



DAML-S, Service Markup for the Semantic Web

Christoph Gesperger¹, Gernot Goluch²

¹ Matrikelnummer: 0000303, Studienrichtung: Wirtschaftsinformatik (A175)
gesperger@aon.at

² Matrikelnummer: 0026039, Studienrichtung: Wirtschaftsinformatik (E175)
ggoluch@hotmail.com

Inhaltsverzeichnis

1	EINLEITUNG	1
2	SEMANTIC WEB.....	2
2.1	SERVICES IM SEMANTIC WEB.....	2
2.2	EXKURS: WEB AGENTEN	3
3	DAML-S, AUFBAU UND STRUKTUR.....	4
3.1	KLASSEN	5
3.2	BESCHREIBUNG EINES WEB-SERVICE	6
3.2.1	SERVICEPROFIL	6
3.2.2	SERVICEMODELL.....	9
3.2.3	SERVICEGROUNDING	12
4	ANWENDUNG IM B2C-BEREICH	12
4.1	PROZESSE & PROGRAMME EINES BOOKSELLERS	12
4.2	PROZESSE ZUSAMMENFASSEN & VEREINFACHEN.....	14
4.3	WERBUNG FÜR DAS SERVICE	14
5	ZUKUNFTAUSSICHTEN	16
	LITERARURLISTE	17

DAML-S, Service Markup for the Semantic Web

Christoph Gesperger¹, Gernot Goluch²

¹ Matrikelnummer: 0000303, Studienrichtung: Wirtschaftsinformatik (A175)
gesperger@aon.at

² Matrikelnummer: 0026039, Studienrichtung: Wirtschaftsinformatik (E175)
ggoluch@hotmail.com

Abstract. The world-wide-web, as we know it now, could become obsolete in the nearest future, because it seems that it is gradually changing into a semantic web. This new extension of the internet offers lots of new and interesting possibilities. Web agents should help us with our daily routines, which are supported by web-services. It is important, that they are able to perform specific tasks, currently performed manually by human beings, completely on their own. Such tasks include automated web service discovery, automated invocation, automated interoperation, automated selection and composition and automated execution monitoring. Therefore the services on the web need to be described in a language that machines can read and understand. DAML-S provides such a web service description, which describes what a service can do and not just how it does it. In this paper we focus on the description language DAML-S, how it is related to DAML+OIL and compare it to other description standards. In the main part of the paper we describe the semantics of DAML-S in detail. Furthermore we explain, how DAML-S can be used, by presenting an example of a typical web service.

1 Einleitung

Das Internet, wie wir es heute kennen, wurde als Informationsraum entwickelt. Es diente einst nur dazu, Texte und Bilder darzustellen. Zur Zeit gibt es über 300 Millionen statische Dokumente, die von mehr als 100 Millionen Benutzern genutzt werden können. Das exponentielle Wachstum des World-Wide-Webs erschwert jedoch das Auffinden von Informationen. Da die meisten Webseiten in HTML geschrieben sind, können Browser ihren Inhalt dem Benutzer anzeigen. Um Informationen aus dem Internet zu erhalten, ist es notwendig, dass diese vorhanden und auf verschiedenste Wege erreichbar ist. Benutzer können Informationen im World-Wide-Web entweder über Kataloge, wie zum Beispiel YAHOO, oder über eine Suchmaschine, wie zum Beispiel GOOGLE, finden. Suchmaschinen sind in der Lage, das Internet zu durchsuchen und die gefundenen Seiten zu indizieren. Zum Leidwesen des Benutzers kann es beim Suchen aber zu Problemen kommen. Es kommt sehr häufig vor, dass die Benutzer entweder kein Ergebnis oder viele irrelevante Ergebnisse präsentiert bekommen. Dies kommt dadurch zustande, weil Suchmaschinen nicht fähig sind, den Kontext von Wörtern sowie ihre Beziehung zueinander festzustellen.

Ein Grund für dieses Problem ist, dass Seiten im World-Wide-Web meist nur aus Inhalten bestehen, die für den Menschen verständlich sind, wie zum Beispiel Texte und Bilder, aber auch Audio- und Videostreams. Unglücklicherweise verstehen nun Maschinen, insbesondere Computer, diese Inhalte nicht. Sie sind zwar in der Lage, die Inhalte zu lesen und darzustellen, sie können sie jedoch nicht verarbeiten oder verstehen. Im Semantic Web werden nun zu den Inhalten, die nur für Menschen verständlich sind, maschinenlesbare Informationen hinzugefügt. Es ist nun möglich, einzelnen Textabschnitten, ganzen Texten, Bildern oder sogar ganzen Seiten eine Bedeutung zuzufügen. Die Bedeutung von Services im Semantic Web, sowie die Funktion von Web-Agenten werden im folgenden Teil der Arbeit genauer beschrieben.

2 Semantic web

2.1 Services im semantic web

Das Semantic Web ist kein eigenständiges Web, sondern eine Erweiterung des bestehenden World-Wide-Web. Der Aufbau des Semantic Web ist bereits im Gang und wird durch die Entwicklung von Markup-Standards, wie zum Beispiel OIL, DAML+OIL, oder DAML-S, vorangetrieben. Eine der wichtigsten Komponenten des Semantic Web ist natürlich die Beschreibung von Web-Services, um sie für Computer und Web-Agenten interpretierbar zu gestalten. Agenten, wie sie weiter unter in dieser Arbeit beschrieben werden, sollen dadurch in der Lage sein, selbständig Web-Services im Internet ausfindig zu machen, diese in Anspruch zu nehmen und auszuführen. Um dies zu ermöglichen müssen Aufgaben, wie das Auffinden, das Ausführen und die Komposition von Web-Services automatisiert werden.

In der Vergangenheit konzentrierte man sich primär darauf, das Auffinden und Ausführen von Web-Services zu verbessern. Dazu versuchte man Standards für E-Commerce Applikationen, genauer gesagt für die Beschreibung von Web-Services zu entwickeln. Das Ergebnis waren Web-Service Standards, wie UDDI, WSDL oder ebXML.

UDDI (Universal Description, Discovery and Integration) ist eine globale Verzeichnisstruktur für Web-Services. Diese soll dazu dienen, um Web-Services zu finden und zu veröffentlichen. Es stellt gewissermaßen eine erweiterte Version des „Telefonbuchs“ dar, mit dem Ziel, die Registrierung und Suche jeglicher Information zu unterstützen. Ohne einen solchen Dienst, mit dem auch Programme nach Web-Services suchen können, wäre es kaum möglich, Web-Services in andere Anwendungen ohne menschliches Eingreifen, dynamisch zu integrieren. An der Entwicklung und Spezifikation von UDDI sind vor allem Microsoft und IBM beteiligt. UDDI ermöglicht es, anhand von Services einzelner Firmen, potentielle Geschäftspartner zu finden und neue Geschäftsbeziehungen zu knüpfen. Das oberste Ziel von UDDI ist die Automatisierung von Geschäftstransaktionen in B2B E-Commerce Applikationen.

WSDL (Web Service Description Language) ist eine Sprache, welche auf XML basiert und die Funktionalität von Web-Services beschreibt. Mit Hilfe von WSDL ist es möglich, die Operationen, welche Web-Services ausführen und die Parameter, die diese Operationen akzeptieren und wiedergeben, zu beschreiben. DAML-S steht eine Stufe über WSDL, da es nicht nur beschreibt, welche Informationen über das World-Wide-Web gesendet werden und warum, sondern auch wie dies geschieht.

