

VIA WEB-SERVICES EN SEMANTIEK NAAR FLEXIBELE INTEGRATIE

Frank Goethals, Antoon Goderis, Jacques Vandenbulcke

SAP-leerstoel 'Extended Enterprise Infrastructures', F.E.T.E.W. - K.U.Leuven
Naamsestraat 69, 3000 Leuven, Belgium
email: {frank.goethals, antoon.goderis,
jacques.vandenbulcke}@econ.kuleuven.ac.be

Abstract: Veel bedrijven zien economische waarde in een integratie van computersystemen, zowel onder de vorm van Enterprise Application Integration als onder de vorm van Business-to-Business Integration (B2Bi). De XML web-services technologie belooft veel goeds voor de toekomst van applicatie-integratie. Deze technologie staat echter nog maar in de kinderschoenen. Er zal bijgevolg nog veel onderzoek en overleg nodig zijn om het volledig potentieel van de web-services technologie te realiseren. Momenteel zijn er veel web-service-gerelateerde standaarden in ontwikkeling. In dit artikel pogen we ondermeer deze standaarden te plaatsen in het probleemdomen. XML web-services alleen volstaan niet om een vlotte integratie van systemen te realiseren. Eén van de belangrijkste moeilijkheden bij integratie, en zeker bij B2Bi, betreft immers semantiek. In dit kader is het interessant een blik te werpen op de visie van het *semantisch web* en de gerelateerde standaarden.

Inhoudstafel

Inhoudstafel.....	2
Inleiding.....	3
1 Business opportuniteiten.....	3
2 Web-services	8
2.1 <i>Wat zijn web-services?</i>	9
2.2 <i>Uitdagingen voor web-services technologie</i>	9
2.3 <i>Bestaande standaarden</i>	14
2.3.1 De basisstandaarden SOAP, WSDL en UDDI.....	14
2.3.2 Andere web-service standaarden	17
2.4 <i>Het ontwikkelen van web-services</i>	20
3 Het semantisch web	23
3.1 <i>Wat is het semantisch web?</i>	23
3.2 <i>RDF als basisstandaard voor het creëren van gestructureerde verzamelingen van informatie</i>	26
3.3 <i>Andere standaarden</i>	28
3.3.1 Een stap verder met DAML+OIL.....	28
3.3.2 Het krachtiger beschrijven van web-services met DAML-S.....	29
3.4 <i>Eindbemerkingen</i>	31
4 De basis voor verder onderzoek	32
5 Conclusie	33
Dankbetuiging.....	34
Referenties.....	34
Appendix A: Belangrijke Organen	37
Appendix B: Fragment van een BPML-document.....	39
Appendix C: Fragment van een WSFL-document.....	39
Appendix D: Een SOAP-bericht	40
Appendix E: Een WSDL-document.....	41
Appendix F: Het gebruik van RDF en RDFS	42
Appendix G: Het gebruik van DAML	42

Inleiding

Terwijl enkele jaren geleden iedereen jubelde bij woorden als ‘fusie’ en ‘overname’ ligt het concept ‘partnering’ nu beter in de markt. Een belangrijk verschil tussen beide, fusie en overname enerzijds, en partnering anderzijds, ligt in de sterkte van de koppeling tussen beide bedrijven. De lichtere koppeling die we terugvinden bij partnering moet zich ook uiten in de ICT die gebruikt wordt om de computersystemen van partners met elkaar te verbinden.

SAP, vooral gekend als toonaangevend bedrijf op het vlak van Enterprise Resource Planning (ERP)-toepassingen, heeft zich de laatste jaren opgewerkt tot een belangrijke leverancier van Supply Chain Management (SCM)-software. SAP onderkent – net als de rest van de SCM-markt – het belang van een lichte koppeling van ICT-systemen. In het kader van de SAP-leerstoel ‘Extended Enterprise Infrastructures’ gaan we dan ook op zoek naar een manier om een lichte koppeling tussen systemen te bewerkstelligen om zodoende adaptieve systemen te kunnen realiseren.

In dit artikel gaan we in op een aantal ontwikkelingen en trends die relevant zijn als basis voor het toekomstig onderzoek. De besproken concepten beslaan twee domeinen: SCM en ICT. In Sectie 1 bespreken we de recente evoluties in SCM. Hierbij bouwen we een business-case op ter verantwoording van het pompen van geld en tijd in het integreren van systemen van verschillende bedrijven (de zogenaamde Business-to-Business integration, B2Bi). Om systemen te integreren steunen we op de mogelijkheden die ICT ons biedt. In Secties 2 en 3 onderzoeken we respectievelijk web-services en het semantisch web. De bedoeling is de betekenis van beide concepten te verduidelijken, de huidige stand van zaken weer te geven op beide vlakken en de link met B2Bi duidelijk te maken. Wanneer we al het voorgaande samenbrengen, zien we de mogelijkheid om tot een krachtig, wereldwijd semantisch web van services te komen dat toelaat systemen (en bedrijven) via een lichte koppeling te verbinden. De realisatie van deze mogelijkheid vereist echter nog enorm veel onderzoek. Een aantal onderzoeksvragen komen aan bod in Sectie 4. Terwijl dit artikel conceptueel opgevat is als literatuuroverzicht, laat vooral deze laatste sectie ruimte voor het inbrengen van eigen ideeën.

1 Business opportunities

Sedert decennia zoeken bedrijven naar manieren om hun processen te optimaliseren. Eén stap hierin was de introductie van Enterprise Resource Planning (ERP)-software, welke toelaat de resources waarover een bedrijf beschikt efficiënter in te zetten [1]. Daarnaast probeerden bedrijven ook flexibeler te worden, o.a. door de organisatie platter te maken, business-process reengineering door te voeren, multifunctionele projectteams op te richten, enz. [2].

Gaandeweg is het bewustzijn gegroeid dat er ook veel voordelen liggen in de optimalisatie van processen over de grenzen van het eigen bedrijf heen. Supply Chain Management (SCM) is een discipline die zich hierop concentreert. Ook de term ‘ERP II’ wordt regelmatig in deze context gebruikt. Met ERP II bedoelt men de uitbreiding van ERP in die zin dat ERP II doet op het niveau van een hele Supply Chain, wat ERP doet op het niveau van een individuele onderneming [3]. SCM is niet enkel nuttig om tot efficiëntere, maar ook om tot effectievere processen te komen. Dit laatste wordt ondermeer duidelijk wanneer we kijken naar Customer Relationship Management (CRM). De klant is de bron van inkomsten voor elk bedrijf (en voor elke Supply Chain), en een goede relatie met de klant is dan ook van groot belang. Het spreekt voor zich dat bedrijven (en Supply Chains) grote voordelen kunnen behalen indien ze (tijdig) producten en diensten kunnen leveren die gepersonaliseerd zijn op basis van de specifieke eisen van de klant. Veel kennis omtrent de wensen van de eindklant zit echter verspreid in de Supply Chain. Er wordt vaak gesteld dat Supply Chains die deze informatie kunnen uitbaten betere resultaten zullen boeken dan andere Supply Chains [zie bv. 4]. Hayes komt zo tot het concept van de e-CRM (electronic Customer Relationship Management) extended enterprise: *“the extension of a company’s business model through process automation to customers and channel partners”* [5].

In het vervolg van deze sectie concentreren we ons volledig op SCM: eerst verduidelijken we het concept SCM en vervolgens werpen we een blik op het veranderde gezicht dat SCM krijgt door nieuwe technologieën.

D. Simchi-Levi, P. Kaminsky and E. Simchi-Levi [6] definiëren SCM als

“a set of approaches utilized to efficiently integrate suppliers, manufacturers, warehouses, and stores, so that merchandise is produced and distributed at the right quantities, to the right locations, and at the right time, in order to minimize systemwide costs while satisfying service level requirements”.

De integratie waarvan sprake in de definitie kan slechts goed verlopen mits een goede integratie van de computersystemen van de partijen. Gezien de moeilijkheden die bedrijven momenteel ondervinden bij de integratie van hun eigen systemen, hoeft het geen betoog dat B2Bi een decennium geleden evenmin voor de hand liggend was. De B2Bi beperkte zich vroeger dan ook tot de automatisering van informatie-uitwisselingen. Zo kon het aankoopproces bijvoorbeeld geautomatiseerd worden door het gebruik van Electronic Data Interchange (EDI). Alhoewel EDI een aantal voordelen had – RJR Nabisco schat bijvoorbeeld dat het verwerken van een papieren aankooporder \$70 kost, tegenover 93 cent bij EDI [7] – is het zeker niet de perfecte techniek voor integratie. Zo is EDI gebaseerd op batch-verwerking, is het niet flexibel en is het duur om te implementeren [8]. Een neveneffect hiervan is dat het enkel voor grote

bedrijven economisch verantwoord was EDI te gebruiken. Web-services technologie belooft de belangrijkste EDI-nadelen aan te pakken, en nieuwe mogelijkheden te openen.

Zoals gezegd willen bedrijven meer en beter samenwerken. Deze samenwerking uit zich in concepten als e-procurement, Vendor Managed Inventory (VMI), Supply Chain Event Management (SCEM), Collaborative Planning, Forecasting and Replenishment (CPFR) e.d.¹ Opmerkelijk aan deze concepten is dat bedrijven tegenwoordig hechte partnerships vormen die streven naar een goede uitkomst voor *beide* partijen. Dit staat in schril contrast met de vroegere manier van zakendoen, waarbij de tegenpartij als een soort tegenstander beschouwd werd onder het motto dat ‘de ene wint wat de ander verliest’ [10]. Bedrijven zijn er zich bewust van geworden dat ze door samen te werken grotere winsten kunnen creëren voor de groep van samenwerkende bedrijven, dit dankzij een verlaging van de voorraden, een automatisering van processen, een betere kennis van de wensen van de klant, etc. De extra winst moet dan natuurlijk wel fair verdeeld worden over de deelnemers (wat op zich een interessante problematiek vormt). Merk op dat de bedrijven nog steeds afzonderlijke entiteiten blijven die nog steeds hun eigen winsten trachten te maximaliseren. Desalniettemin leeft er in een collaboratie toch de visie dat de samenwerking voor een lange tijd in leven geroepen is en dat bedrijven *echt* willen samenwerken om tot een betere service voor de klant te komen, en dus niet louter samenwerken om hun eigen winst te maximaliseren (en te profiteren) op korte termijn. Vaak wordt dan ook geopperd dat het vormen van hechte partnerships ertoe leidt dat de concurrentie zich verplaatst van op het niveau van individuele bedrijven naar het niveau van Supply Chains (oftewel ‘extended enterprises’) [11].

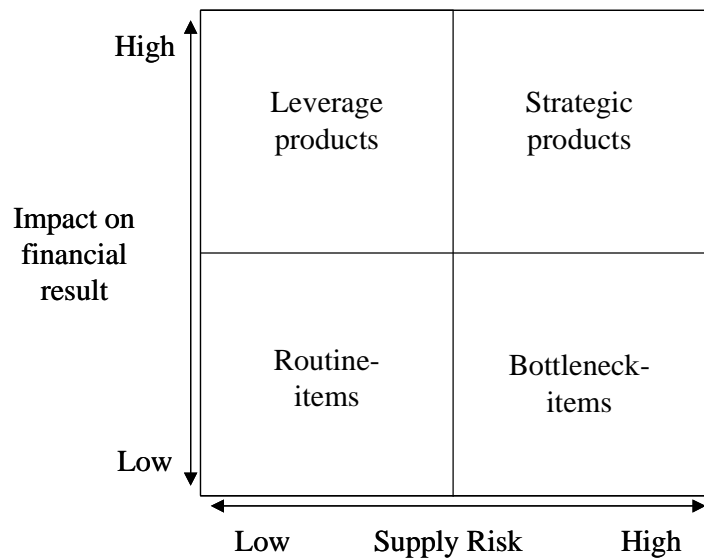
De verwachting is dat ICT-systemen van verschillende bedrijven in de toekomst flexibel integreerbaar zullen zijn, dit o.a. door nieuwe ontwikkelingen op het vlak van integratietechnologieën. Deze gemakkelijke integratie zal zich - ons inziens - niet alleen laten merken op het vlak van *hechte* partnerships met *leveranciers* en *klanten*, maar ook op twee andere vlakken: we verwachten enerzijds een toename in outsourcing en anderzijds een stijging van het aantal partijen waarmee contacten geautomatiseerd worden. In wat volgt verduidelijken we onze visie.

Onze verwachting betreffende het toenemende belang van outsourcing is gebaseerd op de wet van Coase [12]: “*A firm will tend to expand until the costs of organizing an extra transaction within the firm become equal to the costs of carrying out the same transaction on the open market.*” Het belangrijke punt is dat een flexibele integratietechnologie de transactiekosten laat dalen. Zodoende laat

¹ We gaan hier niet dieper in op deze specifieke concepten. Een meer volledig overzicht van SCM-evoluties kan evenwel gevonden worden op de website van de SAP-leerstoel [77].

een betere integratietechnologie bedrijven toe zich meer te concentreren op hun kerncompetenties. Dit uit zich bijvoorbeeld in het feit dat nogal wat bedrijven (zoals Nike en Coca-Cola) die vroeger bekend stonden als productieonderneming tegenwoordig al wat productie betreft uitbesteden en zich nog louter bezighouden met merchandising. Gezien het feit dat een focus op kerncompetenties ervoor zorgt dat een bedrijf deze competentie (bv. boekhouding) kan leveren tegen lagere prijs dan een bedrijf waarvoor deze competentie geen sterk punt is (bv. boekhouding voor een productieonderneming), is het aangeraden zich enkel met de kerncompetenties bezig te houden, en andere competenties in te kopen voor zover de transactiekosten dit toelaten.

Ten tweede merken we dat dure integratietechnologieën (zoals EDI) een beperking plaatsen op het aantal, en vooral de soort, partijen waarmee integreren verantwoord is. Het integreren (met dure technologieën) was enkel *financieel verantwoord* en *praktisch mogelijk* wanneer het ging om contacten met *hechte* partners waarbij de automatisering van de contacten *grote financiële voordelen* opleverde. Met goedkopere, flexibelere integratietechnologieën kan de integratie met andere partijen evenwel ook verantwoord worden. Om deze visie te verduidelijken vertrekken we van het referentiekader dat Dignum [13] ons aanreikt². Dit wordt afgebeeld in Figuur 1.



Figuur 1. Classificatie van elektronische aankopen [13]

² Merk op dat Dignum dit kader hanteert om het gebruik van software-agents te situeren, eerder dan om integratiemogelijkheden *an sich* te bespreken.

Dit referentiekader maakt duidelijk dat bij het aankopen van producten een onderscheid gemaakt kan worden tussen aankopen op basis van twee (voor ons) relevante kenmerken. Ten eerste is het zo dat het financieel niet verantwoord is grote integratiekosten te maken wanneer de automatisering slechts een kleine impact heeft op het financiële resultaat (kenmerk weergegeven op de verticale as van Figuur 1). Een goedkopere integratietechnologie (dan EDI bijvoorbeeld) laat bedrijven toe ook financieel-minder-belangrijke contacten te automatiseren. Bijgevolg is automatisering niet langer iets dat enkel grote, financieel sterke bedrijven aanbelangt. Ook KMO's kunnen aan integratie denken.

Een tweede kenmerk dat een invloed heeft op integratie-mogelijkheden betreft het risico-karakter van de aankoop. Dit laatste bepaalt of een aankopend bedrijf gemakkelijk kan overstappen op een andere toeleverancier zonder grote risico's te nemen betreffende product-specificaties, product-kwaliteit, leveringstermijn e.d. Het tweede criterium laat ons m.a.w. toe een onderscheid te maken tussen toeleveranciers die (praktisch) gemakkelijk vervangbaar (moeten) zijn door andere leveranciers enerzijds, en leveranciers die als een vaststaande bron aanzien mogen worden anderzijds. Strategische goederen en bottleneck-items hebben een hoog supply risico. Voor deze producten kan het overstappen op een andere leverancier onmogelijk zijn (omdat er bv. geen andere producenten van deze goederen bestaan) of enorm veel risico's met zich meebrengen zodat overstappen onverantwoord is (en flexibiliteit overbodig?). Voor leverage-producten en routine-items is het wel verantwoord over te stappen op een andere leverancier. Zoals gezegd lieten vroegere integratie-technologieën (zoals EDI) dit vlotte switchen tussen leveranciers echter niet toe. Merk op dat deze argumentatie niet noodzakelijk indruist tegen de idee van partnerships. Er kunnen immers ook partnerships gevormd worden met leveranciers die goederen leveren met een lage supply-risk. We kunnen voorgaande discussie dan ook interpreteren in die zin dat, wanneer er zich een probleem voordoet bij een partner-leverancier (bijvoorbeeld een probleem met de lopende band, een brandje,...), er dan snel een geautomatiseerde communicatie opgesteld kan worden met een andere partij die (voor korte termijn) de rol van de partner overneemt. Dit was met minder flexibele integratietechnologieën (zoals EDI) ondenkbaar.

Terwijl voorgaande bespreking zich concentreerde op de goederenmarkt, kan de lijn ons inziens doorgetrokken worden naar de inkoop van diensten, zoals goederentransport bijvoorbeeld.

We concluderen dat een technologie die gemakkelijke integratie van computersystemen mogelijk maakt, veel positieve effecten met zich meebrengt, zowel voor bedrijven (en volledige Supply Chains) als voor de eindconsument.

2 Web-services

Sedert een paar jaar steekt de term ‘web-services’ meer en meer de kop op. Deze technologie biedt sterke mogelijkheden op het vlak van applicatie-integratie. Het idee hierachter is dat ICT-systemen van het ene bedrijf via het Internet diensten kunnen aanbieden aan systemen (of mensen) van een ander bedrijf. Deze diensten kunnen dan heel gemakkelijk geïntegreerd worden in bestaande systemen en daarenboven kunnen bestaande systemen gemakkelijk aangepast worden zodat hun functionaliteit aangeboden kan worden via web services. De verhoopde toekomst van deze technologie is de verplaatsing van de controle over processen van de ICT-afdeling naar de eigenlijke zakenlui van het bedrijf. Business-mensen zouden zodoende in staat zijn processen flexibel aan te passen en nieuwe services en partners in het proces op te nemen.

Web-services technologie vormt de tweede stap (na EDI) in de richting van B2Bi. Met B2Bi bedoelen we de integratie van computersystemen die tot verschillende ondernemingen behoren. Web-services kunnen echter ook gebruikt worden voor Enterprise Application Integration (EAI), de integratie van systemen die alle tot eenzelfde bedrijf behoren. Veel bedrijven lijken web-services trouwens *in eerste instantie* te gebruiken voor EAI [14]. Dit lijkt goed overeen te komen met de visie dat bedrijven eerst intern orde op zaken moeten zetten alvorens processen en data naar buiten toe te openen [15, 16]. Linthicum [17] stelt trouwens dat B2Bi voorafgegaan moet worden door EAI. Samtani en Sadhwani [18] delen zijn visie: “[...] *companies should first start using Web Services for their internal integration projects [...], before they venture using Web Services in B2B integration projects.*” Er bestaan evenwel ook tegenstanders van deze manier van handelen. De Patricia Seybold Group bijvoorbeeld raadt aan ‘*to start from the outside in*’ met web-services technologie en deze technologie dus in eerste instantie te gebruiken om de bedrijfssystemen te openen naar buiten toe [19]. Hun argument is dat het eerst intern implementeren van web-services ervoor zorgt dat dergelijke bedrijven een te grote achterstand hebben op andere bedrijven die wel onmiddellijk web-services gebruiken om een aantal (goeddraaiende) kernsystemen naar buiten toe te openen.

In dit artikel zullen we web-services vooral belichten vanuit het standpunt van B2Bi. In wat volgt maken we eerst (in 2.1) duidelijk wat we onder web-services verstaan. Huidige web-services worden vooral gebruikt voor het opvragen van informatie (bijvoorbeeld het opvragen van een aandelenkoers), zonder effectief bedrijfs-transacties uit te voeren. Mits verdergezette ontwikkeling van de web-services technologie, zullen web-services echter veel krachtigere mogelijkheden bieden voor B2Bi. In 2.2 bespreken we kort de uitdagingen die de web-services technologie met zich meebrengt. Vervolgens (in 2.3) geven we een overzicht van de bestaande standaarden op het vlak van web-services technologie. Uit het overzicht zal blijken dat de web-services technologie zich nog volop aan het

ontplooiën is, en dat web-services momenteel zeker nog niet op volle kracht benut kunnen worden. In 2.4 werpen we een blik op de ontwikkeling van web-service-georiënteerde systemen.

2.1 Wat zijn web-services?

Echte eenduidigheid over wat een ‘web service’ juist is, bestaat niet. Er circuleren vele definities van de term die niet altijd precies hetzelfde weergeven. Wij beperken ons tot twee belangrijke definities. Een eerste definitie vinden we (ondermeer) terug bij IBM:

“Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web.”

In de voorgaande definitie komt het belang van XML - typisch aanzien als één van de basisfactoren bij de realisatie van B2Bi - niet tot uiting. Dit is heel anders bij de definitie van het W3C (World Wide Web Consortium) [20]:

“A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols”

Eén van de grote sterktes van de web-services technologie is de realisatie van fysieke interoperabiliteit: dankzij web-services en SOAP (wat vroeger stond voor Simple Object Access Protocol, maar tegenwoordig als een woord *an sich* bekeken wordt) over HTTP worden communicatieproblemen tussen JavaBeans, CORBA- en DCOM-componenten opgelost [21]. Web-services kunnen dan ook opgeroepen worden onafhankelijk van de taal waarin ze geschreven zijn en van het platform waarop ze draaien [22]. Om tot een goeddraaiend gedistribueerd systeem te komen zijn er echter nog een aantal andere uitdagingen die aangepakt moeten worden.

2.2 Uitdagingen voor web-services technologie

Bij het doorbreken van een nieuwe technologie is het altijd zoeken naar de mogelijkheden van die nieuwe technologie en naar de manier waarop die mogelijkheden effectief gerealiseerd kunnen worden. Verschillende organisaties zijn daarom op zoek naar een framework (een ‘kapstok’) om de mogelijkheden en de bijhorende uitdagingen te identificeren. Zo is de Web Services Architecture Working Group van het W3C een specificatie met ‘web services architecture requirements’ aan het voorbereiden (momenteel een Working Draft). Deze is nog volop in ontwikkeling. Meer uitgewerkte ideeën vinden we terug in het WSMF (Web Service Modeling Framework [23]) en bij Glass [24]. In wat volgt geven we een overzicht van de uitdagingen, gebaseerd op hun ideeën en aangevuld met

eigen inzichten. Het doel van het overzicht is tweeledig. Enerzijds willen we effectief een beeld scheppen van de (toekomstige) mogelijkheden van de web-services technologie. Anderzijds pogen we ook een kader te bieden waarbinnen gekende standaarden een plaats kunnen krijgen. Er zijn immers dermate veel standaarden (m.b.t. web-services) dat het niet altijd onmiddellijk duidelijk is wat hun plaats is in het geheel. In het overzicht vermelden we dan ook kort een aantal relevante standaarden. Deze zullen in 2.3 dieper besproken worden. Merk op dat er in het web-services domein heel wat spelers actief zijn en dat standaarden, gespecificeerd door deze spelers, heel snel evolueren. Daarom verdiepen we ons in Appendix A in de belangrijke organen ter zake.

Glass [24] somt een aantal uitdagingen op die aangepakt zouden moeten worden.

1. Om web-services van een andere partij te kunnen gebruiken is het vooreerst nodig te weten welke web-services er bestaan en hoe die services aangeroepen kunnen worden.

We kunnen een onderscheid maken tussen services waarvan een bedrijf wel weet heeft (bijvoorbeeld doordat het aan een partner gevraagd heeft een dergelijke service te creëren), en ongekende services. Laatstgenoemde, ongekende web-services moeten ontdekt kunnen worden. Hiertoe is het natuurlijk nodig dat de service-aanbieder zijn web-services kan beschrijven en deze beschrijving (al dan niet globaal) toegankelijk kan maken. Twee basis-standaarden voor dergelijke web-service *description* en *discovery* zijn WSDL (Web-Service Description Language) en UDDI (Universal Description, Discovery, and Integration). WSDL laat toe een voor mensen (niet voor computers!) verstaanbare beschrijving te geven van de service, en te definiëren hoe de service effectief opgeroepen kan worden. UDDI kan gebruikt worden voor het zoeken naar web-services in een globale repository die voor iedereen toegankelijk is. Om een dergelijke DB (DataBase) te doorzoeken zijn er natuurlijk krachtige zoekmogelijkheden nodig. DAML-S (DARPA Agent Markup Language-Services) is een standaard die toelaat een semantisch sterkere beschrijving te geven van web-services dan mogelijk is met WSDL [25], wat zich kan vertalen in betere zoekmogelijkheden. In de realiteit is het natuurlijk goed mogelijk dat men enkel geïnteresseerd is in services van partners, en dus niet in services die door willekeurige partijen aangeboden worden in een globale UDDI-registry. WSIL (Web-Services Inspection Language) is een standaard die toelaat web-services te vinden waarvan de aanbieder een gekende partij is. Ook 'private UDDI' kan hier een rol spelen.

2. *Betrouwbaarheid* is een kernbegrip indien web-services effectief bruikbaar moeten zijn voor B2Bi.

Betrouwbaarheid kan hierbij op twee manieren bekeken worden. Ten eerste moet een partij die een web-service wenst te gebruiken, weten dat de partij die de service aanbiedt te vertrouwen is. Het moet bijgevolg niet alleen mogelijk zijn services te vinden, maar ook om de betrouwbaarheid van de aanbieders van services te achterhalen indien het onbekende aanbieders betreft. Ten tweede moet de uitvoering van de web-service zelf (en dus de onderliggende infrastructuur) betrouwbaar zijn. Een bedrijf dat een vrachtwagen wil reserveren via een web-service wil geen bericht krijgen dat alles in orde is, terwijl in de achtergrond de gegevens niet opgeslagen kunnen worden in de DB van de vervoersmaatschappij.

3. Gezien het feit dat de communicatie verloopt via het publieke Internet, tussen partijen die elkaar niet noodzakelijk kennen, is er een hoge graad van *beveiliging* nodig. Er is bijvoorbeeld authenticatie nodig om zeker te zijn dat de andere partij effectief is wie ze zegt te zijn. Ook encryptie van de boodschappen mag niet ontbreken.
4. Eén van de factoren die de toepasbaarheid van web-services bij het zakendoen bepaalt, is de geboden ondersteuning voor de realisatie van *zakelijke transacties*.

Onder een transactie verstaan we bijvoorbeeld de situatie waarbij een reiziger zowel een vliegtuig als een hotel wil boeken. De reiziger wil natuurlijk enkel een hotel boeken indien hij zeker is dat hij ook een vliegtuig kan boeken, en vice versa. Het realiseren van transacties is in het B2B-domein niet altijd eenvoudig. De klassiek gebruikte locking-protocollen zijn immers niet altijd bruikbaar, aangezien transacties niet altijd binnen afzienbare tijd afgerond zullen worden, maar meerdere dagen of zelfs weken in beslag kunnen nemen. We spreken dan van long-lived (ook wel long-running) transacties: *'business processes that run over an extended time period'* [26]. Zo kan het bijvoorbeeld 24 uur duren alvorens een hotelboeking bevestigd kan worden, waardoor de vliegtuigboeking ook nog uitgesteld moet worden.

Alhoewel transacties traditioneel gekarakteriseerd worden door de ACID (Atomicity, Consistency, Isolation en Durability)-eigenschappen, moeten deze eigenschappen wellicht wat gelost worden in een web-services wereld. Zo geldt de Isolation-eigenschap niet voor long-lived transacties (wel voor short-lived transacties) [27]. Een eerste standaard voor de realisatie van transacties was XAML (Transaction Authority Markup Language). Deze poging werd echter opgegeven in 2001. Een meer recente standaard (mei 2002) is het Business Transaction Protocol (BTP). In augustus 2002 brachten IBM, Microsoft en BEA samen de specificatie uit van 'WS-transaction'.

5. *Schaalbaarheid* van het systeem lijkt op twee vlakken van groot belang. Ten eerste moet het effectieve gebruik van de service schaalbaar zijn. Het is echter heel moeilijk te voorspellen hoe vaak een bepaalde dienst gebruikt zal worden en bovendien kan dit aantal op een heel korte tijd exploderen of imploderen. Ons inziens heeft de web-services technologie hierbij inherent het voordeel dat ze verder bouwt op CBD (Component Based Development, zie 2.4), zodat web-services door meerdere application servers aangeboden kunnen worden. Ten tweede is schaalbaarheid ook relevant wanneer we het hebben over het zoeken naar services (dus niet louter bij het gebruik van services). Het mechanisme om te zoeken naar web-services zal in de toekomst ook moeten toelaten generieke web-services te customiseren op maat van individuele gebruikers (zie infra), wat een invloed kan hebben op de schaalbaarheid van de oplossing.
6. Het volledige ICT-systeem moet *beheersbaar* blijven. Bijkomende problemen kunnen de kop opsteken door het gedistribueerde karakter van de verschillende systemen en doordat systemen (services) gebruikt worden die onder controle zijn van andere organisaties. Automatisch genegotieerde Service Level Agreements (SLAs) zullen hier hun nut kunnen bewijzen.
7. Een systeem dat gebruik maakt van één of meerdere web-services (aangeboden door andere partijen) is moeilijker te *testen en debuggen*. De performantie (bv. antwoordtijd) van een dergelijk systeem is moeilijk te voorspellen. SLAs zullen ons inziens ook hier hun rol spelen.

Deze zeven aandachtspunten vinden we min of meer ook terug in het WSMF. Het WSMF haalt daarenboven nog de volgende uitdaging aan:

8. Het *definiëren van processen*, opgebouwd uit componenten die met elkaar interageren en die eventueel (elk op zich) als web-services bekeken kunnen worden. Stel dat bedrijf A binnen 24 uur goederen nodig heeft die door bedrijf B geproduceerd worden. A moet dan niet alleen weten of B deze goederen in voorraad heeft, maar ook of een vervoersmaatschappij deze goederen tijdig zal kunnen transporteren van B naar A. Indien A een web-service gebruikt om de goederen te kopen, zou ineens een web-service opgeroepen moeten worden bij de transportonderneming waarbij het vervoer van die goederen geregeld wordt. De twee web-services, de aankoop van goederen en het boeken van transport, kunnen hier gecombineerd worden tot één krachtigere web-service. Web-services van verschillende partijen moeten dus serieel, parallel of eventueel op basis van een aantal condities met elkaar verbonden kunnen worden om zodoende een proces te implementeren. Het spreekt voor zich

dat web-services technologie meer nut heeft wanneer deze koppeling *dynamisch* kan verlopen.

Tegenwoordig worden er heel wat 'talen' ontwikkeld om de choreografie (het samenwerken) van web-services uit te drukken. Merk op dat een choreografie uitgevoerd kan worden door boodschappen uit te wisselen, waarbij web-services opgeroepen worden en/of feedback gegeven wordt. Dit wetende merken we dat er twee klassieke manieren zijn om een choreografie te beschrijven. Een eerste manier definieert het proces vanuit de gezichtshoek van een buitenstaander. Hierbij worden de volgende zaken gedefinieerd:

- de rollen van de deelnemers aan het proces (bijvoorbeeld een reiziger, een reisagentschap en een luchtvaartmaatschappij),
- de web-services die deze rollen aanbieden, zonder de concrete implementatie van de web-services te bepalen (bijvoorbeeld een reisboeking en een vliegtuigboeking),
- de boodschappen die uitgewisseld worden tussen partijen om web-services op te roepen en feedback te geven (bijvoorbeeld een reisaanvraag en een vluchtaanvraag).

Bij de tweede manier wordt het proces meerdere keren beschreven, namelijk vanuit het standpunt van elke deelnemer aan het proces (zo bijvoorbeeld vanuit het standpunt van de reiziger, die enkel weet dat hij een reisaanvraag moet versturen naar een reisagentschap en een antwoord van dit agentschap moet ontvangen). Zodoende wordt het duidelijk welke boodschappen een partij moet ontvangen en sturen en welke web-services een partij moet implementeren om een choreografie te realiseren.

Naast de acht reeds vermelde uitdagingen willen we ook de aandacht vestigen op twee andere zaken:

- 9 Benevens processen moeten ook *zakelijke documenten*, zoals facturen, bestelorders e.d gedefinieerd worden. In een web-services wereld zullen zakelijke documenten immers onder digitale vorm verspreid worden. Het is dan ook belangrijk een eenduidige manier te hebben om dergelijke documenten te beschrijven. Merk op dat het irrealistisch is te eisen dat iedereen dezelfde documenten gebruikt. Verschillende industrieën hebben daarenboven verschillende eisen op dit vlak. Het is daarom interessant te voorzien in een algemene taal en algemene constructies die gebruikt kunnen worden om zakelijke documenten op te stellen die door iedereen verstaanbaar zijn. Belangrijke initiatieven op dit vlak zijn xCBL (XML Common Business Library) en UBL (Universal Business Language, welke verder bouwt op xCBL). Merk op dat het de bedoeling is van xCBL en UBL om bruikbaar te zijn voor alle industrieën. Dergelijke standaarden, die door alle industrietakken gebruikt kunnen worden, worden *horizontale*

standaarden genoemd [30]. Deze staan in contrast tot *verticale* standaarden (zoals bijvoorbeeld RosettaNet) die op een specifieke industrietak toegespitst zijn (de high-tech industrie in het geval van RosettaNet). Om algemeen toepasbaar te zijn zouden de horizontale standaarden de verticale standaarden dus moeten omvatten.

- 10 Oplossingen voor bovenstaande uitdagingen worden krachtiger naarmate ze meer met elkaar gecombineerd kunnen worden. Dit wordt verwezenlijkt door B2B-frameworks. Deze frameworks zijn verzamelingen van standaarden die samen instaan voor verschillende aspecten van B2Bi, zoals de communicatiewijze, de structurering van boodschappen, het registreren van services, e.d. Op dit vlak verdient ebXML (e-business XML) zeker een vermelding. Zo worden ebXML-berichten bijvoorbeeld gestructureerd volgens de SOAP-with-attachments specificatie (een variante op SOAP, zie 2.3.1). ebXML is een horizontaal framework dat aangevuld kan worden met verticale frameworks. Hamilton [31] stelt bijvoorbeeld dat RosettaNet van plan is in de toekomst ebXML te ondersteunen, om zodoende interoperabiliteit over verschillende industrietakken heen te waarborgen.

2.3 Bestaande standaarden

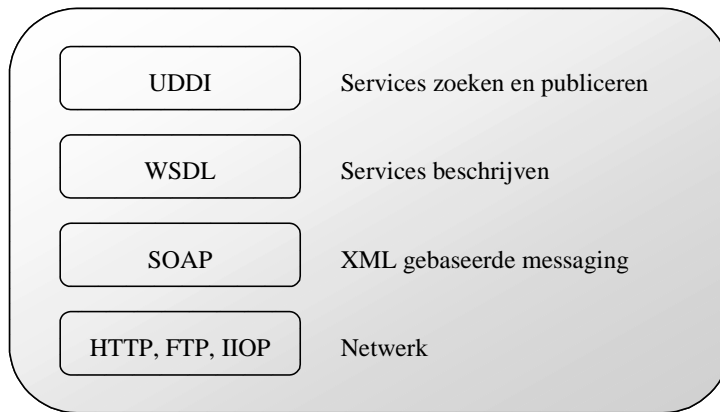
In het web-services domein kunnen we drie basisstandaarden onderkennen: SOAP, WSDL (Web Service Description Language) en UDDI (Universal Description, Discovery, and Integration). Deze standaarden vormen de basis voor de verdere ontwikkeling van andere beloftevolle standaarden. In wat volgt bespreken we kort eerst (in 2.3.1) de drie basisstandaarden. Vervolgens (in 2.3.2) komen de minder stabiele, nieuwere standaarden aan bod. Deze standaarden evolueren momenteel enorm snel en zijn groot in getale. We beperken ons dan ook tot een korte bespreking van de belangrijkste standaarden en tonen enkel het basisprincipe en de mogelijkheden van elke standaard.

2.3.1 De basisstandaarden SOAP, WSDL en UDDI

Zoals gezegd worden web-services doorgaans geassocieerd met drie industriestandaarden: SOAP, WSDL en UDDI. Deze standaarden worden als dermate belangrijk aanzien dat Leymann web-services zelfs omschrijft als '*self-contained, modular business process applications that are based on the industry standard technologies of WSDL (to describe), UDDI (to advertise and syndicate), and SOAP (to communicate)*' [32].

Kort samengevat kunnen we dus stellen dat

- SOAP-berichten verstuurd worden om te communiceren met web-services,
- WSDL een taal biedt om web-services te beschrijven en
- UDDI de infrastructuur voorziet om web-services te publiceren en te zoeken.



Figuur 2. De klassieke web-services stack

In Figuur 2 wordt de klassieke *web-services stack* afgebeeld. Een netwerk is hierbij de basis voor het versturen van SOAP-berichten. Zo'n bericht kan verstuurd worden om een web-service op te roepen, of om informatie te versturen. Alhoewel SOAP gebaseerd is op het synchrone XML-RPC protocol, kan het ook asynchroon gebruikt worden³. SOAP versie 1.2 is momenteel een working draft bij het W3C. Versie 1.2 zal echter niet backward compatibel zijn met versie 1.1. Merk op dat een SOAP-bericht een geldig (well-formed) XML-bericht is. In Appendix D wordt een eenvoudig SOAP-bericht gepresenteerd.

SOAP berichten kunnen verstuurd worden naar gekende web-services. Om te weten te komen welke web-services bestaan, kan er bijvoorbeeld gezocht worden in een UDDI (Universal Description, Discovery, and Integration)-registry. Wanneer men het over UDDI heeft, bedoelt men doorgaans de wereldwijde UDDI-registries waarin iedereen web-services kan publiceren en naar gepubliceerde web-services kan zoeken. Merk op dat er ook iets bestaat als 'private UDDI', waarbij geen globale - maar een private registry gebruikt wordt, die slechts voor een specifieke groep entiteiten toegankelijk is (bv. voor een aantal partners in een Supply Chain). Er wordt verwacht dat dit principe van private UDDI de eerstkomende jaren meer gebruikt zal worden dan globale UDDI [33].

³ Het verschil tussen asynchroon en synchroon berust hierop dat bij een synchrone oproep het oproepende systeem wacht op een antwoord van de web-service alvorens verder te werken (zoals bij een klassieke Remote Procedure Call, RPC); terwijl bij een asynchrone oproep het oproepende systeem onmiddellijk verder werkt na het versturen van het request.

UDDI is gebouwd rond drie soorten informatie:

- *white pages* met informatie (zoals naam en adres) over het bedrijf dat de service levert,
- *yellow pages* die een categorisatie geven van bedrijven op basis van industrie, lokatie, producten en diensten, en
- *green pages* met meer technische details (zoals service beschrijvingen en *binding* informatie), nodig om de service effectief op te roepen.

Het zoeken in een UDDI-registry gebeurt op basis van de bedrijfsnaam, de industrie-code, de geografische lokatie van het bedrijf of de aangeboden goederen en diensten.

In een registry zijn de web-services in één of andere taal beschreven, bv. in WSDL (Web Service Description Language). WSDL 1.1 was een voorstel van IBM, Microsoft en Ariba, en is momenteel een Note van het W3C (zie <http://www.w3.org/TR/wsdl>). WSDL 1.2 is ondertussen een Working Draft bij het W3C.

WSDL is een XML grammatica voor het specificeren van eigenschappen van web-services, zoals de lokatie van de service, de functie van de service en de manier om de service op te roepen. Een voorbeeld van een WSDL document is te vinden in Appendix E. WSDL wordt vaak vergeleken met de Interface Definition Languages (IDL) die we terugvinden bij klassieke Remote Procedure Call (RPC) mechanismen [34]. Gudgin en Ewald [35] argumenteren dat deze vergelijking slechts opgaat in de zin dat beide gebruikt worden om poorttypes en interfaces te beschrijven. Gudgin en Ewald stellen immers dat programmeurs WSDL niet willen gebruiken als vertrekpunt: “*Modern web service tools do not expect developers to write WSDL definitions. From the complexity of the language specification, one might assume that the authors of WSDL have no such expectation, either.*” Dit is echter wat typisch wel gebeurt bij IDL. Programmeurs schrijven eerst IDL, laten vervolgens een IDL compiler draaien en verkrijgen zo broncode en meta-data die gebruikt kunnen worden om clients en servers te ontwikkelen die *compliant* zijn met de interface. WSDL-definities worden echter meestal gegenereerd uit bestaande broncode. Volgens Gudgin en Ewald zou dit zwaar kunnen wegen op de snelheid waarmee de web-services technologie geadopteerd wordt.

Bovenop de basis-stack (Figuur 2) kunnen nog vele lagen komen. Er zou bijvoorbeeld een *flow-laag* toegevoegd kunnen worden die een choreografie van web-services weergeeft [36]. Omgekeerd kan er echter ook gesteld worden dat een verzameling van web-services, met elkaar verbonden via een choreografie, eveneens aangeboden kan worden als één enkele web-service, die gepubliceerd en gevonden kan worden in een UDDI-registry. Zodoende zou de flow-laag eventueel tussen UDDI en WSDL geplaatst kunnen worden [37]. Verschillende

organisaties (zoals IBM, W3C, Microsoft, Sun, enz.) stellen verschillende web-services stacks voor. Een overzicht wordt gegeven door Myerson [38].

Merk op dat er op elke laag eventueel nog protocols toegevoegd kunnen worden (bv. DAML-S voor het beschrijven van web-services).

2.3.2 Andere web-service standaarden

Voor het **beschrijven** van web-services kan niet alleen gebruik gemaakt worden van WSDL, maar ook van *RDF* en *DAML-S*. Wegens het grotere belang van dergelijke standaarden voor ons verder onderzoek, bespreken we ze diepgaander in een aparte sectie (zie 3.3.2).

Naast UDDI zijn er nog andere initiatieven om het **vinden** van web-services mogelijk te maken. Zo bijvoorbeeld *WSIL* (Web-service Inspection Language), een initiatief van IBM en Microsoft. *WSIL* verschilt van UDDI in die zin dat er geen globale registries meer gebruikt worden: waar UDDI een gecentraliseerd model gebruikt, steunt *WSIL* op een gedistribueerd model [39]. *WSIL* vereist dan ook dat iemand die informatie over een service zoekt vooraf al weet wie die service aanbiedt. Op de website van de gekende service provider staat dan een *WSIL* document. Dit is een XML-document met pointers naar documenten die web-services beschrijven (bv. naar WSDL documenten) [40]. *WSIL* biedt dus minder functionaliteit dan UDDI, maar is dan ook minder complex.

Om de juiste web-services te vinden is het belangrijk te kunnen steunen op een goede omschrijving van welke service gezocht wordt en van welke services aangeboden worden. Wanneer web-services op een krachtiger manier (met meer semantiek) beschreven worden dan met WSDL, kunnen de gewenste web-services wellicht gemakkelijker gevonden worden. Paolucci en zijn collega's [41] werken hiertoe bijvoorbeeld een *matching engine* uit die werkt op basis van DAML-S (zie 3.3.2).

Voor het beschrijven van een **choreografie** van web-services is er een hele reeks standaarden in ontwikkeling, waaronder BPML, WSCI, WSCL en BPEL4WS (zie infra). Ons inziens verliest een choreografie-taal veel van haar nut indien ze niet toelaat transacties te realiseren. Vanuit dit oogpunt is het dan ook relevant eerst een blik te werpen op BTP, een protocol dat specifiek gericht is op de realisatie van **transacties** via web-services.

Het *Business Transaction Protocol (BTP)* wordt gestandaardiseerd door OASIS. Sinds mei 2002 is de specificatie van BTP 1.0 beschikbaar. Bij BTP wordt een onderscheid gemaakt tussen *atomic* business transacties en *cohesive* business transacties. Atomic transacties zijn transacties waarbij ofwel alle werk bevestigd wordt, ofwel helemaal geen werk bevestigd wordt (het alles-of-niets karakter van transacties). Bij cohesive transacties is het mogelijk slechts een deel van het werk te bevestigen. Opdat het systeem zich steeds in een consistente toestand zou

bevinden, wordt gebruik gemaakt van *provisional effects*, *final effects*, en *counter effects* om een two-phase systeem te creëren. Merk op dat BTP steunt op een coördinerende entiteit die bepaalt welke acties bevestigd zullen worden en welke geannuleerd. De coördinerende entiteit verstuurt ook effectief boodschappen om de gewenste final – en counter effects te bekomen.

De *Web Services Conversation Language* (WSCL 1.0, W3C Note) is bedoeld als een eenvoudige choreografietaal voor het definiëren van legale volgordes waarin boodschappen verstuurd mogen worden. Momenteel voorziet deze taal geen ondersteuning voor transacties.

De *Business Process Execution Language For Web Services* (BPEL4WS, ook wel BPEL) combineert de ideeën van - en is bedoeld als vervanger van - WSFL (Web-services Flow Language, IBM) en XLang (Microsoft). Met BPEL kunnen bedrijven business-processen specificeren en expliciteren hoe deze processen gerelateerd zijn tot web-services. Deze relatie uit zich op twee vlakken. Ten eerste wordt duidelijk gemaakt welke web-services een proces aanbiedt en ten tweede wordt bepaald welke web-services het proces zelf gebruikt. Merk op dat business-processen op twee manieren kunnen gespecificeerd worden met BPEL: (1) in een uitvoerbare (executable) vorm en (2) in de vorm van een *business protocol*. Met een 'business protocol' bedoelt men de declaratie van de boodschappen die opeenvolgend verstuurd moeten worden om een businessdoel te bereiken. De PIPs (Partner Interface Processes) van RosettaNet kunnen bijvoorbeeld als een business protocol bekeken worden. Applicaties die ontwikkeld worden in BPEL zijn proces-gebaseerd. Dit wil zeggen dat de applicatie in twee strikt gescheiden lagen gesplitst wordt: een toplaag die het business-process voorstelt, geschreven in BPEL, welke de flow van de applicatie weergeeft, en een onderste laag die bestaat uit de implementatie van de web-services zelf. Deze splitsing zorgt voor een grote flexibiliteit bij het wijzigen van de applicatie. BPEL voorziet het principe van *fault handlers*. Deze definiëren een set van fouten met bijhorende activiteiten (bv. compenserende activiteiten) die uitgevoerd moeten worden wanneer een specifieke fout zich voordoet. Op deze manier kunnen transacties gerealiseerd worden. Merk op dat hier dus geen gebruik gemaakt wordt van een locking-protocol voor het realiseren van transacties, maar wel van compenserende activiteiten. Zodoende kunnen long-running transacties gerealiseerd worden [42]. De manier waarop met transacties wordt omgegaan berust eigenlijk op het WS-transaction protocol (een specificatie die tegelijkertijd met die van BPEL werd uitgebracht door dezelfde partijen: IBM, Microsoft en BEA). We wijzen erop dat WS-transaction - in tegenstelling tot BPEL - ook ondersteuning biedt voor gedistribueerde atomaire transacties (dit zijn short-lived transacties, doorgaans voorzien via een two-phase commit protocol). Leymann [42] meldt dat het mogelijk is dat BPEL in de toekomst wel ondersteuning zal bieden voor

gedistribueerde atomaire transacties. Merk op dat het realiseren van transacties momenteel totaal anders verloopt bij BPEL dan bij BTP.

De *Business Process Modeling Language (BPML) 1.0* is een specificatie van het BPMI (Business Process Management Initiative). Het BPMI aanziet e-business processen (waarin twee partijen betrokken zijn) als een combinatie van drie elementen: één gemeenschappelijke publieke interface en twee private implementaties. De publieke interface wordt ondersteund door protocollen als ebXML en RosettaNet, welke gebaseerd zijn op het uitwisselen van boodschappen. De twee private implementaties zijn partner-specifiek en zijn beschreven in een uitvoerbare taal, zoals BPML bijvoorbeeld [43]. BPML ondersteunt twee transactiemodellen: ‘gecoördineerde’ transacties en ‘uitgebreide’ transacties. Bij gecooördineerde transacties worden de ACID eigenschappen van transacties gerespecteerd via een two-phase commit protocol. Het uitgebreide transactiemodel daarentegen lost de isolation-eigenschap, terwijl het alles-of-niets karakter van transacties wel bewaard blijft via het roll-back mechanisme en via compenserende transacties. Noteer dat BPML, net als BPEL, niet enkel toelaat uitvoerbare processen – maar ook abstracte processen – te beschrijven.

De *Web Service Choreography Interface (WSCI)* was een initiatief van SAP AG, Sun Microsystems, BEA Systems en Intalio, en is nu een W3C note. WSCI is er, net als WSCL, op gericht boodschapuitwisselingen te definiëren. Het opvallende aan WSCI is dat alles belicht wordt vanuit de invalshoek van één specifieke web-service. Zo kan bijvoorbeeld vastgelegd worden welke choreografie van boodschappen een specifieke web-service aankant, welke operaties uitgevoerd kunnen worden als transacties of hoe een web-service zal werken in verschillende omstandigheden. WSCI verstrekt als dusdanig informatie die nuttig is bij het opstellen van een choreografie van web-services. WSCI kan in de web-services stack dan ook gezien worden als een laag onder de proces-modellering [44].

Omtrent de choreografie van web-services wordt veel discussie gevoerd. Dubray [45] meent bijvoorbeeld dat WSCI weinig interessants te bieden heeft. WSCI gaat er immers van uit dat de choreografieën van interfaces los van elkaar gespecificeerd kunnen worden, en dat operaties dan twee-per-twee met elkaar verbonden kunnen worden. Er wordt dus vanuit gegaan dat de interfaces van los van elkaar ontworpen web-services compatibel zullen zijn. Dubray stelt dat het logischer is dat handelspartners vertrekken van een bestaande choreografie (die ze eventueel samen ontwikkelden) en vervolgens elk de benodigde interfaces implementeren (zoals bij de ebXML Business Process Specification Schemes). Deze manier van werken gaat ons inziens in tegen de flexibelere (doch vanuit een choreografie-standpunt minder nuttige?) manier van werken waarbij partijen los van elkaar de services die ze aanbieden publiceren zodat die gevonden en aangeroepen kunnen worden.

2.4 Het ontwikkelen van web-services

Doorheen de jaren werden verschillende architecturen gebruikt om een ICT-systeem op poten te zetten. Vele organisaties berusten nu op een 3-tier architectuur, waarbij presentatielogica (op PC), applicatielogica (op een application-server), en databaselogica (op een DB-server) gescheiden zijn. Vaak wordt tussen de gebruiker en de application-server nog een web-server geplaatst.

Application-servers kunnen tegenwoordig omgaan met de notie van componenten. Vandenbulcke [46] beschrijft componenten als *'een coherent geheel van objecten, die noodzakelijk zijn voor de uitvoering van een eenduidig te bepalen functionaliteit, die op een bepaalde manier aan te roepen moet zijn.'* Lindgren [47] en Vandenbulcke [46] stellen dat een object-georiënteerde ontwikkeling van software essentieel is om te komen tot componentensoftware. Beide merken echter op dat een object-georiënteerde ontwikkeling niet noodzakelijk leidt tot componentensoftware, aangezien niet noodzakelijk herbruikbare code gecreëerd wordt. Merckx [48] merkt hieromtrent op dat het streven naar herbruikbare code een edel doel is, doch heel moeilijk te realiseren. Voor Merckx is de beoogde lichte koppeling tussen componenten veeleer de sleutel tot vervangbaarheid van componenten. Componenten moeten dan ook opzichzelfstaande elementen zijn die geen deel uitmaken van een hiërarchie van klassen. Lindgren [47] stelt hieromtrent dat *"a component may be very complex internally and be built using class structures and inheritance; but to the rest of the world, the component is only viewed by a clean, external interface that is not interdependent upon other components."*

De vier meest bekende componentenmodellen zijn (Meyer [49] en Merckx [48]):

- het EJB (Enterprise JavaBeans) componentenmodel,
- .Net en COM+ (Component Object Model +) van Microsoft, en
- CORBA (Common Object Request Broker Architecture) van de OMG (Object Management Group).

Slechts twee van deze lijken voor de toekomst nog relevant: .Net en EJB (met het J2EE platform).

Een open architectuur van componenten is één van de hoekstenen van Gartner's ERP II-systeem (zie surpa) [3]. Vele bedrijven (waaronder SAP) zijn dan ook hun systemen aan het opsplitsen in componenten. Componenten vormen eveneens de basis voor web-services. Rick Van der Lans [33] beschrijft een web-service trouwens als *'a component that is callable by anyone in a loosely coupled style over the Internet using XML and SOAP'*. Het behoeft weinig betoog dat .Net en J2EE de voornaamste platformen zijn voor het ontwikkelen van web-service gebaseerde systemen.

Veel seminars en artikels (bv.[50], [51], [52]) gaan over de vergelijking van .Net en J2EE, al dan niet op basis van hun potentieel op het vlak van web-services. De eindconclusie van dergelijke vergelijkingen is doorgaans dat beide platformen naast elkaar zullen blijven bestaan en dat bedrijven zelfs vaak de twee platformen zullen gebruiken. Hanson [51] maakt een vergelijking van beide op basis van hun mogelijkheden met betrekking tot web-services. Hanson meent dat .Net het voordeel heeft dat het gebouwd werd met als doel de web-services-droom te realiseren, terwijl het J2EE platform slechts aangepast werd aan het web-services-idee door een aantal APIs (zoals JAX-RPC en JAXR) toe te voegen. Anderzijds heeft het J2EE-platform wel al bewezen de nodige robuustheid en schaalbaarheid te kunnen leveren, terwijl dit nog niet het geval is voor .Net [52]. Bovendien is het zo dat er voor J2EE software een aantal verschillende vendors bestaat waaruit gekozen kan worden, terwijl Microsoft de enige vendor is van het .Net platform. Het belangrijkste punt bij de keuze van het platform is echter het bestaande computersysteem van het bedrijf [51]. Samtani en Sadhwani [52] stellen bijvoorbeeld dat J2EE-gebaseerde application servers eerder duur zijn ten opzicht van het .Net-product, doch dat het logisch is dat bedrijven die reeds over een J2EE-gebaseerde application server beschikken deze infrastructuur zullen blijven gebruiken. Immers, het J2EE platform laat o.a. toe bestaande EJBs te *wrappen* en open te stellen als web-services.

Indien het bestaande computersysteem omgevormd moet worden naar een web-services-gebaseerd systeem, is het aangeraden gebruik te maken van ontwikkelingsmethodologieën. In dit kader wordt vaak gesproken over de Unified Modeling Language (UML), een modelleringstaal die gebruikt kan worden bij het toepassen van een ontwikkelingsmethodologie.

Software systemen worden tegenwoordig vaak gemodelleerd in UML. Geïntegreerde ontwikkelingsomgevingen (IDEs, Integrated Development Environments) worden bovendien regelmatig geïntegreerd met UML-tools. Oracle9i JDeveloper bijvoorbeeld voldoet aan de J2EE standaarden en laat toe klassen en activiteiten te modelleren in UML [53]. Zodoende probeert men de productiviteit van software-ontwikkelaars op te drijven.

Ook voor het ontwikkelen van web-service-georiënteerde systemen kan vertrokken worden van UML modellen. Merckx [54] maakt hierbij een onderscheid tussen de top-down benadering en de bottom-up benadering. Bij de top-down benadering wordt gestart vanaf de business interacties en worden eerst services en data-uitwisselingen geïdentificeerd. Bij de bottom-up benadering daarentegen wordt vertrokken van bestaande transacties en wordt bestaande software zoveel mogelijk hergebruikt. Dit komt dan neer op het *componentizeren* van bestaande software.

Ambler [55] past de bottom-up benadering toe om web-services te ontwikkelen. Volgens Ambler moeten hierbij domein-pakketten⁴ geïdentificeerd worden en moet voor elk pakket bepaald worden welke services het voorziet. Het komt er dus op neer de klassen in het UML-diagram te verdelen over pakketten, en die pakketten vervolgens te integreren via web-services. