Treffer in der Dienstkategorie zählt genauso viel wie ein Treffer bei den Nachbedingungen.

Anhand der somit definierten Vektoren für Suchergebnis und Suchanfrage kann man die Relevanz als Korrelation der beiden Vektoren auffassen und beispielsweise als den Kosinus des von den Vektoren eingeschlossenen Winkels berechnen. Dieser Wert liegt (bei ausschließlich positiven Gewichten) zwischen 0 und 1 und ergibt sich als normiertes Skalarprodukt der beiden Vektoren [BR99:27,28].

5.3 Realisierung des Prototyps

Nach der analytischen Vorarbeit in den vorigen Kapiteln dieser Diplomarbeit soll das Dienstverzeichnis nun prototypisch umgesetzt werden. Dazu schließt sich im Folgenden ein softwaretechnischer Entwurf gefolgt von einem Unterkapitel zur Implementierung und zum Betrieb des Prototyps an.

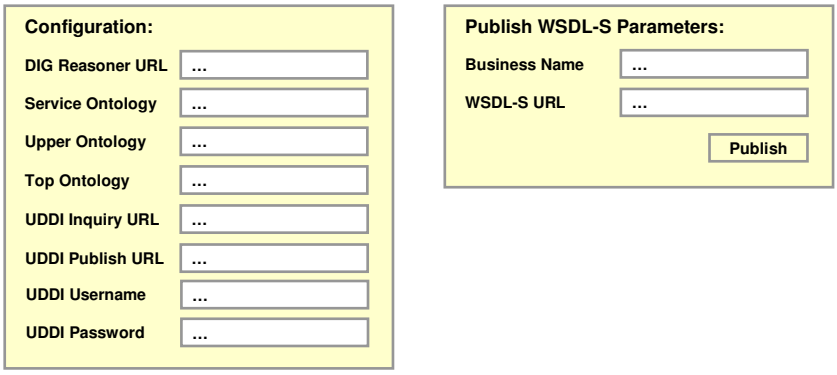
5.3.1 Entwurf

Benutzeroberfläche

Die Benutzeroberfläche der Dienstpublizierung und -Suche könnte beliebig trickreich und interaktiv gestaltet werden, um ein möglichst reibungsloses und Aufwand schonendes Arbeiten mit dem Dienstverzeichnis zu ermöglichen. Auch wenn dies in einer endgültigen Version für den Wirkbetrieb durchaus wünschenswert ist, ist ein solches Vorhaben mit hoher Komplexität und einem Zeitaufwand behaftet, die der Umsetzung während dieser Arbeit entgegenstehen. Der zu erarbeitende Prototyp wird daher neben einer API, die die benötigte Funktionalität enthält, lediglich auf eine einfach zu realisierende GUI beschränkt. Mögliche Verbesserungen, Umstrukturierungen und Erweiterungen zu einer finalen Version werden in den nächsten Unterkapiteln zusätzlich angedeutet.

Konfiguration

Die angestrebte Publizierung und Suche von Webservices kann nicht ohne einige Konfigurationsparameter realisiert werden. Für diese ist eine einfache Benutzerschnittstelle vorgesehen (Information 61, links). Die benötigten Parameter sind im Einzelnen die URLs des DIG *reasoners* und der Ontologien (Dienstbeschreibungsontologie, Dienstbeschreibungs-Ober-Ontologie und *Top-Level*-Ontologie) sowie die Zugangsdaten des zu Grunde liegenden UDDI-Verzeichnisses (*Inquiry*- und *Publish*-URL, Benutzername und Passwort). Als zukünftige Ergänzung wäre hier sowohl für die Ontologien als auch für das UDDI-Verzeichnis das Erstellen, Speichern und die Auswahl von mehreren Konfigurationsprofilen denkbar.



Information 61: GUI Konfiguration und Publizierungsdienst

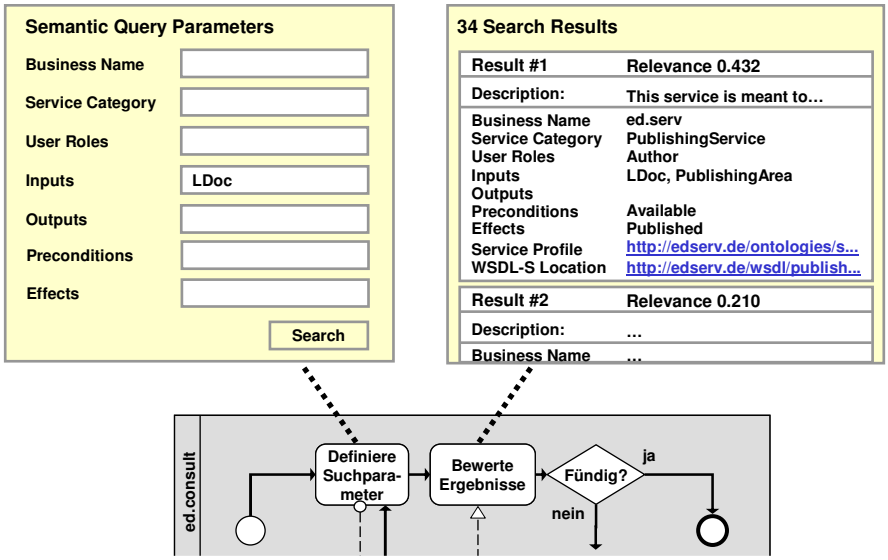
Publizierungsfunktion

Die Benutzeroberfläche für die Publizierungsfunktion wird im Prototyp einfach ausgestaltet (Information 61, rechts). Neben einem Eingabefeld, das die URL der zu publizierenden WSDL-S-Datei aufnimmt, ist lediglich ein weiteres Feld für den Namen der *business entity* vorgesehen, unter der die Webservices aus dem WSDL-S veröffentlicht werden sollen.

Suchfunktion

Die Benutzerschnittstelle der Suchfunktion spaltet sich entlang der vorgesehenen Aktivitäten mit Benutzerinteraktion in zwei Teile (Information 62). Zur Aktivität “Definiere Such-

parameter" existiert eine Eingabemaske, in der die Suchparameter erfasst werden können. Für alle Suchparameter beinhaltet diese Textfelder, in die die Namen einzelner Konzepte beziehungsweise wenn vorgesehen (Kapitel 5.2.3) eine durch Komma getrennte Liste von Konzeptnamen eingegeben werden kann. In eine finale Version könnten folgende Verbesserungen eingearbeitet werden: Eine Auswahl passender Konzepte mittels *drag'n'drop* aus einem Ontologieeditor wäre ebenso denkbar wie eine ontologiegestützte automatische Vervollständigung bei der Eingabe über die Tastatur.

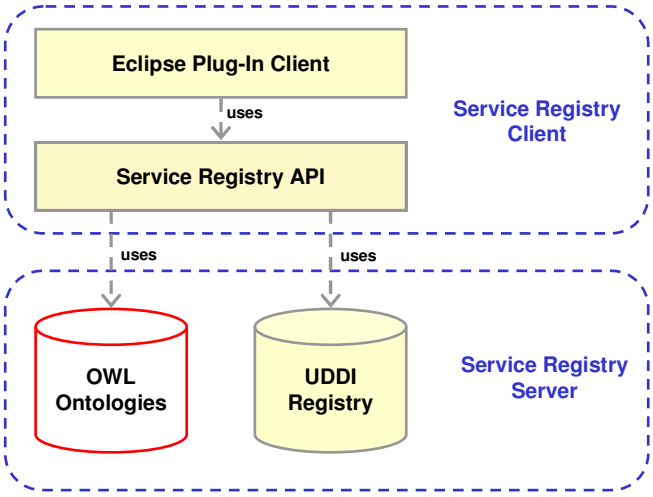


Information 62: GUI für den Suchdienst

Für die Aktivität "Bewerte Ergebnisse" ist im Prototyp eine nach Relevanz sortierte Liste der Ergebnisse vorgesehen. Sie beinhaltet pro Dienst Informationen über dessen Eigenschaften sowie zur Dienstprofilinstanz und eine Verknüpfung zur WSDL-S-Datei, die die technische Definition und das *grounding* des Dienstes enthält. Auch hier sind noch einige Verbesserungen möglich. Denkbar wären eine direkte Verknüpfung der Parameter der Ergebnisse mit einem geeigneten Ontologieeditor sowie eine visuelle Anzeige der Ontologiekonzepte des jeweiligen Dienstprofils. Auch die Anzeige in welcher Unschärfestufe die Dienste gefunden wurden, beziehungsweise welche Konzepte direkte Treffer sind, welche konform passen und welche ignoriert wurden, wäre interessant.

Anwendungsarchitektur

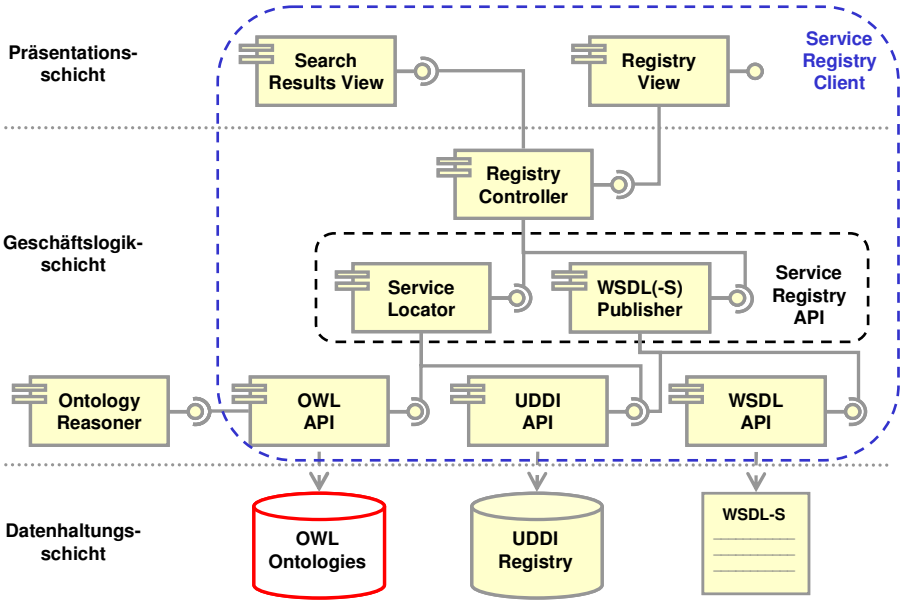
Der Prototyp zeichnet sich durch eine einfache Softwarearchitektur aus. Er besteht aus einem Server- und einem Clientanteil. Der Client enthält eine *Application Programming Interface* (API), die *Service Registry* API , die die Zugriffe auf die darunter liegenden OWL-Ontologie und das UDDI-Verzeichnis kapselt. Zur Nutzung durch Softwareingenieure bei der Dienstverschaltung dient ein *plug-in* für die Entwicklungsumgebung Eclipse [URL-22], das die oben vorgestellte Benutzeroberfläche beinhaltet und auf die Funktionen der API zurückgreift. Da der Dienstverzeichnis-Client mit der *Service Registry* API einen Großteil der Geschäftslogik enthält, kann man die Architektur auch als *Rich-Client*-Architektur bezeichnen. Mittels entsprechender Server-Softwarekomponenten ist es durchaus denkbar die Fachfunktionalität in einer späteren Version des Verzeichnisses auf die Seite des Servers zu verlagern. Eine Darstellung der Zusammenhänge des aktuellen Prototyps bietet Information 63.



Information 63: Anwendungsarchitektur

Initiales Komponentenmodell

Der Dienstverzeichnis-Prototyp ist als Drei-Schichten-Architektur aufgebaut und verfügt über eine Präsentations-, eine Geschäftslogik- und eine Datenhaltungsschicht.



Information 64: Initiales Komponentenmodell

Auf die oberen beiden Schichten verteilen sich neun Komponenten:

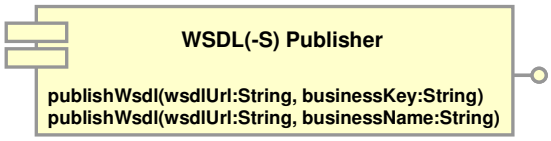
1. *Search Results View* ist eine Komponente, die der Anzeige der Suchergebnisse dient.
2. Die *Registry View* bietet dem Benutzer die Möglichkeit, die vorgesehenen Aktionen Suche und Publizierung zu aktivieren und die dafür nötigen Parameter anzugeben.
3. Die Komponente *Registry Controller* dient der Koordination der Anzeigekomponenten und der Geschäftslogikfunktionen.
4. Der *Service Locator* ist die erste Komponente der *Service Registry* API. Sie beinhaltet die semantische Suche nach Webservices.

5. Der WSDL(-S) *Publisher* gehört ebenfalls zur *Service Registry* API und enthält die Funktionalität zur Publizierung der WSDL-S-Dateien.
6. Der *Ontology Reasoner* wird von der OWL API benutzt, um die Ontologieinhalte zu durchsuchen.
7. OWL API, UDDI API und WSDL API dienen jeweils der Bearbeitung der unterschiedlichen Datenobjekte. Hierfür können nach Möglichkeit externe Komponenten wiederverwendet werden.

Komponentenmodelle

In diesem Unterkapitel werden die beiden wichtigsten Komponenten des Dienstverzeichnis-Prototyps vorgestellt.

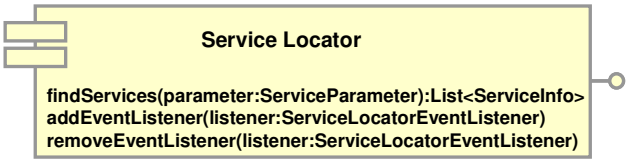
WSDL(-S) Publisher



Information 65: WSDL(-S) Publisher

Der WSDL(-S) *Publisher* bietet die Möglichkeit, alle Webservices, die in einer WSDL- oder WSDL-S-Datei definiert sind, im Dienstverzeichnis zu publizieren. Dazu benötigt er die URL, über die die WSDL-Datei abgerufen werden kann sowie Informationen über die *businessEntity*, unter der die Webservices veröffentlicht werden sollen. Dazu kann man wahlweise den UDDI-Schlüssel der *businessEntity* oder ihren Namen angeben. Im letzteren Fall wird eine neue *businessEntity* mit dem gegebenen Namen angelegt, falls keine gefunden werden konnte.

Service Locator

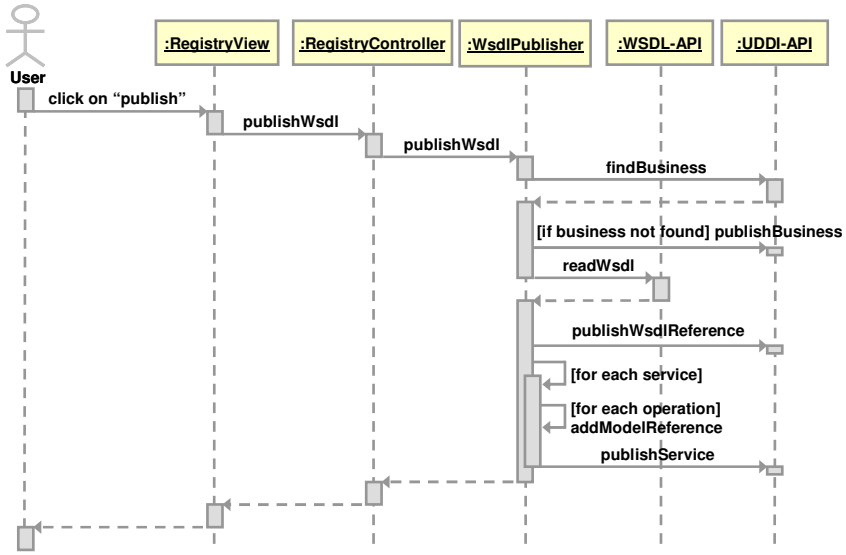


Information 66: Service Locator

Mit Hilfe des *Service Locators* kann nach Webservices im Dienstverzeichnis gesucht werden. Dazu bietet er eine Schnittstellenfunktion, die für eine gegebene *ServiceParameter*-Instanz die Suche durchführt und die ermittelten Ergebnisse als eine Liste von *ServiceInfos* zurückliefert. Über zwei weitere Funktionen können *ServiceLocactorEventListener* registriert und entfernt werden, die über Ereignisse, die während der Suche auftreten, informiert werden.

Ablaufbeschreibungen

Publizierungsfunktion

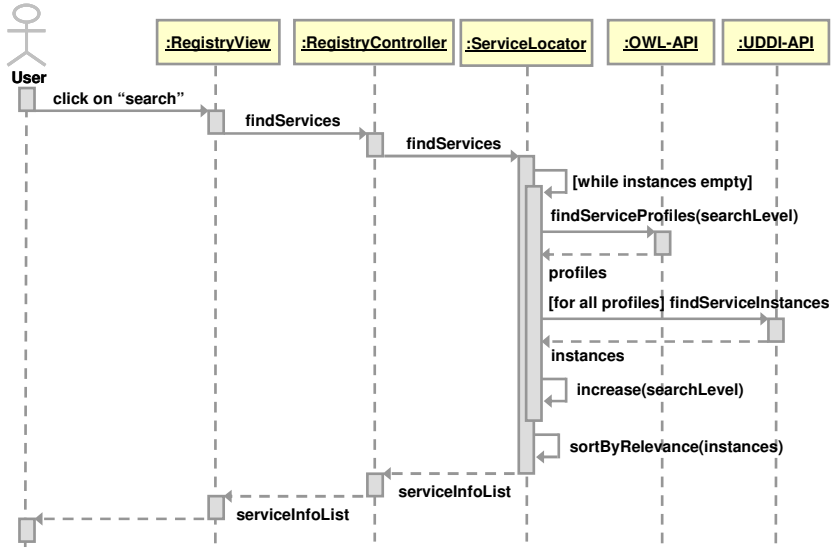


Information 67: WSDL(-S)-Publizierung

Information 67 zeigt den Ablauf der WSDL(-S)-Publizierung, der im Wesentlichen aus vier Schritten besteht.

1. Im ersten Schritt wird im UDDI nach der *businessEntity* gesucht, zu der die Dienste veröffentlicht werden sollen. Falls keine gefunden wurde, wird eine neue mit dem angegebenen Namen im UDDI angelegt.
2. Jetzt wird im UDDI eine Referenz auf die WSDL(-S)-Datei gespeichert.
3. Anschließend werden alle Webservices aus der zu publizierenden WSDL(-S)-Datei ermittelt. Dabei wird auch untersucht, welche *operations* durch die Webservices realisiert werden, und falls vorhanden werden deren Referenzen auf Dienstprofile in der Ontologie ermittelt.
4. Zum Schluss werden die Webservices mit allen Verknüpfungen zu den semantischen Informationen veröffentlicht.

Suchfunktion



Information 68: Semantische Dienstsuche

Die semantische Dienstsuche gemäß dem Suchalgorithmus aus Kapitel 5.2.3 ist in Information 68 dargestellt. Solange keine Ergebnisse vorhanden sind, werden iterativ Dienstprofile und Dienstinstanzen gesucht und schrittweise die Suchungenauigkeit (*searchLevel*) erhöht. Die gefundenen Dienstinstanzen werden nach Relevanz sortiert zurückgegeben.

5.3.2 Implementierung

Zur gewählten Softwaretechnologie

Die eingesetzte Softwaretechnologie wird für den Prototyp so gewählt, dass sich der Umfang der selbst zu implementierenden Teile gering halten lässt. Dies wird durch die Wahl von Java als Implementierungssprache erreicht. In diesem Bereich gibt es zahlreiche *Open-Source*-Komponenten, so dass die Anbindung der Datenhaltung quasi vollständig über externe Bibliotheken realisiert werden kann. Auch die Erstellung einer Anwendungs-GUI lässt sich mit dem Eclipse Framework [URL-22], das sich in der Softwareentwicklung als De-facto-Standard unter den Entwicklungsumgebungen herauskristallisiert hat und auch hier zur Entwicklung verwendet wird, hervorragend umsetzen. Einen Überblick über die verwendeten externen Komponenten gibt Information 69. Dabei ist jeweils auch der Name der Komponente aus dem Entwurf in Klammern angegeben.