Die Beschreibungssprache ebXML wurde von OASIS (Organisation for the Advancement for Structured Information Standards) und den Vereinten Nationen entwickelt. ebXML verwendet zwei Sichtweisen, um Geschäftsinteraktionen zu beschreiben. Eine Sichtweise beschäftigt sich mit der Semantik von Geschäftsdatentransaktionen, die andere mit den Services, welche diese unterstützen. Obwohl sich ebXML nicht allein auf Web-Services konzentriert, ist das Ziel das gleiche wie bei DAML-S. Solange es jedoch nicht Information verwendet, die von DAML-S bereitgestellt werden, wird ebXML nicht in der Lage sein, die automatische Interaktion von Web-Services zu unterstützen.

2.2 Exkurs: Web Agenten

Web-Agenten sind autonome Programme, die Informationen selbständig aus dem World-Wide-Web beziehen und Internetservices ausführen können. Die Beschreibung von Web-Services durch DAML-S bildet die Grundlage für die Nutzung von Web-Services durch Web-Agenten. Die Semantik, mit der Markup-Sprachen, wie DAML-S, den Inhalt von Internetseiten versehen, ermöglicht es Web-Agenten, sich von Seite zu Seite zu bewegen, um wichtige Informationen für seinen Benutzer zu sammeln oder komplizierte Aufträge auszuführen.

Damit ein Web-Agent auch die an ihn gestellten Aufgaben erfüllen kann, muss er gewisse Eigenschaften [1] haben: Er muss kommunikativ, leistungsfähig, selbstständig und anpassungsfähig sein.

- **Kommunikativ:** Damit es möglich ist, mit dem Web-Agenten zu kommunizieren, ist es notwendig, dass dieser als Grundvoraussetzung das Vokabular des Themengebietes beherrscht. Es ist jedoch nicht ausreichend, die Wörter an sich zu kennen. Vielmehr ist es wichtig, in einem gewissen Sinne deren Bedeutung zu verstehen.

Dieses Manko ist an den heutigen Suchmaschinen deutlich zu erkennen. Sie betrachten lediglich das Auftauchen bestimmter Schlüsselwörter als Grundlage für die Relevanz eines Dokumentes und liefern aus diesem Grund leider oft falsche Ergebnisse. Web-Agenten dagegen greifen bei ihrer Suche auf Ontologien zurück, welche ein einheitliches Vokabular für den Benutzer und den Web-Agenten als Grundlage für die Kommunikation bieten.

- **Leistungsfähig:** Web-Agenten müssen in Lage sein, bestimmte Aktionen auszuführen, zum Beispiel Kinokarten reservieren. Es ist daher wichtig, dass sie nicht nur geeignete Web-Services auffinden, sondern diese auch gleich ausführen oder anwenden können. Als Grundlage machen sie Gebrauch vom Semantic Web, welches hierfür eine Schnittstelle darstellt.
- **Selbständig:** Web-Agenten sind durch ihre Leistungsfähigkeit in der Lage, Aktionen auszuführen, bei denen möglicherweise Ressourcen verbraucht werden. Dies können entweder finanzielle Ressourcen des Benutzers, zum Beispiel beim Kauf einer Kinokarte, aber auch Hardwareressourcen, wie zum Beispiel der Platz auf der Festplatte, beim Download einer großen Datenmenge, sein. Es ist daher notwendig, im vor hinein festzulegen, inwieweit Web-Agenten diese Aktionen selbständig, ohne weiter nachzufragen, durchführen dürfen. Für diesen Zweck wäre es denkbar Modelle zur Einschränkung einzusetzen, um einen gewissen, sinnvollen Grad an Selbständigkeit zu erreichen. Beide Extreme, das heißt, vollständige Kontrolle oder vollständige Unabhängigkeit, sind sicher nicht wünschenswert. Der Web-Agent wäre einerseits keine große Hilfe, andererseits auch ein Risiko, weil er durch Fehlinterpretationen oder Fehlentscheidungen Ressourcen verschwenden könnte. Eine sinnvolle Möglichkeit, um dieses Problem zu lösen, wäre, eine Grenze bezüglich bestimmter Faktoren, wie Kosten oder Datenmengen, festzulegen, bis zu denen der Web-Agent frei agieren kann. Sollte diese Grenzen überschritten werden, muss aber eine Rücksprache mit dem Benutzer gehalten werden.
Ein völlig anderer Ansatz wäre, den Web-Agenten vom Benutzer lernen zu lassen, damit er sein eigenes Verhalten dem des Benutzers anpassen kann.
- **Anpassungsfähig:** Eine weitere Eigenschaft, die ein Web-Agent haben sollte, ist Anpassungsfähigkeit. Darunter versteht man, dass er die Präferenzen des Benutzers berücksichtigen kann, oder in der Lage ist, das Verhalten des Benutzers vorauszusagen und daraus zu lernen. Die Lernfähigkeit des Web-Agenten könnte sich zum Beispiel dadurch realisieren lassen, indem ein neuer Benutzer ein Formular mit Fragen zu bestimmten Bereichen ausfüllt. Anschließend kann er anhand der Antworten auf die Fragen klassifiziert werden. Die Präferenzen von anderen Nutzer derselben Klasse könnten dann als Ausgangspunkt genommen und während der Benutzung an das spezifische Verhalten angepasst werden.

Um den oben genannten Anforderungen und Eigenschaften gerecht zu werden, muss ein Web-Agent eine Reihe von Aufgaben [1] erfüllen können.

- **Automatisches Auffinden von Web-Services:** Web-Agenten müssen im Stande sein, Web-Services, die eine bestimmte Funktion anbieten und auch gewisse Nebenbedingungen, im World-Wide-Web ausfindig zu machen. Der Benutzer könnte zum Beispiel Karten für eine Kinovorstellung reservieren wollen, die er mit einer bestimmten Kreditkarte bezahlen möchte. Sind nun die nötigen Informationen eines solchen Web-Services mit Hilfe des Serviceprofils von DAML-S innerhalb der entsprechenden Webseite kodiert, kann sich der Web-Agent nun mit einer Anfrage an Serviceverzeichnisse, oder Ontologie-erweiterte Suchmaschinen wenden, um einen geeigneten Web-Service zu finden.
- **Automatisches Aufrufen von Web-Services:** Wenn der Web-Agent ein bestimmtes Service gefunden hat, ist es für ihn wichtig, zu wissen, welche Funktionen des Web-Services er aufrufen muss, und wie dies funktioniert. Jetzt kommen das Servicemodell und das Servicegrounding von DAML-S ins Spiel, welche weiter unten in dieser Arbeit noch ausführlich beschrieben werden. Hier sind auf abstrakter Ebene die einzelnen Funktionen des Web-Services und auf konkreter Ebene die Konstruktion eines entsprechenden Aufrufes beschrieben.

