Semantische Web Services

Een suggestie voor de verbetering van web service compositie

Bachelorscriptie Informatiekunde
Door Pepijn Arts
0412570

Begeleiding door Patrick van Bommel

Inhoudsopgave

Voorwoord

Nijmegen, Juni 2009

Geachte lezer,

Voor u ligt het document wat dient tot afsluiting van mijn bacheloropleiding Informatiekunde aan de Radboud Universiteit Nijmegen. Deze scriptie is het resultaat van de kennis en vaardigheden die ik heb opgedaan in de tijd dat ik deze opleiding volg. Vanwege het feit dat het schrijven van een scriptie niet enkel een kwestie van tekst uitschrijven is, kent de aanloop naar en het uitwerken van de scriptie vele up's en down's. Ervan uitgaande bij het juiste station geëindigd te zijn toon ik graag mijn dank aan Patrick van Bommel voor de begeleiding en het sturen gedurende de weg naar het eind.

Ik wens u veel leesplezier.

Pepijn Arts

1. Inleiding

Het internet bestaat niet langer uit enkel een verzameling van pagina's. Via het internet wordt er steeds meer functionaliteit aangeboden in de vorm van web services, ofwel diensten via internet. Eén van de krachten van web services is de toegankelijkheid. De platformonafhankelijkheid van web services en de standaardisering van protocollen als SOAP zorgen ervoor dat iedereen in de web service wereld met elkaar kan praten. Door het toenemende gebruik (met name in de E-Commerce) is er behoefte aan een geautomatiseerde vorm van deze internetdiensten. Met name in de *business to business* is dit erg van belang. De business to business wil zeggen dat web services worden gebruikt om bedrijfsapplicaties te ondersteunen of ze te voorzien van de juiste informatie. Dit gaat als volgt in zijn werk.

Met de Web Service Description Language (WSDL) worden web services beschreven en via een register worden de services aangeboden. Een gebruiker van de web service zoekt de service op in een register en een communicatieproces met informatie uitwisseling volgt. De beschrijving van web services vindt echter plaats op syntactisch niveau dat de structuur van een web service beschrijft. Het aanbieden, vinden en gebruiken van web services kan een stuk doelgerichter als men gericht kan zoeken op functionaliteit, semantiek van de web service. Met name in initiatieven om het zoeken naar en het communiceren met web services te automatiseren, is deze semantische beschrijving van groot belang [1]. De recente versies van het Universal Description Discovery Integration (UDDI) register erkennen de behoefte om services in te delen in verschillende registers die met elkaar kunnen interacteren [2]. Belangrijk hierbij is dat de schaalbaarheid van deze registers moet worden gewaarborgd [1].

Er zijn een aantal initiatieven om semantiek toe te kennen aan de WSDL-beschrijving van web services. Deze initiatieven richten zich niet alleen op de (semantische) beschrijving van web services, maar ook op het domein waarin ze zich bevinden. Op het moment dat men in staat is semantiek toe te kennen aan de beschrijving van web services kan men makkelijker zoeken naar functionaliteit en kan deze gerichter aangeboden worden. Essentieel hierbij is dat de beschrijvingen kunnen worden ondergebracht in (domeinspecifieke) categorieën, zodanig dat de explosieve groei van het gebruik van Web Services niet leidt tot een onhanteerbare structuur.

1.1 Probleemstelling

Bij een volledig geautomatiseerde vorm van het gebruik van web services weet de gebruiker van de services bij voorbaat niet welke web services er beschikbaar zijn. De gebruiker doet slechts een verzoek bij het register en die bepaalt met welke web service aan het verzoek kan worden voldaan. Het kan dus voorkomen dat er niet één web service is die aan het verzoek kan voldoen, maar dat meerdere web services hun krachten moeten bundelen. Dit proces heet de compositie van web services. De huidige vorm van compositie, zoals deze later in dit verslag zal worden beschreven, is niet dynamisch en krachtig genoeg om ten tijden van uitvoeren geschikte composities te maken [3]. Het doel van deze bachelorscriptie is daarom om inzicht te krijgen in de semantiek van Web Services, in hoe deze semantiek past in de structuur van registers en op basis van deze inzichten suggesties te doen om semantische compositie van web services naar een hoger niveau te tillen. De onderzoeksvraag luidt daarom als volgt:

"Hoe kunnen web services op semantische basis worden beschreven en gematcht, en hoe kunnen deze inzichten worden gebruikt om de compositie van web services te verbeteren?"

Ter illustratie van de problematiek een kort voorbeeld. Stel we zijn op zoek naar een service die ons kan voorzien van een weerbericht. We zoeken hier natuurlijk geen service die helderziend is maar een die zicht heeft op een aantal informatiebronnen, namelijk de kans op zon, de kans op regen, de kracht en richting van de wind, het waterpeil en natuurlijk de temperatuur. Laten we aannemen dat deze service zich richt op een enkele en vaststaande regio. Deze informatie hoeft wat ons betreft natuurlijk niet in één service te zitten, zolang wij ons gewenste resultaat maar krijgen. Als we een register vragen om een weerbericht zijn we op zoek naar een volgende situatie.

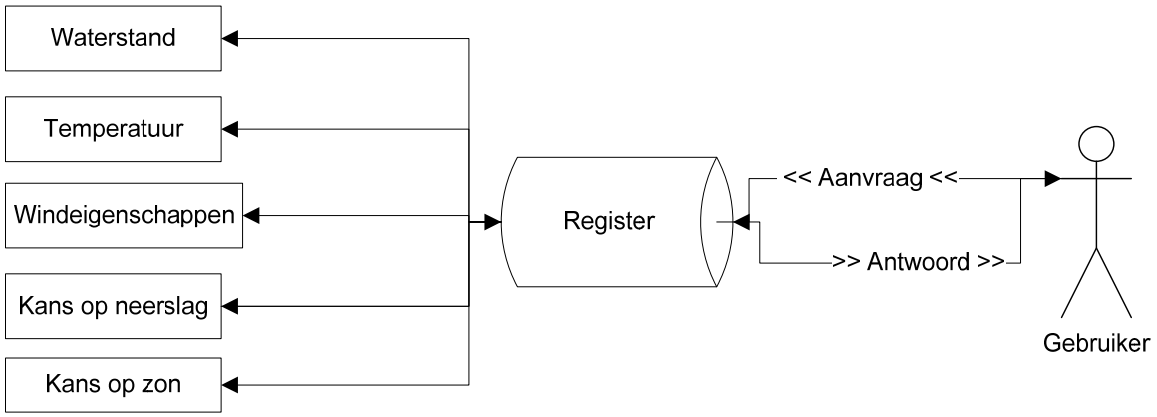

Fig. 1 Weerbericht web service

Een gebruiker doet de aanvraag bij een register dat de benodigde web services bij elkaar raapt. Deze geven elk hun resultaat terug aan het register dat een antwoord kan geven aan de gebruiker. Echter, in een situatie waarin de web services op syntactisch niveau beschreven worden is de functionaliteit van de services niet inzichtelijk. Het register kan daarom niet weten of het de goede services te pakken heeft en of deze set van services een volledig antwoord oplevert. Daarnaast is het een hele kunst om ten tijde van uitvoering de goede web services bij elkaar te vinden.

In het resterende deel van dit verslag zal het voorbeeld nog een aantal keer terug komen om te zien hoe deze problematiek kan worden aangepakt. Om de onderzoeksvraag met tevredenheid te kunnen beantwoorden, zal gebruik worden gemaakt van een aantal deelvragen. Nadat er in het volgende hoofdstuk uitleg wordt gegeven over de gebruikte concepten en begrippen, zullen in de daarop volgende hoofdstukken de deelvragen worden behandeld en beantwoord. Deze deelvragen zijn de volgende.

- Hoe kunnen web services op semantische basis worden beschreven?
- Hoe worden deze semantische web service verwerkt in registers?
- Welke knelpunten kunnen er worden geïdentificeerd aan de hand van een ORM model van web services?
- Hoe kan de compositie van semantische web services worden verbeterd?

Belangrijk hierbij is te kennen te geven dat de eerste twee deelvragen zullen worden beantwoord aan de hand van de concepten die bij web services een rol spelen. Deze concepten worden uitgelegd aan de hand van erkende literatuur over deze onderwerpen. Het antwoord op de derde en met name vierde deelvraag komt voort uit eigen redenatie en afleiding op basis van de literatuur besproken bij beantwoording van de eerste twee deelvragen en de modellen waarin de concepten en hun onderlinge relaties schematisch en formeel worden weergegeven.

2. Uitleg concepten en begrippen

Zoals in de inleiding al is genoemd krijgen Web Services een steeds grotere rol binnen de wereld van het internet. Dit brengt ons tot de vraag: “Wat zijn Web Services?”. Er zijn vele definities gevormd, waaronder één door het UDDI consortium. Zij beschrijven Web Services als *“self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces”* [4]. Deze definitie zal in dit verslag worden gehanteerd vanwege het concrete en volledige karakter. Om duidelijkheid te verschaffen met betrekking tot de termen en concepten die in dit verslag gebruikt worden, volgt nu een korte uitleg.

Binnen de web service wereld is er gewoonlijk sprake van drie partijen, namelijk de vragende partij, de aanbiedende partij en een tussenpartij die de eerste twee met elkaar in contact kan brengen. De aanbiedende partij, vanaf nu de service provider genoemd, kan alleen aanvragen verwerken die voldoen aan een vooraf bepaalde structuur. De vragende partij, vanaf nu de service requester genoemd, moet dus weten hoe hij de service provider dient aan te spreken en hoe hij het antwoord moet interpreteren. Soms is dit bekend maar veelal voorziet de derde partij, een register, de service requester van deze informatie. Het meest bekende register is het Universal Description Discovery Integration (UDDI) register [5].
In een UDDI register kunnen service providers hun service publiceren door de service met de Web Service Description Language (WSDL) te beschrijven. Hierin staat beschreven hoe de service aangesproken kan worden, via welke URL en wat de service aan input verwacht en hoe de output eruit ziet, zodanig dat de service requester deze juist kan gebruiken. De totale beschrijving van de web service noemt men het profiel. De service requester kan op basis van dit profiel een web service opzoeken en vervolgens met de provider van de service interacteren.

De volgende afbeelding geeft het bovenstaande schematisch weer.

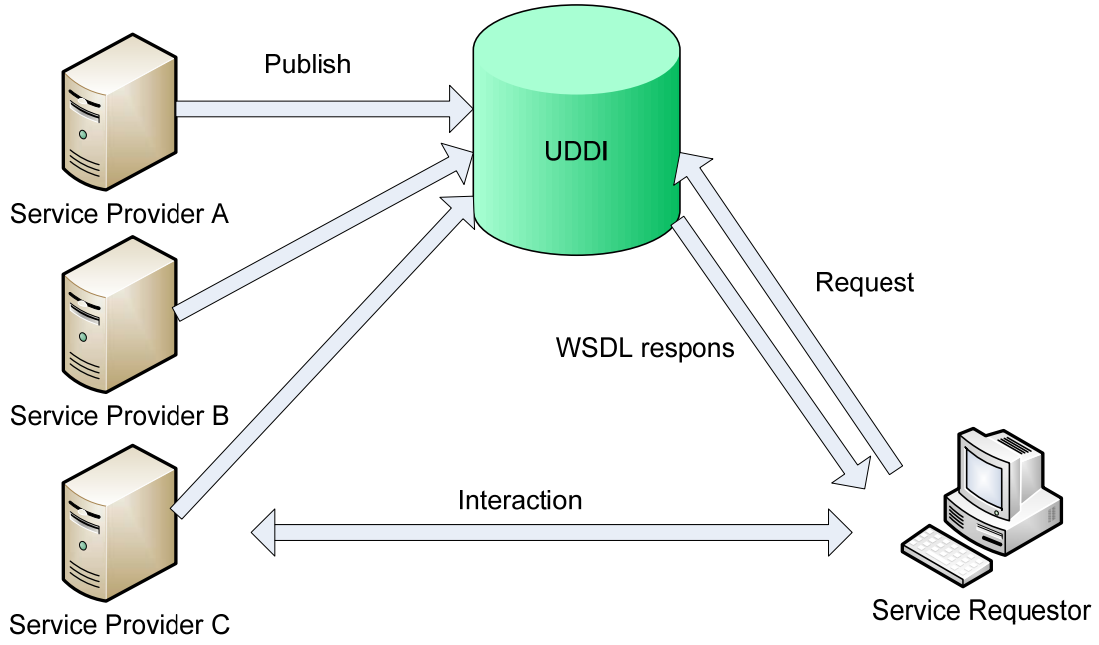

Fig. 2 Web Service Architectuur

Het vinden van een bepaalde service wordt de discovery genoemd. Er heeft veel onderzoek plaatsgevonden naar de automatisering van de discovery van web services. De grote vraag op dit punt van schrijven is hoe de juiste vraag bij het juiste antwoord gekoppeld kan worden. Het antwoord: agents. Agents zijn software componenten die de juiste service aan een bepaalde request koppelen. Op het moment dat agents een service profiel koppelen aan een service request spreken we van matching.