WSDL4J 1.6.1	Javabibliothek für die Bearbeitung von WSDL-Dateien (Komponente WSDL API) [URL-23]
UDDI4J 2.0.5	Javabibliothek für die Kommunikation mit UDDI-Verzeichnissen (Komponente UDDI API) [URL-24]
Protégé-OWL Beta 2.3 Build 339	Javabibliothek für die Bearbeitung von OWL-Dateien und Kommunikation mit dem Ontologie-Reasoner (Komponente OWL API) [URL-25]
Eclipse Framework 3.2.0	Javaframework für die Erstellung von GUI-Applikationen [URL-22]
RacerPro 1.9	Ontologie-Reasoner (Komponente Ontology Reasoner) [URL-26]

Information 69: Externe Komponenten

Eine Besonderheit liegt bei der Protégé OWL API vor, die für das Einlesen der OWL-Dateien und die Kommunikation mit einem externen Ontologie-*Reasoner* verwendet wird. Um den nötigen Zugriff auf die *Reasoner*-Funktionalität zu haben, musste bei drei Javakonstrukten die Sichtbarkeit erhöht werden. Dies waren im Einzelnen die Methode `createQueryElement` der Klasse `DefaultDIGTranslator`, die Methode `synchronizeReasoner` der Klasse `DefaultProtegeOWLReasoner` und das Attribut `translator` der Klasse `DefaultDIGReasoner`.

Zur Kommunikation mit dem Ontologie-*Reasoner* bedient sich Protégé-OWL der DIG-Schnittstelle, die momentan erst in der Version 1.1 [Be03] verfügbar ist. Als *reasoner* wurde bei der Umsetzung RacerPro 1.9 getestet, andere DIG 1.1 kompatible *reasoner* sollten jedoch auch funktionieren. Problematisch bei der Verwendung von DIG 1.1 sind die Einschränkungen bei den Anfragen. So ist es momentan nicht möglich, nach Instanzen einer Klasse zu fragen, ohne die Instanzen ihrer Unterklassen mitgeliefert zu bekommen. Auch eine Anfrage nach Instanzen der Oberklassen einer Klasse gibt es nicht. Dadurch ergeben sich überzählige Ergebnisse bei den Anfragen für die verschiedenen Unschärfestufen aus Information 60, die in der aktuellen Version des Prototyps programmatisch aussortiert werden. Dies führt zwar zu einem korrekten Ergebnis, mindert allerdings die Verarbeitungsgeschwindigkeit. Mit der Verabschiedung und Umsetzung von DIG 2.0 wird diese Sonderbehandlung voraussichtlich hinfällig und kann durch geeignete Anfragen ersetzt werden. Entsprechende Stellen sind im Quellcode gekennzeichnet.

Struktur und Umfang der Implementierung

Ontologieanteile

Gemäß Kapitel 4 besteht die Ontologie aus vier Teilen, die jeweils als OWL-DL-Ontologien realisiert sind und sich gegenseitig wie erforderlich referenzieren. Die URLs zur Identifizierung und Beschaffung der Ontologien verweisen für den Prototyp zunächst auf `localhost` und können später angepasst werden. Die verwendeten Domänenontologie und Dienstbeschreibungsontologie enthalten nur Testdaten und sind daher als vorläufig anzusehen. Sie sollten für die Beschreibung tatsächlich zu realisierender Dienste ersetzt werden.

URL	Teil-Ontologie
http://localhost/ontologies/top.owl#	Top-Level-Ontologie
http://localhost/ontologies/upperService.owl#	Dienstbeschreibungs-Ober-Ontologie
http://localhost/ontologies/serviceProfile.owl#	Dienstbeschreibungsontologie
http://localhost/ontologies/eduDomain.owl#	Domänenontologie der Aus- und Weiterbildung

Information 70: OWL-Ontologien

Java-Anteile

Information 71 zeigt tabellarisch alle Javapakete der aktuellen Implementierung des Prototyps nebst der zugehörigen Komponenten aus dem Entwurf. Die *Service Registry* API macht einen Großteil der Implementierung aus. Die Komponente WSDL(-S) *Publisher* enthält ca. 1500 *Lines of Code* (LOC) und verteilt sich nach Zuständigkeit auf mehrere Javapakete. Die aktuelle Implementierung veröffentlicht die Webservices nach den *best practices* von OASIS [CJ02a] und mit der in Kapitel 5.2.2 beschriebenen Technik für die Verknüpfung zu den semantischen Informationen. Der *Service Locator*, verteilt sich ebenfalls auf mehrere Javapakete, die insgesamt ca. 2000 LOC umfassen. Die restlichen Komponenten und ihre Javapakete sind relativ klein (700 LOC) und enthalten das *plug-in* für Eclipse. Der Quellcode liegt in zwei

Eclipse-Projekten, je eines für die *Service Registry* API und das Eclipse-*Plug-In*, mit insgesamt 34 mit Code- und Dokumentationskommentaren versehenen Javaklassen vor.

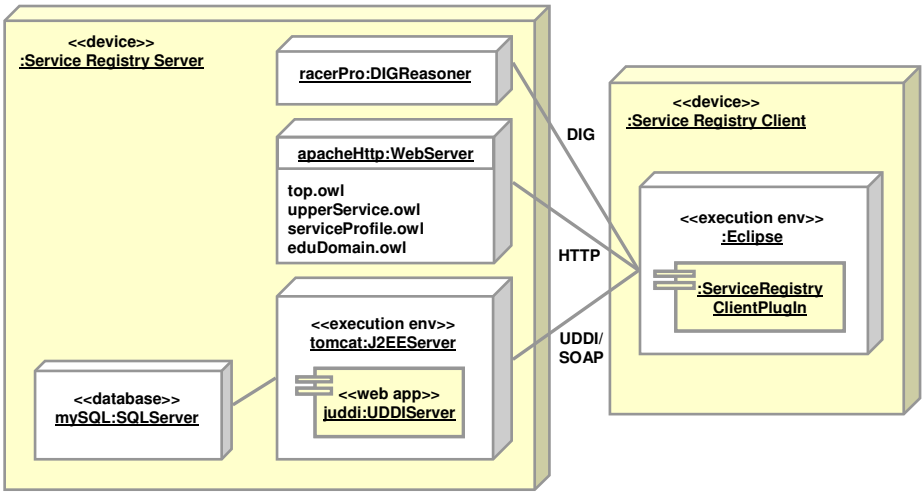
Name des Pakets	Name der Komponente
serviceregistry	Registry Controller
serviceregistry.plugin.editors	Search Results View
serviceregistry.plugin.views	Registry View
serviceregistry.publish.uddi	WSDL(-S) Publisher (Service Registry API)
serviceregistry.publish.uddi.wsdl	
serviceregistry.publish.uddi.wsdls	
serviceregistry.query	Service Locator (Service Registry API)
serviceregistry.query.protege	
serviceregistry.query.protege.dig	
serviceregistry.query.uddi	

Information 71: Softwarepakete

5.3.3 Einsatz

Inbetriebnahme

Um den Prototyp nutzen zu können, müssen zunächst einige Server und Anwendungen installiert werden. Die Verteilung der Server auf Netzwerkkomponenten ist prinzipiell beliebig. In dieser Beschreibung werden alle Server auf dem lokalen Rechner (`localhost`) installiert. Ebenso beliebig ist die Wahl der Systeme für Webserver, Datenbank, UDDI, *web application server*, DIG-Ontologie-*Reasoner*, solange von diesen die jeweiligen Standards eingehalten werden. In den folgenden Unterkapiteln wird die Installation einer Beispielzusammenstellung gezeigt, die auch für die Implementierung und *screen shots* verwendet wurde. Ein UML-Verteilungsdiagramm dieser Installation ist in Information 72 gegeben.



Information 72: Verteilungsdiagramm des Prototyps

Der Dienstverzeichnis-Prototyp besteht aus dem Dienstverzeichnis-Server und dem Dienstverzeichnis-Client. Die einzelnen Softwarekomponenten können aus dem Internet heruntergeladen werden (sie befinden sich aber auch auf der beigelegten CD im Unterverzeichnis `Software`). Sowohl auf dem Server als auch auf dem Client wird ein Java-2-

Platform Standard Edition (J2SE) Runtime Environment [URL-36] benötigt, damit alle Server und Anwendungen lauffähig sind.

XAMPP 1.5.4 (Apache HTTP, MySQL)

XAMPP [URL-30] ist eine zum Paket geschnürte Distribution verschiedener Serversysteme und beinhaltet unter anderem den Apache HTTP-Server [URL-31] und die MySQL-Datenbank [URL-32]. XAMPP enthält ein Installationsprogramm für Windows, mit dem es eingerichtet werden kann. Am Ende der Installation genügt es, Apache HTTP und MySQL als Dienst zu aktivieren. Der mitgelieferte FTP-Server wird nicht benötigt. Mit dem automatisch installierten XAMPP-Kontrollprogramm können die einzelnen Server gestartet und angehalten werden.

Um die Ontologien und WSDL-Dateien über HTTP verfügbar zu machen, müssen diese in das Verzeichnis, das vom Apache HTTP-Server veröffentlicht wird, kopiert werden. Bei einer gewöhnlichen Installation ist dies das Verzeichnis `C:\Programme\xampp\htdocs`. Die beiden Verzeichnisse `EclipseWorkspace\ontologies` und `EclipseWorkspace\wsdl` müssen dorthin kopiert werden. Fortan sind die enthaltenen Dateien mittels HTTP erreichbar, beispielsweise die *Top-Level*-Ontologie über `http://localhost/ontologies/top.owl`.

RacerPro 1.9

RacerPro [URL-26] ist ein Ontologie-*Reasoner*, der die DIG-Schnittstelle in der Version 1.1 implementiert, die vom Dienstverzeichnis-Prototyp verwendet wird. RacerPro wird über das mitgelieferte Installationsprogramm eingerichtet und ist danach sofort startbar und einsatzbereit. Die DIG-Schnittstelle kann bei laufendem Server über die Adresse `http://localhost:8080` angesprochen werden.

Apache Tomcat 5.5.20

Apache Tomcat [URL-33] ist der freie *Servlet Container* der Apache Software Foundation [URL-34]. Auch Tomcat lässt sich über sein Installationsprogramm installieren. Da Tomcat für seine Webanwendungen standardmäßig *port* 8080 verwendet, dieser *port* allerdings schon von RacerPro verwendet wird, muss bei der Installation 8000 als *port* angegeben werden. Alternativ kann nach bereits erfolgter Installation die Tomcat-Konfiguration angepasst werden. Sie befindet sich in der Datei `C:\Programme\Apache Software Foundation\Tomcat 5.5\conf\server.xml`. Im *Connector*-Element muss dazu der Wert des Attributs *port* auf den Wert 8000 angepasst werden. Wenn Apache Tomcat gestartet ist, kann er anschließend im Browser über `http://localhost:8000` aufgerufen werden.

jUDDI 0.9rc4

jUDDI [URL-35] ist eine UDDI-Implementierung, die auf einem beliebigen *Servlet Container* eingesetzt werden kann. Bei jUDDI gestaltet sich die Installation etwas komplizierter. Nach dem Auspacken der jUDDI-ZIP-Datei sind noch drei Schritte nötig: Einrichtung der Datenbank, Konfiguration der Webapplikation und Einsetzen der Webapplikation.

1. Zum Einrichten der Datenbank muss diese zunächst gestartet werden. Dann werden die SQL-Dateien aus dem Unterverzeichnis `sql\mysql` der jUDDI Verzeichnisstruktur in das MySQL-Unterverzeichnis `C:\Programme\xampp\mysql\bin` kopiert. Die Datei `insert_publishers.sql` muss editiert werden, so dass sie die Zeilen

   ```
   INSERT INTO PUBLISHER (PUBLISHER_ID, PUBLISHER_NAME,
   EMAIL_ADDRESS, IS_ENABLED,IS_ADMIN)
   VALUES ('juddi','juddi','juddi@localhost','true','true');
   ```

   enthält. Dadurch wird später ein Benutzer mit dem Namen und Passwort “juddi” angelegt, der die Rechte zum Publizieren im UDDI-Verzeichnis besitzt. Jetzt kann die Datenbank für jUDDI vorbereitet werden. Dazu müssen die beiden Dateien in MySQL ausgeführt werden. Dies geschieht durch die folgenden beiden Commandozeilenbefehle, die im `bin`-Verzeichnis von MySQL auszuführen sind:

```
mysql --user=root < create_database.sql
mysql --user=root < insert_publishers.sql
```

2. Zur Konfiguration der jUDDI-Webapplikation muss wieder die Tomcat-Konfiguration in der XML-Datei `C:\Programme\Apache Software Foundation\Tomcat 5.5\conf\server.xml` editiert werden. Innerhalb des *Host*-Elements wird der folgende *Context*-Abschnitt als Unterelement eingefügt.

```
<Context path="/juddi" docBase="juddi" debug="5"
         reloadable="true" crossContext="true">

      <Resource name="jdbc/juddiDB" auth="Container"
                type="javax.sql.DataSource" maxActive="10"
                maxIdle="5" maxWait="100"
                username="root" password=""
                driverClassName="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/juddi?autoReconnect=true"/>

</Context>
```

   Dadurch wird Tomcat mitgeteilt, wie er jUDDI veröffentlichen soll und welche Datenbankverbindung er jUDDI zur Verfügung stellen muss. Damit das funktioniert muss in Tomcat allerdings noch der JDBC-Treiber MySQL Connector/J 5.0.3 [URL-37] eingebunden werden. Dieser muss nach dem Herunterladen aus dem Internet oder Kopieren von der CD aus dem Verzeichnis `JavaBibliotheken\MySQLConnectorJ` zunächst ausgepackt werden. Anschließend kann man die im Paket enthaltene JAR-Datei in das Verzeichnis `C:\Programme\Tomcat55\common\lib` kopieren.

3. Die jUDDI-Webapplikation wird in Apache Tomcat eingesetzt, indem das Webarchiv `juddi.war` in das Verzeichnis `C:\Programme\Tomcat55\webapps` kopiert wird.

Sind alle Schritte erfolgreich erledigt, kann Tomcat mit Hilfe der Anwendung "Monitor Tomcat" gestartet und jUDDI unter der URL `http://localhost:8000/juddi/` aufgerufen werden. Weitere wichtige URLs sind `http://localhost:8000/juddi/happyjuddi.jsp` zur Überprüfung der jUDDI-Installation (hier dürfen keine roten Einträge existieren, sonst funktioniert jUDDI nicht richtig) sowie `http://localhost:8000/juddi/inquiry` und `http://localhost:8000/juddi/publish` zum Aufruf von UDDI-Operationen.

Dienstverzeichnis-Client-Plug-In

Nachdem alle Serverkomponenten installiert sind, hier noch ein kurzer Hinweis wie der Client in Form des Eclipse-*Plug-In*s in Betrieb genommen werden kann. Selbstverständlich muss zunächst Eclipse [URL-22] installiert werden. Dies geschieht durch Auspacken der Eclipse-ZIP-Datei nach `C:\Programme`. Das *plug-in* befindet sich auf der CD im Verzeichnis `Software\SeviceRegistryClient`. Die JAR-Datei, die sich dort befindet, muss in das Eclipse-Unterverzeichnis `C:\Programme\eclipse\plugins` kopiert werden. Das *plug-in* wird erst nach einem Neustart von Eclipse aktiviert, falls Eclipse bereits gestartet war.

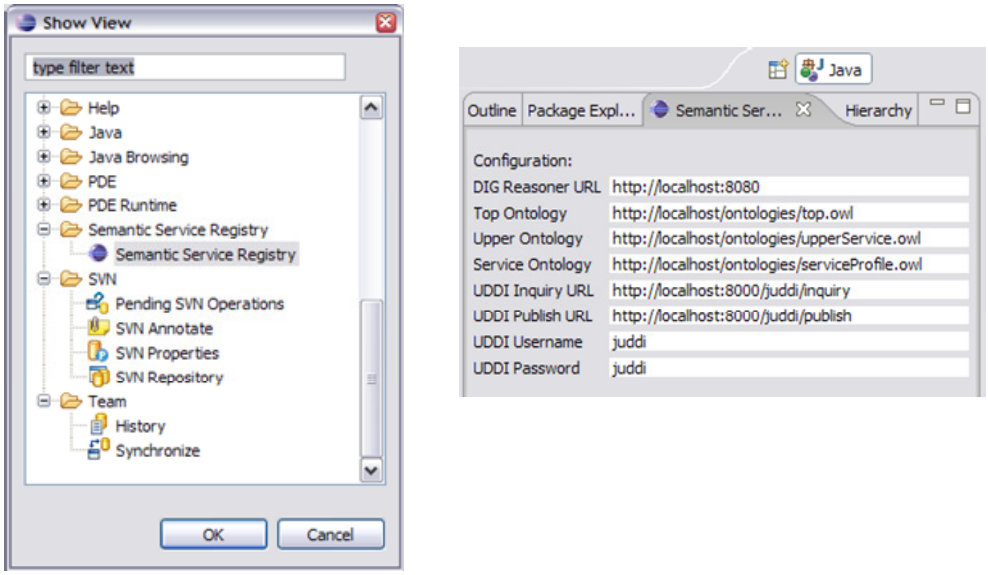
Betrieb

Dienstverzeichnis-Server

Der Server befindet sich in Betrieb, wenn alle Serverkomponenten erfolgreich gestartet sind. Dazu zählen MySQL, Apache Tomcat mit jUDDI, Apache HTTP und RacerPro.

Dienstverzeichnis-Client

Der Client kann benutzt werden, wenn die Eclipse-Entwicklungsumgebung mit installiertem *plug-in* ausgeführt wird. Dazu muss der Menüpunkt `Window→Show View→Other…` aufgerufen werden. Im erscheinenden Auswahldialog ist der Punkt `Semantic Service Registry` zu wählen, um die GUI des Clients anzuzeigen.



Information 73: Aufruf und Konfiguration des Dienstverzeichnis-Clients

Die Konfiguration des Clients erfolgt über die Eingabefelder in der GUI. Die standardmäßige Konfigurationsbelegung entspricht der obigen Beispielinstallation des Servers für den Fall, dass der Client auf demselben Rechner ausgeführt wird. Andernfalls muss `localhost` durch den jeweiligen Servernamen ersetzt werden.

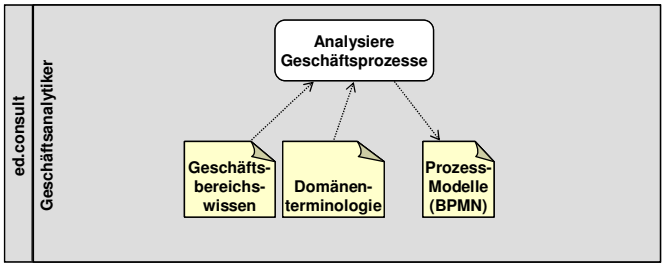
Bedienung

Die Bedienung des Dienstverzeichnis-Clients wird zusammen mit seiner Anwendung beim Programmieren im Großen in Kapitel 6.7 betrachtet.

6 ONTOLOGIEBASIERTES DIENSTORIENTIERTES ENTWICKLUNGSVORGEHEN

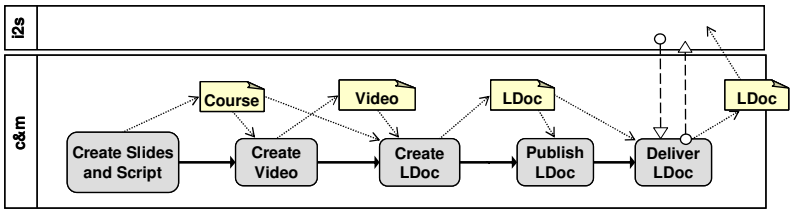
Im Kapitel zum übergeordneten Rahmen dieser Diplomarbeit (Kapitel 1.2) und in Information 1 wurden die zentralen Problembereiche, die bei der dienstorientierten Rechnerunterstützung der Aus- und Weiterbildung betrachtet werden sollen, dargestellt. Dabei wurden auch auf hohem Abstraktionsniveau die Tätigkeiten identifiziert, die zu einer zielgerichteten Befüllung des Dienstverzeichnisses führen. Es stellt sich die Frage wie an die dort genannten Aufgabenstellungen bei einer Arbeitsteilung im Team herangegangen werden kann. In diesem Kapitel soll nicht erläutert werden, welche eine optimale Methodik für den gemeinsamen Entwicklungsprozess wäre, es wird vielmehr ein Vorschlag gegeben, wie man anhand von den zu erstellenden Zwischenergebnissen zum Ziel kommen kann. Dieser Vorschlag kann dann vom Team in die Tat umgesetzt und als praktikabel verifiziert werden. Da der Fokus auf den zu erstellenden und weiterzuverarbeitenden Artefakten liegt, werden in den folgenden BPMN-Diagrammen keine kompletten Geschäftsprozesse skizziert, sondern lediglich einzelne Aktivitäten, die durchgeführt werden müssen, anhand ihrer ein- und ausgehenden Informationen erläutert.