- **Automatische Komposition von Web-Services:** Damit der Web-Agent auch komplexere Aufgaben, wie zum Beispiel die Planung einer Reise, erledigen kann, ist es notwendig, einzelne Web-Services zu kombinieren. Nachdem der Web-Agent beispielsweise einen Flug über einen Web-Service gebucht hat, muss er noch ein Busticket über einen anderen Web-Service buchen, damit der Reisende vom Flughafen zum Hotel fahren kann, das wiederum über einen anderen Web-Service gebucht wurde. Die einzelnen Buchungen müssen hier natürlich auf einander abgestimmt sein. Das Busticket sollte an dem Tag gültig sein, an dem der Reisende am Flughafen eintrifft, usw... Um dies erfüllen zu können, muss der Web-Agent über die Vorbedingungen und Effekte der Ausführung der einzelnen Web-Services Bescheid wissen. Diese Information liefert das entsprechende Servicemodell des Web-Services.

Web-Agenten mit den oben beschriebenen Eigenschaften sind jedoch noch Zukunftstechnologie. Das semantische Web, Web-Services und Web-Agenten gehören zu den Forschungsbereichen, die sich zur Zeit am schnellsten weiter entwickeln. Web-Agenten, wie oben beschrieben, sind im Moment noch Visionen oder existieren nur in Testumgebungen von Forschungseinrichtungen. Die Eigenschaften und Anforderungen für die Agenten sind wünschenswert, jedoch noch nicht realisierbar, da noch kein einheitlicher Standard für die Beschreibung von Web-Services existiert. Ein Werkzeug zur Beschreibung von Web-Services ist die Markup-Sprache DAML-S, auf deren Aufbau und Semantik nun im folgenden Teil der Arbeit näher eingegangen wird.

3 DAML-S, Aufbau und Struktur

Um angebotene Web Services zu nutzen, ist es notwendig, dass für einen Software Agent eine Computer interpretierbare Beschreibung des Services vorhanden ist. DAML soll die Modellierungswerkzeuge bereitstellen, mit welchen diese Beschreibungen gemacht und kommuniziert werden.

Web Services können entweder einfach oder komplex sein. Einfache Services sind Services, für welche es nicht notwendig ist, dass vorher bereits ein anderer Service durchgeführt wurde. Sie bestehen meist nur aus einer einzigen Transaktion zwischen Client und Server. Ein Beispiel wäre ein Service, welcher die Postleitzahl einer übergebenen Adresse zurückliefert. Komplexe Services sind nichts anderes, als die Durchführung von mehreren einfachen Services hintereinander. Außerdem interagieren bei komplexen Services Client und Server stark miteinander. Ein Beispiel für ein komplexes Service wäre eine Bestellung bei Amazon. Zuerst wird nach dem Buch gesucht, dann kann der Benutzer entscheiden, ob ihn das Buch interessiert. Entweder kauft er es oder er kauft es nicht. Wenn er es kauft, teilt er Kreditkarteninformationen und Informationen für die Versendung des Buches mit.

DAML kann beide Service Arten abbilden. Damit der Prozess schlussendlich funktioniert, sind vier Modellierungsbereiche notwendig.

- **Automatisiertes Auffinden von Web Services**
- **Automatisiertes Aufrufen von Web Services**
- **Automatisierte Komposition und Interoperation von Web Services**
- **Automatisierte Überprüfung der Durchführung von Web Services**

DAML+OIL ist eine Implementierung von DAML, welche OIL als Ontologiesprache verwendet. DAML+OIL beschreibt abstrakte Kategorien von Dingen und Ereignissen mit Hilfe von Klassen und Eigenschaften.

DARPA Agent-Markup Language for Services (DAML-S) ist eine ausdrucksvolle semantische Markup-Sprache mit einer wohl definierten Semantik, die auf DAML+OIL basiert. DAML-S wurde entwickelt, um Web-Services zu beschreiben. DAML-S legt eine Menge von Klassen und Eigenschaften zur Beschreibung von Web-Services fest, indem es eben diese Sprache benutzt. Bei DAML-S handelt es sich also um eine DAML+OIL Ontologie zur Beschreibung von Web-Services.

Ontologien sind im Grunde nichts anderes, als Repräsentationen von gemeinsamen Konzeptionalisierungen einer bestimmten Domäne. Sie wurden im Rahmen der künstlichen Intelligenz entwickelt. Ontologien stellen ein gemeinsames Verständnis einer Domäne bereit, das dann zwischen Personen und Anwendungssystemen ausgetauscht werden kann. Es ist üblich, dass Ontologien hierarchische Beschreibungen von wichtigen Konzepten in einer Domäne beinhalten.

Eines der Ziele von DAML-S ist es, einen Web-Service zu finden, der einen speziellen Service anbietet, aufgrund von genau spezifizierten Suchkriterien. Die Benutzung des Services sollte möglich sein, ohne die Beschreibung zu lesen und danach den Service aufzurufen. Mit einer Suchmaschine, die die Ontologien berücksichtigt, kann ein Service automatisch gefunden werden. Als Alternative dazu kann auch die UDDI benutzt werden, um einen Anbieter zu suchen und dann kann innerhalb des Anbieters im DAML-S Profil nach einem bestimmten Service gesucht werden.

Die Beschreibung von Web-Services durch DAML-S bildet die Grundlage für die Nutzung von Web-Services durch Web-Agenten (Fig. 1).

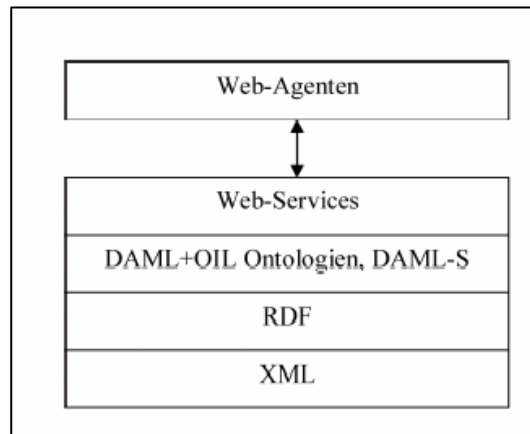


Fig.1 Gesamtüberblick: Semantisches Web und Web-Agenten [1]

Es liegt nun eine maschinenlesbare Beschreibung des Ein- und Ausgabeverhaltens, der Vorbedingungen und Effekte und der Funktion der Web-Services vor. All dies beruht auf einem einheitlichen Vokabular, festgehalten in Ontologien.

DAML-S ist also wie vorher erklärt eine DAML+OIL Ontologie, die zur Beschreibung von Web Services dient. Im Folgenden werden wir nun den Aufbau und die Struktur von DAML-S beschreiben.

3.1 Klassen

Die Oberklasse von DAML-S heißt SERVICE. Diese Klasse enthält nur sehr wenige konkrete Details über die Implementierung des Web Service.

Es wird hier auch noch keine konkrete Aussage über die Anzahl oder die Art der Unterklassen gemacht die verschiedene Arten von Web Services beschreiben werden.

Die Klasse SERVICE beschreibt also eigentlich nur den organisatorischen Weg für eine Beschreibung eines Web-Service, auf den wir im Folgenden noch ausführlich eingehen werden (siehe 3.2).

Wahrscheinlich wird die Klassifizierung dieser Subclasses durch die Bedürfnisse des Marktes entschieden. Realistisch wäre eine baldige Konkretisierung einer Subclass B2C-TRANSACTIONS, die die Services rund um Internetshops beschreiben würden.

3.2 Beschreibung eines Web-Services

DAML-S definiert nun für einen Web-Service folgende Eigenschaften:

- Jene Daten die ein Web-Service vom jeweiligen Web-Agenten verlangt und jene die er die dem zur Verfügung stellt.
Dies erfolgt durch das so genannte SERVICEPROFIL das die Klasse SERVICE präsentiert. Das SERVICEPROFIL ist eine Klasse die jene Parameter und Leistungen angibt die ein Web-Service beschreiben (Fig. 2).

- Die Arbeit des Web-Service im Detail.
Dies erfolgt wiederum durch das SERVICEMODELL das ebenfalls die Klasse SERVICE beschreibt. Das SERVICEMODELL ist eine Klasse die die möglichen Pfade und den Arbeitsablauf durch den Web-Service beschreibt (Fig. 2).
- Die reale Verwendung des Web-Service.
Dies erfolgt durch das SERVICEGROUNDING das so wie die anderen zwei ebenfalls die Klasse SERVICE beschreibt (Fig. 2).

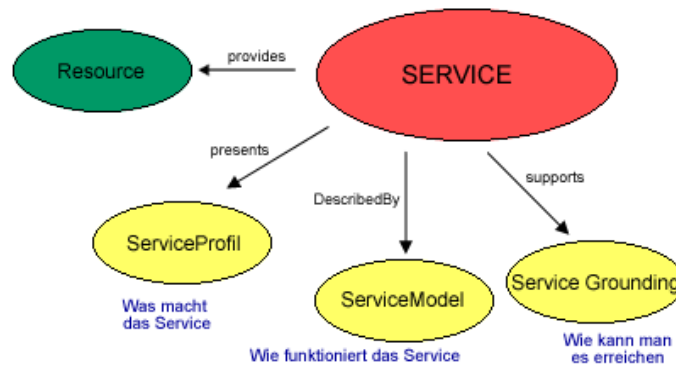


Fig. 2. Top Level of the service ontology [2]

„Each instance of SERVICE will present a descendant class of SERVICEPROFIL be describedBy a descendant class of SERVICEMODEL, and support a descendant class of SERVICEGROUNDING“ [2]

3.2.1 SERVICEPROFIL

Das SERVICEPROFIL erfüllt die Funktion der Repräsentation des Web-Service nach außen.

Ein Web-Agent benützt es um festzustellen um ein Service den gewünschten Anforderungen des Kunden entspricht. Dieses Anforderungsprofil enthält gewisse Nebenbedingungen bezüglich der Qualität, der Preise, etc.

Zur Beschreibung eben dieser erforderlichen Informationen besitzt die Klasse SERVICEPROFIL eine Reihe von Eigenschaften die wir in Folge aufzeigen wollen.

Wir unterteilen diese in vier Klassen¹:

1. Service Profil:

Diese Klasse gibt uns die Basisinformationen die benötigt werden um das Profil mit dem Service zu „verbinden“ (Fig. 3). Ein Service muss mit dem Profil in Verbindung gebracht werden können und umgekehrt. Zu diesem Zwecke stellt die Klasse folgende Eigenschaften zur Verfügung:

Table 1. Service Profil, Eigenschaften

presents	beschreibt die Relation zwischen dem Service und dem Profil. Im Prinzip wird hierdurch nur ausgedrückt das ein gewisses Service durch dieses Profil beschrieben wird.
-----------------	---

¹ Anmerkung: Die hier aufgezählten Eigenschaften sind eine Variante der Implementierung der Klasse SERVICEPROFIL die zur Zeit als jene angesehen wird die vermutlich für die Beschreibung der Klasse adoptiert wird. Es wären auch andere Varianten vorstellbar.

presentedBy	ist genau das Umgekehrte der Aussage presents. Hier wird gesagt dass ein gewisses Profil ein Service beschreibt.
--------------------	--

2. Service Name, Kontakte und Beschreibung:

Ein Service Profil muss mindestens einen Namen und eine Beschreibung besitzen. Diese Klasse bietet sozusagen „human-readable information“ (Fig. 3). Folgende Eigenschaften werden in dieser Klasse definiert:

Table 2. . Service Name, Kontakte und Beschreibung, Eigenschaften

serviceName	bezieht sich auf den Namen des Services und kann somit auch zum identifizieren des Web-Service verwendet werden.
textDescription	Liefert eine kurze Beschreibung des Services. Es beinhaltet eine kurze Zusammenfassung der Angebote, der benötigten Informationen und jegliche zusätzliche Information die der Betreiber des Services seinen Kunden zukommen lassen will.
contactInformation	bietet Informationen die der Benutzer nützen kann um mit dem Betreiber des Services Kontakt aufzunehmen. Jeder zur Verfügung gestellte Kontakt ist eine Instanz der Klasse Actor , die im Folgenden behandelt wird.

Unterklasse: **Actor**:

Die Klasse Actor hat folgende Eigenschaften, die mehr oder weniger selbsterklärend sind:

Table 3. Eigenschaften der Unterklasse Actor

name	title	phone	Fax	email	physicalAddress	webURL
-------------	--------------	--------------	------------	--------------	------------------------	---------------

3. Funktionelle Beschreibung des Service:

Eine der wichtigsten Komponenten des Profils ist die Spezifikation der Funktionalität des Services und die Spezifikation der Bedingungen die erfüllt sein müssen um ein erfolgreiches Resultat z.B.: der B2C-Transaction zu erhalten (Fig. 3).

Der Besitzer des Web-Services muss auch entscheiden welche Funktionalitäten er der Öffentlichkeit zugänglich machen will. Ein Buchhändler kann z.B.: das Browsen nach Büchern als internen Prozess der Öffentlichkeit verschließen den eigentlichen Kaufvorgang jedoch nicht. So würde ein Web-Agent der nach der Möglichkeit des Browsens in einem Bücherbestand sucht von diesem Web-Service nicht angesprochen werden.

Das DAML-S Profil beinhaltet zwei Aspekte der Service Funktionalität: die Transformation der Information und der Zustandswandel der durch die Ausführung des Service erzeugt wird.

Der erste Aspekt der Funktionalität, die Transformation der Information, wird durch die In- bzw. Output Eigenschaften des Web-Services dargestellt.

Table 4. .funktionelle Beschreibung, Eigenschaften 1

input	beschreibt einen Input des Services. Als Wert des Inputs wird eine Instanz der Unterklasse ParameterDescription verwendet
output	beschreibt einen Output des Services. Als Wert des Outputs wird eine Instanz der Unterklasse ParameterDescription verwendet.

Der zweite Aspekt, der Zustandswandel, wird durch die Vorbedingungen und die Effekte des Service Profils spezifiziert.

Table 5. funktionelle Beschreibung, Eigenschaften 2

precondition	beschreibt eine Vorbedingung des Services. Als Wert der Vorbedingung wird eine Instanz der Unterklasse ParameterDescription verwendet.
effect	beschreibt einen Effekt des Services. Als Wert des Effekts wird eine Instanz der Unterklasse ParameterDescription verwendet.

Unterklasse: **ParameterDescription**

Diese Unterklasse sammelt Informationen bezüglich der Parameter: den Namen des Parameters der als ID verwendet werden kann, den Wert des Parameters und die Referenz zu dem korrespondierenden Parameter im Prozess Model, zu dem wir noch später kommen werden.

Table 6. Eigenschaften der Unterklasse ParameterDescription

parameterName	gibt den Namen des Parameters.
restrictedTo	bietet gewisse Einschränkungen auf die Werte der Parameter.
refersTo	beinhaltet die Referenz zum Parameter im Prozess Model.