Zoals in de inleiding al kort is aangehaald wordt de matching van services gebaseerd op beschrijvingen van syntactisch niveau. Echter, als een service requester een request doet die niet met één web service is op te lossen, zeker wanneer de discovery van services zoveel mogelijk wordt geautomatiseerd, ontstaan er problemen. Het verwerken van semantiek in dit proces kan uitkomst bieden. Alvorens deze oplossingen worden aangedragen, is het verstandig eerst helder te krijgen wat wordt bedoeld met syntax en wat met semantiek.

Syntax

Syntax, afgeleid van het taalkundige Syntaxis, is een taalkundig concept dat de opbouw en grammatica van een zin omvat. De syntax van een taal, zowel een natuurlijke taal als een machine- of programmeertaal, is gekoppeld aan regels. De zin *"Ik ga naar huis"* noemen we syntactisch correct terwijl we de zin *"Ga nar hus ik"* syntactisch incorrect noemen. Niet alleen de zinsbouw, maar ook de spelling van de individuele woorden vallen onder het begrip syntax. Hoewel de syntax van een taal de grammaticale correctheid waarborgt, heeft het geen invloed op de betekenis van een zin. De Nederlandse zin *"Ik heb geen appel, maar wel een appel"* is syntactisch correct terwijl deze inconsistent is. De zin heeft geen betekenis. Syntax heeft dus enkel betrekking op de grammatica maar laat de betekenis en interpretatie buiten beschouwing.

Semantiek

Semantiek houdt zich bezig met de betekenis van taalkundige constructies zoals zinsbouw en de betekenis van individuele woorden. Zoals we gezien hebben kunnen zinnen syntactisch correct zijn, maar semantisch incorrect en kan eenzelfde syntactische zin meerdere betekenissen hebben. Semantiek is daarom gevoelig voor interpretatie. Waar grammaticaregels vast liggen, zijn interpretaties subjectief en contextafhankelijk. Het klassieke voorbeeld is de Engelse zin *"Time flies like an arrow"*. Deze zin is op zeer veel manieren te interpreteren. Met deze zin wordt meestal bedoel dat tijd snel passeert, zoals een pijl dat doet. Echter, het kan zomaar zijn dat iemand bedoelt dat tijdvliegjes van een pijl houden. Een tweede voorbeeld is de Engelse zin *"I never said she stole my money"*. Afhankelijk van op welk woord de klemtoon ligt krijgt de zin haar betekenis. Ligt de klemtoon op "*I*", dan betekent de zin dat iemand de uitspraak heeft gedaan, maar niet de ik-persoon. Ligt de klemtoon op "*my*" dan betekent de zin dat *"she"* het geld van iemand anders dan de ik-persoon heeft gestolen. Ambiguïteit heeft daarom grote impact bij semantische beschrijvingen. In dit verslag kijken we naar de semantiek van web services waarmee we de betekenis en dus de functie van de web service bedoelen.

Overige begrippen en concepten

Naast de twee leidende begrippen syntax en semantiek worden er in het resterende deel van dit hoofdstuk nog een aantal begrippen uitgelegd om onduidelijkheid zoveel mogelijk uit te sluiten.

Ontologie

An ontology is a formal explicit specification of a shared conceptualization. Deze definitie van Gruber [6] slaat de spijker op zijn kop. Een ontologie is een formele expliciete specificatie van een gemeenschappelijke conceptualisatie. Met andere woorden, een ontologie is een concrete specificatie van een abstracte perceptie die mensen van een concept hebben. Binnen de web service wereld is een sterke ontologie onmisbaar. Eerder is gesproken over het koppelen van de juiste service aan de juiste vraag. Dit is onmogelijk op het moment dat er verschillen in interpretatie aanwezig zijn. Een ontologie biedt eenduidigheid en laat geen of weinig ruimte over voor ambiguïteit. Er zijn verschillende ontologieën ontworpen in de web service wereld. De meest bekende voorbeelden hiervan zijn RDF, RDF-S, DAML-S en OWL. Hoewel er vele variaties en uitbreidingen zijn voorgesteld op deze ontologieën, zijn deze in basis redelijk als standaard geaccepteerd. Deze ontologieën zullen uitgebreider worden besproken in het volgende hoofdstuk.

XML & SOAP

In deze sectie worden de begrippen XML en SOAP kort aangehaald. XML staat voor Extensible Markup Language [7] en is een van de meest gebruikte standaarden op het internet. Evenals het bekende Hypertext Markup Language (HTML) is XML een markup language. Daar waar HTML gericht is op de markup van visuele representatie (bijvoorbeeld van webpagina's), is XML gericht op de structurering van data. De syntax en structuur van XML is zo flexibel opgesteld dat het ontwerpers in staat stelt een eigen specificatietaal te bouwen middels XML. XML kent namelijk net als HTML *tags* om informatie te voorzien van een label. Hoewel HTML een vaste set van tags kent, is XML zo gericht dat ontwerpers hun eigen tags kunnen definiëren, wat zorgt voor de genoemde flexibiliteit. Mede omdat XML breed geaccepteerd is als standaard voor de representatie van datastructuren zijn de bovengenoemde ontologieën syntactisch gebaseerd op XML. Dit verslag zal niet verder ingaan op XML, voor meer informatie over XML zie de w3c documentatie [7].

Het Simple Object Access Protocol (SOAP) is een XML-gebaseerd communicatieprotocol. Het protocol schrijft voor hoe actoren op het web met elkaar kunnen communiceren. SOAP is in de web service wereld als standaard geaccepteerd voor het uitwisselen van berichten. Eerder in dit hoofdstuk is bijvoorbeeld gesproken over het aanspreken van een service provider door de service requester en het ontvangen van het antwoord als reactie op de aanvraag. Deze uitwisseling van informatie gebeurt typisch via het SOAP protocol.

3. Hoe wordt semantiek toegekend aan web services?

Verscheidene initiatieven zijn erop gericht de web services wereld te voorzien van een semantische uitdrukkingskracht. In dit hoofdstuk wordt op drie niveaus bekeken hoe dit verwezenlijkt kan worden. Het eerste niveau is het beschrijvingsniveau van een web service. Deze beschrijving geschiedt middels de eerder besproken ontologieën. Deze ontologieën hebben elk een aantal concepten die de semantische uitdrukkingskracht tot stand brengen en zijn onmisbaar bij het semantisch beschrijven van web services [8]. Het tweede niveau is het uitsplitsen van de functionaliteit en de intentie van een web service. Hierdoor is het mogelijk om domein specifieke zaken van domein neutrale te onderscheiden en is het mogelijk om de capaciteiten van een web service uit te drukken. Met name dit laatste is erg belangrijk voor een semantische match. Het derde niveau is het niveau van compositie. Het combineren van functionaliteit biedt mogelijkheden voor het automatiseren van de discovery van web services op basis van semantische informatie.

3.1 Web Service ontologieën

Er zijn verschillende methoden waarmee geprobeerd wordt om semantiek aan web services toe te kennen. Het slagen van deze methoden hangt af van de mate waarin ze als standaard worden geaccepteerd. Op het moment dat dit niet het geval is ontstaat er geen draagvlak voor een methode en zal deze niet ver komen. In dit eerste stuk van dit hoofdstuk wordt er gekeken naar een viertal ontologieën die het meeste potentie hebben om als standaard geaccepteerd te worden of al acceptatie kennen. Per ontologie wordt eerst gekeken naar het conceptueel model dat achter de ontologie schuilt, vervolgens wordt er gekeken naar de (semantische) uitdrukkingskracht.

3.1.2 Resource Description Framework (RDF)

Het Resource Description Framework (RDF) [9] is een techniek om metadata van resources te beschrijven. Het wordt gebruikt om metadata zodanig te structureren dat de informatie interpreteerbaar wordt voor machines. De volgende informatie is gebaseerd op de w3c documentatie betreffende RDF en wordt gebruikt om RDF kort toe te lichten als basis voor RDF Schema. Voor een uitgebreide specificatie van RDF zie de w3c documentatie [9].

Conceptueel Model

Het conceptueel model van RDF bestaat uit drie componenten: *Resources*, *Properties* en *Statements*. Resources zijn de objecten die beschreven worden door RDF. Een voorbeeld hiervan is een webpagina of web service. De properties vormen de eigenschappen van de resources. Een statement brengt een resource met een property tot uitdrukking door deze te combineren en een waarde toe te kennen aan de eigenschap. Een statement bestaat op uit drie componenten te noemen *Subject*, *Predicate* en *Object*.

Ter toelichting van het bovenstaande een voorbeeld.
Gegeven is het volgende statement: "*Ora Lassila is the creator of the resource* *http://www.w3.org/Home/Lassila*". In dit statement is te zien dat de http://www.w3c.org/Home/Lassila het subject, ofwel vrij vertaald het onderwerp, van het statement is. Merk op dat het onderwerp op zich een resource is namelijk een webpagina. Dit onderwerp kent een property, vrij vertaal een eigenschap, namelijk dat het een schepper heeft.

Deze eigenschap vormt het predicaat (predicate) van het statement. Wederom is de connectie te zien tussen een concept op het niveau van properties en op het niveau van predicates. Het predicaat koppelt een object aan het onderwerp waarmee de eigenschap van het statement een waarde krijgt toegekend. In dit voorbeeld is “Ora Lassila” de waardetoekenning aan de eigenschap “creator”. In dit geval is het object een persoon, al is het ook mogelijk dat deze rol wordt vervuld door een resource op het web, bijvoorbeeld een webpagina of een web service. Op deze manier kan er niet alleen een beschrijving worden gegeven van entiteiten en hun relaties, maar ook van de waarde van de relatie. Naast het feit dat statements iets kunnen zeggen over resources kunnen ze ook iets zeggen over andere statements. De methode om dit in RDF te doen is om een statement te benoemen tot resource die vervolgens weer als subject in een statement voor kan komen.

Vaak is het nodig om resources te groeperen. RDF biedt hier een aantal concepten voor, bijvoorbeeld de *container*. Een container kan gezien worden als een verzameling resources die afhankelijk van een type container een rangorde kennen of niet. Dit geeft RDF een krachtige methode om de metadata van resources te groeperen en een hiërarchische structuur te creëren. Deze structuur kan tot uitdrukking worden gebracht door de syntax van RDF. Om hier inzicht in te verschaffen wordt de syntax van RDF kort besproken.

RDF Syntax

Zoals eerder aangegeven maakt de RDF specificatie gebruik van een XML-gebaseerde syntax. RDF stelt geen eisen aan het grammaticaal gebruik - anders dan die door XML worden opgelegd - binnen een specificatie waardoor er een sterke vrijheid en flexibiliteit ontstaat voor degene die een RDF specificatie ontwerpt. Om te zorgen voor een gemeenschappelijke opvatting over de betekenis van dergelijke structuren maakt RDF gebruik van een namespace waarin beschreven staat op welke manier XML gebruikt wordt en welke tags van toepassing zijn. Op deze manier ontstaat er een eenduidige manier om een RDF-specificatie te ontwerpen en kan iedereen de specificatie correct interpreteren.

Om meer inzicht te verschaffen in de RDF specificatie en het gebruik van XML en namespace volgt hier nogmaals een kort voorbeeld uit de w3c documentatie.

Gegeven het statement…
“The individual referred to by employee id 85740 is named Ora Lassila and has the email address lassila@w3.org. The resource http://www.w3.org/Home/Lassila was created by this individual.”

… ziet de RDF specificatie er als volgt uit:

```
<?xml version="1.0"?>
<RDF
xmlns="http://www.w3.org/TR/WD-rdf-syntax#"
xmlns:s="http://description.org/schema/">
        <Description about="http://www.w3.org/Home/Lassila">
        <s:Creator rdf:resource=http://www.w3.org/staffId/85740/>
        <v:Name>Ora Lassila</v:Name>
        <v:Email>lassila@w3.org</v:Email>
</Description>
</RDF>
```

In dit voorbeeld is te zien hoe RDF de XML structuur overneemt en eigen tags gebruikt zoals <v:Name> en <v:Email> De eerste tag geeft aan van welke XML versie gebruik wordt gemaakt en in de tweede tag is te herleiden welke namespace wordt gebruikt.

Nu we zicht hebben hoe RDF ongeveer te werk gaat kunnen we kijken naar de semantische uitdrukkingskracht van deze ontologie.

Semantische uitdrukkingskracht

Het moge duidelijk zijn dat RDF echt bedoeld is voor het beschrijven van metadata en niet specifiek voor het beschrijven van semantische data. De beschrijvingen zijn namelijk erg gericht op wat de resource is, maar niet wat de resource doet. Functionaliteit wordt buiten beschouwing gelaten. Wat wel een interessant punt is, is dat men door middel van RDF domein specifieke informatie kan beschrijven. Men kan bijvoorbeeld aangegeven in welk domein de resource zich bevindt en kan een label toekennen aan een resource.

Persoonlijk vind ik het grootste gebrek van het gebruik van RDF voor het beschrijven van semantische data het gemis van relatie aanduiding. Hoewel RDF in staat is hiërarchische structuren te definiëren is de aard van de relaties tussen resources niet duidelijk. Zelf ben ik van mening dat dit erg belangrijk is voor de uitdrukking van de betekenis, omdat de aard van de relatie invulling geeft aan een stukje context. De context bepaald namelijk in meer of mindere maten de interpretatie van een web service. Desalniettemin is biedt RDF een solide en eenduidige basis en is het niet verwonderlijk dat RDF als fundament dient van veel andere ontologieën.

3.1.2 RDF Schema

RDF Schema (RDF-S) vormt een laag boven op RDF om te zorgen voor meer kracht en uitdrukkingsvermogen. Dit houdt in dat RDF-S gebruik maakt van het RDF framework maar hier concepten aan toevoegt. Dit is overigens vaak het geval bij ontologieën of variaties van c.q. uitbreidingen op ontologieën. RDF-S vormt een basis voor nog geavanceerdere specificatietalen als OWL die semantiek daadwerkelijk tot uitdrukking brengen. Hierdoor is het interessant RDF Schema kort te bespreken.

RDF-S maakt gebruik van een tweetal concepten om de uitdrukkingskracht te vergroten. Deze zijn *Classes* en *Properties*.

Classes (klassen) kunnen gezien worden als groepen resources met gelijke eigenschappen. De individuele resources worden *instances* (instanties) van de betreffende klasse genoemd. Classes zijn niet te verwarren met de eerder genoemde Containers. Het verschil is dat het bij een Container puur en alleen om de groepering en ordening als verzameling gaat, terwijl het bij klassen met name gaat om de gedeelde eigenschappen.

RDF-S definieert een basis structuur van klassen waarin alle overige klassen kunnen worden ondergebracht. Deze structuur ziet er als volgt uit:

De groep van RDF Schema klassen vormen samen de klasse *rdfs:Class*. Deze groep bestaat uit een zestal instanties te noemen *rdfs:Resource, rdfs:Class, rdfs:Literal, rdfs:Datatype, rdf:XMLLiteral* en *rdf:property*. Elke instantie is zelf ook een klasse wat rdfs:Class de superklasse maakt van alle klassen binnen RDF-S. Deze klassen hebben naast generieke eigenschappen ook eigenschappen die enkel voor die klasse gelden. Alle entiteiten binnen RDF-S moet ondergebracht kunnen worden in één van deze klassen. Zoals we bij de sectie over RDF gezien hebben vormen properties in RDF de relatie tussen subject resources en object resources. De klasse properties in RDF Schema bevatten alle properties die de basis vormen voor RDF Schema.

Het tweede belangrijke concept is de proprty. De scherpe lezer dat er een klasse rdf:property is waarin alle de properties worden ondergebracht. Deze properties kunnen worden gebruikt om eigenschappen aan een entiteit mee te geven. Zo is het mogelijk om een descriptie toe te kennen, aan te geven in welk domein de entiteit zich bevindt, welke relatie het heeft met andere entiteiten en kunnen er constraints (restricties) worden opgelegt. Met name het opleggen van deze restricties kan sterk bijdragen aan de uitdrukkingskracht van RDF-S bijvoorbeeld door af te dwingen dat een entiteit een geldige waarde kent.

Klassen en eigenschappen (properties) zijn de belangrijkste concepten uit RDF Schema. Hiernaast maakt RDF-S een vertaalslag van de eerder genoemde Containers naar Container Classes en Properties. Ook introduceert RDF-S het concept Collections. Collections zijn ook verzameling maar in tegenstelling tot Containers kunnen er op collections restricties met betrekking tot aantallen worden gezet. Hierdoor ontstaat er als het ware een vaste set items dat kenmerken vertoont van een lijst. Het laatste concept dat kort aangehaald dient te worden is het *reification* principe. Een reification is het benoemen van een statement tot een resource zodat men statements over statements kan uitvoeren. Hoewel dit bij RDF al mogelijk was, maakt RDF Schema het principe expliciet om de werkbaarheid te vergroten. Meer informatie over deze concepten is te vinden in de w3c documentatie [20] en dit verslag zal hier niet verder op in gaan.

Syntax

De syntax lijkt sterk op de van RDF al zijn er nieuwe tags toegevoegd op de boven genoemde concepten tot uitdrukking te brengen. Ter illustratie een klein voorbeeld van een klasse en een property.

```
<rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Resource">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>Resource</rdfs:label>
  <rdfs:comment>The class resource, everything.</rdfs:comment>
</rdfs:Class>

<rdf:Property rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <rdfs:label>type</rdfs:label>
  <rdfs:comment>The subject is an instance of a class.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:domain rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Property>
```

Semantisch uitdrukkingskracht

Het is al snel duidelijk dat RDF meer semantische uitdrukkingskracht heeft dan RDF. Door middel van de klasse structuur is men in staat met behulp van RDF Schema context te geven aan resources en de relatie met andere resources vorm te geven. Ditzelfde geldt voor het classificeren van eigenschappen. De aard van de eigenschappen wordt inzichtelijk en eigenschappen worden op een uniforme manier gepresenteerd. Verder zorgt de gegeven set van eigenschappen ervoor dat de relaties tussen resources wordt vastgelegd en kan er informatie worden gespecificeerd over het domein,zie de tag *<rdfs:domain>* in het voorbeeld. Ondanks deze voorwaartse stap is de uitdrukkingskracht nog steeds te zwak voor echte semantische gegevens op basis waarvan een web service aan een request gekoppeld kan worden. Met name het gemis aan het stellen van restricties is hier oorzaak van.

3.1.3 Web Ontology Language for Semantic Web Services (OWL-S)

De Web Ontology Language for Semantic Web Services (OWL-S) is ontology specifiek ontworpen voor een semantische benadering van web services [10]. OWL-S is gebaseerd op de ontologie OWL (Web Ontology Language). OWL maakt gebruik van het RDF schema en is bedoeld voor het beschrijven van resources op het web. OWL-S spitst zich specifiek toe op web services en de discovery van web services op basis van semantisch informatie. OWL-S is momenteel het meest belovend als ontologie in pogingen om de discovery en interactie met web services te automatiseren. OWL-S blijft niet alleen bij een beschrijvende ontologie, maar presenteert ook een matching engine. Deze engine, genaamd OWL-S/UDDI Matchmaker is een service register gericht op het semantisch matchen van een service request bij een service provider. Hierover meer in het volgende hoofdstuk. In deze paragraaf worden de concepten besproken die OWL-S typeren als ontologie. Vanwege het feit dat OWL-S gebaseerd is op OWL kan het gebruik maken van klassen en andere hiërarchische structuren.

OWL-S maakt een belangrijk onderscheid tussen de wat een service doet, hoe deze gebruikt wordt en op welke manier deze aangesproken moet worden. Respectievelijk worden deze componenten het service profiel, service proces en service grounding genoemd. Waarom dit belangrijk is voor de semantiek van web services wordt besproken in de volgende paragraaf, voor nu is het voldoende om te weten dat OWL-S dit onderscheid maakt. Elk van deze drie componenten kent een eigen subontologie binnen OWL-S, waardoor het mogelijk is gericht en specifiek specificaties toe te kennen. Het service proces en de service grounding bevatten voornamelijk informatie over hoe een web service doet wat hij doet en hoe interactie kan worden bewerkstelligd.

OWL-S richt zich dus specifiek op semantische web services. Dit komt al tot uiting in de visie van OWL-S op web services. Zo stelt Martin in zijn artikel *"Whereas interoperability is the primary motivation for Web Services, automation of information use and dynamic interoperability are the primary objectives of Semantic Web Services."* [11]. Tevens komt dit tot uiting in de functionele benadering van web services. In deze visie wordt een service request als één of meerdere doelen gezien. Deze doelen moeten bereikt worden. De doelen worden daarom uitgedrukt in vorm van proposities die waar gemaakt moeten worden.
Om dit voor elkaar te krijgen kent OWL-S de concepten *inputs*, *outputs*, *preconditions* en *results*. Preconditions zijn voorwaarden waaraan voldaan moet worden wil de web service van start gaan. Mocht aan deze voorwaarden zijn voldaan benodigd de web service één of meedere inputs, al hoeft dit niet altijd het geval te zijn. Inputs zijn gegevens welke de web service nodig heeft om zijn taak te volbrengen en leiden naar de uitkomst, ofwel het resultaat (results). Het resultaat bestaat uit de output van de web service (het daadwerkelijke resultaat) en een effect. Het effect kan gezien worden als een propositie aan welke is voldaan wanneer de web service zijn taak heeft volbracht. Merk op dat de gestelde doelen ook gezien konden worden als proposities. Met andere woorden, een effect kan zijn dat een doel is behaald.

Syntax

Zoals gezegd kent OWL-S verschillende syntaxen voor het service profiel, het service proces en de service grounding. Binnen de subontologie voor het service profiel kent OWL-S nog een *presentation syntax*, ook *surface syntax* genoemd. Aangezien het bij het service proces gaat om hoe web service doen wat ze doen, hebben we te maken met processen. De presentation syntax toont deze processen als programma's in een hoge mate van abstractie. Het voordeel van deze syntax is dat het eenvoudiger te begrijpen is voor mensen en dat de specificatie zich makkelijk laat vertalen naar een RDF/XML syntax [10].

Semantische uitdrukkingskracht

Naast de functionele benadering van web services is OWL-S ook erg implementatie gericht. Dat blijkt wel uit de OWL-S/UDDI matchmaker en het extraheren van de capaciteiten van de web service. Naarmate de ontologie sterker semantisch van aard wordt, lijken deze capaciteiten een steeds voornamere rol te krijgen. De capaciteiten geven namelijk weer wat een web service doet en sterker nog, ze geven weer wat het transformatieproces van input naar output inhoudt. Dit is cruciaal voor de semantische uitdrukkingskracht omdat het verder reikt dan terminologie waarvan we hebben gezien dat het problemen op kan leveren. De betekenis van een web service kan nu worden afgeleid uit de transformatie of zelfs uit het transformatieproces. In combinatie met de andere semantische factoren als de presentatie van het domein en het classificeren van web services vormt inzicht in de capaciteiten van een web service een sterke impuls voor de semantische uitdrukkingskracht.

Het idee van het opsplitsen van het service profiel, service proces en de service grounding komt oorspronkelijk van de ontologie DAML-S [12]. Het idee is overgenomen in de OWL-S ontologie. Vanwege de sterke overlap tussen OWL-S en DAML-S is het overbodig DAML-S apart te bespreken al is DAML-S wel noemenswaardig vanwege de ideeën die hierin gepresenteerd zijn en overgenomen worden door OWL-S. Op dit punt zijn er vier ontologieën gepresenteerd en is inzicht verkregen in hoe semantische informatie wordt toegekend aan web services. Naast dit viertal zijn er nog enkele andere ontologieën en kent elk geheid één of meerdere variaties of extensies. Duidelijk naar voren is gekomen dat de representatie van de capaciteiten en het transformatieproces een vooraanstaande rol spelen. De volgende paragraaf zal daarom verder in gaan op de opsplitsing van het service profiel en het service proces en op twee verschillende methoden om capaciteiten te representeren.