6.1 Geschäftsprozessanalyse und -modellierung



Information 74: Geschäftsprozessanalyse und -modellierung

Am Anfang der Entwicklung der Rechnerunterstützung steht die Analyse des betrachteten Geschäftsbereichs. Diese manifestiert sich im Wesentlichen in zwei Blöcken: zum einen die Analyse der Geschäftsprozesse der beteiligten Akteure des Geschäftsbereichs und zum anderen die Entwicklung der Domänenontologie (Kapitel 4.5). Für die Erstellung einer serviceorientierten Anwendung sind vor allem die Arbeitsabläufe innerhalb eines Geschäftsbereichs von Interesse. Diese werden auch Geschäftsprozesse (*business processes* oder auch *workflows*) genannt. Ausgangspunkt bei der Analyse der Geschäftsprozesse ist das Wissen über den betrachteten Geschäftsbereich, das bereits niedergeschrieben in Dokumentation über Vorgänge und Vorschriften innerhalb des Bereichs vorliegen kann oder gegebenenfalls durch Interviews mit beteiligten Personen erfasst werden muss. Weiteren Einfluss auf die Analyse der Geschäftsprozesse und deren Ergebnisse hat die Domänenterminologie, ein allgemein anerkannter Satz von Begriffsfestlegungen, der alle in der Domäne auftretenden relevanten Bezeichnungen definiert. Dieser wird parallel zur (oder vor der) Prozessanalyse in Form einer Domänenontologie erarbeitet. Als Ergebnisartefakt der Geschäftsprozessanalyse sind Geschäftsprozessmodelle vorgesehen. Zur Modellierung der Geschäftsprozesse gibt es unterschiedliche Methoden, Notationen und Werkzeuge, sodass hier wie folgt eine Auswahl stattfinden muss. Während als Notation aus dem Kontext der Forschungsgruppe heraus die BPMN angedacht ist und das methodische Vorgehen auf Vorarbeiten anderer Diplomanden [Ba05a] [Ru06] gestützt werden kann, könnte die Wahl eines geeigneten Modellierungswerkzeuges durch Effizienz motiviert werden: Wie kommt man später möglichst einfach vom Modell zu einem funktionstüchtigen Softwaresystem? Konkret stellt sich hier die Frage, inwieweit potenziell geeignete Werkzeuge den Export des Prozessmodells in eine ausführbare Prozessdefinition für eine im Prototyp einzusetzende Laufzeitumgebung beherrschen.

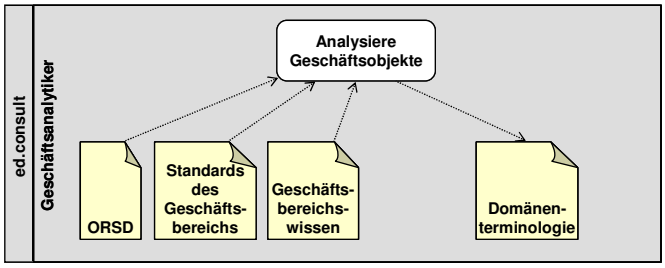


Information 75: Beispielhaftes Geschäftsprozessdiagramm

Als beispielhafter Ausschnitt aus den Ergebnissen der Geschäftsprozessanalyse kann das BPMN-Diagramm aus Information 75 dienen. Es zeigt eine erste Version des in Kapitel 2.3.2 vorgestellten Geschäftsprozesses mit dem Namen *Provide Courseware*.

6.2 Entwicklung der Domänenontologie

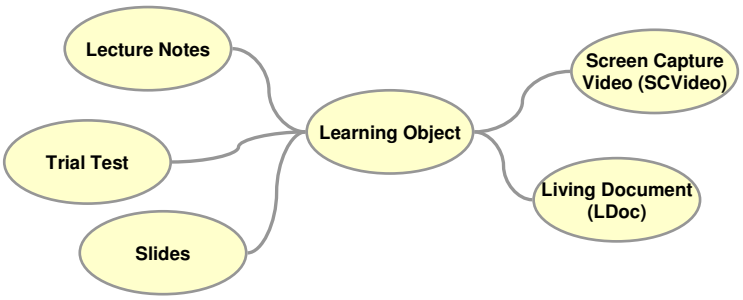
Die Entwicklung einer Terminologie für die Aus- und Weiterbildung wird zeitgleich mit der Geschäftsprozessanalyse in Angriff genommen, um soweit möglich von vornherein eine begriffliche Konsistenz der Prozessmodelle mit der betrachteten Domäne herzustellen und anerkannte Begriffe aus bereits gebräuchlichen Begriffsdefinitionen in die Prozessmodellierung einfließen zu lassen. Dabei spielen die Gegenstände und Artefakte, mit denen im Geschäftsbereich umgegangen wird, die so genannten Geschäftsobjekte, eine wichtige Rolle, da sie ein wesentliches Element der Dienstbeschreibung sind. Später werden die erhaltenen Fachwörter mit Informationen über ihre Bedeutung (Semantik) angereichert, indem sie in Klassen eingeteilt und diese in eine hierarchische Beziehung gesetzt werden. Man spricht dann von einer Klassifizierung der Begriffe oder auch von einer Taxonomie. Für die semantische Dienstbeschreibung wird diese als OWL-Ontologie kodiert und bereitgestellt.



Information 76: Entwicklung der Domänenontologie (Kickoff)

Zur Entwicklung der Ontologie soll ein angepasstes Verfahren des Knowledge Meta Process [SS+04a] Anwendung finden. Dieses Verfahren teilt die eigentliche Ontologieextraktion auf die beiden Schritte *Kickoff* und *Refinement* auf. Während des *Kickoff* (Information 76) wird eine initiale (semi-formale) Ontologie erstellt, die gerade die in die Ontologie aufzunehmenden Begriffe wiedergibt. Dieses Artefakt lässt sich auch als Domänenterminologie bezeichnen und muss im Falle einer zu erstellenden Domänenontologie für die Dienstbeschreibung in der Hauptsache die Bezeichnungen der Geschäftsobjekte der Domäne enthalten (erste Anforderung aus Kapitel 4.5.1). Als Ausgangsmaterial dienen bei der Erstellung das bereits bei der Prozessanalyse angesprochene Geschäftsbereichswissen sowie möglicherweise zu berücksichtigende Standards, die Teile des Geschäftsbereichs abdecken. Welche Quellen das genau sind, ist im zuvor zu erstellenden Ontology Requirements Specification Document festzuhalten, ebenso wie sonstige Anforderungen an die Ontologie.

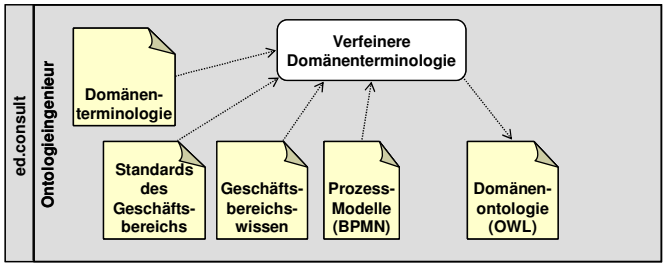
Die Ergebnisse dieser Analyse werden semi-formal notiert. Dazu können beispielweise einfache ungerichtete Graphen, *mind maps* oder ein Glossar verwendet werden. Ein stark eingegrenzter Ausschnitt einer solchen Domänenterminologie ist in Information 77 abgebildet. Er beinhaltet den Begriff *Learning Object* und einige weitere Konzepte, die möglicherweise mit *Learning Object* in Verbindung stehen.



Information 77: Beispielhafte Domänenterminologie

Nach diesem initialen Schritt (wenn nicht schon unterdessen) kann ein Abgleich mit den parallel modellierten Geschäftsprozessen stattfinden, indem diese einerseits soweit nötig an die Domänenterminologie angepasst und andererseits während des *Refinement* der Ontologie berücksichtigt werden.

Im oben begonnenen Beispiel (Information 75 und Information 77) stellt sich beim Abgleich heraus, dass die Bezeichnungen *Script* und *Video* im Geschäftsprozessdiagramm unglücklich gewählt wurden und zur Vereinheitlichung stattdessen *Lecture Notes* und *SCVideo* gewählt werden sollten. Auch für die Initialontologie gibt es neue Erkenntnisse. Im Prozessdiagramm taucht der Begriff *Course* auf, für den sich in der Domänenterminologie kein entsprechendes Konzept finden lässt. Nach gemeinsamer Diskussion der Beteiligten wurde beschlossen, dass der Begriff missverständlich ist, durch *Courseware* ersetzt und in die Ontologie aufgenommen werden muss. Damit ergibt sich auf der Seite des Geschäftsprozesses das aus Information 17 bekannte Diagramm.



Information 78: Entwicklung der Domänenontologie (Refinement)

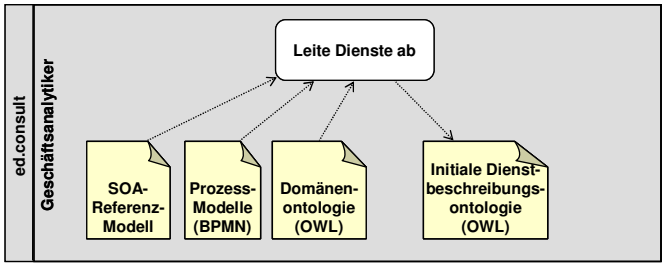
Während des *Refinement* wird die Domänenterminologie um fehlende Einträge ergänzt und durch Anreicherung mit weiteren Informationen wie beispielsweise Hierarchienbildung und weiteren Beziehungen unter den Konzepten ausgebaut. Dann wird sie mit einem passenden Werkzeug erfasst und kodiert, so dass eine Domänenontologie im OWL-Format entsteht. Die Artefakte, die in diesem Schritt benötigt und erstellt werden, zeigt Information 78. Wie ein mögliches Ergebnis in UML-Darstellung aussehen kann, wurde bereits in Information 23 vorgestellt.

6.3 Ableitung der Dienste

Das Interesse gilt nun der Erstellung einer serviceorientierten Anwendung aus den Ergebnissen der Geschäftsprozessmodellierung heraus. Um eine solche zu verwirklichen, werden die Beschreibungen jener Dienste benötigt, die für die Geschäftsprozesse charakteristisch sind. Es muss also ermittelt werden, welche Dienste zur Realisierung der Geschäftsprozesse benötigt werden, in welche Ebenen des SOA-Referenzmodells [C&M-P] von C&M diese jeweils einzuordnen sind und wie die Dienste zur Realisierung der Prozesse zu kombinieren sind. Die

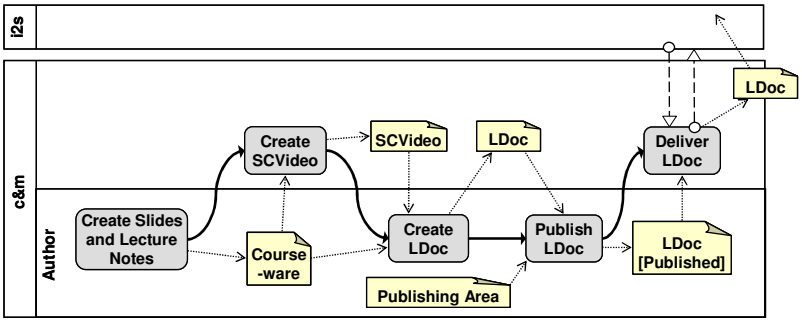
Beschreibungen der Dienste sollen letztendlich in einem Dienstverzeichnis strukturiert abgelegt werden, so dass sie später einfach und flexibel wiederverwendet werden können (Kapitel 6.7).

Für die Ableitung der Dienste, basierend auf den Geschäftsprozessmodellen und dem SOA-Referenzmodell, kann ebenfalls auf die bereits entwickelten Techniken der Forschungsgruppe zurückgegriffen werden [Ba05a] [Ru06]. Vereinfacht kann das Spezifizieren der Dienste oder Webservices dadurch werden, dass bereits bei der Modellierung der Geschäftsprozesse der Abstraktionsgrad und die modellierten Abläufe SOA-gerecht gewählt werden. Ein solches Vorgehen vermeidet einen Bruch zwischen den Geschäftsprozessmodellen und den abgeleiteten Diensten dadurch, dass eine Abbildung zwischen Aktivitäten im BPMN-Diagramm und Diensten dergestalt ermöglicht wird, dass jede automatisierbare Aktivität komplett durch einen Dienst erbracht werden kann.



Information 79: Ableitung der Dienste

Um bei der Dienstableitung und anschließenden -beschreibung Mehraufwand zu sparen, können die identifizierten Dienste direkt in einer initialen Form in die Dienstbeschreibungsontologie aufgenommen werden. Um ein pragmatisches Vorgehen bei der Dienstableitung zu ermöglichen, ist es sinnvoll Mindestvoraussetzungen an die BPMN-Diagramme zu stellen. Es ist hilfreich, wenn die bei der Dienstbeschreibung benötigten Aspekte in den BPMN-Diagrammen modelliert sind oder die BPMN-Diagramme wo erforderlich erweitert werden. Zumindest die Informationen zu den eingehenden und ausgehenden Geschäftsobjekten und deren Zustände sowie die organisatorische Einheit, die die jeweiligen Aktivitäten durchführt, wären angebracht.



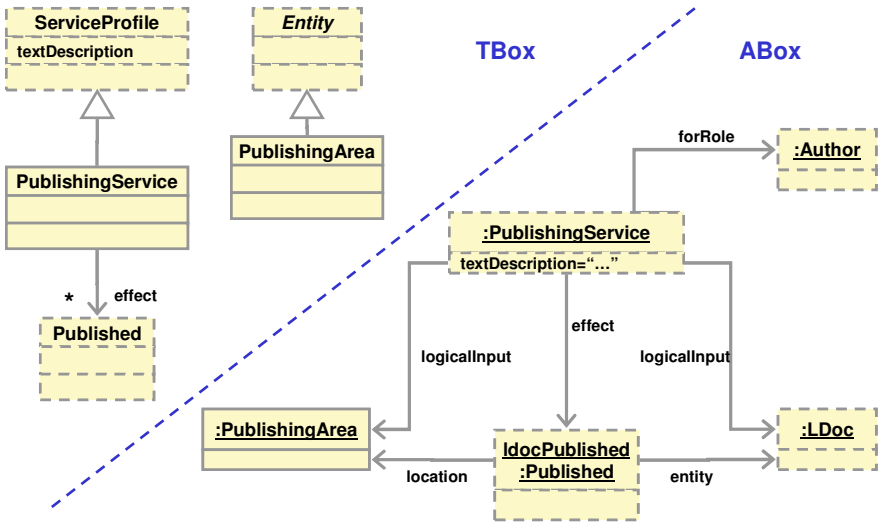
Information 80: Ausführliches BPMN-Diagramm

Wie ein solches ausführliches BPMN-Diagramm aussehen könnte, zeigt Information 80. Im Gegensatz zum Originaldiagramm aus Information 17 enthält es zusätzliche Informationen über den Zustand des LDoc nach der Publizierung, den Ort der Publizierung (*PublishingArea*) und über die Rolle des Autoren, die einige der modellierten Aktivitäten durchführt.

6.4 Entwicklung der initialen Dienstbeschreibungsontologie

Die Entwicklung der Dienstbeschreibungsontologie läuft in drei Stufen ab, die die Phasen zwei bis vier des Verfahrens nach [KK03] (Kapitel 3.1.2) abdecken. Im Gegensatz zu diesem Verfahren werden die ersten beiden Stufen jedoch anders aufgeteilt. Anstatt streng zwischen

Erstellung der Dienstbeschreibungsontologie und Erweiterung derselben um Referenzen auf Domänenontologien zu unterscheiden, ist es zweckmäßig eine Aufteilung nach Abstraktionsgrad vorzunehmen und auch im ersten Schritt schon die Konzepte der Domänenontologie zu berücksichtigen. Die erste Stufe kann wie in Information 79 angedeutet bereits während der Ableitung der Dienste durchgeführt werden und führt zu einer initialen Dienstbeschreibungsontologie. Diese initiale Ontologie enthält bereits die Kategorisierung des Dienstes als *PublishingService*, ein entsprechende Instanz und alle Informationen über den Dienst, die aus den BPMN-Diagrammen abgeleitet werden können. Dies sind in der Regel die logischen Ein- und Ausgaben sowie die wichtigsten Zustände und eventuell die Rolle, für die der Dienst gedacht ist. Auch eine textuelle Beschreibung sollte bereits hier eingefügt werden. Information 81 zeigt die initiale Dienstbeschreibungsontologie für die LDoc-Publizierungsdienst mit ihren ABox- und TBox-Anteilen.

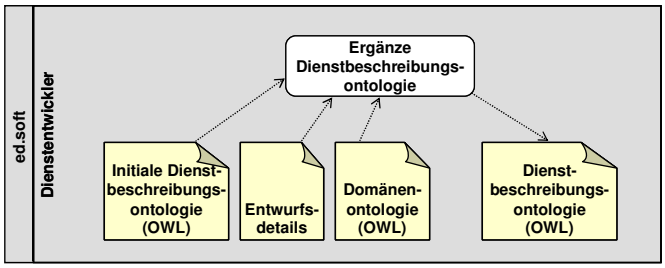


Information 81: Initiale Dienstbeschreibungsontologie

Eine interessante Rolle spielt in diesem Beispiel das Konzept *PublishingArea*. Da es in der Domänenontologie nicht vorhanden ist, muss es neu erstellt werden. Hier muss eine Abwägung getroffen werden, ob die Domänenontologie zu erweitern ist oder eine Klasse in der Dienstbeschreibungsontologie erstellt werden soll. Prinzipiell sollten Konzepte, die Geschäftsobjekte repräsentieren, eher in die Domänenontologie aufgenommen werden. Ein Änderungsantrag ist entsprechend zu stellen. Eine zu starke Belastung der Domänenontologie mit Details erhöht allerdings die Komplexität und vermindert die Übersichtlichkeit und Wiederverwendbarkeit. Im gezeigten Fall wurde festgestellt, dass sich unter *PublishingArea* ein Detail der Dienstimplementierung verbirgt und daher beschlossen, eine entsprechende Klasse in die Dienstbeschreibungsontologie aufzunehmen.

6.5 Ergänzung der Dienstbeschreibungsontologie

In der zweiten Stufe wird die initiale Beschreibung der funktionalen Dienstaspekte in der Ontologie vervollständigt (Information 82). Es müssen die funktionalen Merkmale ergänzt werden, die zur Zeit der Erstellung der initialen Ontologie noch nicht bekannt waren oder nach festgestellter Notwendigkeit unterdessen in die Domänenontologie aufgenommen wurden. Ebenso ist es erforderlich sich Gedanken über die Entwicklung der Dienste zu machen. Mit der Beschreibung der tatsächlichen Ein- und Ausgabeparameter, die in diesem Schritt vorgenommen werden soll, fließen bereits die ersten Entwurfs- und eventuell Implementierungsdetails des Dienstes beziehungsweise seiner Schnittstellendefinition ein. Dies führt zu einer ersten Eingrenzung der möglichen Implementierung.



Information 82: Ergänzung der Dienstbeschreibungsontologie

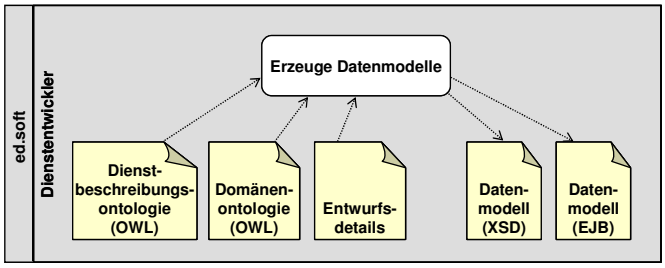
Wieder wird für jeden Ein- oder Ausgabeparameter eine Klasse aus der Domänenontologie herangezogen oder falls erforderlich in der Dienstbeschreibungsontologie erstellt. Diese Klassen werden instantiiert und über passende (eventuell zu erstellende) Zustände oder sonstige Relationen mit den bereits vorhandenen Instanzen der Dienstbeschreibung, in der Regel mit den logischen Ein- und Ausgaben oder den Vor- und Nachbedingungen, verknüpft. Als Ergebnis dieser Stufe erhält man eine Ontologie, die alle Teile der fertigen Dienstbeschreibungsontologie aus Information 50 und Information 51 enthält.

6.6 Realisierung von Diensten

Bei der Realisierung von Diensten können neben dem softwaretechnischen Entwurf und der Implementierung zwei wesentliche Teilziele identifiziert werden. Dies sind zum einen die Modelle der Daten, die beim Dienstaufruf transferiert werden, und zum anderen die fertige Dienstinstanz. Die folgenden Unterkapitel zeigen auf, wie der Einsatz der Dienstbeschreibungs- und Domänenontologie bei der Erlangung dieser Zwischenergebnisse aussehen könnte.

6.6.1 Erzeugung von Datenmodellen

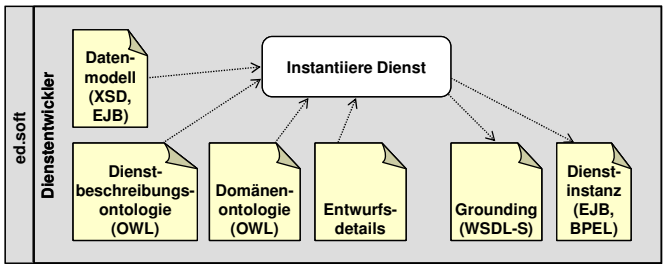
Die zu realisierende Dienstinstanz muss beim Aufruf Ein- und Ausgaben durch ihre Kommunikationsschnittstelle übermitteln. Dazu werden Datenmodelle beziehungsweise (möglicherweise abstrakte) Datentypen benötigt, die die Struktur der übertragenen Daten festlegen. Wenn auch wünschenswert, ist es momentan noch nicht möglich, dass die Dienstimplementierung die in der Ontologie festgelegte Typenspezifikation direkt als Datenmodell für ihre Kommunikation nutzen kann. Es müssen passende Datenmodelle für die jeweilige Implementierung erstellt werden.



Information 83: Erzeugung der Datenmodelle

Bei Kenntnis der nötigen Entwurfs- und Implementierungsdetails des zu implementierenden Dienstes können Datenmodelle aus den Informationen aus der Domänenontologie und Dienstbeschreibungsontologie abgeleitet werden. Für spätere Verwendung im WSDL-S-Dokument werden XML-Schema-Definitionen benötigt. In Richtung der Implementierung ist es zudem auch denkbar, die Quellcoderümpfe von Enterprise Java Beans (EJB) zu erzeugen. Zukünftig könnte mit entsprechenden Werkzeugen die Erstellung solcher Datenmodelle eventuell teilautomatisiert ablaufen.

6.6.2 Instantiierung der Dienste



Information 84: Instantiierung der Dienste

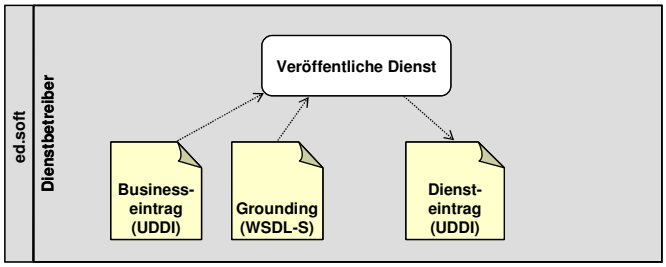
Um den Dienst in Betrieb nehmen zu können, muss eine Instanz des Webservice erzeugt werden. Diese kann unterschiedlich realisiert werden. So ist es zum Beispiel möglich, die Dienstimplementierung durch EJB- oder .NET-Komponenten zur Verfügung zu stellen oder aber durch eine BPEL-Verschaltung vorhandener Dienste. Unabhängig davon wird ein WSDL-S-*grounding* erstellt, um die syntaktischen Schnittstelleninformationen mit ihrer semantischen Annotierung zu versehen. Wie ein solches WSDL-S-Dokument aussieht, wurde bereits in Information 52 gezeigt.

6.7 Nutzung und Nutzen des Dienstverzeichnisses

Anknüpfend an die vorangegangenen Unterkapitel 6.1 bis 6.6, soll nun der Bogen zurück zum Demonstrator und seiner Beschreibung in Kapitel 1.5 gespannt werden. Es soll aufgezeigt werden, dass das ursprüngliche Vorhaben umgesetzt werden konnte. Dazu folgt eine Beschreibung des Einsatzes des Dienstverzeichnis-Prototyps bei der Programmierung im Großen, wobei auch auf die Bedienung der Funktionen des Dienstverzeichnis-Clients eingegangen wird.

6.7.1 Veröffentlichung der Dienste

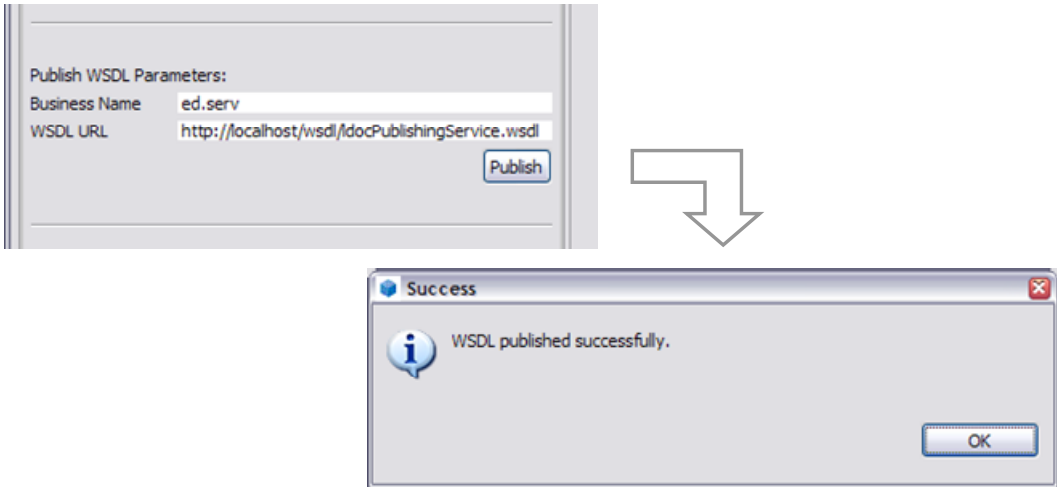
Dienste, die realisiert und in Betrieb genommen wurden, können im Dienstverzeichnis veröffentlicht werden. Wie Information 85 zeigt, wird die Dienstinstanz dazu gar nicht benötigt. Das bedeutet, dass zum Beispiel zu Testzwecken auch Dienste veröffentlicht werden können, die sich gar nicht in Betrieb befinden. Es muss lediglich ihr *grounding* vorliegen. Für die folgenden Demonstrationen wurde genau dies getan, da die hier verwendeten Beispieldienste bisher nur beschrieben und noch nicht umgesetzt wurden.



Information 85: Veröffentlichung der Dienste

Für die Veröffentlichung bietet der Dienstverzeichnis-Client eine einfache Oberfläche mit zwei Eingabefeldern an (Kapitel 5.3.1). Ein Beispiel für die Anwendung gibt Information 86. Darin werden nach einem Klick auf die *Publish*-Schaltfläche alle in der WSDL-Datei `http://localhost/wsdl/ldocPublishingService.wsdl` enthaltenen Dienste unter der *businessEntity* mit dem Namen ed.serv veröffentlich. Hinter dieser Datei verbirgt sich bei der Standardinstallation des Prototyps gerade der LDoc-Publizierungsdienst aus Kapitel 2.3.3.

Damit die Veröffentlichung so erfolgreich ablaufen kann wie gezeigt, muss der Client allerdings je nach Installation richtig konfiguriert sein (Kapitel 5.3.3).



Information 86: Publizierungsfunktion des Dienstverzeichnis-Clients

Auf der beigelegten CD befinden sich im Verzeichnis `EclipseWorkspace\wsdl` noch weitere WSDL-S-Dateien mit Diensten, die auch in der mitgelieferten Dienstbeschreibungsontologie spezifiziert sind, darunter auch der Auslieferungsdienst für Lernobjekte aus Kapitel 2.3.3. Die Dienste lassen sich auf gleiche Art und Weise wie eben beschrieben veröffentlichen.

6.7.2 Dienstsuche und Einbindung

Zur Vorführung der Dienstsuche werden selbstverständlich Dienste im Verzeichnis benötigt.

Service Name	Service Category	User Roles	Inputs	Outputs	Preconditions	Effects
ldocCreationService	AuthoringService	Author	Slides, SCVideo, LectureNotes	LDoc		
courseMaterialMailing Service	DeliveryService	Provider	CourseMaterial, Learner	CourseMaterial	Published	Mailed
exerciseDelivery Service	DeliveryService	Learner	Exercise	Exercise	Published	Transferred
learningObjectDelivery Srevice	DeliveryService	Learner	LearningObject	LearningObject	Published	Transferred
scheduleDelivery Service	DeliveryService	Learner	Schedule	Schedule	Published	Transferred
ldocPublishingService	PublishingService	Author	PublishingArea, LDoc			Published
lectureNotesPublishing Service	PublishingService	Author	PublishingArea, LectureNotes			Published
slidesPublishingService	PublishingService	Author	PublishingArea, Slides			Published

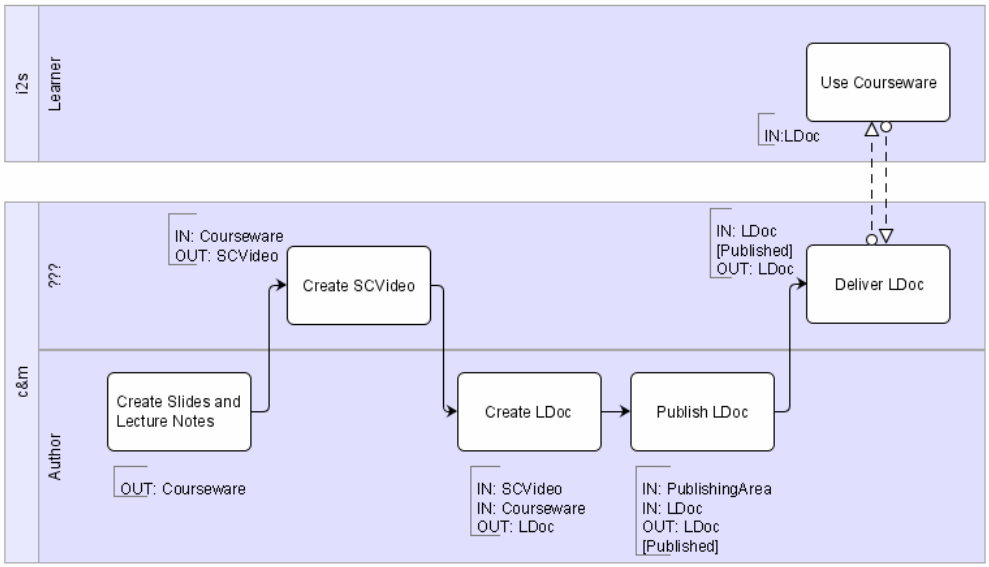
Information 87: Dienstbestand für die Vorführung

Die folgenden Beispiele beruhen auf einem Verzeichnis, in dem alle acht auf der CD mitgelieferten Dienste wie oben beschrieben veröffentlicht wurden. Die folgenden Beispiele sind daher mit der Standardinstallation des Prototyps nachvollziehbar. Die Beispieldienste sind aus den Kategorien Autorendienste, Auslieferungsdienste und Publizierungsdienste. Eine Aufstellung inklusive der verknüpften semantischen Konzepte ist in Information 87 dargestellt.

Zum Zwecke der Veranschaulichung wird das BPMN-Modellierungswerkzeug Intalio Designer [URL-38] verwendet. Dieses Eclipse-*Plug-In* ermöglicht die Erstellung von BPMN-Diagrammen, die Einbindung von Webservices und, was hier jedoch nicht weiter betrachtet wird, die spätere Übersetzung der BPMN-Diagramme in BPEL-Code, der auf den zugehörigen

Intalio Server übertragen und dort ausgeführt werden kann. Das Werkzeug richtet sich daher an Prozessanalytiker, die Geschäftsprozessmodellierung zum Programmieren im Großen nutzen.

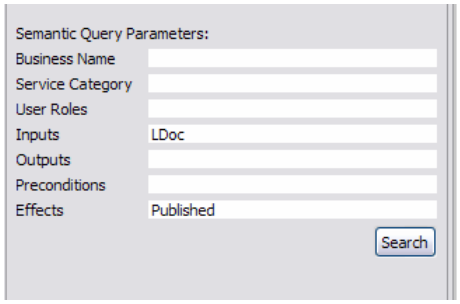
Genau für diese Nutzung ist auch der Dienstverzeichnis-Client ausgelegt. Ein grundlegender Nutzungsprozess wurde bereits in Information 3 dargelegt. Darin wird die Dienstsuche erst nach der initialen Prozessmodellierung verwendet, wovon auch im Folgenden ausgegangen wird. Die Suche ließe sich alternativ auch schon direkt während der Modellierung nutzen.



Information 88: Der Provide-Courseware-Prozess in Intalio Designer

Information 88 zeigt den *ProvideCourseware*-Prozess aus den Kapiteln 1.5 und 6.3 als *screen shot* in Intalio Designer. Da Intalio Designer die Modellierung von Datenobjekten nicht unterstützt (sie sind laut Hersteller irrelevant für die Generierung von BPEL-Code), wurden ein- und ausgehende Geschäftsobjekte in den Beispielen als Kommentare modelliert.

Wenn nun die Aktivität *Publish LDoc* durch einen IT-Dienst realisiert werden soll, kann man mit Hilfe der implementierten Dienstsuche passende Kandidaten ausfindig machen. Der Dienstverzeichnis-Client bietet dazu zwei GUI-Elemente an. Das erste ist zur Erfassung der Suchparameter gedacht und in Information 89 abgebildet. Gemäß dem Entwurf des Dienstverzeichnis-Prototyps gibt es darin Texteingabefelder für die verschiedenen Eigenschaften der Dienste. Analog zu Kapitel 1.5 wird nun nach Diensten mit der Klasse *LDoc* als Eingabe und der Klasse *Published* als Nachbedingung gesucht. Ein Klick auf die Schaltfläche *Search* startet die Suchfunktion. Sollten Namen eingegeben worden sein, die nicht zu Klassen aus der Ontologie gehören, so wird die Suche dennoch ausgeführt, wobei die entsprechenden Werte ignoriert werden. Auf diesen Umstand weist der Client durch einen Informationsdialog hin.



Information 89: Suche nach einem LDoc-Publizierungsdienst

Das zweite GUI-Element, das bei der Suche eine wesentliche Rolle spielt, ist die Ergebnisliste (Information 90). Sie erscheint im Editorbereich von Eclipse und enthält alle zu den Suchparametern passenden Dienste, die anhand des Suchalgorithmus aus Kapitel 5.2.3 gefunden wurden. Im gegebenen Beispiel waren die semantischen Suchparameter ausreichend, um zu genau einem Treffer zu führen. Der gefundene Dienst ist der *LDocPublishingService*, wie der Zeile *Service Profile* entnommen werden kann. Die Anzeige der Suchergebnisse erfolgt tabellarisch und enthält außer dem *Service Profile* und dessen Beschreibung noch die URL der zugehörigen WSDL-Datei und alle Diensteigenschaften, die aufbauend auf der Dienstbeschreibungs-Ober-Ontologie spezifiziert wurden.

Search Results

1 Search Result

#1	Relevanz 0,632
Description:	This service can be used to publish a LDoc in the courseware repository, which will then be available for retrieval in the given Publishing Area.
Business Name:	ed.serv
Service Category:	PublishingService
User Roles:	Author
Inputs:	LDoc, PublishingArea
Outputs:	
Preconditions:	
Effects:	Published
Service Profile:	http://localhost/ontologies/serviceProfile.owl#ldocPublishingService
WSDL-S Location:	http://localhost/wsdl/ldocPublishingService.wsdl

Information 90: Suchergebnisse für den LDoc-Publizierungsdienst

Information 91 zeigt als Alternative die Ergebnisse bei einer Suche mittels UDDI. Diese wurde mit dem Web Services Explorer der Eclipse Web Tools Platform [URL-39] durchgeführt.

(1) **%Published%**
Keine Ergebnisse

(2) **%Publish%**

Name	Description
ldocPublishingService	This service can be used to publish a LDoc in the courseware repository, which will then be available for retrieval in the given Publishing Area.
lectureNotesPublishingService	This service can be used to publish LectureNotes in the courseware repository, which will then be available for retrieval in the given Publishing Area.
slidesPublishingService	This service can be used to publish Slides in the courseware repository, which will then be available for retrieval in the given Publishing Area.

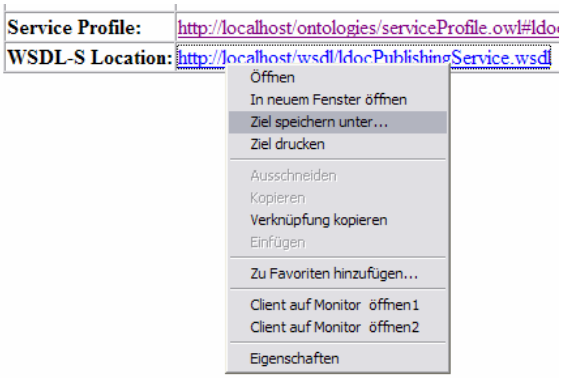
(3) **%LDoc%**

Name	Description
ldocCreationService	This service takes Lecture Notes a SCVideo and Slides and creates a LDoc.
ldocPublishingService	This service can be used to publish a LDoc in the courseware repository, which will then be available for retrieval in the given Publishing Area.

Information 91: UDDI-Suchergebnisse für den LDoc-Publizierungsdienst

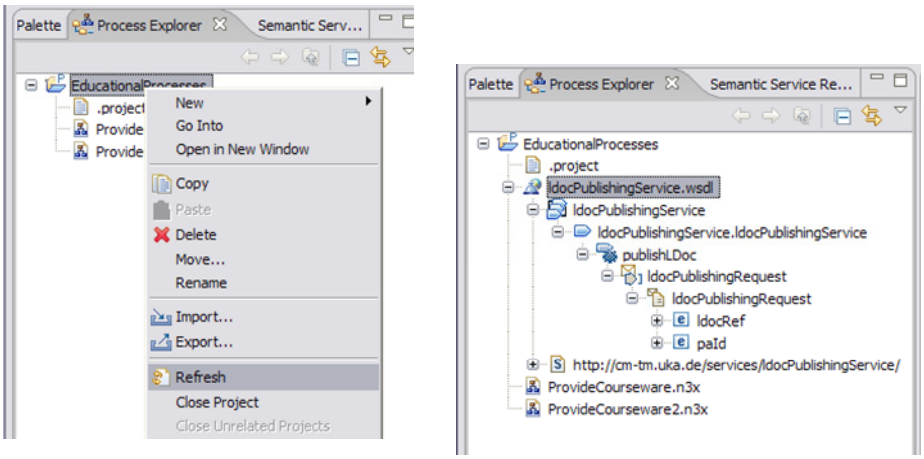
Es wurde die einfache Suche verwendet, was bedeutet, dass lediglich in den Namen der Webservices gesucht wird. Eine Suche, die technische Details beinhaltet, ist in UDDI per Hand nur sehr schwierig durchführbar und faktisch nicht möglich, wenn die Webservices wie im vorgestellten Fall nach den *best practices* von OASIS [CJ02a] veröffentlicht wurden, da dabei außer dem Ort der WSDL-Datei und den Zugriffs-URLs keine weiteren Informationen ins UDDI übertragen werden. Generell basiert die Suche bei UDDI auf Vergleichen von Zeichenketten. Die verwendeten Suchparameter sind mit %-Zeichen umrahmt, damit sie an beliebiger Stelle in den Namen der Webservices gefunden werden und nicht nur am Anfang. Bemerkenswert ist, dass die UDDI-Suche für den gegebenen Fall entweder eine größere Ergebnismenge oder gar kein Ergebnis liefert. Schon bei den acht Diensten der Beispielinstallation werden bei einem schlichten Suchbegriff wie "Publish" zwei unpassende Dienste gefunden. Bei einem Verzeichnis mit etlichen hundert Diensten würde die Ergebnismenge ungleich größer ausfallen. Die Eingabe des Suchbegriffs "LDocPublish" hätte bei der UDDI-Suche zu einem direkten Treffer geführt, "PublishLDoc" jedoch zu keinem Ergebnis. Die Verwendung der semantischen Suche führt hingegen zu einer klaren abgegrenzten Ergebnisliste mit dem relevanten Dienst, da hierbei durch die unterschiedlichen Suchparameter direkt nach einzelnen funktionalen Eigenschaften des Dienstes gesucht werden kann.

Dieser soll nun eingebunden werden. Dazu ist der Ort der WSDL-Datei im Ergebnis mit einem Hyperlink unterlegt, so dass die Datei wie üblich über Rechtsklick und "Ziel speichern unter…" im Arbeitsbereich von Intalio Designer abgelegt werden kann (Information 92).



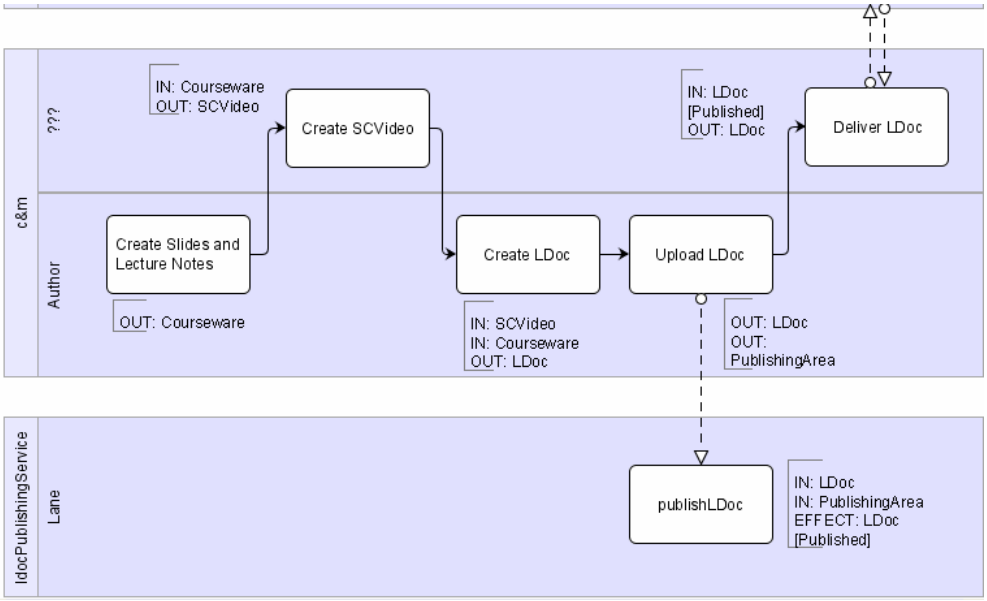
Information 92: Speichern der WSDL-S-Datei

In der aktuellen Version des Dienstverzeichnis-Clients wird die Datei dann leider noch nicht direkt von Intalio Designer angezeigt. Es ist zunächst erforderlich, die Inhalte des *Process Explorer* zu erneuern, was mit dem Menüpunkt *Refresh* aus dem Kontextmenü des Projektes erreicht wird (Information 93). Anschließend wird die WSDL-Datei in den *Explorer*-Baum eingeblendet und kann ausgeklappt werden, so dass ihre Inhalte, wie unter anderem Webservices und Operationen, sichtbar werden.



Information 93: WSDL-Datei in Intalio Designer anzeigen

Die Einbindung des Webservice, der durch die WSDL-Datei beschrieben wird, ist bei Intalio Designer auf einfache Weise möglich. Dazu muss zunächst das *Service*-Element aus dem WSDL-Baum im *Process Explorer* mittels *drag'n'drop* in das Diagramm gezogen werden. Es erscheint ein neuer Pool, der den Titel des Webservice trägt. Anschließend kann man auf gleiche Weise die gewünschte Operation des Dienstes im Diagramm hinzufügen. Sie erscheint dort als Aktivität und kann durch Nachrichtenfluss mit anderen Aktivitäten des Diagramms in Verbindung gesetzt werden. Das Ergebnis zeigt Information 94.



Information 94: Einbindung des Publizierungsdienstes in Intalio Designer

Die zweite Suche in Kapitel 1.5 gilt einem Auslieferungsdienst für LDocs, die als nächstes nachvollzogen wird. Diesmal wird zunächst davon ausgegangen, dass zur Veröffentlichung der Webservices ausschließlich das Standardvorgehen mit WSDL und UDDI gewählt wurde und nur die schlichte UDDI-Suche zur Verfügung steht. Es muss also ein passender Suchbegriff gefunden werden. Der erste Versuch "LDocDelivery" und der zweite "DeliverLDoc" führen zu keinem Ergebnis. Der dritte"LDoc" führt zwar zu zwei Ergebnissen, die aber beide nicht brauchbar sind. Alle Suchbegriffe, die "Published" enthalten führen ebenfalls zu keinem Ergebnis. Der Suchbegriff "Delivery" schließlich führt zu einer Liste mit drei Diensten, wovon einer der eigentlich gesuchte *LearningObjectDeliveryService*. ist (Information 95).

(1) **%LDoc%**

Name	Description
IdocCreationService	This service takes Lecture Notes a SCVideo and Slides and creates a LDoc.
IdocPublishingService	This service can be used to publish a LDoc in the courseware repository, which will then be available for retrieval in the given Publishing Area.

(2) **%Delivery%**

Name	Description
exerciseDeliveryService	This service can be used by a client to fetch a exercise that is available to the public. You can use this as a means for provision of downloads for example.
learningObjectDeliveryService	This service can be used by a client to fetch a learning object that is available to the public. You can use this as a means for provision of downloads for example.
scheduleDeliveryService	This service can be used by a client to fetch a schedule that is available to the public. You can use this as a means for provision of downloads for example.

(3) **%Published%**
Keine Ergebnisse

Information 95: UDDI-Suchergebnisse für den LDoc-Auslieferungsdienst

Generell liegt wieder das Problem vor, dass der gewählte Suchbegriff Bestandteil des Namens des Webservice sein muss. Daher laufen viele Suchbegriffe in diesem Fall ins Leere. Beim Suchbegriff *Delivery* liegt das gleiche Problem vor wie bei *Publish* bei der letzten Suche. Er beschreibt eine ganze Kategorie von Diensten. Bei einer solchen Such- und Dienstkonstellation offenbart die semantische Dienstsuche ihr volles Potential. Jetzt kommt es nicht wie bei der letzten Suche nur mehr auf die Diversität aufgrund der unterstützten Suchparameter an, sondern tatsächlich auf die semantische Information, die hinter den eingegebenen Klassennamen steckt. Mit Hilfe der durch die Ontologie einheitlich festgelegten Begriffe lassen sich die Parameter als die Namen der Klassen *DeliveryService* für die Dienstkategore, *LDoc* für die Eingabe und *Published* für die Vorbedingung ausmachen. Da *LearningObject* eine Oberklasse der gesuchten Eingabe *LDoc* ist, wird der richtige Dienst gefunden, sobald die Suchunschärfe Konformität für die Ein- und Ausgaben zulässt (Information 96). Interessanterweise taucht in der Ausgabe noch ein weiterer Dienst auf. Da *CourseMaterial* ebenfalls Oberklasse von *LDoc* ist, wurde auch der *CourseMaterialMailingService* gefunden. Er hat gegenüber dem ersten Dienst eine geringere Relevanz, da er zwei Eingabeparameter spezifiziert, der erste Dienst und die Suchanfrage jedoch nur einen. Trotz mehrerer Ergebnisse kann der passende Dienst aufgrund der Beschreibung leicht ermittelt werden.

#1	**Relevanz 0,456**
Description:	This service can be used by a client to fetch a learning object that is available to the public. You can use this as a means for provision of downloads for example.
Business Name:	ed.serv
Service Category:	DeliveryService
User Roles:	Learner
Inputs:	LearningObject
Outputs:	LearningObject
Preconditions:	Published
Effects:	Transferred
Service Profile:	http://localhost/ontologies/serviceProfile.owl#learningObjectDeliveryService
WSDL-S Location:	http://localhost/wsdl/learningObjectDeliveryService.wsdl

#2	**Relevanz 0,423**
Description:	With this service Course Material can be mailed to a student. The desired Course Material and the receiving Learner must be given.
Business Name:	ed.serv
Service Category:	DeliveryService
User Roles:	Provider
Inputs:	Learner, CourseMaterial
Outputs:	CourseMaterial
Preconditions:	Published
Effects:	Mailed
Service Profile:	http://localhost/ontologies/serviceProfile.owl#courseMaterialMailingService
WSDL-S Location:	http://localhost/wsdl/courseMaterialMailingService.wsdl

Information 96: Suchergebnisse für den LDoc-Auslieferungsdienst

7 ZUSAMMENFASSUNG UND AUSBLICK

Bei der Entwicklung von SOA-Anwendungen gehören die Realisierung und Einbindung von IT-Diensten zum Alltagsgeschäft. Um die Arbeit und Zusammenarbeit von Geschäftsanalytikern und Softwareentwicklern auf diesem Gebiet einfacher zu gestalten, lohnt es sich, diese mit geeigneten Mitteln zu unterstützen. Eine wesentliche Voraussetzung für die erfolgreiche Wiederverwendung von Diensten ist, dass potentielle Nutzer in einem vorhandenen Dienstportfolio nach funktional passenden Angeboten suchen können. Die Verwirklichung einer halbmanuellen Suche in einem Dienstverzeichnis ist vielschichtig und wird durch die Bedürfnisse der Anwender bestimmt.

Als Grundlage einer solchen Dienstsuche wird eine für den vorgesehenen Zweck passende Suchdatenbasis benötigt. Dazu wurde in dieser Arbeit ein semantisches Dienstbeschreibungskonzept vorgestellt, mit dem die aus Sicht der Suche wichtigsten funktionalen Diensteigenschaften wie unter anderem Ein- und Ausgabeparameter und Vor- und Nachbedingungen erfasst werden können. Das Konzept kombiniert Aspekte bereits existierender semantischer Dienstbeschreibungsverfahren. Es sieht eine Aufteilung der Dienstbeschreibung in semantische Informationen, und ihre Verknüpfung mit syntaktischen Operationen und Übergabeparametern vor. Dazu werden die semantischen Konzepte in einer Ontologie spezifiziert, die Verknüpfungen und syntaktischen Parameter in WSDL erfasst. Die Möglichkeit der Anknüpfung einer Domänenontologie, die das Wissen über die Begriffe eines Geschäftsbereichs enthält, trägt der Notwendigkeit Rechnung, ein einheitliches Begriffsverständnis zwischen Dienstanbietern und Dienstnutzern bezüglich der angebotenen Leistung herzustellen. Die Komplexität der Dienstbeschreibung konnte im Vergleich zu anderen semantischen Dienstbeschreibungsverfahren erheblich verringert werden, da als Ziel nicht die vollautomatische Verschaltung von Diensten vorgesehen wurde, wie es bei den anderen Verfahren der Fall ist. Damit einher geht reduzierter Aufwand bei der Erstellung der Dienstbeschreibung und der Anfragen bei der Dienstsuche.

Die Dienstsuche, die umgesetzt wurde, ist über eine einfache Benutzeroberfläche ansprechbar, in der die Suchparameter für die gewünschten Diensteigenschaften als Texteingaben erfasst werden können. Sie wurde durch ein semantisches Dienstverzeichnis realisiert. Dieses wurde als UDDI-Verzeichnis mit Semantikerweiterung konzipiert und mittels einer Client-Server-Architektur umgesetzt. Dazu wurde das UDDI-Verzeichnis um eine Ontologie und Funktionen zur Veröffentlichung und Suche von semantisch beschriebenen Diensten ergänzt. Bei der Veröffentlichungsfunktion wurde Wert darauf gelegt, dass Interoperabilität zu gängigen Praktiken bei der Veröffentlichung von Diensten gewahrt bleibt, damit diese auch durch etablierte Suchmethoden aufgefunden werden können. Die neue Suchfunktion macht von der semantischen Dienstbeschreibung Gebrauch, um optimal passende Ergebnisse zu liefern. Es wurde ein Suchalgorithmus entworfen, der bei ausbleibenden Suchergebnissen die eingesetzte Unschärfe iterativ anpasst, um auch weniger relevante Dienste zu finden. Dank der semantischen Interpretation der Eingaben ist es dabei möglich, konforme Suchergebnisse auch aufgrund von Vererbungshierarchien zu erzielen.

Um Nutzen aus dem Dienstverzeichnis und dem zugehörigen Dienstbeschreibungskonzept ziehen zu können, ist ein gewisser Grundbestand an Diensten erforderlich. Mit einem ontologiebasierten dienstorientierten Entwicklungsvorgehen wurde in dieser Arbeit eine erste Möglichkeit angedeutet, wie man ausgehend von Geschäftsprozessen zu einer semantisch beschriebenen Dienstinstanz gelangen kann. Dabei wurde der Fokus nicht auf den Gesamtprozess gelegt, sondern es wurden anhand der bearbeiteten und erstellten Artefakte einzelne Teilschritte betrachtet, die von unterschiedlichen Beteiligten bei der SOA-Anwendungsentwicklung durchgeführt werden. Abschließend wurde aufgezeigt, wie das Dienstverzeichnis im Rahmen einer solchen Anwendungsentwicklung bei der Modellierung von Geschäftsprozessen genutzt werden kann, um passende Dienste zu finden und einzubinden. Dabei wurde deutlich, dass die semantische Suchfunktion im Vergleich zur simplen UDDI-Suche ein deutlich besseres Ergebnis bezüglich Quantität und Qualität der Resultate erzielt.

Das vorgeschlagene Entwicklungsvorgehen für die Dienstbeschreibung steht vor der Erprobung im C&M-Projektteam (Kapitel 1.2). Die Erfahrungen daraus könnten genutzt werden, um Methoden und Werkzeuge zur Erstellung der Dienstbeschreibungen gemäß dem entworfenen Konzept und zur Umsetzung der zugehörigen Dienstinstanzen zu entwickeln. Dabei ist insbesondere von Interesse, welche Modellierungstechniken für Geschäftsprozesse und zusätzlich benötigte Informationen eingesetzt und wie die erhaltenen Modelle in Richtung der semantischen Dienstbeschreibung umgewandelt werden können. Ebenso kann es lohnenswert sein, eine weitere Unterteilung der Domänen- und Dienstbeschreibungsontologie voranzutreiben, um eine für den Geschäftsbereich der Aus- und Weiterbildung angepasste und übersichtliche Diversifizierung der Entitäten und Zustandsklassen zu erzielen. Für das Dienstverzeichnis selbst und dessen Nutzung bietet sich eine Ausweitung der Zielgruppe an. Die bisher umgesetzte Funktionalität richtet sich im Wesentlichen an Geschäftsanalytiker. Eine bessere Unterstützung der Softwareentwickler, die die tatsächliche Verschaltung der eingebundenen Dienste zu einem ausführbaren Prozess realisieren, wäre wünschenswert. Die dafür nötigen syntaktischen Diensteigenschaften könnten in der verwendeten Entwicklungsumgebung zusammen mit ihrer Semantik visualisiert werden. Der Grundstein dazu wurde durch die Spezifikation sowohl der logischen als auch der tatsächlichen Ein- und Ausgabeparameter bereits gelegt.

ANHÄNGE

Abkürzungen und Glossar

Abkürzung oder Begriff	Langbezeichnung und/oder Begriffserklärung
ABox	*assertional components* Teil einer Ontologie, in dem die Instanzen der Konzepte aus der TBox und ihre Beziehungen untereinander modelliert werden.
API	*Application Programming Interface* Definierte Stelle, an der eine Anwendung festgelegte Funktionen (z.B. zur Datenübertragung, Transaktionskontrolle, Namenauflösung) in Anspruch nehmen kann.
BPEL	Business Process Execution Language Die Business Process Execution Language (BPEL) ist eine XML-basierte Sprache, mit der Geschäftsprozesse durch Verschaltung von Webservices beschrieben werden können.
BPMN	Business Process Modeling Notation Wichtiger Standard, der Diagramme zur Modellierung von Geschäftsprozessen spezifiziert. Entwickelt von der Business Process Management Initiative (BPMI).
C&M	Cooperation & Management Name der an der Universität Karlsruhe (TH) angesiedelten Forschungsgruppe.
c&m	cooperation & more Im Szenario des IT-unterstützten Wissenstransfers auftretendes Schulungsunternehmen.
CRUD-Operationen	*create*, *read*, *update*, *delete* Standardoperationen, die bei den meisten Datenhaltungssystemen nutzbar sind.
DAML-S	DARPA Agent Markup Language Services Eine Markupsprache, die zur semantischen Beschreibung von Webservices verwendet und inzwischen durch OWL-S abgelöst wurde.
Dienst	In dieser Arbeit wird das Wort Dienst als Synonym für Operation eines webservicebasierten IT-Dienstes verwendet. (Siehe Kapitel 2.2.1)
Dienstinstanz	Für den Betrieb geeignete Realisierung eines Dienstes.
Dienstverzeichnis	Eine meist durch ein oder mehrere Server realisiertes Registrierungssystem zur Veröffentlichung und Suche von Diensten.
DIG	Description Logic Implementation Group Interessengemeinschaft, die standardisierte Schnittstellen für Ontologie-*Reasoner* erarbeitet.

Domänenontologie	Eine Ontologie die darauf ausgelegt ist, das Wissen eines speziellen Geschäftsbereichs (Domäne) zu modellieren.
DSD	Diane Service Description Eine auf Ontologien basierende Sprache zur semantischen Beschreibung von Diensten.
ed.consult	educational.consulting Ein Beratungsunternehmen mit der Kernkompetenz der Konzeption von dienstorientierten Lösungen für die Aus- und Weiterbildung.
ed.serv	educational.services Der IT-Dienstleister educational.services ist Spezialist für IT-gestützte Schulungen.
ed.soft	educational.software Ein Unternehmen, das E-Learning-Software herstellt.
ed.tec	educational.technology Name eines Softwaresystems, das das die Geschäftsprozesse von IT-gestützten Schulungen unterstützt.
FTP	File Transfer Protocol Von der IETF standardisiertes Protokoll zur Übertragung von Dateien im Internet.
Geschäftsobjekt	Ein Geschäftsobjekt ist ein in der Geschäftsprozessmodellierung eingesetztes Element zur Repräsentation physischer Objekt oder Konzepte, die in der realen Geschäftswelt vorkommen.
Geschäftsprozess	Ein Geschäftsprozess ist eine Abfolge von Aktivitäten, die von Mitarbeitern oder IT-Systemen eines oder mehrerer beteiligter Unternehmen durchgeführt werden um ein Geschäftsziel zu erreichen. Ein Geschäftsprozess kann sich in mehrere Unterprozesse aufgliedern.
grounding	Ein *grounding* dient der Verknüpfung einer semantischen Dienstbeschreibung mit der syntaktischen Beschreibung des spezifizierten Dienstes
GUI	Graphical User Interface Grafische Benutzeroberfläche
halbmanuelle Dienstsuche	Art der Dienstsuche bei der sowohl Menschen als auch IT-Systeme beteiligt sind. Eine Person stellt eine Suchanfrage an ein IT-System und erhält eine Liste mit passenden Ergebnissen unter denen sie nochmals wählen kann.
Homonym	Ein Homonym ist ein Wort, das für mehrere Begriffe steht. Homonyme werden umgangssprachlich auch Teekesselchen genannt.
HTTP	HyperText Transfer Protocol Internet-Anwendungsprotokoll, das innerhalb des World Wide Web benutzt wird.

i2s	intellligent internet solutions Name des als Kunden einer Schulung auftretenden Unternehmens, das im Szenario der IT-gestützen Schulung auftritt.
Information Retrieval	(Informationswiedergewinnung) Fachgebiet, das sich mit der meist durch IT-Systeme gestützten, textbasierten Suche von Informationsinhalten beschäftigt.
IOPEs	*input*, *output*, *preconditions*, *effects* Funktionale Eigenschaften eines Dienstes
ISO	International Organization of Standardization Organisation, die u.a. Standards zur Rechnerkommunikation (Open System Interconnection OSI) erarbeitet.
ISWA	Internet-Systeme und Web-Applikationen Das in Lehre und Forschung von C&M vertretene Informatik-Gebiet. Schwerpunkte sind die Entwicklung (Anwendungsmigration, Dienstorientierung, Anwendungsintegration, Sicherheit) und das Management (Anwendungsmanagement, Betriebskonzepte, Dienstmanagement) von Internet-Systemen und Web-Applikationen.
IT-Dienst	Ein IT-Dienst ist ein immaterielles Produkt eines IT-Dienstleisters, das entweder auf IT basiert oder im Zusammenhang mit IT steht und dem Zweck dient, den Zustand des IT-Dienstnehmers unmittelbar zu verbessern. Der Zustand des Dienstnehmers bezieht sich dabei auf seinen Informationsstand, sein Wissen, seine wirtschaftliche Situation usw. Ein IT-Dienst zeichnet sich durch seine Funktionalität bzw. Leistung sowie durch eine Menge von Dienstmerkmalen aus.
IT-Dienstleister	Die Rolle des IT-Dienstleisters wird von jeder (internen oder externen) Organisationseinheit besetzt, die IT-Dienste für einen Kunden bereitstellt und betreibt.
IST	*IT-Supported Training*, IT-gestützte Schulungen Beispielszenario, in dem Informationssysteme zur Unterstützung der Aufbereitung, der Vermittlung und der Aneignung von digitalisierten und über das Internet verfügbaren Wissensinhalten dienen.
J2SE	Java-2-Platform Standard Edition Java-*Framework* für die Entwicklung von Client-Anwendungen.
JDBC	Java Database Connectivity API, die in Java-Programmen zur Unterstützung des Zugangs zu Datenbanken genutzt wird.
Kardinalität	Die Kardinalität gibt den Grad einer Relation an. Für eine Instanz eines Ontologiekonzepts legt sie fest mit wie vielen Partner diese durch die Relation in Verbindung stehen kann.
Komponente	Eine Komponente ist eine Einheit bzgl. unabhängiger Verteilung und bzgl. Komposition durch Dritte und besitzt keinen (externen beobachtbaren Zustand).

Kopplung, lose Kopplung	Kopplung ist die Abhängigkeit zwischen interagierenden Systemen. Diese Abhängigkeit lässt sich in zwei Anteile zerlegen, die reale Abhängigkeit und die künstliche Abhängigkeit. 1. Reale Abhängigkeit ist die Menge von Fähigkeiten und Diensten eines Systems, die vom anderen System beansprucht wird. Reale Abhängigkeit existiert immer und kann nicht reduziert werden. 2. Künstliche Abhängigkeit ist die Menge von Faktoren, denen ein System entsprechen muss, um Fähigkeiten oder Dienste des anderen Systems zu beanspruchen. Typische Faktoren der künstlichen Abhängigkeit sind: Sprachabhängigkeit, Platformabhängigkeit, API-Abhängigkeit usw. Künstliche Abhängigkeit existiert immer, sie oder ihre Kosten können jedoch reduziert werden. Lose Kopplung beschreibt eine Konstellation bei der die künstliche Abhängigkeit auf ein Minimum reduziert wurde. [W3C Web Service Glossary, http://www.w3.org/TR/ws-glossary/]
LDoc	Living Document Ein in Standard-HTML-Format vorliegendes Textdokument (*Document*) mit eingebundenen Grafiken, die durch Anklicken um (i.d.R. multimediales) Material ergänzt werden können und damit zum Leben (*Living*) erweckt werden. Wenn es sich bei diesem Material um ein *ScreenCapture*-Video handelt, das die die Präsentation der zugehörigen Folien umfasst, wird das LDoc als C&M-LDoc bezeichnet.
OASIS	Organization for the Advancement of Structured Information Standards,
Ontologie	Eine Ontologie ist ein Mittel zur objektorientierten Modellierung eines Geschäftsbereichs. In ihr können Konzepte und Instanzen des betrachteten Bereichs sowie deren Beziehungen untereinander erfasst und formal spezifiziert werden.
OWL	Web Ontology Language Die vom W3C definierte OWL Web Ontology Language (2004) ist eine auf RDF aufbauende semantische Markupsprache, um gemeinsame Ontologien im WWW zu definieren. Sie kann zur semantischen Beschreibung der Funktionalität von Webservices genutzt werden.
OWL-S	OWL-S ist eine OWL-Ontologie, die ein Grundgerüst für die semantische Beschreibung von Webservices darstellt.
RDF	Resource Description Framework Eine XML-Sprache zur Bereitstellung von Metainformationen im World Wide Web.
reasoner	Ein *reasoner* ist eine Inferenzmaschine zur Berechnung von logischen Schlussfolgerungen aus formal spezifiziertem Wissen. *Reasoner* werden zur Abfrage von Informationen aus Ontologien eingesetzt.
Relevanz	Unter Relevanz wird im *Information Retrieval* der Grad der Übereinstimmung eines Suchergebnisses mit der Suchanfrage verstanden.

SC-Video	*ScreenCapture*-Video Aufzeichnung des Bildschirminhalts (*Screen*) und des gesprochenen Worts; besonders geeignet bei Verwendung eines eingabesensitiven Bildschirms (Grafiktablett, Plasmabildschirm).
SMTP	Simple Mail Transfer Protocol Internet-Anwendungsprotokoll, das zur Übermittlung von E-Mails verwendet wird.
SOA-Anwendungs-entwicklung	SOA-Anwendungsentwicklung Dienstorientierte Anwendungsentwicklung ist ein Softwareentwicklungsvorgehen zur Erstellung von Anwendungen auf Basis einer serviceorientierten Architektur
SOA	Serviceorientierte Architektur Die Serviceorientierte Architektur (SOA) ist ein Konzept für die Erstellung von Software, dass es ermöglicht nach Fachfunktionalität getrennte wiederverwendbare Softwarekomponenten lose gekoppelt in ein Gesamtsystem zu integrieren.
SOAP	Simple Object Access Protocol Ein auf XML basierendes plattformunabhängiges Kommunikationsprotokoll, das für Webservice-Aufrufe verwendet wird.
Softwarearchitektur	Diese Architekturausprägung beschreibt die Bausteine (Komponenten, deren Eigenschaften, Funktionen, Klassen und Relationen) von Programmen oder Programmteilen.
SWRL	Semantic Web Rule Language XML-basierte Sprache, die es ermöglicht mit Hilfe von Hornlogiken Aussagen über OWL-Konstrukte zu machen.
Taxonomie	Eine Taxonomie ist eine Wissensrepräsentation, mit der Konzepte in Klassen eingeteilt werden.
TBox	*terminological components* Teil einer Ontologie, in dem Konzepte und deren Relationen spezifiziert sind.
Terminologie	Eine Terminologie ist eine Festlegung von Begriffsdefinitionen wie beispielsweise durch ein Glossar oder Wörterbuch.
Top-Level-Ontologie	Eine *Top-Level*-Ontologie ist ein Ontologieteil mit hohem Abstraktionsgrad, der direkt unter der Urklasse angesiedelt ist und Aspekte modelliert, die in verschiedenen Ontologien wiederverwendet werden.
UDDI	Universal Description, Discovery and Integration of Web Services. UDDI ist ein Standard, der OASIS der Dienstverzeichnisse für die Veröffentlichung und Suche von Webservices beschreibt.
UML	Unified Modeling Language Sprache, die aus graphischen Elementen besteht und zur semi-formalen Beschreibung von beliebigen Gegenständen (z.B. Software-Systemen oder Geschäftsbereichen) geeignet ist.

URL	Uniform Resource Locator Eine Zeichenkette, die eine Ressource, wie z.B. eine Web-Seite, eindeutig im World Wide Web identifiziert.
W3C	World Wide Web Consortium Organisation, durch die mit dem Web verbundene Technologien und Konzepte standardisiert werden.
Webservice	Webservices sind eine Technologie für die Erstellung von Softwarekomponenten mit Schnittstellen, die mittels standardisierter Kommunikationsprotokolle über das Internet angesprochen werden können. Kommunikation und Datenformate basieren auf gängigen Standards, denen oftmals XML zu Grunde liegt.
WSDL	Web Service Description Language XML-basierte Sprache zur Spezifikation von Webserviceschnittstellen
XML	eXtensible Markup Language Vom W3C standardisierte Auszeichnungssprache, durch die sich die Struktur eines Dokuments beschreiben lässt.
XML-Schema	XML-basierte Sprache zur Spezifikation der Grammatik anderer XML-basierter Sprachen.

Index

Informationen

Diese Diplomarbeit

Inhalte

Was sind die zentralen Inhalte (Themen, interessante Aussagen, Botschaften, Fragestellungen), die in der Arbeit behandelt werden?

(I1) Wie sieht eine semantische Dienstbeschreibung aus, die eine gute Sucheffizienz bei der halbmanuellen Suche von Diensten bietet? (D1) (D3)

(I2) Wie kann eine solche Dienstbeschreibung mit einem Dienstverzeichnis verknüpft werden, um dadurch eine halbmanuelle Suche zu realisieren? (D2)

(I3) Wie kann das Dienstverzeichnis mit entsprechend beschriebenen Diensten gefüllt und beim Programmieren im Großen genutzt werden? (D3)

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) Domänenunspezifische und eher auf die Syntax fokussierte Dienstbeschreibungen sind unzureichend für eine hohe Sucheffizienz.

(D2) Die Verwendung semantischer Dienstbeschreibungen und zugehöriger Suche zur Unterstützung halbmanueller Dienstsuche wurde bisher unzureichend betrachtet.

(D3) Es gibt bisher wenige konkrete Beispiele für die Ontologienutzung und das Aussehen der Ontologieanteile bei der semantischen Dienstbeschreibung.

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Die erarbeiteten Methoden sollen bei der halbmanuellen Dienstsuche Unterstützung bieten. Nicht betrachtet werden Anforderungen und Realisierung einer vollautomatisierten Dienstsuche oder -verschaltung.

(P2) Es wird ausschließlich die Spezifikation funktionaler Aspekte der Dienste betrachtet.

(P3) Beispiele und Untersuchungen beschränken sich auf den Bereich der Aus- und Weiterbildung.

(P4) Es soll nach Möglichkeit auf gängige oder zukünftige Standards aufgebaut werden.

Lösungen

Was sind die eigenen Lösungen?

(L1) Dienstbeschreibungskonzept: Es wird aufbauend auf Ontologien und gängigen Standards ein Konzept zur semantischen Beschreibung von Diensten erarbeitet, ausgerichtet an der Veröffentlichung und halbmanuellen Suche von Diensten. (I1) (D1) (D3)

(L2) Dienstverzeichnis: Es wird ein Dienstverzeichnis zur Veröffentlichung und halbmanuellen Suche von Diensten entworfen und prototypisch implementiert, das sich die semantischen Dienstbeschreibungen zunutze macht. (I2) (D2)

(L3) Ontologiebasiertes, dienstorientiertes Entwicklungsvorgehen: Es wird ein mögliches Vorgehen zur Erstellung der Verzeichnisinhalte und der Suchdatenbasis aufgezeigt sowie die Nutzung des Verzeichnisses beim Programmieren im Großen. (I3) (D3)

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

(N1) Das erarbeitete Dienstbeschreibungskonzept wird an passenden Beispielen umgesetzt und verdeutlicht. (L1)

(N2) Die Einbringung und Suche von semantisch beschriebenen Diensten wird anhand des Dienstverzeichnis-Prototyps demonstriert. (L2)

(N3) Es wird aufgezeigt, wie das Dienstverzeichnis beim Programmieren im Großen eingesetzt werden kann. (L3)

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Welche Modellierungstechniken und Werkzeuge können zu einer effizienteren Entwicklung der semantischen Dienstbeschreibung verwendet werden?

(O2) Wie können die Domänen- und Dienstbeschreibungsontologie angepasst an den Geschäftsbereich der Aus- und Weiterbildung sinnvoll weiter unterteilt und diversifiziert werden?

(O3) Wie kann das Dienstverzeichnis und seine Funktionen zur besseren Unterstützung von Softwareentwicklern erweitert werden, die die tatsächliche Verschaltung gefundener Dienste zu einem ausführbaren Prozess auf syntaktische Ebene realisieren?

Literaturanalysen

Automatisierung dienstorientierten Rechnens durch semantische Dienstbeschreibung

[Kl06]	Michael Klein: Automatisierung dienstorientierten Rechnens durch semantische Dienstbeschreibung, Dissertation, Friedrich-Schiller-Universität Jena, Fakultät für Mathematik und Informatik, 2006.

Inhalte

Was sind die zentralen Inhalte (Themen, interessante Aussagen, Botschaften, Fragestellungen), die in der Arbeit (d.h., in der analysierten Literatur) behandelt werden?

(I1) Zusammenfassung der Anforderungen an semantische Dienstbeschreibungssprachen zur automatischen Dienstsuche (S.40)

(I2) Eine semantische Dienstbeschreibungssprache für die automatisierte Dienstnutzung

(I3) Ein automatischer Vergleicher für die neue Dienstbeschreibungssprache

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) Manuelle Dienstbindung bei wechselnden Anforderungen ist weder effizient noch skalierbar in dynamischen Umgebungen. (S.9)

(D2) Bisherige semantische Dienstbeschreibungsmethoden erfüllen nur einen Bruchteil der Anforderungen. (I2) (S.80)

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Es werden nur funktionale Eigenschaften der Dienste betrachtet. (S.12)

(P2) Es werden nur atomare Dienstaufrufe betrachtet, keine Choreographien (S.12)

(P3) Dienste werden als Blackbox betrachtet (S.13)

Lösungen

Was sind die eigenen Lösungen?

(L1) Entwurf der Dienstbeschreibungssprache Diane Service Description (DSD). Dabei Aufteilung der Sprache in die drei Teile Generische Ontologiesprache, Dienstspezifische Ontologiesprache und Dienstontologie und Aufteilung der Dienstbeschreibung in operationale, aggregierende, selektierende und bewertende Elemente. (S.14-15,94) (I2) (D2)

(L2) Vergleichsalgorithmen für DSD-Beschreibungen und -Anfragen (S.217) (D1) (D2)

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

(N1) Evaluation der DSD hinsichtlich der festgelegten Anforderungen (S.247) (I1) (L1)

(N2) Einige mit DSD beschriebene Beispieldienste. (L1) (L2)

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Wie lassen sich Domänenontologien integrieren, die nicht primär für die DSD geschrieben wurden?

(O2) Wie könnte die DSD in OWL transformiert werden?

Importing the Semantic Web in UDDI

[PK+02]	Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, Katia Sycara: Importing the Semantic Web in UDDI, Web Services E-Business and the Semantic Web CAiSE 2002 International Workshop (WES'02), Toronto, LNCS 2512 S.225-236, Springer, Mai 2002.

Inhalte

Was sind die zentralen Inhalte (Themen, interessante Aussagen, Botschaften, Fragestellungen), die in der Arbeit (d.h., in der analysierten Literatur) behandelt werden?

(I1) Wie können DAML-S-Dienstbeschreibungen in UDDI repräsentiert werden? (S.224)

(I2) Wie können UDDI-Verzeichnisse erweitert werden, so dass die DAML-S-Dienstbeschreibungen zur Suche verwendet werden können? (S.224)

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) In UDDI werden die Fähigkeiten der Dienste nicht gespeichert. Daher kann man in UDDI-Verzeichnissen nicht anhand von gewünschter Funktionalität nach Diensten suchen. (S.224)

(D2) Die XML-basierten Webservice-Standards bieten keine Möglichkeit zur Spezifikation von expliziter Semantik. (S.224)

(D3) Die Dienstsuche mittels UDDI ist Schlüsselwort-basiert und nutzt keine logische Deduktion. (S.231)

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Die automatische Dienstkomposition wird nicht betrachtet. (S.226)

(P2) Das UDDI-Verzeichnis soll zur Dienstveröffentlichung benutzt werden. Alternativen werden nicht betrachtet. (S.229)

Lösungen

Was sind die eigenen Lösungen?

(L1) Es wird eine mögliche Abbildung von DAML-S-Beschreibungen auf UDDI-Einträge vorgestellt. (D1, D2) (I1) (S.231)

(L2) Es wird eine Architektur und ein Vorgehen zur Dienstveröffentlichung und -suche aufgezeigt, das die DAML-S-Beschreibungen mit einbezieht. (D3) (I2) (S.233)

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

(N1) Implementierung der Abbildung und der eines *Matchmakers* ist vorhanden (L1) (L2) (S.235)

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Welche Schwierigkeiten ergeben such aus der redundanten Datenhaltung in DAML-S und UDDI?

(O2) Können konventionell beschriebene Dienste über die erweiterte Suchfunktion gefunden werden; wenn ja wie?

A Process and a Tool for Creating Service Descriptions Based on DAML-S

[KK03]	Michael Klein, Birgitta König-Ries: A Process and a Tool for Creating Service Descriptions Based on DAML-S, 2003 VLDB Workshop on Technologies for E-Services (TES'03), Berlin, September 2003.

Inhalte

Was sind die zentralen Inhalte (Themen, interessante Aussagen, Botschaften, Fragestellungen), die in der Arbeit (d.h., in der analysierten Literatur) behandelt werden?

(I1) Welche Bedingungen muss eine „gute" Dienstbeschreibungssprache erfüllen? (S.144)

(I2) Wie sieht das Verhältnis gängiger Dienstbeschreibungssprachen zu diesen Bedingungen aus? (S.145)

(I3) Welche Schwierigkeiten bestehen bei DAML-S und wie kann man mit DAML-S eine Dienstbeschreibung erreichen, die den geforderten Bedingungen entspricht? (S.146) (D2)

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) Dienstbeschreibungssprachen mit geringer semantischer Aussagekraft sind für die automatische Dienstsuche nicht geeignet. (S.145)

(D2) DAML-S bietet sehr generische Möglichkeiten der Dienstbeschreibung und keine konkreten Anhaltspunkte, wie es sinnvoll verwendet werden kann. Daraus folgen Schwierigkeiten bei der Erstellung und beim Vergleich von Dienstbeschreibungen. (S.143,146)

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Das aufgezeigte Vorgehen zur Entwicklung der Dienstbeschreibung wird nur anhand des *Service Profile* aufgezeigt. *Service Model* und *Service Grounding* bleiben unbetrachtet. (S.147)

(P2) Um Probleme mit der Klassenhierarchie in der Ontologie zu umgehen wird eine Anpassung der DAML-S-Spezifikation in Kauf genommen (S. 149)

(P3) Es wird davon ausgegangen, dass zu verwendende Domänenontologien bereits existieren. Die Erstellung einer solchen im Rahmen der Dienstbeschreibung wird nicht thematisiert. (S. 150)

Lösungen

Was sind die eigenen Lösungen?

(L1) Wesentliche Bedingungen, die an Dienstbeschreibungen zu stellen sind, sind die folgenden: Semantische Ausdruckskraft, automatisch vergleichbar, flexibel, editierbar. (S.144) (I1)

(L2) Nur Ontologie-basierte Beschreibungsarten erfüllen die gesetzten Bedingungen. (S.145f) (I2) (D1)

(L3) Es wird ein auf DAML-S basierendes Dienstbeschreibungsverfahren vorgestellt, das die vier Schritte Übernahme der Ober-Ontologie, Definition der Dienstkategorie, Einfügen von Domänenontologien, Erstellung der Beschreibungsinstanz enthält. Das Verfahren

baut auf eine dreiteilige Ontologie auf und wird durch ein ebenfalls vorgestelltes Werkzeug unterstützt.

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

(N1) Die Durchführung des vorgestellten Beschreibungsverfahrens wird am Beispiel eines Druckdienstes aufgezeigt.

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Sind die vorgestellten Bedingungen an die Dienstbeschreibung ausreichend? (L1)

(O2) Wie können *Service Model* und *Service Grounding* sinnvoll für eine Dienstbeschreibung genutzt werden? (P1)

(O3) Wie können zahlreiche verteilte Ontologien bei der Veröffentlichung und Suche von Diensten zugänglich gemacht werden? (S.150)

(O4) Wie können die *Input*- und *Output*-Parameter sinnvoll semantisch beschrieben werden? (S.149)

Adding Semantics to Web Services Standards

[SV+03]	Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth, John Miller: Adding Semantics to Web Services Standards, Proceedings of The 2003 International Conference on Web Services (ICWS'03) S.395-401, Las Vegas, Juni 2003.

Inhalte

Was sind die zentralen Inhalte (Themen, interessante Aussagen, Botschaften, Fragestellungen), die in der Arbeit (d.h., in der analysierten Literatur) behandelt werden?

(I1) Wie kann der gängige Webservice-Standard WSDL um Semantikzuordnungen ergänzt werden? (S.1)

(I2) Wie kann darauf aufbauend mit UDDI eine Registrierung und eine (automatisierte) Suche nach passenden Webservices durchgeführt werden? (S.1)

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) Der Suchmechanismus von UDDI ist nicht für die Suche von Webservices zur automatischen Verschaltung geeignet, da er keine Semantikinformationen und Elemente der Dienstbeschreibung verwendet. (S.1)

(D2) XML-Schema kann nicht für einen Austausch von Semantikinformationen verwendet werden, da es oft in die Dienstbeschreibung eingebettet wird und daher schwer zu verteilen und wieder verwenden ist. (S.2)

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Die Einbettung von Semantik in Dienstbeschreibungen soll rückwärtskompatibel zu den gängigen Standards für Webservices sein. (S.2)

(P2) Die Semantikeinbettung ist beschränkt auf die Veröffentlichung und Suche von Webservices. (S.1)

Lösungen

Was sind die eigenen Lösungen?

(L1) Mit Hilfe von WSDL-Erweiterungselementen kann eine Verknüpfung von WSDL-Konstrukten und DAML+OIL- (OWL-) Ontologien erreicht werden. (S.2) (I1) (D2)

(L2) Durch die Verwendung von tModels und keyedReferenceGroups (UDDI v3) kann die Zuordnung von Webservices zu den zugehörigen Ontologiekonzepten ins Verzeichnis aufgenommen werden. (S.3) (I2) (D1).

(L3) Gegeben wird ein dreistufiger Algorithmus zur Suche von Diensten unter Verwendung des UDDI und der Ontologie. (S.4) (I2) (D1)

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

(N1) Die Semantikeinbettung ins WSDL wird anhand eines Beispiels vorgestellt. (S.7) (L1)

(N2) Die Semantikeinbettung ins UDDI wird anhand eines Beispiels vorgestellt (S.6) (L2)

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Reichen die vorgestellten Suchmöglichkeiten aus, die in UDDI nur schlüsselwortbasiert funktionieren und sich ausschließlich auf den Diensttyp beziehen?

(O2) Wie sieht die Struktur der Ontologie aus, die mit den WSDL-Konstrukten verknüpft wird.

Semi-automatic Composition of Web Services using Semantic Descriptions

[SH+03]	Evren Sirin, James Hendler, Bijan Parsia: Semi-automatic Composition of Web Services using Semantic Descriptions, 2003 Web Services: Modeling, Architecture and Infrastructure Workshop (WSMAI'03), 2003 International Conference on Enterprise Information Systems (ICEIS'03), Angers, April 2003.

Fragestellungen

Was sind die zentralen Fragen (Themen, Probleme), die in der Arbeit behandelt und gelöst werden sollen?

(F1) Wie kommt man zu kombinierten Webservices, wenn die vorhandenen nicht die gewünschte Funktionalität abdecken? (S. 1 Abs. 3)

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) Vollautomatische Verschaltung von Webservices bisher unausgereift (S. 2 Abs. 1)

(D2) Vollautomatische Verschaltung aufgrund der Anzahl und Art der zu Verfügung stehenden Webservices nicht immer möglich (z.B. Sensornetze)

(D3) UDDI für die Suche ungenügend, da nur Zeichenkettenvergleich mit Namen (S. 5 Abs. 2)

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Webservice-Beschreibungen und die zugrunde liegenden Ontologien sind vorhanden.

Lösungen

Was sind die eigenen Lösungen?

(L1) Verwendung von OWL und DAML-S zur Erstellung eines Frameworks für die Verschaltung von Webservices. (S. 2 Abs. 2)

(L2) Abgleich der Webservices für die Eingaben mittels semantischer Klassen. (S. 4 Abs. 5)

(L3) Zusätzliche Verwendung von nicht funktionalen Eigenschaften zur Auswahl der geeigneten Webservices. (S. 5 Abs. 2)

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

(N1) Framework bestehend aus einer *inference enginge* (basierend auf Prolog) und einem *composer*, der eine GUI zur Verschaltung zur Verfügung stellt. (S. 4 Abs. 2)

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Wie kann man effizient und standardisiert vermeiden, dass die verschalteten Services über den Client aufgerufen werden? (S. 6 Abs. 4)

(O1) Die vorgestellten Such- und Kombinationsmöglichkeiten beschränken sich auf bereits entdeckte Webservices. Wie kann man sie auf die Webservice-Suche erweitern? (S. 7 Abs. 4)

Semantic Web Services – Fundamentals and Advanced Topics

[Bu04]	Christoph Bussler: Semantic Web Services – Fundamentals and Advanced Topics, Net.ObjectDays 2004 (NODe'04), Erfurt, LNCS 3263 S.1-8, Springer, 2004.

Fragestellungen

Was sind die zentralen Fragen (Themen, Probleme), die in der Arbeit behandelt und gelöst werden sollen?

(F1) Wie werden Webservices mit Semantik angereichert und wo sind dafür Ergänzungen zu traditionellen Webservices nötig? (S. 1)

(F2) Welche zusätzlichen Erweiterungen für Webservices sind nötig, um diese für eine echte Verflechtung von Geschäftspartnern einzusetzen? (S. 1)

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) EAI und B2B sind mit traditionellen nur auf syntaktischer Ebene möglich. Gängige Standards (WSDL, UDDI, BPEL) klammern Semantikfragestellungen aus. (S. 1-3)

(D2) Aktuelle Ansätze für die Verknüpfung von Semantik mit Webservices betrachten nicht die Anforderungen, die ein tatsächlicher Einsatz in der Geschäftswelt fordert. (S. 4)

(D3) Aktuelle Ansätze, die Anforderungen der Geschäftswelt anzugehen, sind meist isoliert auf das jeweilige Problem und berücksichtigen umgekehrt die Notwendigkeit der Semantikintegration nicht. Ein umfassender Ansatz existiert nicht. (S. 7)

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Zur Beschreibung der Einbringung von Semantik in Webservices wird beispielhaft der WSMO-Vorstoß betrachtet. (S. 3)

Lösungen

Was sind die eigenen Lösungen?

(L1) Es wird aufgezeigt, wo innerhalb der WS-Standards Semantik berücksichtigt werden muss und wie dies im Fall von WSMO angegangen wird. (S. 1-4)

(L2) Es werden zahlreiche offene Fragen bezüglich Anforderungen der Geschäftswelt an Webservices formuliert und erläutert. Z.B.: Prozessentdeckung, vertragliche Verpflichtungen, Rückabwicklung, … (S. 4-7)

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Siehe (L2)

What Are Ontologies, and Why Do We Need Them?

[CJ+99]	B. Chandrasekaran, R. Josephson, V. Richard Benjamins: What Are Ontologies, and Why Do We Need Them?, Intelligent Systems and Their Applications Band 14 Heft 1 S.20-26, IEEE Intelligent Systems, 1999.

Fragestellungen

Was sind die zentralen Fragen (Themen, Probleme), die in der Arbeit behandelt und gelöst werden sollen?

(F1) Welche Rolle spielen Ontologien in informationsverarbeitenden und wissensbasierten Systemen und beim Verstehen natürlicher Sprachen? (S. 20)

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) Die gängige Literatur enthält Inkonsistenzen bezüglich der Bedeutung des Plurals „Ontologies“. (S. 21)

(D2) Es gibt keine einheitliche und/oder standardisierte Ontologie erster Ebene. (S. 22)

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Der Artikel betrachtet ausschließlich Ontologien über Faktenwissen und nicht Ontologien über methodisches Wissen. (S. 20)

Lösungen

Was sind die eigenen Lösungen?

(L1) Die Bedeutung von Ontologien bezüglich (F1) wird herausgearbeitet.

(L2) Gemeinsamkeiten der vielen unterschiedlichen Ontologien werden aufgezeigt (S. 22)

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Wie sieht ein konkretes Einsatzszenario von Ontologien aus?

A Practical Guide to Building Ontologies Using the Protégé OWL Plugin and CO-ODE Tool

[HK+04]	Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, Chris Wroe: A Practical Guide to Building Ontologies Using the Protégé OWL Plugin and CO-ODE Tool, http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf, Juni 2004.

Fragestellungen

Was sind die zentralen Fragen (Themen, Probleme), die in der Arbeit behandelt und gelöst werden sollen?

(F1) Was sind Ontologien und wie können sie mit OWL und Protégé erstellt werden?

(F2) Wie kann eine Ontologie durch Einsatz eines Reasoners zur Ableitung von Wissen verwendet werden?

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) Die Ontologieerstellung ist komplex, vor allem ohne geeignetes Werkzeug.

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Die Anleitung bezieht sich auf OWL-DL Ontologien, und ihre Erstellung mit Protégé.

Lösungen

Was sind die eigenen Lösungen?

(L1) Eine ausführliche Demonstration der Erstellung am Beispiel einer Pizza-Ontologie

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Wie kann eine Ontologie für einen bestimmten Geschäftsbereich oder Verwendungszweck gezielt erstellt werden?

Wissensmanagement mit Ontologien und Metadaten

[St02]	Steffen Staab: Wissensmanagement mit Ontologien und Metadaten, Habilitationsschrift, Universität Karlsruhe (TH), AIFB, Informatik Spektrum Band 25 Heft 3 S.194-209, Springer, 2002.

Fragestellungen

Was sind die zentralen Fragen (Themen, Probleme), die in der Arbeit behandelt und gelöst werden sollen?

(F1) Wie können Wissensmanagementsysteme aussehen, gebaut und genutzt werden, die auf Ontologien und Metadaten für die Wissensrepräsentation zurückgreifen? (S. 1 Abs. 1)

(F2) Wie sehen Methoden und Werkzeuge aus um die Prozesse aus (F1) zu unterstützen?

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) Konventionelle Wissensmanagementsysteme decken häufig nur Infrastruktur, organisatorische Abläufe und Kontrollfunktionen ab. Semantische Verknüpfung der Inhalte findet nicht statt. (S. 16 Abs. 8).

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Effektives Wissensmanagement ist eine für den wirtschaftlichen Erfolg notwendige Disziplin. (S. 1. Abs. 1)

Lösungen

Was sind die eigenen Lösungen?

(L1) Vorgestellte Methoden zur Ontologieerstellung (S. 11), (Meta-)Datenkreierung (S. 14) und Wissensnutzung (S. 15)

(L2) Unterscheidung zwischen Wissensprozess und Wissensmetaprozess (s. 10 Abs. 3)

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

(N1) Beispiel für mögliche Realisierung des zweigeteilten Wissenszyklus (Prozess und Metaprozess) (S. 18. Abs 2).

(N2) Verschiedene Algorithmen für die halbautomatische Entwicklung einer Baseline-Ontologie (S. 18. Abs. 4)

(N3) Ein Beispielprojekt für ein Wissensportal (S. 19. Abs. 1).

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Wie kann man aus der Kunst Ontologieerstellung eine Ingenieurstätigkeit machen? Wie lässt sich generell der Entwicklungsprozess verbessern? (S. 13 Abs. 6)

Overview and analysis of methodologies for building ontologies

[FG02]	Mariano Fernández-López, Asunción Gómez-Pérez: Overview and analysis of methodologies for building ontologies, The Knowledge Engineering Review Band 17 Heft 2 S.129-156, Cambridge, Cambridge University Press, April 2002.

Fragestellungen

Was sind die zentralen Fragen (Themen, Probleme), die in der Arbeit behandelt und gelöst werden sollen?

(F1) Welche Methoden zur Entwicklung von Ontologien gibt es? Was leisten diese, wie ausgereift sind sie? (S. 1 Abs. 2)

(F2) Kann der IEEE *Standard for Developing Software Life Cycle Processes* (1074-1995) auch für die Entwicklung von Ontologien angewandt werden?

(F3) Wie sehen die Entwicklungsmethoden für Ontologien im Vergleich zu diesem IEEE Standard aus?

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) Es gibt viele verschiedene Entwicklungsmethoden, Ontologieentwicklung ist eine Kunst. (S. 2 Abs. 2)

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Es werden nicht sämtliche Entwicklungsmethoden untersucht, sondern nur eine Auswahl.

Lösungen

Was sind die eigenen Lösungen?

(L1) Der IEEE Standard kann auch auf die Ontologieentwicklung angewandt werden. (S. 21 Abs. 6)

(L2) Die Methode METHONTOLOGY ist am meisten ausgereift bezüglich des IEEE-Standards. (S. 153 Abs. 4)

(L3) Die Methoden sind (berechtigterweise) sehr unterschiedlich, es gibt keine wirklich ausgereifte Referenz-Methode. (S. 153 Abs. 12)

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

(N1) Vergleichstabelle der Methoden für die Unterprozesse des IEEE-Standards (Table 1)

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Wie können Ontologien durch verteilte Parteien entwickelt werden? (S. 24 Abs. 11)

(O2) Können verschiedene Methoden kombiniert werden? (S. 153 Abs. 14)

Service-Oriented Provisioning of Learning Objects

[VW05a]	Gottfried Vossen, Peter Westerkamp: Service-Oriented Provisioning of Learning Objects, Arbeitsbericht Nr. 2 des E-Learning Kompetenzzentrums Münster, ERCIS – European Research Center for Information Systems, Universität Münster, 2005, Hrsg.: H. L. Grob, J. vom Brocke, Münster 2005.

Fragestellungen

Was sind die zentralen Inhalte (Themen, interessante Aussagen, Botschaften, Fragestellungen), die in der Arbeit (d.h., in der analysierten Literatur) behandelt werden?

(F1) Wie kann die zur Auslieferung von *learning objects* benötigte Funktionalität standardisiert bereitgestellt werden?

(F2) Wie kann die Darstellung von *learning objects* realisiert werden?

Defizite

Welche Defizite bestehender Arbeiten und Lösungen werden als Motivation der eigenen Lösungen genannt?

(D1) Die bereits vorhandenen Vorschläge für die Standardisierung des Austauschs von Lernmedien sind mangelhaft oder werden nicht in der Breite akzeptiert.

(D2) Wartung und Verteilung von Lerninhalten ist komplex und aufwändig. (S. 3 Abs. 4)

(D3) Möglichkeiten für Inhalte von Lerneinheiten sind von der jeweiligen (proprietären) Lösung abhängig, die zum Einsatz kommt. (S. 3 Abs. 5)

Prämissen

Welche Einschränkungen und Vorgaben werden hinsichtlich der eigenen Lösungen gemacht?

(P1) Der Client muss WSRP verstehen (S. 22 Abs. 7)

Lösungen

Was sind die eigenen Lösungen?

(L1) LernServe eine dienstorientierte Lernplattform (S. 10 Abs. 4). Die Plattform soll auf Standards aufbauen (SCROM, WSRP).

(L2) Die Plattform interagiert mit dem Client mittels Webservices. Dabei wird die Inhaltsverarbeitung und -Integration weitestgehend auf der Serverseite gehandhabt. (S. 14 ff)

(L3) Die Plattform bietet darüber hinaus weit reichende Tracking- und Profilingmöglichkeiten. (S. 17 ff)

Nachweise

Welche Nachweise (*Evidence*) werden hinsichtlich der Tragfähigkeit der eigenen Lösungen geliefert?

(N1) LernServe eine dienstorientierte Lernplattform, die sich angeblich bei der Universität Münster im Bau befindet. (S. 10 Abs. 4)

Offene Fragen

Welche Fragen sind noch ungelöst geblieben bzw. stellen sich dem Leser?

(O1) Wie geschieht die Erstellung von Lernmaterial für eine solche Plattform?

(O2) Wie können Semantikbeschreibungen für die Lerninhalte eingeflochten werden und welchen Mehrwert bringen sie? (S. 24. Abs. 2)

(O3) Wie können Semantikbeschreibungen für die Dienste eingeflochten werden und welchen Mehrwert bringen sie? (S. 24. Abs. 2)

Literatur

[AB+02] Anupriya Ankolenkar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Drew McDermott, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry R. Payne, Katia Sycara: DAML-S: Web Service Description for the Semantic Web, 2002 International Semantic Web Conference (ISWC'02), Sardinia, LNCS 2342 S.348-363, Springer, Juni 2002.

[AB+04] Ali Arsanjani, Bernhard Borges, Kerrie Holley: Service-oriented Architecture, Artikel veröffentlicht im DM Direct Newsletter, http://www.dmreview.com/article_sub.cfm?articleId=1013602, 2004

[Ab05b] Sebastian Abeck: KURSBUCH INFORMATIK I – Formale Grundlagen und Programmierkonzepte am Beispiel von Java, Universitätsverlag Karlsruhe, http://www.uvka.de/univerlag/volltexte/2005/73/, 2005.

[AF+05] Grigoris Antoniou, Enrico Franconi, Frank van Harmelen: Introduction to Semantic Web Ontology Languages, Reasoning Web Summer School 2005, Msida, LNCS 3564 S.1-21, Springer, Juli 2005.

[AF+05a] Rama Akkiraju, Joel Farrell, JohnMiller, Meenakshi Nagarajan, Marc-Thomas Schmidt, Amit Sheth, Kunal Verma: Web Service Semantics – WSDL-S, W3C Member Submission Version 1.0, http://www.w3.org/Submission/WSDL-S/, November 2005.

[AG+04] Sebastian Abeck, Markus Gebhard, Karsten Krutz, Klaus Scheibenberger, Niko Schmid: Ein Portal zur Lehrunterstützung, Arbeitskonferenz „Elektronische Geschäftsprozesse", Klagenfurt, 2004

[AH04] Grigoris Antoniou, Frank van Harmelen: Web Ontology Language: OWL, Handbook On Ontologies S.67-92, International Handbooks on Information Systems, Springer, 2004.

[Ba02] Sandra Bartl: Spezifikation und prototypische Implementierung eines Werkzeugs zur Planung von Schulungen, Studienarbeit, Universität Karlsruhe (TH), C&M (Prof. Abeck), 2002.

[Ba05a] Sandra Bartl: Methode zur Abbildung von Geschäftsprozessen auf serviceorientierte Architekturen, Diplomarbeit, Universität Karlsruhe (TH), C&M (Prof. Abeck), 2005.

[Be03] Sean Bechhofer: The DIG Description Logic Interface: DIG/1.1, http://dl-web.man.ac.uk/dig/2003/02/interface.pdf, Februar 2003.

[BH+04] Franz Baader, Ian Horrocks, Ulrike Sattler: Description Logics, Handbook On Ontologies S.3-28, International Handbooks on Information Systems, Springer, 2004.

[BM+03] Sean Bechhofer, Ralf Möller, Peter Crowther: The DIG Description Logic Interface, 2003 International Workshop on Description Logics (DL2003), Rome, CEUR Workshop Proceedings, September 2003.

[BPMI-BPMN1.0] Business Process Management Initiative (BPMI): Business Process Modeling Notation (BPMN) Version 1.0, http://www.bpmn.org/, Mai, 2004.

[BR99] Ricardo Baeza-Yates, Berthier Ribeiro-Neto: Modern Information Retrieval, Addison-Wesley Longman, 1999.

[Bu04] Christoph Bussler: Semantic Web Services – Fundamentals and Advanced Topics, Net.ObjectDays 2004 (NODe'04), Erfurt, LNCS 3263 S.1-8, Springer, 2004.

[BV+04] Sara Brockmans, Raphael Volz, Andreas Eberhart, Peter Löffler: Visual Modeling of OWL DL Ontologies Using UML, 2004 International Semantic Web Conference (ISWC'04), Hiroshima, LNCS 3298 S.198–213, Springer, 2004.

[C&M-P] Cooperation & Management: Die Forschungsgruppe Cooperation & Management (C&M), http://www.cm-tm.uka.de, Menüpunkt "Profil", Universität Karlsruhe (TH), C&M (Prof. Abeck).

[CJ+99] B. Chandrasekaran, R. Josephson, V. Richard Benjamins: What Are Ontologies, and Why Do We Need Them?, Intelligent Systems and Their Applications Band 14 Heft 1 S.20-26, IEEE Intelligent Systems, 1999.

[CJ02a] John Colgrave, Karsten Januszewski: Using WSDL in a UDDI Registry, Version 1.08, OASIS Best Practices Document, http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.htm, 2002

[CJ04] John Colgrave, Karsten Januszewski: Using WSDL in a UDDI Registry, Version 2.0.2, OASIS Technical Note, http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v202-20040631.htm, 2004.

[Cl06] Luc Clément: The Top-Seven Risks to SOA Without a Registry, Business Integration Journal Online, http://www.bijonline.com/index.cfm?section=article&aid=735, 2006.

[CP99] Stephen Cranefield, Martin Purvis: UML as an Ontology Modelling Language, Proceedings of the Workshop on Intelligent Information Integration, 1999 International Joint Conference on Artificial Intelligence (IJCAI'99), Stockholm, August 1999.

[CT+04] Peter Clark, John Thompson, Bruce Porter: Knowledge Patterns, Handbook On Ontologies S.191-207, International Handbooks on Information Systems, Springer, 2004.

[CT+04a] Sam Chung, Lai Hong Tang, Sergio Davalos: A Web Service Oriented Integration Approach for Enterprise and Business-to-Business Applications, 2004 International Conference on Web Information Systems Engineering (WISE'04), Brisbane, LNCS 3306 S.510-515, Springer, 2004.

[DP98] T. H. Davenport, L. Prusak: Working knowledge: how organizations manage what they know, Boston, Harvard Business School Press, 1998

[DW+01] Aseem Das, Wei Wu, Deborah L. McGuinness: Industrial Strength Ontology Management, Stanford Knowledge Systems Laboratory Technical Report KSL-01-09 2001, 2001 International Semantic Web Working Symposium (SWWS'01) S.17-37, Stanford, Juli 2001.

[Ec02] Matthias Eck: Wissensnetze zur Navigation und semantischen Suche in der Internet-basierten Aus- und Weiterbildung, Diplomarbeit, Universität Karlsruhe (TH), 2002.

[EM+05] Christian Emig, Christof Momm, Jochen Weisser, Sebastian Abeck: Programming in the Large based on the Business Process Modeling Notation, Jahrestagung der Gesellschaft für Informatik (GI), Bonn, September 2005.

[EU99] Die Ausbildungsminister der Europäischen Union: The Bologna Declaration, Bologna, Juni 1999.

[EW+06] Christian Emig, Jochen Weisser, Sebastian Abeck: Development of SOA-Based Software Systems – an Evolutionary Programming Approach, 2006 International Conference on Internet and Web Applications and Services (ICIW'06), Guadeloupe, Februar 2006.

[Fe01] Dieter Fensel: Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce, Springer, 2001.

[FF99] Richard Fikes, Adam Farquhar: Distributed Repositories of Highly Expressive Reusable Ontologies, Intelligent Systems and Their Applications Band 14 Heft 2 S.73-79, IEEE Intelligent Systems, 1999.

[FG02] Mariano Fernández-López, Asunción Gómez-Pérez: Overview and analysis of methodologies for building ontologies, The Knowledge Engineering Review Band 17 Heft 2 S.129-156, Cambridge, Cambridge University Press, April 2002.

[FG97] Mariano Fernández-López, Asunción Gómez-Pérez: METHONTOLOGY: From ontological art towards ontological engineering, AAAI 1997 Spring Symposium on Ontological Engineering S.33-40, Stanford, AAAI Press, 1997.

[FH97] Natalya Fridman Noy, Carole D. Hafner: The State of the Art in Ontology Design – A Survey and Comparative Review, AI Magazine Band 18 Heft 3 S.53-74, AAAI Press, 1997.

[FL06] Joel Farrell, Holger Lausen: Semantic Annotations for WSDL, W3C Working Draft, http://www.w3.org/TR/2006/WD-sawsdl-20060928/, September 2006.

[GH+96] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software, Addison-Wesley-Longman, 1996.

[GM+03] John H. Gennari, Mark A. Musen, Ray W. Fergerson, William E. Grosso, Monica Crubézy, Henrik Eriksson, Natalya F. Noy, Samson W. Tu: The Evolution of Protégé: An Environment for Knowledge-Based Systems Development, International Journal of HumanComputer Studies Band 58 Heft 1 S.89-123, Elsevier ScienceDirekt, 2003.

[Gó99] Asunción Gómez-Pérez: Ontological Engineering: A State of the Art, Expert Update Band 2 Heft 3 S.33-43, The University of Liverpool, The Specialist Group on Artificial Intelligence, 1999.

[Gr93] Thomas R. Gruber: A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition Band 5 Heft 2 S.199-220, Academic Press, April 1993.

[HB04] Hugo Haas, Allen Brown: Web Services Glossary, W3C Working Group Note, http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/, Februar 2004.

[HK+04] Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, Chris Wroe: A Practical Guide to Building Ontologies Using the Protégé OWL Plugin and CO-ODE Tool, http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf, Juni 2004.

[HP+04] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean: SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/, Mai 2004.

[IEEE-DSLCP-95] IEEE Standard for Developing Software Life Cycle Processes (1074-1995), http://ieeexplore.ieee.org/servlet/opac?punumber=3513, April 1996.

[IETF-HTTP1.1] IETF Request for Comments 2616: Hypertext Transfer Protocol – HTTP/1.1 http://tools.ietf.org/html/rfc2616, June 1999.

[IETF-SMTP] IETF Request for Comments 2821: Simple Mail Transfer Protocol, http://tools.ietf.org/html/rfc2821, April 2001.

[JM+03] Stefan Jablonski, Christian Meiler, Ilia Petrov: Web-Services und Semantic Web, Handbuch der Maschinellen Datenverarbeitung HMD – Prasix der Wirtschaftsinformatik Heft 234 S.78-86, dpunkt, Dezember 2003.

[KE+03] Karsten Krutz, Matthias Eck, Christian Mayerl, Matthias Riechmann, Sebastian Abeck: Semantische Suche zur Unterstützung des Internet-basierten Wissenstransfers, Kommunikation in Verteilten Systemen 2003 (KiVS'03) S.257-268, Leipzig, VDE Verlag, Februar 2003.

[KK+05a] Michael Klein, Birgitta König-Ries, Michael Müssig: What is Needed for Semantic Service Descriptions? – A Proposal for Suitable Language Constructs, International Journal of Web and Grid Services 2005 Band 1 Heft 3/4 S.328-364, Inderscience Publishers, 2005.

[KK03] Michael Klein, Birgitta König-Ries: A Process and a Tool for Creating Service Descriptions Based on DAML-S, 2003 VLDB Workshop on Technologies for E-Services (TES'03), Berlin, September 2003.

[Kl04] Michael Klein: Handbuch zur DIANE Service Description, Technical Report TR 2004-17, ISSN 1432-7864, Fakultät für Informatik, Universität Karlsruhe (TH), Karlsruhe, Dezember 2004.

[Kl06] Michael Klein: Automatisierung dienstorientierten Rechnens durch semantische Dienstbeschreibung, Dissertation, Friedrich-Schiller-Universität Jena, Fakultät für Mathematik und Informatik, 2006.

[La05] Lee W. Lacy: OWL: Representing information using the Web ontology language, Victoria, Trafford, 2005.

[Li04] Stefan Link: Service-Design-Methode zur Leistungsbeschreibung eines ITDienstleisters, Diplomarbeit, Universität Karlsruhe (TH), C&M (Prof. Abeck), 2004.

[Li05] Ke Li: Lumina: Using WSDL-S For Web Service Discovery, Master Thesis, University of Georgia, LSDIS (John A. Miller), 2005.

[MB+04] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, Katia Sycara: OWL-S: Semantic Markup for Web Services, W3C Member Submission, http://www.w3.org/Submission/OWL-S/, November 2004.

[Mc04] Brian McBride: The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS, Handbook On Ontologies S.51-65, International Handbooks on Information Systems, Springer, 2004.

[Mi04] Riichiro Mizoguchi: Ontology Engineering Environments, Handbook On Ontologies S.275-295, International Handbooks on Information Systems, Springer, 2004.

[MP+04] David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, Katia Sycara: Bringing Semantics to Web Services: The OWL-S Approach, 2004 International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'04), San Diego, LNCS 3387 S.26-42, Springer, 2004

[MS+01] Alexander Mädche, Steffen Staab, Rudi Studer: Ontologien, Witschaftsinformatik Band 43 Heft 4 S.393-395, Wiesbaden, Vieweg Verlag, Februar 2001.

[NF+91] Robert Neches, Richard Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, William R. Swartout: Enabling Technology for Knowledge Sharing, AI Magazine Band 12 Heft 3 S.36-56, Menlo Park, AAAI Press, 1991.

[OASIS-BPEL1.1] Organization for the Advancement of Structured Information Standards (OASIS): Web Services Business Process Execution Language (WSBPEL), Version 1.1, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, Mai 2003.

[OASIS-UDDI2.0-API] Organization for the Advancement of Structured Information Standards (OASIS): Universal Description, Discovery and Integration Version 2.04 API Specification, http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf, Juli 2002.

[OASIS-UDDI2.0-DS] Organization for the Advancement of Structured Information Standards (OASIS): Universal Description, Discovery and Integration Version 2.03 Data Structure Reference, http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.pdf, Juli 2002.

[OASIS-UDDI2.0-OS] Organization for the Advancement of Structured Information Standards (OASIS): Universal Description, Discovery and Integration Version 2.01 Operator's Specification, http://uddi.org/pubs/Operators-V2.01-Published-20020719.pdf, Juli 2002.

[OASIS-UDDI2.0-RS] Organization for the Advancement of Structured Information Standards (OASIS): Universal Description, Discovery and Integration Version 2.03 Replication Specification, http://uddi.org/pubs/Replication-V2.03-Published-20020719.pdf, Juli 2002.

[OASIS-UDDI3.0] Organization for the Advancement of Structured Information Standards (OASIS): Universal Description, Discovery and Integration Version 3.0.2, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, Februar 2005

[OMG-UML2.0-IS] Object Management Group (OMG): Unified Modelling Langueage Version 2.0 Infrastructure Specification, http://www.omg.org/technology/documents/formal/uml.htm, Juli 2005.

[OMG-UML2.0-SS] Object Management Group (OMG): Unified Modelling Langueage Version 2.0 Superstructure Specification, http://www.omg.org/technology/documents/formal/uml.htm, Juli 2005.

[OR49] Charles K. Ogden, Ivor A.Richards: The Meaning of Meaning, London, Routledge & Paul, 1949.

[PK+02] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, Katia Sycara: Importing the Semantic Web in UDDI, Web Services E-Business and the Semantic Web CAiSE 2002 International Workshop (WES'02), Toronto, LNCS 2512 S.225-236, Springer, Mai 2002.

[PO+04] Abhijit Patil, Swapna Oundhakar, Amit Sheth, Kunal Verma: METEOR-S Web service Annotation Framework, 2004 International Conference on World Wide Web S.553-562, New York, ACM Press, Mai 2004.

[Pr03] Rubén Prieto-Díaz: A Faceted Approach to Building Ontologies, IEEE 2003 International Conference on Information Reuse and Integration (IRI '03) S.458-465, Las Vegas, IEEE Systems, Oktober 2003.

[Pr04] Chris Preist: A Conceptual Architecture for Semantic Web Services (Extended version), HP Laboratories, Bristol, November 2004.

[PU+03] Lutz Prechelt, Barbara Unger, Michael Philippsen, Walter Tichy: A controlled experiment on inheritance depth as a cost factor for code maintenance, Journal of Systems and Software Band 58 Heft 1 S.115-126, Elsevier ScienceDirekt, 2003.

[RM+05] Preeda Rajasekaran, John Miller, Kunal Verma, Amit Sheth: Enhancing Web Services Description and Discovery to Facilitate Composition, 2004 International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'04) S.34-47, 2004 International Conference on Web Services (ICWS'04), San Diego, Juli 2004.

[Ru06] Erik Rull: Serviceorientierte Lehrveranstaltungsunterstützung im Kontext der Lehrmaterialerstellung, Diplomarbeit, Universität Karlsruhe (TH), C&M (Prof. Abeck), 2006.

[Sa06] Reka Marta Sabou: Building Web Service Ontologies, Ph.D. Thesis, Vrije Univeristeit Amsterdam, Knowledge Representation and Reasoning Group, 2006.

[SB+98] Rudi Studer, V. Richard Benjamins, Dieter Fensel: Knowledge Engineering: Principles and Methods, Data & Knowledge Engineering Band 25 Heft 1-2 S.161-198, Elsevier ScienceDirekt, 1998.

[SH+03] Evren Sirin, James Hendler, Bijan Parsia: Semi-automatic Composition of Web Services using Semantic Descriptions, 2003 Web Services: Modeling, Architecture and Infrastructure Workshop (WSMAI'03), 2003 International Conference on Enterprise Information Systems (ICEIS'03), Angers, April 2003.

[SM87] Gerard Salton, Michael J. McGill: Information Retrieval – Grundlegendes für Informationswissenschaftler, McGraw-Hill, 1987

[SS+04a] York Sure, Steffen Staab, Rudi Studer: On-To-Knowledge Methodology (OTKM), Handbook On Ontologies S.117-132, International Handbooks on Information Systems, Springer, 2004.

[SS02] York Sure, Rudi Studer: On-To-Knowledge Methodology – Final Version, On-To-Knowledge Deliverable 18, Universität Karlsruhe (TH), AIFB, http://www.ontoknowledge.org/, 2002.

[St02] Steffen Staab: Wissensmanagement mit Ontologien und Metadaten, Habilitationsschrift, Universität Karlsruhe (TH), AIFB, Informatik Spektrum Band 25 Heft 3 S.194-209, Springer, 2002.

[St02a] Michael Stollberg: Ontologiebasierte Wissensmodellierung – Verwendung als semantischer Grundbaustein des Semantic Web, Magisterarbeit, Freie Universität Berlin, Institut für Publizistik- und Kommunikationswissenschaft, 2002.

[SV+03] Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth, John Miller: Adding Semantics to Web Services Standards, Proceedings of The 2003 International Conference on Web Services (ICWS'03) S.395-401, Las Vegas, Juni 2003.

[UK95] Mike Uschold, Martin King: Towards a Methodology for Building Ontologies, Workshop on Basic Ontological Issues in Knowledge Sharing, 1995 International Joint Conference on Artificial Intelligence (IJCAI'95), Montreal, August 1995.

[UM+03] Mike Ullrich, Andreas Maier, Jürgen Angele: Taxonomie, Thesaurus, Topic Map, Ontologie – ein Vergleich, Ontoprise Whitepaper Series, Ontoprise GmbH Karlsruhe, http://www.ontoprise.de/content/e1276/e1358/e1362/TaxonomieThesaurusTopicMapOntologiev13_ger.pdf, 2003.

[VW05a] Gottfried Vossen, Peter Westerkamp: Service-Oriented Provisioning of Learning Objects, Arbeitsbericht Nr. 2 des E-Learning Kompetenzzentrums Münster, ERCIS – European Research Center for Information Systems, Universität Münster, 2005, Hrsg.: H. L. Grob, J. vom Brocke, Münster 2005.

[W3C-OWL-O] W3C Recommendation: OWL Web Ontology Language Overview, http://www.w3.org/TR/2004/REC-owl-features-20040210/, Februar 2004.

[W3C-RDF-P] W3C Recommendation: RDF Primer, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, Februar 2004.

[W3C-RDF-S1.0] W3C Recommendation: RDF Vocabulary Description Language 1.0 RDF Schema, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/, Februar 2004.

[W3C-SOAP1.2] W3C Recommendation: SOAP Version 1.2 Part 1: Messaging Framework, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/, Juni 2003.

[W3C-WSDL1.1] W3C Note: Web Services Description Language (WSDL) Version 1.1, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, März 2001.

[W3C-XML] W3C Recommendation: XML Version 1.0 (Fourth Edition), http://www.w3.org/TR/2006/REC-xml-20060816, August 2006.

[W3C-XS-DT] W3C Recommendation: XML Schema Part 2: Datatypes Second Edition, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/, Oktober 2004.

Webreferenzen

[URL-01] Arbeitsgruppe Lehrunterstützung der Fakultät für Informatik http://alfi.ira.uka.de

[URL-02] Karlsruher integriertes Informationsmanagement
http://www.kim.uni-karlsruhe.de

[URL-03] World Wide Web Consortium (W3C)
http://www.w3.org

[URL-04] Organization for the Advancement of Structured Information Standards (OASIS)
http://www.oasis-open.org/http://www.oasis-open.org/

[URL-05] W3C Recommendation: XML Schema 1.0 Specification
http://www.w3.org/XML/Schema#dev

[URL-06] W3C Working Group: Semantic Annotations for Web Services Description Language
http://www.w3.org/2002/ws/sawsdl/

[URL-07] W3C Submission Request: Web Service Description Language - Semantics (WSDL-S)
http://www.w3.org/Submission/2005/10/

[URL-08] W3C Submission Request: OWL Web Ontology Language for Services (OWL-S)
http://www.w3.org/Submission/2004/07/

[URL-09] ESSI Working Group: Web Service Modeling Ontology (WSMO)
http://www.wsmo.org

[URL-10] W3C Submission Request: Web Service Modeling Ontology (WSMO)
http://www.w3.org/Submission/2005/06/

[URL-11] Semantic Web Services Initiative
http://www.swsi.org

[URL-12] W3C Submission Request: Semantic Web Services Framework (SWSF)
http://www.w3.org/Submission/2005/07/

[URL-13] Universität Karlsruhe (TH), Friedrich-Schiller-Universität Jena: Dienste in Ad-hoc-Netzen (DIANE)
http://hnsp.inf-bb.uni-jena.de/DIANE/

[URL-14] DARPA Agent Markup Language (DAML): Web Ontology Language for Services
http://www.daml.org/services/owl-s/

[URL-15] W3C Member Submission: OWL-S' Relationship to Selected Other Technologies
http://www.w3.org/Submission/OWL-S-related/

[URL-16] University of Georgia, Large Scale Distributed Information Systems (LSDIS): METEOR-S – Semantic Web Services and Processes
http://lsdis.cs.uga.edu/projects/meteor-s/

[URL-17] University of Georgia, Large Scale Distributed Information Systems (LSDIS): Web Service Description Language - Semantics (WSDL-S)
http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/

[URL-18] University of Georgia, Large Scale Distributed Information Systems (LSDIS): Lumina – Semantic Web Service Discovery
http://lsdis.cs.uga.edu/projects/meteor-s/downloads/Lumina/

[URL-19] Information Society Technologies (IST) Program for Research: On-To-Knowledge
http://www.ontoknowledge.org

[URL-20] Information Society Technologies (IST)
http://cordis.europa.eu/ist/

[URL-21] Stanford University: Protégé Ontology Editor
http://protege.stanford.edu

[URL-22] The Eclipse Framework
http://www.eclipse.org/eclipse/

[URL-23] WSDL4J – Java-Klassenbibliothek zur Bearbeitung von WSDL-Dateien
http://sourceforge.net/projects/wsdl4j

[URL-24] UDDI4J – Java-Klassenbibliothek zur Interaktion mit UDDI-Verzeichnissen.
http://uddi4j.sourceforge.net/

[URL-25] Protégé-OWL API – Java-Klassenbibliothek für die Bearbeitung von OWL-Dateien und Kommunikation mit Ontologie-Reasonern
http://protege.stanford.edu/plugins/owl/api/index.html

[URL-26] RacerPro Ontologie-Reasoner
http://www.racer-systems.com/

[URL-27] Description Logic Implementation Group
http://dl.kr.org/dig/

[URL-28] DIG 2.0: The DIG Description Logic Interface – Document Index
http://dig.cs.manchester.ac.uk/

[URL-29] Pellet OWL-Reasoner
http://www.mindswap.org/2003/pellet/

[URL-30] XAMPP
http://www.apachefriends.org/de/xampp.html

[URL-31] Apache HTTP-Server
http://httpd.apache.org/

[URL-32] MySQL
http://www.mysql.org/

[URL-33] Apache Tomcat
http://tomcat.apache.org/

[URL-34] The Apache Software Foundation
http://www.apache.org/

[URL-35] jUDDI
http://ws.apache.org/juddi/

[URL-36] Java 2 Standard Edition Runtime Environment
http://java.sun.com/j2se/desktopjava/jre/

[URL-37] MySQL Connector/J
http://www.mysql.com/products/connector/j/

[URL-38] Intalio Designer
http://www.intalio.com/products/designer/

[URL-39] Eclipse Web Tools Platform
http://www.eclipse.org/webtools/