4. Funktionelle Profil-Attribute:

Diese Attribute beschreiben Qualitätsgarantien, mögliche Klassifikationen des angebotenen Service und weitere mögliche Eigenschaften und Informationen die der Kunden bzw. sein Web-Agent bei der Entscheidungshilfe verwenden kann (Fig. 3).

Table 7. Funktionelle Profil-Attribute, Eigenschaften

serviceParameter	ist eine erweiterbare Liste von Eigenschaften des Web-Service. Der Wert solch einer Eigenschaft ist eine Instanz der Unterklasse
ServiceParameter	
serviceParameterName	ist der Name des jeweiligen Parameters
sParameter	gibt den Wert des Parameters
QualityRanking	ist selbsterklärend und hat die Eigenschaften:
ratingName	gibt den Namen des Rating-Services an
rating	speichert den Wert des momentanen ratings
intendedPurpose	gibt die Information die benötigt wird um ein Service korrekt abzuschließen
providedBy	verbindet das Service Profil mit einem Actor der das Service zur Verfügung stellt
requestedBy	verbindet das Service Profil mit einem Actor der das Service nachfragt
domainResource	spezifische Ressourcen die benötigt werden um die Aufgabe korrekt zu erfüllen
geographicRadius	bezieht sich auf den geographischen Standpunkt des Services
degreeOfQuality	bietet ein Qualitätsmaß des Service an
qualityGuarantee	beinhaltet Garantien die der Anbieter des Service verspricht einzuhalten

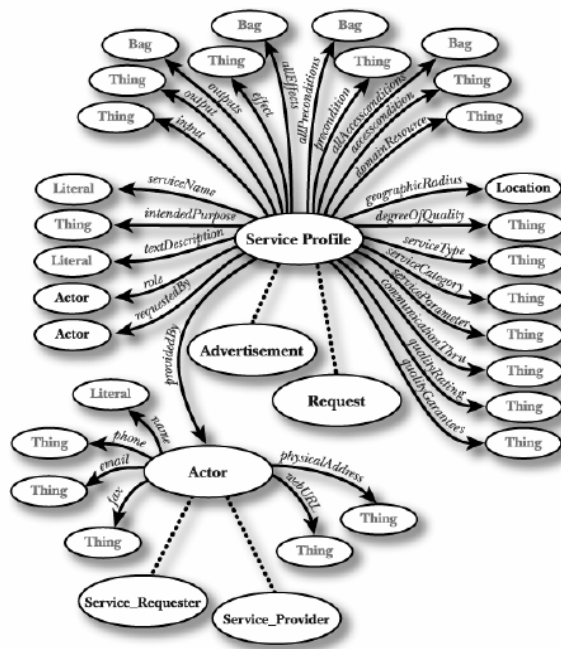


Fig. 3. Upper Ontology of ServiceProfile [6.4]

3.2.2 SERVICEMODELL

Das Service Modell gibt uns an wie das Service nun genau funktioniert, sprich aus welchen einzelnen Schritten es besteht. Zu diesem Zwecke wird der Web-Service als Prozess modelliert. Um diese Modellierung des Prozesses zu verwirklichen wird eine neue Klasse eingeführt: **ProcessModel** (Fig.4).

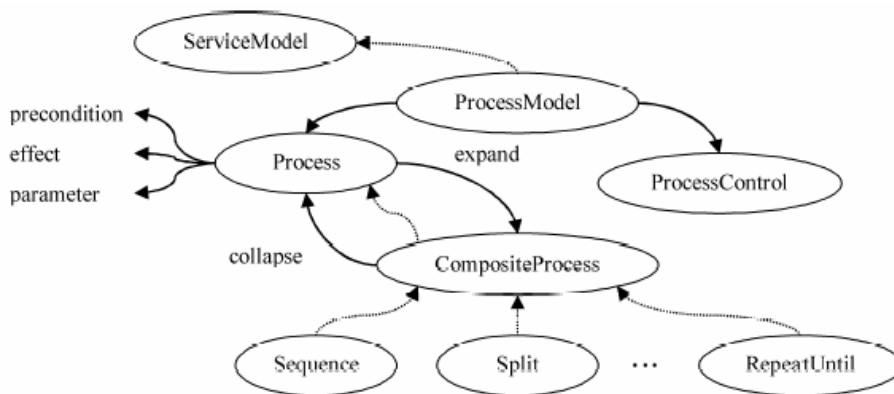


Fig. 4. Das Prozessmodell [1]

Diese Klasse spiegelt mit ihren Eigenschaften und Unterklassen den Datenfluss und die Kontrollstruktur eines Web-Services wieder.

Das Prozessmodell wird nun in zwei Komponenten, die Prozessontologie und die Prozesskontrollontologie, aufgeteilt:

Prozessontologie:

Diese Komponente des Prozessmodells gibt uns Auskunft über das Ein- und Ausgabeverhalten eines Web-Services, deren Unterprozesse und über diverse Vorbedingungen und Effekte.

Die Klasse **PROCESS** ist der Hauptbestandteil der Prozessontologie. Ein Prozess kann beliebig viele Eingabeparameter erwarten und somit auch eine beliebig große Anzahl von Ausgaben produzieren. Ebenfalls können beliebig viele Vorbedingungen die zur Ausführung des Prozesses benötigt werden definiert werden. Und schlussendlich kann ein Prozess beliebig viele Effekte besitzen die auf der Ausführung des Web-Service basieren (z.B.: die Belastung des Kontos nach einem Buchkauf).

Das folgende Beispiel [2] zeigt die Definition eines Prozesses einen zugehörigen Parameters und dessen Input Klasse:

```
<daml:Class rdf:ID="Process">
  <rdfs:comment> The most general class of processes </rdfs:comment>
</daml:Class>
```

Parameter:

```
<rdfs:Property rdf:ID="parameter">
  <rdfs:domain rdf:resource="#Process"/>
  <rdfs:range rdf:resource="http://www.daml.org/2001/03/daml+oil#Thing"/>
</rdfs:Property>
```

Input:

```
<daml:Property rdf:ID="input">
  <rdfs:subPropertyOf rdf:resource="#parameter"/>
  <rdfs:range rdf:resource="http://www.daml.org/2001/03/daml+oil#Thing"/>
</daml:Property>
```

DAML-S bietet nun drei verschiedene Arten von Prozessen an:

- Atomare Prozess:
Diese Prozesse können direkt aufgerufen werden, beinhalten keine Unterprozesse und sind in einem Schritt ausführbar.

```
<daml:Class rdf:ID="AtomicProcess">
  <daml:subClassOf rdf:resource="#Process" />
</daml:Class>
```

- Einfache Prozesse
Diese sind nicht mehr direct ausführbar die Ausführung kann jedoch in einem Schritt stattfinden. Durch einfache Prozesse lässt sich eine abstrakte Sicht auf atomare oder zusammengesetzte Prozesse erstellen.

```
<daml:Class rdf:ID="SimpleProcess">
  <daml:subClassOf rdf:resource="#Process" />
</daml:Class>
```

Im folgenden Fall wird ein einfacher Prozess durch einen atomaren realisiert:

```
<rdfs:Property rdf:ID="realizedBy">
  <rdfs:domain rdf:resource="#SimpleProcess"/>
  <rdfs:domain rdf:resource="#AtomicProcess"/>
  <daml:inverseOf rdf:resource="#realizes"/>
</rdfs:Property>
```

Im zweiten Fall wird der einfache Prozess zu einem zusammengesetzten erweitert:

```
<rdfs:Property rdf:ID="expandsTo">
  <rdfs:domain rdf:resource="#SimpleProcess"/>
  <rdfs:domain rdf:resource="#CompositeProcess"/>
  <daml:inverseOf rdf:resource="#collapsesTo"/>
</rdfs:Property>
```

- Zusammengesetzte Prozesse

Diese Prozesse sind schon wie der Name andeutet aus einzelnen, atomaren oder anderen zusammengesetzten Prozessen zusammengesetzt. Die Zusammensetzung bedient sich diverser Kontrollstrukturen (siehe Tabelle 2).

```
<daml:Class rdf:ID="CompositeProcess">
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Process"/>
    <daml:Restriction daml:minCardinality="1">
      <daml:onProperty rdf:resource="#components"/>
    </daml:Restriction>
  </daml:intersectionOf>
</daml:Class>
```

Zusammengesetzte Prozesse besitzen eine composedOf Eigenschaft mit dieser lässt sich die Kontrollstruktur des zusammengesetzten Prozesses erkennen. Die composedOf Eigenschaft lässt sich von der Klasse ControlConstruct ableiten

Die Klasse ControlConstruct gibt im Folgenden jeder Kontrollstruktur eine Eigenschaft components, In dieser wird die Reihenfolge und die bedingte Ausführung der Unterprozesse festgelegt.

DAML-S bietet (siehe oben) verschiedene Detailstufen in denen Prozesse abgebildet werden können. Soll nun ein zusammengesetzter Prozess² als einfacher Prozess³ dargestellt werden so wird die Beziehung zwischen diesen zwei Prozessen mit Hilfe der Eigenschaften expand und collapse dargestellt.

Table 8. Kontrollstrukturen zur Beschreibung von zusammengesetzten Prozessen [1]

Konstruktor	Bedeutung
Sequence	Führe eine Liste von Prozessen nacheinander aus
Concurrent	Führe eine Menge von Prozessen gleichzeitig aus.
Split	Rufe Elemente einer Menge von Prozessen auf
Split+Join	Rufe Elemente einer Menge von Prozessen auf und synchronisiere sie
Unordered	Führe eine Menge in beliebiger Reihenfolge aus
Choice	Führe einen beliebigen Prozess aus einer Menge von Prozesse aus
If-Then-Else	Wenn die Bedingung erfüllt ist, führe den Prozess im Then-Teil aus, ansonsten den im Else-Teil
Repeat-Until	Führe eine Menge von Prozessen so lange aus bis die Bedingung erfüllt ist
Repeat-While	Führe eine Menge von Prozessen aus, solange die Bedingung erfüllt ist

Prozesskontrollontologie:

In der Prozesskontrollontologie wird der Prozess als Zustand dargestellt. Dies unterstützt die Ausführungsüberwachung. Um die Ausführung eines Prozesses zu überwachen muss dem Agenten ein Modell zur Verfügung stehen das Prozesse adäquat interpretiert. Zu diesem Zwecke sollte die Prozesskontrollontologie 3 Eigenschaften besitzen:

1. Sie sollte Richtlinien für die verschiedenen Input-Eigenschaften (Inputs, Vorbedingungen) zu den entsprechenden Output-Eigenschaften zur Verfügung stellen
2. Sie sollte ein Modell der Abhängigkeiten zur Verfügung stellen, die durch Konstruieren wie sequence, split, split+join, etc. beschrieben werden
3. Sie sollte den Durchführungszustand der atomaren und zusammengesetzten Prozesse anzeigen die benötigt werden um die Durchführungsüberwachung zu tun. Dieses erlaubt einem Agenten, den Status von Durchführungen, einschließlich der erfolgreichen, verlassenen und unterbrochenen Prozesse zu verfolgen, und auf jedes adäquat zu reagieren.

² auch Glass-Box genannt

³ auch Black-Box genannt

In der aktuellen Version von DAML-S ist noch keine Definition einer Prozesskontrollontologie enthalten, da diese sich noch in der Entwicklung befindet.

3.2.3 SERVICEGROUNDING

Das SERVICEGROUNDING bietet nun sozusagen die Schnittstelle zwischen der abstrakten Sichtweise des Web-Service, die durch das SERVICEPROFIL und das SERVICEMODEL gegeben wird, und der konkreten Realisierung (Fig. 5). Um das zu erreichen bedient sich DAML-S schon vorhandener „Sprachen“ wie WSDL (Web Service Description Language) um die konkrete Realisierung eines Web-Service zu erreichen.

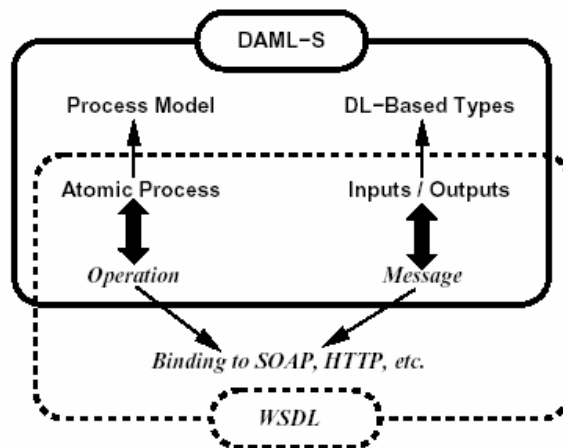


Fig. 5. Servicegrounding mit WSDL [1]

Wir wollen jedoch hier nicht weiter auf diesen Punkt eingehen da dies den Rahmen dieser Seminararbeit sprengen würde. Für genauere Informationen zu dieser Thematik empfehlen wir [2] Punkt 5 bzw. 6.

4. Anwendung im B2C-Bereich

Die Anwendung von DAML-S soll mit Hilfe eines einfachen Beispiels [3] erklärt werden. Das Beispiel gibt einen Einblick in die Beschreibungssprache DAML-S anhand eines Buchhandelsservice, welcher von der frei erfundenen Firma Congo GmbH [3] angeboten wird. Congo stellt eine Reihe von Programmen über das World-Wide-Web dem Benutzer des Service zu Verfügung. Diese Programme sind LocateBook, PutInCart, SignIn, CreateAcct, CreateProfile, LoadProfile, SpecifyDeliveryDetails und FinalizeBuy.

Entscheidend beim Beschreiben eines Web-Services ist, eindeutig zwischen der Beschreibung der physischen Programme, welche über das Internet erreicht werden können, und der Beschreibung, die verwendet wird, um die Services dieser Programme zu beschreiben, zu unterscheiden. Die Beschreibung des Web-Services wird in mehreren Schritten durchgeführt.

4.1 Prozesse & Programme eines Booksellers

Der erste Schritt beim Markup eines Web-Services ist, die einzelnen Programme, die der Service umfasst, zu beschreiben. Das Prozessmodell stellt für diesen Zweck eine Beschreibung der Eigenschaften von Programmen, die über das Web erreicht werden können, zu Verfügung.

Das Prozessmodell interpretiert jedes Programm entweder als atomaren Prozess oder als zusammengesetzten Prozess. Ein Programm, welches nicht mehr zerlegt werden kann, wird als atomarer Prozess modelliert. Ein atomarer Prozess ist dadurch gekennzeichnet, dass er durch einen einzigen Aufruf, der eine Antwort zurückgibt, ausgeführt werden kann. Ein Beispiel für einen atomaren Prozess ist das Programm LocateBook, welches als Input den Namen eines Buches erhält und daraufhin eine Beschreibung des Buches und den Preis zurückliefert, falls das Buch überhaupt im Congo Katalog existiert. Die einfachste Möglichkeit, um LocateBook als atomaren Prozess zu kennzeichnen ist, die Konstruktion subClassOf wie folgt zu verwenden.

```

<daml : Class rdf : ID="LocateBook">
  <rdfs : subclassOf rdf : resource="&process; #AtomicProcess"/>
</daml : Class>

```

Mit jedem Prozess ist Reihe von Eigenschaften verbunden. Wenn nun ein Programm benutzt wird, werden mit dem Prozess bestimmte Parameter assoziiert. Zwei Arten von Parametern sind die, zu DAML-S gehörenden Eigenschaften *input* und *output*, welche in der Prozess Ontologie definiert sind. Ein Beispiel für den Input bei LocateBook wäre der Name des Buches. Dies kann beschrieben werden, indem man die Konstruktion *subPropertyOf* verwendet.

```

<rdf : Property rdf : ID="bookName">
  <rdfs : subPropertyOf rdf : resource="&process; #input" />
  <rdfs : domain rdf : resource="#LocateBook" />
  <rdfs : range rdf : resource="&xsd; #string" />
</ rdf : Property>

```

Inputs können entweder verbindlich oder optional sein. Im Gegensatz dazu sind Outputs immer konditional. Wenn man zum Beispiel im Congo Katalog nach einem Buch sucht, ist der Output entweder eine Beschreibung des Buches, falls Congo das Buch führt, oder aber falls dies nicht der Fall ist, eine „Sorry, Buch konnte nicht gefunden werden“ Nachricht. Solche Outputs werden als abhängige Outputs bezeichnet. Um einen konditionellen Output zu beschreiben, existiert die Klasse *ConditionalOutput*, welche sowohl einen Zustand, als auch den Output, der aus diesem Zustand folgt, beschreibt.

Die Klasse *ConditionalOutput*, welche eine Unterklasse von *Thing* ist, hat zwei Eigenschaften, nämlich den Zustand *coCondition*, und den Output *coOutput*.

Ein Beispiel für einen abhängigen Output ist *bookDescription*, welcher nur dann zum Tragen kommt, wenn das Buch im Katalog aufscheint. Falls dies nicht der Fall ist, dann ist der Output eine Nachricht, die diesen Umstand dem Benutzer mitteilt.

Auch LocateBook kann beschrieben werden, indem die Prozessontologie durch die Verwendung von Konstruktionen, wie *subclassOf* und *subPropertyOf*, spezialisiert wird. Ein konditionaler Output für LocateBook ist in Fig. 6 zu sehen.

Die Bestimmung von Inputs und Outputs ermöglicht es, dass Programme und Services, die mittels DAML-S beschreiben wurden, automatisch von zum Beispiel Web-Agenten aufgerufen werden können. Damit die Services auch die automatische Komposition und Interoperation unterstützen, müssen zusätzlich noch die Nebeneffekte des Programms, beschrieben werden, falls welche existieren. Hierfür hat man festgelegt, dass Services die Eigenschaften *preconditions* und *conditional effect* haben. *Preconditions* und *conditional effects* werden analog zu *inputs* und *konditionalen outputs* beschrieben.

Preconditions spezifizieren Dinge, die einen Wahrheitswert liefern müssen, damit ein Web-Agent das Service ausführen kann.

```

<rdf : Property rdf : ID="precondition">
<rdfs : domain rdf : resource="#Process" />
<rdfs : range rdf : resource=http://www.daml.org/2001/03/daml+oil#Thing/>
< /rdf : Property>

```

Effekte, wie *outputs*, sind konditional. Konditionale Effekte charakterisieren die physikalischen Nebeneffekte, welche durch die Ausführung eines Web-Servive entstehen. Man kann sagen, dass alle Services physikalische Nebeneffekte haben, außer Services, die nur Information liefern.

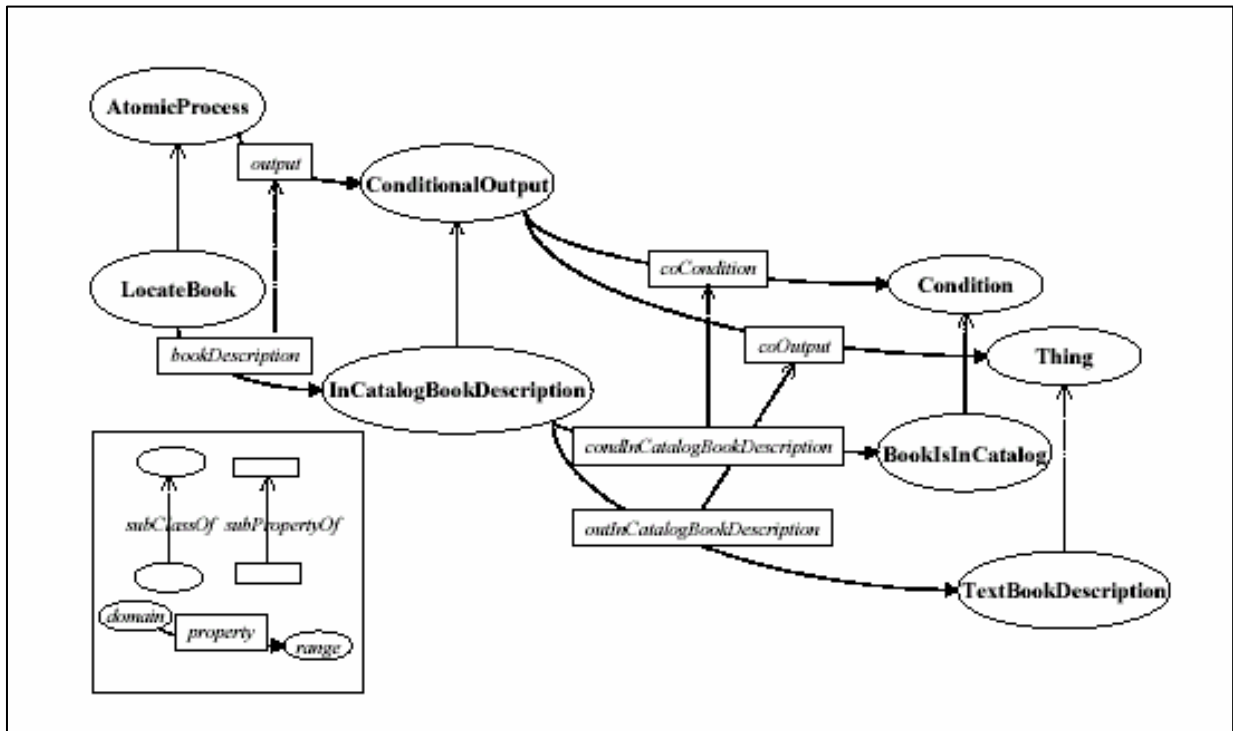


Fig.6 Ein konditionaler Output für LocateBook in DAML-S [2]

4.2 Prozesse zusammenfassen & vereinfachen

Nachdem nun alle atomaren Prozesse und deren inputs, outputs, preconditions und effects beschrieben worden sind, kann man nun Kompositionen dieser Prozesse, die verschiedene Services bereitstellen, beschreiben.

Im Gegensatz zu atomaren Prozessen sind zusammengesetzte Prozesse Kompositionen von anderen zusammengesetzten und atomaren Prozessen. Die einzelnen Prozesse werden mit Hilfe von typischen Kontrollkonstruktionen, wie zum Beispiel Sequenzen, if-then-else-Anweisungen oder while-Schleifen zusammengefügt, damit die Reihenfolge, in welcher die Prozesse ausgeführt werden, festgelegt werden kann. In DAML-S werden zusammengesetzte Prozesse nach dem top-down Prinzip gebildet. Jeder CompositeProcess ist composedOf einem ControlConstruct, welches eine Sequenz, eine if-then-else-Anweisung oder auch eine while-Schleife sein kann.

Bei unserem Buchhandel Congo GmbH, wird der zusammengesetzte Prozess CongoBuy in zwei Hauptschritten zusammengefügt. Zuerst muss das Buch gesucht werden, dann wird es gekauft. Wir nehmen an, dass locate book ein atomarer Prozess ist, während der Buchkauf eine Sequenz von Unterprozessen (atomar und zusammengesetzt) ist. Diese Unterprozesse sind die Spezifikationen der Zahlungsmethode, der Lieferung und der Abschluss des Kaufprozesses.

4.3 Werbung für das Service

DAML-S wurde entwickelt, um das automatische Auffinden von Web-Services zu ermöglichen, indem es DAML+OIL Beschreibungen der Eigenschaften und Fähigkeiten des Services zur Verfügung stellt. Diese Beschreibung kann dazu verwendet werden, um sich in Service-Verzeichnisse einzutragen oder um bessere Anhaltspunkte für Suchmaschinen bereitzustellen.

Die Beschreibung zum Auffinden von Web-Services ist die einfachste Form der Beschreibung, die ein Service-Provider anbietet, da sie nicht von einem Prozessmodell abhängig ist.

Ein Service Profil ist eine Instanz des Klassenprofils, welches in der Profil Ontologie definiert ist. Zu Beginn stellt das Service Profil Informationen über das Service selbst und den Anbieter zur Verfügung.

isPresentedBy

```
<service : isPresentedBy>
  <service : Service df : resource="&congo;#Congo_BookBuying_Service" />
</service : isPresentedBy>
```

serviceName

```
<profile : serviceName>Congo_BookBuying_Agent</profile : serviceName>
```

textDescription

```
<profile : textDescription>
  Diese Service bietet die Möglichkeit um Bücher zu Suche und diese zu
  kaufen.
</profile : textDescription>
```

geographicRadius

```
<profile : geographicRadius rdf : resource="&country;#Austria"/>
```

qualityRating

```
<profile : qualityRating rdf: resource="&concepts ;#qualityRatingGood" />
```

Im letzten Teil des Profils werden einzelne Schlüsselattribute, wie zum Beispiel Input, Output, Preconditions und Nebeneffekte des Services beschrieben.

5 Zukunftsaussichten von DAML-S

Im Wesentlichen hängt die Zukunft von DAML-S wie die aller anderen Markup Languages von der Entwicklung des Semantic Webs ab und somit stark von der Einbindung semantischer Technologien und Ideen in gängigen html Editoren oder anderen Programmen die zur Gestaltung des WWW dienen.

Was sich in ferner Zukunft auf dem Gebiet der Web-Services abspielen könnte möchten wir abschließend im folgenden Beispiel darstellen:

Angenommen man möchte sich ein Auto kaufen. Man hat natürlich gewisse Präferenzen die von Farbe, Sitzbezügen, Klimaanlage bis zu Leistung, Verbrauch und Größe reichen.

Wie beginnt man nun seine Suche nach dem Automobil seiner Wahl?

Heutzutage wird man sämtlich in Frage kommenden Autohändler absuchen und die Angebote miteinander vergleichen, wenn man nicht gerade ein „Markenfetischist“ ist und nur eine Automarke fährt. Hier werden sich erneut Fragen auftun. Gibt es einen günstigen Leasingvertrag? bzw. Soll es doch ein Gebrauchtwagen werden? All diese Fragen müssen beantwortet werden und bis all die lästige Kleinarbeit erledigt ist und das Traumauto gefunden, kann viel Zeit und Nerven verloren werden.

Genau hier kommen die Web-Agenten ins Spiel. Sie sollen all diese lästigen Kleinigkeiten für einen erledigen. Man teilt also nun seinem Web Agenten seine Präferenzen mit, Lederbezüge, dunkle Farbe, mindestens 100 PS, Leasingvariante ab einem gewissen Zinssatz und wenn möglich eine Stereoanlage mit integriertem mp3 Player. Nun begibt sich der Web-Agent auf die Suche im WWW. Hier an der Schnittstelle zwischen Anbietern von Web-Services, wie z.B.: Autohändler, und dem Web-Agenten könnte DAML-S ins Spiel kommen. Denn DAML-S ermöglicht es eben dem Web-Agenten genaue, spezifische und adäquate Information über das Web-Service und somit deren Produkte zu erhalten.

Somit könnte der Web-Agent ganz allein nach der Auffindung des passenden Produktes, in unserem Fall: des Autos, dieses sofort bestellen bzw. bei entsprechender Autorisierung auch sofort per Kreditkarte bezahlen.

Natürlich klingt all dies sehr futuristisch und mehr nach Träumerei als nach baldiger Realität, jedoch wollen wir uns hier eher an John Galsworthy(1867 – 1933), einem englischen Schriftsteller halten der über die Zukunft folgendes meinte „*Wenn Sie nicht über die Zukunft nachdenken, können Sie keine haben.*“.

Literaturliste

1. Sebastian Lohmann.: Web-Agenten: Ideen und Szenarien. Lehrstuhl 15, RWTH Aachen (2002)
2. DAML-S Coalition: DAML-S: Web Service Description for the Semantic Web. BBN Technologies, Carnegie Mellon University, Nokia Research Center SRI International Stanford Universtiy, Yale University (2002)
3. DAML-S Version 0.6.: Walk-Through (2001)
4. Giulio Alighieri.: Global E-Government Web-Services. Institut für Informatik der Universität Zürich (2002)
5. Simon Breuß, Mark Hall, Alexander Ruß.: Semantic Web. Institut für Wirtschaftsinformatik und Anwendersysteme Klagenfurt (2001)
7. Sheila McIlraith, Tran Cao Son, Honglei Zeng: Semantic Web Services (2001). <http://www.daml.org/services/>
8. Anupriya Ankolekar, Frank Huch, Kathie Sycra: Concurrent Execution Semantics for DAML-S with subtypes (2002) <http://www.daml.org/services/>
9. DAML-S Coalition: DAML-S and Related Technologies (2002). <http://www.daml.org/services/>
10. Massimo Paolucci, Takahiro Kawamura, Terry Payne, Kathie Sycra: Semantic Matching of Web Services Capabilities (2002). <http://www.daml.org/services/>
11. Source Codes zum Congo Bsp. : <http://www.daml.org/services/daml-s/0.7/>
<http://www.daml.org/services/daml-s/2001/10/>
<http://www.daml.org/services/daml-s/2001/05/>