3.2 Profiel en proces scheiden

Het expliciet maken van het verschil tussen intentie en functionaliteit kan sterk bijdragen aan de semantische uitdrukkingskracht van web services [8]. Zoals Gibbins in zijn artikel schrijft: *"In the conventional Web Services approach exemplified by WSDL or even by DAML Services, the communicative intent of a message (for example, whether it is a request or an assertion) is not separated from the application domain"* [8]. Ofwel, er wordt onderscheid gemaakt tussen domein specifieke zaken, en domein neutrale zaken of zoals Gibbins het noemt de communicatieve intentie.
Indien dit onderscheid wordt gemaakt met betrekking tot web services, is men in staat om de communicatieve intentie vast te leggen in een Agent Communication Language (ACL). Een ALC is een taal waarin verschillende agents met elkaar kunnen communiceren. Een sterke definitie wordt gegeven door Labrou & Finin [13]: *"Agent communication language allow agents to effectively communicate and exchange knowledge with other agents despite differences in hardwareplatform, operating systems, architectures, programming languages and representation and reasoning systems."*. Zoals de definitie van Labrou & Finin al aangeeft zorgt een ACL voor onafhankelijkheid en eenduidigheid. Momenteel zijn er twee bekende ACL's die als standaard kunne worden beschouwd. De eerste is de Knowledge Query and Manipulation Language (KQML), welke enigszins ingehaald c.q. vervangen is door FIFA-ACL dat staat voor Foundation for Intelligent Physical Agents. Voldoende voor dit verslag is te weten dat deze standaarden bestaan en dat zij voldoende capaciteit bezitten om de communicatieve intenties van web services uit te drukken [8].

Nu we in staat zijn de domein neutrale boodschap, ofwel communicatieve intentie te verwoorden in een gestandaardiseerde taal kunnen we kijken naar de tweede helft van de opsplitsing, de domein specifieke functionaliteit. Dit gedeelte kan worden vastgelegd middels ontologieën, bijvoorbeeld door de ontologieën besproken aan het begin van dit hoofdstuk. Het domein specifieke karakter en met name het expressieve vermogen met betrekking tot relaties tussen entiteiten maakt een ontologie als DAML-S of OWL-S een voor de hand liggende keus. Het opsplitsen van deze twee componenten brengt buiten het vergroten van de semantische uitdrukkingskracht nog een aantal andere voordelen met zich mee. Zo is het mogelijk om enkel het domein specifieke deel te veranderen indien er iets relevants in de wereld veranderd, en kan de beschrijving van de communicatieve intentie intact blijven. Aan de andere kant kunnen in dit domein neutrale deel concepten worden gecombineerd en kunnen er patronen en generieken uit worden afgeleid en gebruikt ten behoeve van de eenduidigheid en hanteerbaarheid.

Het maken van dit onderscheid kan op vele communicatieve berichten worden toegepast. Met betrekking tot web services kunnen we dit onderscheid concreet maken door de web service beschrijving ook op te delen in twee delen. Het eerste deel omvat de domein neutrale informatie en is genaamd het service proces (service process) en het tweede deel omvat de domein specifieke informatie en is genaamd het service profiel (service profile). Eerder in dit verslag werd ook gesproken over het profiel als geheel, vanaf nu zal met de term service profiel de domein specifieke informatie worden bedoeld, met het service proces de domein neutrale informatie en met de term service beschrijving het geheel.

Het concept lijkt veelbelovend, maar wat betekend het nu werkelijk voor de semantische uitdrukkingskracht en wat draagt het bij aan de automatisering van de discovery van web services. Deze bijdrage is vooral terug te vinden in het service profiel. Het service profiel is namelijk in staat om - domein specifiek - capaciteiten te representeren, met andere woorden te representeren wat de functionaliteit van de web service is. Dit is een zeer krachtige en

bruikbare methode om één of meerdere geschikte service provider(s) te vinden bij een bepaalde service request, hierover meer in het volgende hoofdstuk. Er worden in de literatuur twee vormen van representatie van capaciteiten beschreven, een impliciete vorm en een expliciete vorm [14].

Bij een expliciete representatie worden alle taken van een web service als concepten binnen een ontologie beschreven. Deze directe aanpak vergt een grote flexibiliteit van de ontologie, aangezien elke taak uitgedrukt moet worden als een apart concept. Als het aantal taken dat een bepaalde web service heeft stijgt, stijgt de behoefte voor nieuwe concepten evenredig mee. Bij grote aantallen taken leidt tot een schaalprobleem en gaat dit ten koste van de hanteerbaarheid van de ontologie, die in een dergelijk geval erg complex wordt. Bij de tweede representatie - de impliciete representatie - wordt er hoofdzakelijk gekeken naar de transformatie van de input naar de output. Hetgeen wat er in dit proces gebeurd representeert de functionaliteit en dus de capaciteit van de service. Omdat veel verschillende taken gelijklijkende resultaten kunnen opleveren, wordt hier het schaalbaarheids probleem aangepakt. Hiermee kunnen we niet concluderen dat de impliciete aanpak beter is dan de expliciete aanpak, enkel dat bij het construeren van het service profiel overwogen dient te worden voor welke aanpak men kiest. Belangrijke factoren hierbij zijn de taken van de web service, het domein waarin deze zich bevindt en de ontologie die gebruikt wordt.

Naast de capaciteiten van de service worden er ook nog twee andere categorieën aan gegevens meegegeven [14]. De eerste categorie informatie is die van niet functionele parameters en de tweede categorie bestaat uit een beschrijving van een persoon of instantie die verantwoordelijk is voor de service. Hoewel de gegevens uit de laatste categorie niets bijdragen aan de semantische beschrijving, is het wel interessant om naar de tweede categorie te kijken. Hoewel het bij informatie uit deze categorie niet om functionele informatie gaat, is deze wel geschikt overige informatie mee te geven aan een service. Bijvoorbeeld als het een service betreft die banktransacties voltooid, kunnen via deze weg de transactie kosten bekend worden gemaakt zodat deze inzichtelijk zijn voor de service requester. Een andere mogelijkheid is om gegevens betreffende de classificatie van een service mee te geven. Dit kan semantisch gezien zeer waardevol zijn bij zowel de discovery als compositie van web services. Hierover meer in de volgende paragraaf en het volgende hoofdstuk.

De beschrijving van het service proces biedt vooral informatie over hoe de web service aangesproken moet worden en beschrijft gedetailleerder hoe de web service te werk gaat. Het tonen van de stappen die de web service uitvoert is niet verplicht. Sommige services laten alle stappen duidelijk zien, we spreken dan van een *white box* principe, andere geven enkel te kennen wat er aan input komt en hoe de output eruit ziet. In het laatste geval spreekt men van het *black box* principe. Vaak wordt een gedeelte van het transformatieproces getoond wat men het *gray box* principe noemt. Functioneel gezien heeft de service requester weinig aan deze informatie, al is de requester in staat om enige kwalitatieve waarde aan een service toe te kennen. Concreet gezegd kan een service aan de hand van de getoonde stappen - in het geval van een white box of gray box - na gaan of deze stappen leiden tot het gewenst resultaat. Alle benodigde informatie over hoe de communicatie met deze service verder geschied is ook terug te vinden in het service proces.

Tenslotte speelt het begrip *grounding* nog een kleine rol. De daadwerkelijke koppeling tussen een service requester en een service provider wordt het proces van grounding genoemd. Bij het gebruik van ontologieën wordt de benodigde informatie voor de koppeling gespecificeerd.

Het grounding proces zorgt er in dit geval voor dat deze beschrijving wordt geabstraheerd van de rest van het proces model en wordt omgezet in WSDL beschrijvingen.

In het weerbericht voorbeeld kan het zijn dat de web service gebruik maakt van de informatie van andere web services om tot het gewenste resultaat te komen. Zo kan service een web services benaderen die informatie de waterstand, luchtvochtigheid, kans op regen, kans op zon, temperatuur, windkracht en zelfs seismische activiteiten opleveren. Deze informatie kan de service vervolgens bundelen in één antwoord op de vraag wat het weerbericht is. Dit proces wordt de compositie van web services genoemd en zal het onderwerp zijn van de volgende paragraaf.

3.3 Compositie

Zoals het hierboven staande voorbeeld aan aangeeft is het mogelijk om meerdere web services te combineren om tot een gewenst resultaat te komen. In deze paragraaf worden kort de mogelijkheden en knelpunten besproken van deze techniek.

Een geslaagde vorm van de automatisering dient in staat te zijn praktisch te allen tijden de gewenste functionaliteit te verkrijgen. In sommige gevallen kan het zijn dat dit niet het geval is en komt men uit op twee scenario's. In het eerste geval maakt een web service zelf gebruik van andere web services om voldoende informatie te bemachtigen. Het probleem van gebrek aan informatie wordt erkent aan de service provider kant welke vervolgens actie onderneemt. In het tweede geval wordt het probleem erkent door een agent of andere 'matchmaker'. Hier zoekt de agent de benodigde web services bij elkaar.

Wanneer men spreekt over de compositie van web services, heeft men het al snel over processen. Het sequentieel uitvoeren van verschillende services kan namelijk gezien worden als kleine deelprocessen die samen het proces van de web service vormen. Hierdoor ontstaat er een proces dat sterke kenmerken vertoont van een workflow proces [12]. Een workflow proces laat zich namelijk kenmerken door een synchrone of asynchrone opeenvolging van kleine stappen. Het pad dat bewandeld wordt tussen deze deelprocessen kan steeds verschillen, ofwel er kunnen onderweg keuzes gemaakt worden welke richting men op gaat. Deze processen van samengestelde services vormen een belangrijk onderdeel binnen web service compositie, de DAML-S coalitie beschrijft daarom drie vormen van processen [12].

3.3.1 Processen

1. Atomaire processen (atomic processes).

Atomaire processen zijn direct uitvoerbare services die geen subprocessen bevatten en dus geen gebruik maken van andere services. Zij zijn vanuit de service requester gezien met een enkele stap uit te voeren door een het uitwisselen van één of meerdere SOAP berichten. Atomaire processen moeten zijn uitgerust met een service grounding beschrijving om de service requester te voorzien van de benodigde informatie voor het aanspreken van de web service en het interpreteren van het antwoord.

2. Simpele processen (simple processes).

Simpele processen zijn niet direct aanspreekbaar en kennen ook geen service grouding beschrijving. Simpele processen kunnen vaak wel gezien worden als processen die in één stap zijn uit te voeren, vormen vaak een element van een abstract proces. Bijvoorbeeld het proces

dat gerealiseerd wordt door één of enkele atomaire processen kan gezien worden als een simpel proces. Simpele processen dragen qua functionaliteit niet veel bij omdat ze gevormd woorden door atomaire processen, maar worden vaak toegepast bij het plannen en synchroniseren van of redeneren over processen.

3. Samengestelde processen (composite processes)

Samengestelde processen zijn processen die zijn op te delen subprocessen welke samengesteld of niet samengesteld kunnen zijn. Kernmerkend aan deze processen is dat er controlerende structuren worden toegepast. Deze controlerende structuren bepalen uiteindelijk de dataflow van het proces, ofwel het pad dat wordt bewandeld vanaf de input tot de output. Merk op dat het meeste abstract samengestelde proces in feite het transformatieproces van de web service uitbeeld. Ter verduidelijking het onderstaande figuur.

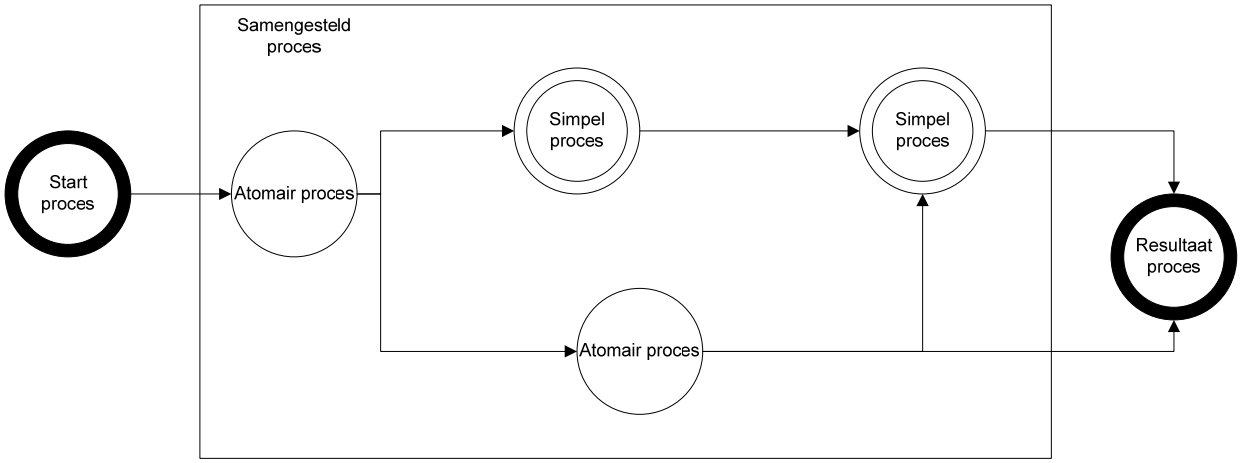

Fig. 3 Samengesteld, simpel en atomair proces

3.3.2 Business Process Execution Language for Web Services (BPEL4WS)

BPEL4WS wat staat voor Business Process Execution Language for Web Services is een specificatietaal waarin web service processen kunnen worden gespecificeerd. Met BPEL4WS ook wel genaamd WS-BPEL kan een ontwerper workflows specificeren in een service georiënteerde omgeving. In termen van web services kan worden beschreven welke web services in welke volgorde aangesproken dienen te worden en hoe de benodigde data als input bij de juiste web service terecht komt. BPEL4WS is flexibel en uitgebreid genoeg om sterke workflow concepten zoals if-then-else statements, herhalingen en samenvoegingen uit te drukken.

Hoewel BPEL4WS een sterke functionele uitdrukkingskracht kent, kan het moeilijk overweg met de conceptuele en semantische kant van web services. Lee e.a. stellen in hun artikel voor om BPEL4WS daarom te combineren met de OWL-S ontologie om semantiek toe te kennen aan web services, bijvoorbeeld door het verschil tussen het service profiel, het service proces en de service grounding te onderkennen [15]. Hoewel dergelijke initiatieven bijdragen aan de semantische uitdrukkingskracht is BPEL4WS nog steeds niet dynamisch en flexibel genoeg voor geautomatiseerde compositie van web services. Het grote struikelpunt is namelijk dat het adaptief vermogen ten tijde van uitvoering onvoldoende is. Hoewel BPEL4WS over enige capaciteit beschikt om fouten te kunnen opvangen en afhandelen, is het niet in staat ‘on the fly’ andere web services aan te spreken op het moment dat informatie niet beschikbaar is. Stel dat in het weerbericht voorbeeld de service die de huidige temperaturen aanlevert ontbreekt, dan is de huidige structuur niet in staat een andere web service te vinden die het wel kan. Het proces waarin staat beschreven welke web services in welke volgorde worden uitgevoerd staat namelijk vast.

3.4 Conclusie eerste deelvraag

In dit hoofdstuk is de nadruk gelegd op het beantwoorden van de vraag *“Hoe wordt semantiek toegekend aan Web Services?”*. We hebben gezien dat de semantische uitdrukkingskracht naar voren komt door drie aspecten, te noemen de representatie, de opsplitsing van het profiel, het proces en de grounding en het samenstellen van verschillende web services om nieuwe of uitgebreidere functionaliteit te creëren. De representatie wordt verzorgd middels ontologieën, specificatietalen die web services beschrijven. Sommige beschrijvend op metadata niveau, sommige beschrijvend op semantisch niveau. Semantische informatie over de functionaliteit, het transformatieproces, het domein en het effect van een web service kan hierdoor tot uitdrukking worden gebracht. De opsplitsing van een web service in de drie genoemde componenten heeft als resultaat dat er gerichte informatie aangeboden kan worden betreffende hetgeen een web service doet, hoe hij dit doet en kan er gerichter om worden gegaan met de input, output en het transformatieproces. Tenslotte biedt het samenvoegen van web services in theorie de mogelijkheid om ten tijden van uitvoering te switchen tussen web services op momenten dat de beschikbaarheid van services of de vraag naar informatie komt te veranderen. Al deze concepten dragen bij aan de mogelijkheid om het zoeken, vinden en koppelen van web services op semantische basis te automatiseren. Dit kan enkel worden bewerkstelligd indien deze initiatieven niet alleen op web service niveau worden ondersteund, maar ook in de registers, daar waar de discovery en het matchen van web services plaatsvindt. Het volgende hoofdstuk zal hier nader op ingaan.

4. Hoe wordt de semantiek van Web Services verwerkt in registers?

Semantische web services in registers, dat is waar het in dit hoofdstuk om draait. Eerst wordt bekeken hoe web services in registers worden gepubliceerd, vervolgens komt de discovery en het matchen van web services aan de orde. Daarna verdient de OWL-S/UDDI Matchmaker het besproken te worden als een echte semantische matchmaker en tenslotte zullen de nadelen van dit register worden besproken.

4.1 Universal Description Discovery Integration

Het Universal Description Discovery Integration (UDDI) register kan gezien worden als het telefoonboek van de web service wereld. Het register is oorspronkelijk opgesteld in een samenwerkingsverband tussen Microsoft, IBM en Ariba. De UDDI standaard is momenteel het meest gebruikt en geaccepteerd als standaard voor service registers. Het register past categorisatie toe om het structureren en organiseren van verschillende services te optimaliseren ten behoeve van de service discovery. Merk op dat het hier gaat over een ‘handmatig’ discovery proces - met zoekmachines die de onderstaande begrippen als zoektermen gebruiken - zonder automatiseringstoepassingen en zonder de beschikking over semantische data. Het UDDI informatiemodel bestaat uit instanties van de volgende types: [2]

- Business Entiteit (businessEntity): Deze entiteit beschrijft een bedrijf of organisatie dat de web service aanbiedt, met andere woorden het beschrijft de service provider.
- Business Service: (businessService): Instanties van deze entiteit beschrijven een collectie van gerelateerde web services aangeboden door een service provider die beschreven wordt in een Business entiteit.
- Binding Template (bindingTemplate): Instanties van deze entiteit beschrijven de technische informatie benodigd voor het gebruik van een bepaalde web service.
- T-Model (tModel): Dit technische model beschrijft een herbruikbaar concept zoals een communicatieprotocol of een categorisatiesysteem.
- Publisher Assertion (publisherAssertion): Hierin wordt een ‘view’ op een bepaalde business entiteit beschreven in termen van relaties tot andere business entiteiten
- Subscription: Hierin wordt een statusrapport met onder andere alle veranderingen in bijgehouden.

In de meest basale vorm wordt een web service in het UDDI register beschreven aan de hand van deze concepten. Op het moment dat een service requester een service zoekt geeft het bepaalde zoektermen door aan het register. Het register levert vervolgens een lijst met geschikte kandidaten op en de service requester kan hier vervolgens uit kiezen en ontvangt de informatie benodigd om gebruik te maken van de service. Vanwege het statische en syntactische karakter van de beschrijving is het enkel mogelijk het selectieproces ten tijde van ontwerp uit te voeren en is dynamische aanpasbaarheid niet haalbaar. Er is behoefte aan verwerking van semantische informatie in registers.

4.2 Discovery en Matching op basis van semantische gegevens

In deze sectie wordt beschreven hoe web services op basis van semantische gegevens kunnen worden gevonden en hoe een service request aan een web service gekoppeld kan worden. Hierbij wordt uitgegaan van de toepassing van ontologieën, het splitsen van het profiel, proces en de grounding en de mogelijkheid tot compositie van web services. Registers krijgen de rol van agents toegewezen die zorgen voor de automatische discovery, dienen als communicatielink tussen de service requester en de service provider en zorgen voor de compositie van web services indien het niet mogelijk is met één web service te voldoen aan een request.

Op het moment dat een service requester een aanvraag doet bij een service register construeert de requester op basis van zijn verzoek het ideale service profiel, met andere woorden er wordt een beschrijving gemaakt van de ideale web service voor dit probleem [14]. Deze beschrijving wordt vervolgens naar het register gestuurd, welke we vanaf nu ook wel matchmaker mogen noemen. Het register heeft immers niet meer enkel de functie van telefoonboek, maar probeert nu de juiste service te vinden voor het beschreven probleem. De matchmaker zoekt op basis van de door de service provider gepresenteerde capaciteiten de beste match en presenteert deze aan de service requester inclusief informatie met betrekking tot de service grounding. De verdere communicatie tussen de service requester en de service provider wordt geleid door de beschreven service grounding en het service proces. Zo weten beide partijen wat er van elkaar verwacht wordt en kan de service provider de kwaliteit van het resultaat in het oog houden.

Een voorbeeld van een vooraanstaande matchmaker is de OWL-S/UDDI Matchmaker. Gebaseerd op OWL-S als leidende ontologie en bewapend met technieken als compositie en het evalueren van het transformatieproces van input tot output komt deze matchmaker sterk tot zijn recht.

4.3 OWL-S/UDDI Matchmaker

De OWL-S/UDDI Matchmaker is een praktische implementatie van de in dit verslag beschreven theoretische concepten. In deze paragraaf zullen we kijken naar hoe de OWL-S/UDDI Matchmaker deze concepten in de praktijk brengt, wat de sterke punten zijn van de matchmaker maar ook waar punten van verbetering liggen.

4.3.1 Methoden en sterke punten

Naast de beschreven methoden in de bovenstaande sectie maakt de OWL-S/UDDI Matchmaker gebruik van specifieke matching algoritmen. Deze algoritmen komen in twee smaken. De eerste soort richt zich op de representatie van capaciteiten, de andere richt zich op het transformatieproces van input naar output. Bij het matchen wordt gelet op de gelijkenis van capaciteiten of de gelijkenis van processen en outputs tussen het service profiel van een web service en het geconstrueerde service profiel op basis van de service request. De web service die de hoogste gelijkenis kent, wordt gepresenteerd als de beste kandidaat. Merk op dat deze web service niet in staat hoeft te zijn een oplossing te bieden voor het gehele probleem, maar dat deze bij kan dragen door aan één van de gevraagde capaciteiten te voldoen.

Twee sterke punten van de OWL-S/UDDI Matchmaker zijn het omgaan met verschillende mate van matches en het bezitten van foutafhandeling capaciteiten. Het eerste punt komt tot uiting door het verschil te onderkennen in een *exact match, plug-in match, subsumes match, subsumed-by match, logic based fail, nearest-neighbor match* en *fail* [19]. Een korte uitleg volgt.

- Exact match
  Service S en request R vormen een exacte match.

- Plug-in match
  De plug-in match is afgeleid van het software engineering domein. Deze match is minder streng dan de exact match, daar waar service S minder input nodig heeft dan request R aanvoert, maar waar de output van S precies of bijna precies overeen komt met de verwachte output van R.

- Subsumes match
  Deze match lijkt sterk op de Plug-in match, maar is ook minder strikt wat betreft de output. De output is in dit geval specifieker dan gevraagd.

- Subsumed-by match
  Deze match is gelijk aan de subsumes match, behalve dat de output meer generiek is dan gevraagd.

- Logic-based fail
  Op basis van de logica afgeleid uit de semantische gegevens is er geen match tussen service S en request R. Merk op dat op basis van input / output wel een match gevonden kan worden, in dit geval is er waarschijnlijk sprake van een terminologieprobleem.

- Nearest-neighbor match
  Service S komt het best overeen met request R.

- Fail
  Er is geen enkele match van één van de bovenstaande typen te vinden tussen Service S en request R.

Het onderscheid tussen de verschillende mate van matches zorgt ervoor dat een service requester een betere inschatting kan maken van het resultaat dat een bepaalde service kan leveren en op basis daarvan kan beslissen welke service hij gaat aanspreken.

Het tweede sterke punt is de foutafhandeling capaciteiten die de matchmaker bezit. Alle stappen van een service request tot de discovery en het matchen met een service krijgen een procesachtig karakter. Als dit proces wordt bekeken als een workflow is het belangrijk dat een proces altijd eindigt in een zinnige eindtoestand. Op het moment dat er gaande het proces iets mis gaat dat moet er voor een oplossing worden gezorgd. In het geval van web services kan het zijn dat een bepaalde – gepubliceerde - service niet gevonden kan worden of dat de verkeerde match wordt gemaakt. De OWL-S/UDDI matchmaker is in staat proces onderbrekingen te signaleren en de communicatie met de service requester af te handelen.

4.3.2 Punten van verbetering

Indien het mogelijk wordt het matchen van web services te automatiseren via semantisch gebaseerde registers, is het aannemelijk dat de register(s) snel in omvang groeien. Dit kan leiden tot problemen met de schaalbaarheid van de registers [1]. De performance kan hierdoor aangetast en hoewel het tegenstrijdig lijkt wordt het moeilijker om de beste match te vinden. Hoewel men in eerste instantie denkt dat deze match beter gevonden kan worden door het ruime aanbod doet er een ander probleem de kop op. Met OWL-S als beschrijvende ontologie is het namelijk erg moeilijk om verschillen tussen web services helder te krijgen indien deze veel op elkaar lijken, bijvoorbeeld omdat ze zich in eenzelfde domein bevinden. De ontologie is in dit geval te abstract. Een mogelijke oplossing voor dit probleem is het ontwikkelen van domeinspecifieke registers die met elkaar communiceren op abstract niveau, maar intern gebruik kunnen maken van een domeinspecifieke ontologie.

Een tweede verbeterpunt van de OWL-S/UDDI matchmaker is het verbeteren van de robuustheid en flexibiliteit van compositietechnieken [16]. Zoals we gezien hebben is de OWL-S/UDDI Matchmaker in staat fouten op te vangen, maar is het lastig om ten tijden van uitvoering stappen te wijzigen en op een andere manier voor het gewenste resultaat te zorgen. Stel dat in het weerbericht voorbeeld de service die zorgt voor informatie over de luchtvochtigheid komt te vervallen, zijn er andere methode nodig om de kans op neerslag te bepalen. Dit kan bijvoorbeeld door informatie uit andere regio's te combineren met windrichting. Hier is specifieke domein- en proceskennis voor nodig. Deze kennis moet kunnen worden beschreven in machinetaal voordat een software agent hiermee kan werken. Dit is erg complex en tot op heden dag is er nog geen methode om dergelijke kennis correct en expliciet te beschrijven.

4.4 METEOR-S Web Service Discovery Infrastructure (MWSDI)

Een mogelijke oplossing voor het schaalbaarheids probleem van web service register wordt gegeven door de METEOR-S Web Service Discovery Infrastructure (MWSDI) [1]. Deze infrastructuur maakt gebruik van een peer-to-peer structuur tussen domeinspecifieke registers. Een peer-to-peer netwerk is een netwerk waarin alle knopen gelijkwaardig zijn. In tegenstelling tot het bekende client/server paradigma, waarin vaak één server informatie voorziet aan meerdere clients, is het vaak zo dat alle deelnemers in een peer-to-peer netwerk zowel informatie vragend als biedend zijn. Als de registers via deze structuur met elkaar communiceren kunnen alle registers een domeinspecifieke ontologie hanteren zodat web services uit dat specifieke domein erg precies kunnen worden beschreven. Tevens kunnen de registers matches maken op basis van gegevens die in dat domein sterke betekenis hebben. Een bijkomend voordeel van de gelijkheid van registers in een peer-to-peer netwerk is dat het relatief makkelijk is van rol te wisselen. De service provider kan indien hij meer informatie nodig heeft van de service requester, de rol van requester op zich nemen en de oorspronkelijke requester kan in deze informatie voorzien door de rol van provider aan te nemen.

4.5 Conclusie tweede deelvraag

In deze paragraaf hebben we gezien hoe web services in registers worden beschreven. Het UDDI register dat service requesters voorziet van informatie over het domein van een web service alsmede de technische informatie voor het benaderen van de web service, is uitgebouwd tot een ware matchmaker. De OWL-S/UDDI matchmaker is een heuse stap voorwaarts in de reis naar een volledig geautomatiseerde vorm van web service discovery en matching. De matchmaker maakt gebruik van de semantische beschrijvingen van services en de representatie van hun capaciteiten om op functioneel niveau de juiste service bij een bepaalde request te vinden. De praktische vaardigheid van foutafhandeling gepaard met de verschillende matching niveau's, maken dat de OWL-S/UDDI matchmaker gezien kan worden als een software agent gebaseerd register waarin OWL-S als ontologie volledig tot zijn recht komt.

Ondanks deze sterke punten kleven er ook verbeterpunten aan deze matchmaker. Zoals genoemd vormen de schaalbaarheid en de flexibiliteit van compositiemethoden de voornaamste issues. Zoals we gezien hebben kan de schaalbaarheid worden opgelost door de verspreiding van de totale verzameling web services over domein specifieke registers die in een peer-to-peer gestructureerde omgeving met elkaar kunnen communiceren. De praktische implementatie van deze structuur heeft hoogstwaarschijnlijk tot gevolg dat er nog meer eisen worden gesteld aan de compositie van services aangezien de resultaten van verschillende registers bijeen moeten worden gebracht. In het volgende hoofdstuk worden web services met alle belangrijke concepten uit de bovenstaande hoofdstukken gemodelleerd om de essentiële concepten helder te krijgen en de onderlinge relaties inzichtelijk te maken. In het laatste hoofdstuk van dit verslag worden deze inzichten vervolgens gebruikt om een voorstel te doen ter verbetering van het compositieproces van web services.

5. Het Object Role Model

In dit hoofdstuk worden de concepten die een belangrijke rol spelen bij semantische web services gemodelleerd. Door middel van deze modellen wordt geprobeerd relaties tussen de concepten zo expliciet mogelijk weer te geven. Op deze manier kunnen entiteiten of relaties tussen entiteiten aan het licht komen die belangrijk kunnen zijn voor de verbetering van de compositie van web services. Met andere woorden, de modellen in dit hoofdstuk dienen als basis voor de suggesties in het volgende hoofdstuk ter verbetering van de compositie van web services. In de modellen wordt uitgegaan van de bovenstaande concepten. Dit houdt in dat de modellen uitgaan van de semantische beschrijvingen van web services, uitgedrukt in een ontologie die afhankelijk is van het domein waarin de web service zich bevindt. Het onderscheid tussen het service profiel, service proces en de service grounding wordt meegenomen evenals het uitgangspunt van een peer-to-peer architectuur.

5.1 De reden voor ORM als modelleertaal

De Object Role Modeling methode is een modelleermethode die oorspronkelijk bedoeld is voor het ontwerpen van database modellen op conceptueel niveau [22]. ORM is zoals zijn naam doet suggereren een modelleermethode die gebruik maakt van objecten en hun relaties. Deze relaties kunnen zeer expliciet worden uitgedrukt en er kunnen beperkingen (constraints) aan deze relaties worden opgelegd. De lezer wordt geacht enige kennis te hebben van ORM als modelleertaal, daar waar ORM in dit verslag niet wordt besproken. Voor een overzicht van ORM zie [22].

Kijkend naar het probleem, het inzichtelijk maken van de concepten die een rol spelen bij de semantische matching van web services en het zoeken naar verbeterpunten van de compositie van web services, is het belangrijk de relaties tussen de verschillende entiteiten expliciet te maken. Juist door deze behoefte is gekozen voor ORM als modelleertaal, daar waar ORM in staat is relaties sterk uit te drukken. Een tweede argument voor een ORM benadering is de overzichtelijkheid van ORM. Zelfs voor degene die de formele notatie en betekenis van ORM niet begrijpen zijn de modellen inzichtelijk en relatief eenvoudig te lezen en begrijpen.

5.2 Twee modellen

Ten behoeve van de overzichtelijkheid is gekozen voor de opdeling in twee modellen in plaats van één groot model. Het eerste model maakt de web service als entiteit inzichtelijk en geeft de relaties met onder andere het domein en de service request weer. Het tweede model maakt de relatie tussen een web service en een register inzichtelijk.

Eerste model

Het eerste model draait met name om de web service zelf en de relatie met het domein waarin deze zich bevindt, de verschillende beschrijvingscomponenten en natuurlijk de relatie met een service request. Uit het model is af te lezen welke relaties een web service met de andere entiteiten heeft en welke beperkingen er gelden. Zo heeft elke web service precies één service profiel en hoort elk service profiel bij een web service. Een minder strenge relatie is die tussen een web service en een ontologie. Elke web service wordt door één of meerdere ontologieën tot uitdrukking gebracht en er kunnen ontologieën zijn die geen, één of meerdere web services tot uitdrukking brengen. Wat opvalt aan het model is de losse relatie tussen een web service en een service request. Deze relaite is met rood aangegeven. De enige beperking in deze relatie is dat elke web service een match moet vormen met een service request. Deze beperking duidt enkel op de relevantie van de web service, zonder een theoretische service request is de web service niet relevant. Daarnaast is deze relatie niet erg standhoudend. Het is een relatie die tijdelijk voort komt uit het discovery en matching proces. In dit proces kan de relatie tussen de entiteit web service en de entiteit service request als de match gezien worden. Opvallend hieraan is dat juist die match een cruciale rol speelt in de automatisering van web services, maar dat deze niet expliciet wordt gemaakt in het huidige proces van discovery en matching van web services.

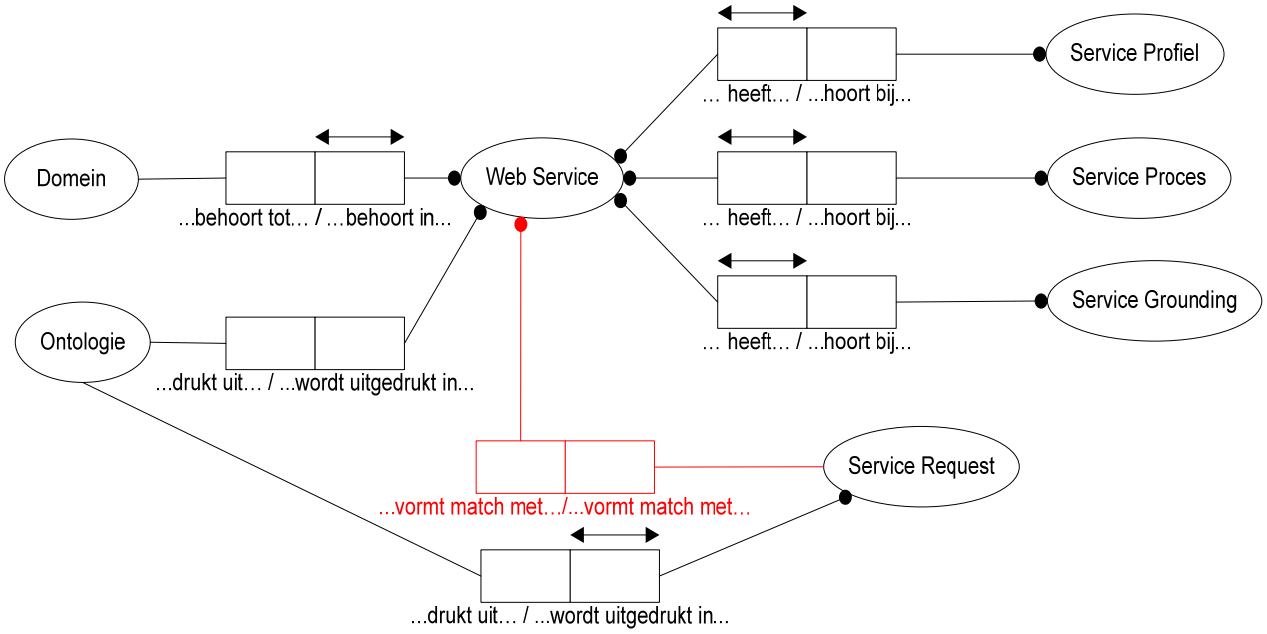

Fig. 4 Web Service ORM model

Tweede model

Het tweede model drukt de relatie tussen web services en een service register uit. De compositie van web services wordt uitgedrukt door een powertype van de entiteit web services. Dit powertype geeft de groepering ofwel samenstelling van verschillende instanties aan. Met andere woorden het beschrijft een groep web services als één object. De compositie van web services kan zelf ook een relatie met het register hebben, namelijk in het geval het register meerdere web services combineert om te voldoen aan een service request. Het volgende model maakt de compositie en de match van een compositie met een register expliciet.

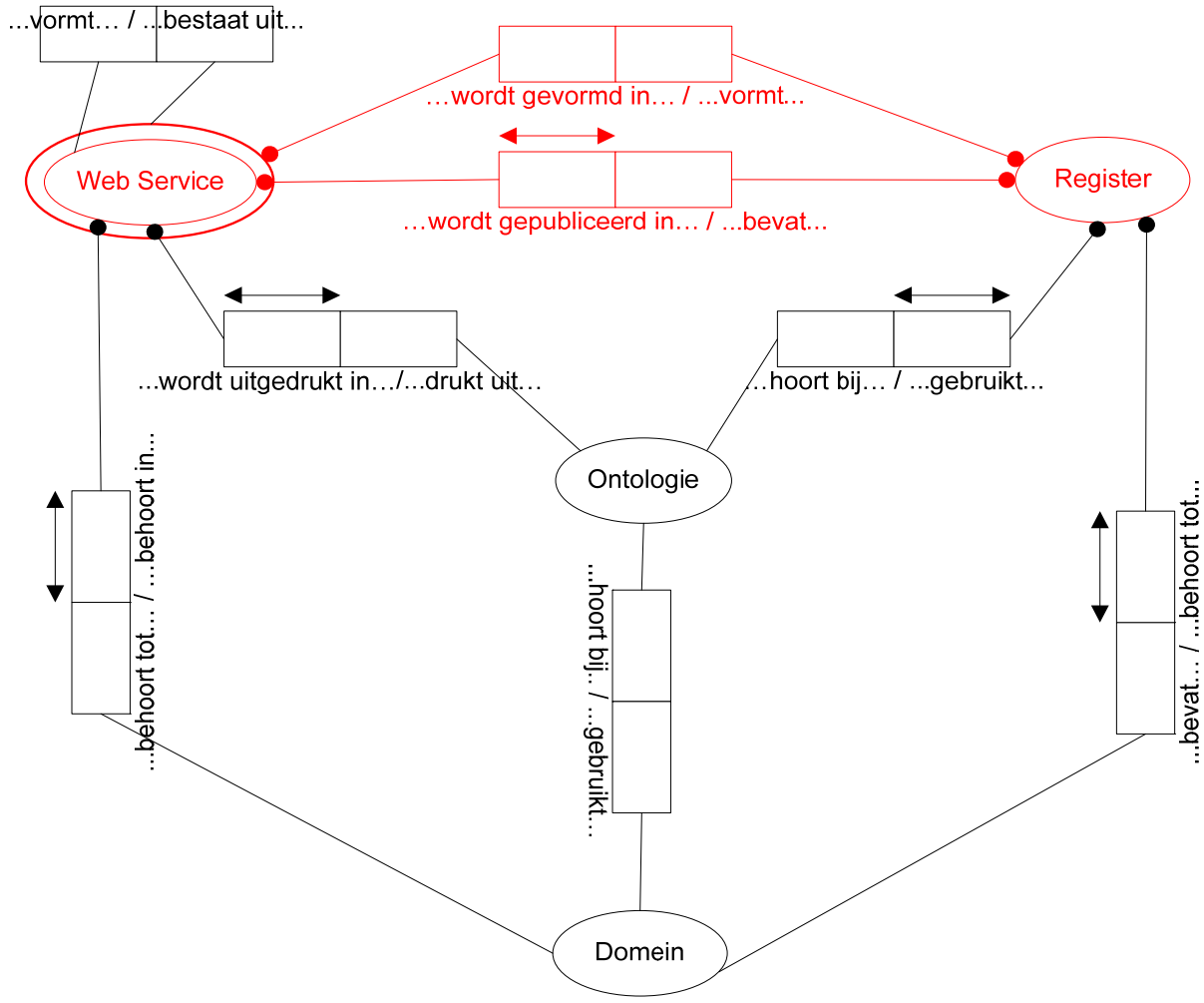

Fig. 5 Web service compositie en register

Op het moment dat de web service compositie wordt gecombineerd met een expliciete objectificatie van een match - een match is nu een entiteit geworden in plaats van een relatie -, ontstaan er twee smaken matches. Een directe match die door één web service wordt gevormd en een compositie match die wordt gevormd door een compositie van web services. De generalisatie van deze twee smaken vormt de gehele entiteit match. Het model ziet er als volgt uit.

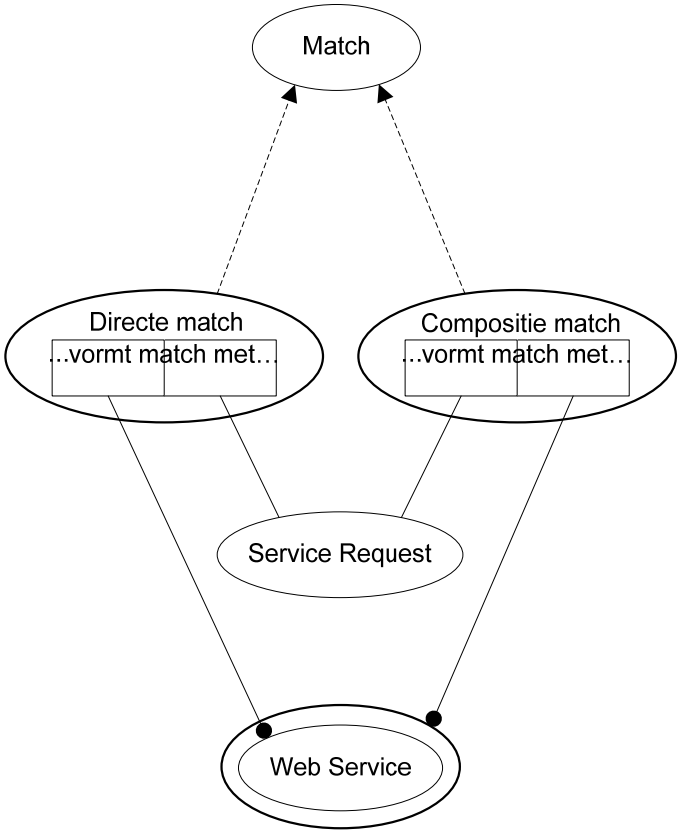

Fig. 6 Objectificatie en generalisatie van een match

Na de objectificatie van de directe match en de compositie match en het samenvoegen tot de entiteit match is het modeltechnisch gezien mogelijk geworden deze entiteit te populeren. Zoals eerder beschreven is ORM oorspronkelijk bedoeld voor het ontwerp van database structuren. Databases kunnen gevuld worden met instanties, welke samen de populatie vormen. In het laatste model is het dus mogelijk matches op te gaan slaan als combinaties van een service request en een bijpassende web service en is het mogelijk deze combinaties relaties met andere entiteiten te laten krijgen. Met deze theoretische benadering van een expliciete vorm van matches wordt in het volgende hoofdstuk bekeken welke relatie(s) een match kan maken met andere entiteiten en wordt dit weer terug vertaald naar een praktische suggestie ter verbetering van het compositieproces.

6. Verbetering compositie op basis van het model

6.1 Verbetering van het model

Nu de relatie match tot een entiteit verheven is wordt er gekeken welke relaties deze aan kan gaan. Kijkend naar de concepten die een belangrijke rol spelen bij semantische web services is te zien dat de mate van matching nog niet terug is gekomen in de modellen. De mate van een match heeft namelijk niet direct betrekking op een web service of een service request, maar op de match tussen die twee. Als de link tussen een match en een bepaalde mate van matching - ter duidelijkheid vanaf nu niveau genoemd - is gelegd kan ook deze relatie worden verheven naar een entiteit waarvan de instanties bestaan uit een match met een bepaald niveau. In het model wordt deze entiteit - vanaf nu te noemen het match record - als volgt weergegeven.

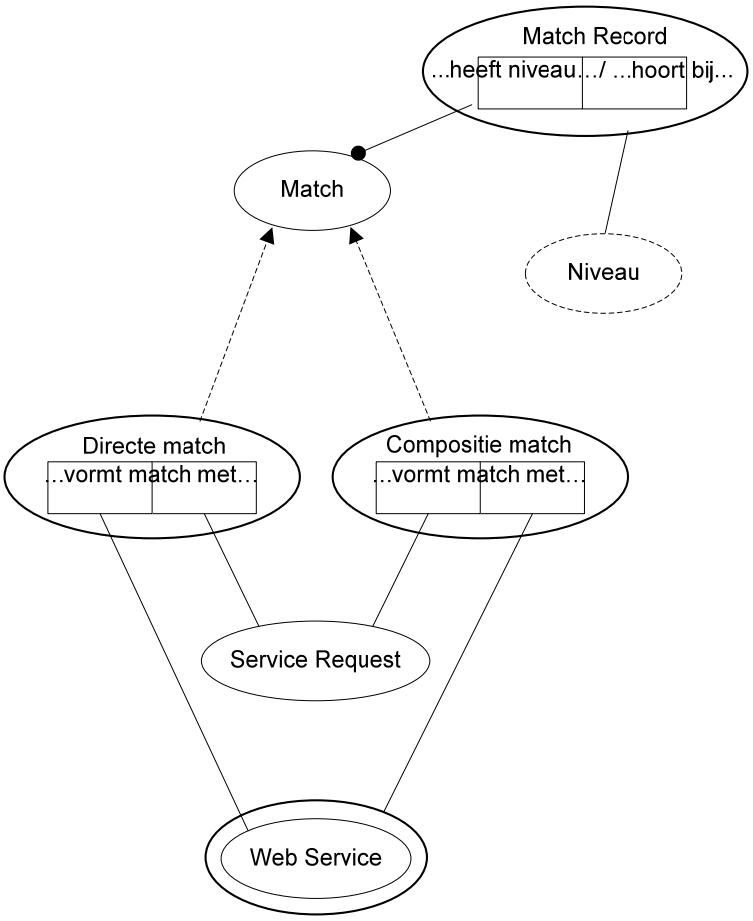

Fig. 7 Objectificatie van een match record

Een match record bevat nu eigenlijk alle informatie die wordt gebruikt in het matchen van een service request aan een web service. Het bestaat uit een match met een bepaald niveau, de match bevat vervolgens informatie over de service request en de gekoppelde web service of compositie van web services. Deze informatie kan zeer bruikbaar zijn bij toekomstige compositieprocessen. Op de huidige manier waarop web services en services requests automatisch worden gekoppeld blijft deze informatie niet beschikbaar, de relatie tussen de service requester en de web service is slechts tijdelijk. Bij het koppelen van een enkele web service met een service request is deze informatie waarschijnlijk nauwelijks relevant, maar bij web service compositie kan deze informatie erg waardevol zijn. Ervaringen uit het verleden gaan nu niet verloren maar kunnen gebruikt worden om het compositieproces te vergemakkelijken. Voordat er gekeken wordt naar een praktische suggestie om dit idee te verwezenlijken, wordt er nog één component toegevoegd aan het model. Persoonlijk ben ik van mening dat de informatie in een match record pas compleet is als er ook feedback van de gebruiker in verwerkt zit, zodat het niveau van een match niet enkel wordt bepaald aan de hand van input en output, maar ook de mate waarin de gevraagde capaciteiten zijn voorzien wordt meegenomen. Het model komt er nu als volgt uit te zien.

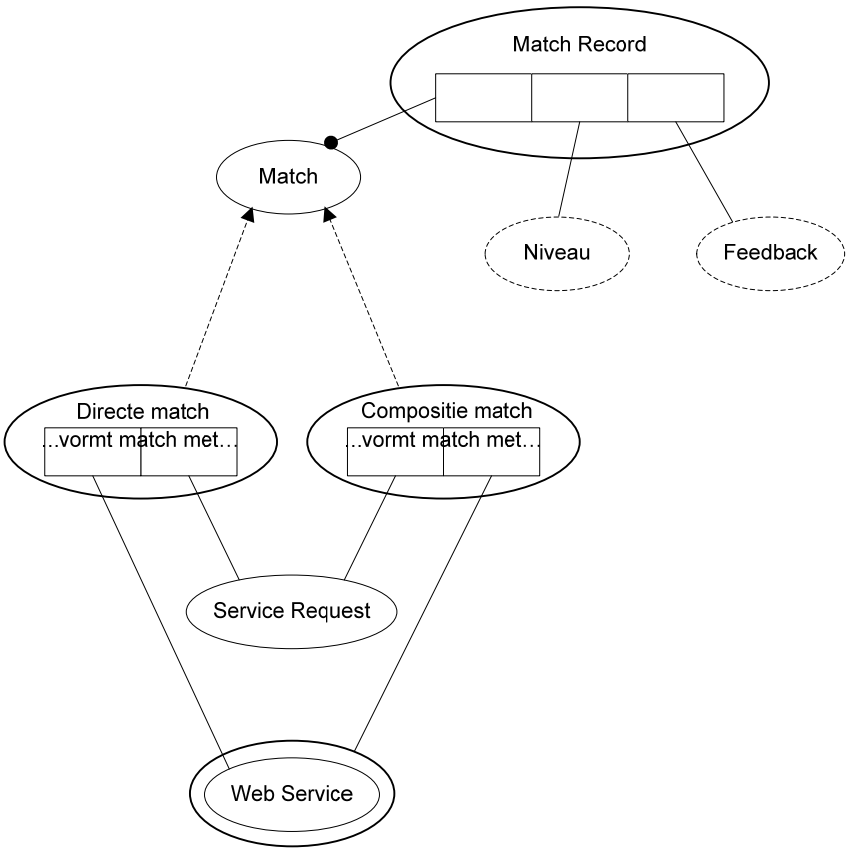

Fig.87 Objectificatie van een match record inclusief feedback

Hiermee is het model compleet en kan deze worden gebruikt voor een praktische suggestie voor het gebruik van een match record ter verbetering van het compositieproces van web services.

6.2 Verbetering van compositie: een feedback mechanisme

De ervaringen uit het verleden inclusief de feedback kunnen werken als een mechanisme waarmee bij de dynamische compositie van web services informatie kan worden geraadpleegd. Door informatie uit het verleden te gebruiken neemt de dynamiek van het compositieproces af, de dynamiek verdwijnt namelijk voor composities die eerder gemaakt zijn. Hierdoor kunnen we wellicht zelfs stellen dat een dergelijke methode niet de dynamische capaciteiten vergroot, maar de behoefte aan dynamiek laat afnemen. Herinner het probleem van een statische compositie beschrijving - bijvoorbeeld door middel van BPEL4WS - in het geval van ontbrekende web services. Om een alternatieve oplossing te vinden is er vaak domeinkennis vereist. Aangezien we redeneren binnen de kaders van een peer-to-peer architectuur waarbij de registers domein specifiek zijn opgericht, is het mogelijk een stuk van deze complexiteit binnen de registers te leggen en vervalt de noodzaak deze generiek op te lossen. Terug naar ons weerbericht voorbeeld. Op het moment dat de web service die de lokale temperatuur door geeft weg valt kan het zijn dat er een vergelijkbare service bestaat met een andere naamgeving of dat de temperatuur kan worden afgeleid uit andere data, bijvoorbeeld door hoge en lage drukgebieden. De kans is zeker aanwezig dat deze domeinkennis beschikbaar is binnen het register, bijvoorbeeld doordat deze compositie eerder is gebruikt of handmatig is gevormd door iemand met domeinkennis. Doordat eerdere composities bewaard blijven, zijn deze makkelijk herbruikbaar.

Een blijvend probleem bij de compositie van web services is de bepaling wanneer is een compositie in staat om in alle gevraagde capaciteit te voorzien. Uit figuur 8 blijkt hoe de feedback van een service requester verwerkt kan worden. De vraag op dit punt is hoe de feedback eruit ziet en wat komt er in komt te staan. Eerder is gesproken over twee smaken matching algoritmes, de eerste die een match zoekt op basis van input en output, de tweede die een match zoekt op basis van de gevraagde capaciteiten. Als we deze tweedeling doorzetten naar ons model dan zien we dat zien we dat de eerste smaak terug komt in de entiteit niveau, herinner de mate van matching in hoofdstuk 4. Met andere woorden in een match record wordt een match gekoppeld aan dit niveau dat een mate van matching aangeeft op basis van input en output. Gezien de theorie en het model is het aannemelijk te suggereren dat de feedback component die een match record de objectificatie van een tertiaire relatie maakt informatie kan bevatten over de gevraagde capaciteiten. Alleen de service requester weet na het afspelen van het hele proces pas of dat het resultaat naar tevredenheid is en in hoeverre is voorzien in de gevraagde capaciteit. Op deze manier is er niet alleen informatie beschikbaar over de web service en in hoeverre deze een match maakt met een service request, maar kan worden aangegeven in hoeverre de gevraagde capaciteiten van de betreffende service request daadwerkelijk zijn voorzien.

In een situatie zoals zojuist beschreven is het mogelijk ten tijde van uitvoering gebruik te maken van ervaringen uit het verleden. In het geval van een ontbrekende functionaliteit, kan eerst worden gekeken of dat de oplossing zich reeds gevormd heeft, in dit register of wellicht in een ander. Aan de hand van de feedback van de service requester kan worden bepaald of de eerdere compositie ook daadwerkelijk resultaat heeft opgeleverd. Zoals vermeld blijft het dynamische karakter van de compositie van web services een probleem. Deze suggestie kan in theorie gezien worden als toevoeging omdat het de behoefte aan dynamiek terug dringt.

7. Conclusie

Na vele woorden zijn we aangekomen bij de conclusie van dit verslag. Deze conclusie zal worden getrokken door het instrumentarium aan theorie en ideeën terug te koppelen aan de probleemstelling. Deze luidt *"Hoe kunnen web services op semantische basis worden beschreven en gematcht, en hoe kunnen deze inzichten worden gebruikt om de compositie van web services te verbeteren?"*

Na een korte inleiding in hoofdstuk 1 en een verduidelijking van begrippen in hoofdstuk 2, is er in hoofdstuk 3 gekeken naar de semantiek van web services. Aan het eind van hoofdstuk 3 is concluderend gesteld dat web services hun semantische waarde krijgen door beschrijvingen op semantisch niveau, het scheiden van de functionaliteit en de domein onafhankelijke intentie en is laten zien dat de compositie van web services sterk bijdraagt aan het voorzien in functionaliteit. In hoofdstuk 4 is laten zien hoe deze semantische web services worden geplaatst in een register, tevens is laten zien dat de rol van registers uitgroeit in die van een software agent. De OWL-S/UDDI matchmaker kan gezien worden als een dergelijke software agent die in staat is om te gaan met de semantische matching van een service request met een web service. Tevens geeft deze matchmaker ondersteuning aan compositieprocessen, echter gaat de matchmaker gebukt onder een potentieel schaalbaarheids probleem en het dynamische karakter van web service compositie. In paragraaf 4.4 wordt een manier beschreven hoe het schaalbaarheids probleem kan worden opgelost middels een peer-to-peer architectuur waarin verschillende registers als peers met elkaar samenwerken. Hiermee is invulling gegeven aan het eerste gedeelte van de probleemstellen, namelijk hoe web services op semantische basis kunnen worden beschreven en gematcht.

Het tweede deel van de problematiek wordt besproken in hoofdstuk 5 en 6 met de twee laatste deelvragen als leidraad. In hoofdstuk 5 zijn met behulp van ORM een aantal modellen ontworpen op basis van de inzichten die zijn verworven in hoofdstuk 3 en 4. Door middel van deze modellen zijn er twee zwakheden aan het licht gesteld. Het eerste zwakke punt is het feit dat een match niet expliciet genoeg wordt benaderd, maar gezien wordt als het resultaat van een dynamisch proces. Het tweede zwakke punt is dat er geen standhoudende relatie is tussen de compositie van web services en de registers, ook hier is het karakter van de relatie slechts tijdelijk. Aan het eind van hoofdstuk 5 is een nieuw model geconstrueerd. Uit dit model blijkt het onderscheid tussen een directe en een samengestelde match die samen leiden tot een expliciete vorm van een match die informatie bevat over zowel de service request als de daaraan gekoppelde web service.

In hoofdstuk 6 is gesuggereerd deze informatie te koppelen aan twee verschillende matching niveaus. De eerste op basis van de gelijkenis tussen de input en output van de web service en de service request, de tweede op basis van feedback van eerdere service requesters. De service requesters geven in deze situatie aan in hoeverre de web service daadwerkelijk heeft voldaan aan de gevraagde informatie. Als de twee stukken informatie worden gekoppeld aan de match zelf ontstaat het match record. Deze bevat alle informatie over geslaagde en minder geslaagde composities. Naarmate er meer composities worden gevormd groeit deze kennisbank en wordt deze steeds waardevoller om informatie uit te halen. Indien de matchmaker tijdens het compositieproces stuit op het gebrek aan functionaliteit kan de agent deze kennisbank raadplegen om te zien of dit probleem in het verleden reeds is opgelost. De opdeling van registers in een domeinspecifieke structuur voorziet in deze gevallen vaak in de domeinkennis die nodig is te komen tot het gewenste resultaat.

Met de modellen en de suggestie, voortkomend uit de modellen en de theoretische concepten, is een mogelijk antwoord gegeven op het tweede stuk van de problematiek. Belangrijk hierbij is de voetnoot dat de uitbreiding van de modellen in hoofdstuk 5 en de suggestie beschreven in hoofdstuk 6 **hypothetisch** van aard zijn. De ideeën komen voort uit redenatie op basis van de literatuur, maar zijn niet aan een toetsing onderworpen. Voordat deze ideeën in werking kunnen worden gesteld dient er eerste onderzoek gedaan te worden naar de validatie van deze ideeën. Hieruit zal blijken of deze methode werkelijk een verbetering vormt en de resultaten positief beïnvloedt. Kortom, in dit verslag zijn semantische web services inzichtelijk gemaakt en is op basis van deze inzichten een suggestie gedaan om het dynamische karakter van compositieprocessen terug te dringen ter bevordering van web service compositie, maar is er nog geen sprake van de validatie van deze ideeën.

Verder onderzoek kan tevens worden gedaan naar de beveiliging van compositieprocessen en de waarborging van kwaliteit. Hoe meer er sprake is van automatisering hoe minder de processen aan menselijk beoordelingsvermogen zijn onderheven. Met name in een omgeving die gekenmerkt wordt door business to business toepassingen is de waarborging van de kwaliteit van de resultaten erg belangrijk, op basis van deze informatie worden immers beslissingen genomen. Een lage kwaliteit van resultaten leidt tot weinig of geen gebruik van een geautomatiseerde vorm van web service discovery, men zal zelf de kwaliteit waarborgen door een handmatige selectie toe te passen. Het gebrek aan menselijk toezicht op de web service communicatie kan ook beveiligingsproblemen opleveren, tenslotte is niet altijd bekend wat web services doen en of web services doen wat ze zeggen te doen.

Een goede omgang met deze problematiek is vereist in een geautomatiseerde omgeving en zolang dit nog niet het geval is raad ik ieder aan voor een betrouwbaar weerbericht gewoon de website van het KNMI te raadplegen.

8. Referenties

8.1 Literatuur

[1] Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar., Miller J., *"METEOR-SWSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services"*, Information Technology and Management, vol. 6 pp. 17-39, Springer Science + Business Media, 2005

[3] Srinivasan, N., Paolucci, M., Sycara, K., *"An Efficient Algorithm for OWL-S Based Semantic Search in UDDI"*, Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition at the 2nd International Conference on Web Services, pp. 96-110, Springer-Verlag Berlin Heidelberg, 2005

[6] Gruber, T.R. *"A Translation Approach to Portable Ontology Specification"*, Knowledge Acquisition, vol. 5 pp. 199-220, 1993

[8] Gibbins, N., Harris, S., Shadbolt, N. *"Agent-based Semantic Web Services"*, Proceedings of the 12th International World Wide Web Conference (WWW) pp. 141–154, Budapest, 2003

[10] Martin, D., Burstein, M., McDemott, D., McIlraith, S., Paolucci, M., Sycara, K., McGuinness, D.L., Sirin, E., Srinivasan, N., *"Bringing Semantics to Web Services with OWL-S"*, World Wide Web, vol. 10 pp. 243-277, Springer Science + Business Media, 2007

[12] DAML-S Coalition: Ankolekar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D., McDermott, D., McIlraith, S. A., Narayanan, S., Paolucci, M., Payne, T., Sycara, K. *"DAML-S: Web Service Description for the Semantic Web"* Springer Berlin / Heidelberg, Volume 2342, pp. 348-363, 2002

[13] Labrou, Y., Finin, T., *"Semantics and conversations for an agent communication language"*, Proceedings of the Fifteenth Internation Joint Conference on Artificial Intelligence, pp. 584 – 591, Springer-Verlag Berlin Heidelberg, 1997

[14] Sycara, K., Paolucci, M., Ankolekar, A., Srinivasa, N., *"Automated discovery, interaction and composition of Semantic Web services"*, Web Semantic: Science, Services and Agents on the World Wide Web, vol. 1 pp. 27-46, Elsevier Journal of Web Semantics, 2003

[15] Lee, M., Yoon, H., Shin, H., Lee, D.G., *"Intelligent dynamic workflow support for a ubiquitous Web service-based manufacturing environment"*, Journal of Intelligent Manufacturing, vol 20, pp. 295-302, Springer Netherlands, 2009

[16] Luo, J., Montrose, B., Kim, A., Khashnobish, A., Kang, M., *"Adding OWL-S Support to the Existing UDDI Infrastructure"*, IEEE International Conference on Web Services, 2006

[17] Alonso, G., Casati, F., Kuno, H., Machiraju, V. *"Web Services, Concepts, Architecture and Applications*", Springer-Verlag, New York 2004

[18] Menasce, D., Almeida, V. "*Capacity Planning for Web Services*", Prentice Hall, New Jersey 2001

[19] Sycara, K., Fries, B., Klusch, M., *"Automated Semantic Web Service Discovery with OWLS-MX"*, Proceedings of the Fifth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 915-922, ACM Press, 2006

8.2 Overig

[2] The Evolution of UDDI, UDDI.org White paper http://www.uddi.org/pubs/the_evolution_of_uddi_20020719.pdf (2002)

[4] UDDI Consortium, *"UDDI Executive White Paper"*, November 2001

[5] Universal Description, Discovery and Integration, *"UDDI Technical White Paper"*, September 2000.

[7] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., *"Extensible markup language (XML) 1.0"*, W3C Eecommendation, http://www.w3.org/TR/2000/REC-xml-20001006, 2000

[9] Lassila, O., Swick, R.R., *"Resource Description Framework Model and Syntax Specification"*, W3C
Recommendation. W3C, 1999.

[11] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K., *"Owl-s: Semantic Markup for Web Services"* W3C Recommendation, http://www.daml.org/services/owl-s/1.1/overview/, 2004

[20] Brickley, D., Guha R.V., *"RDF Vocabulary Description Language 1.0: RDF Schema"*, W3C Recommendation, http://www.w3.org/TR/rdf-schema/, 2004.

[21] McGuinness D.L., Harmelen, van F., "*OWL Web Ontology Language Overview*", W3C Recommendation, http://www.w3.org/TR/OWL-features/, 2004

[22] Halpin, T., *"Object Role Modeling: an overview"*, White paper, http://www.orm.net, 1998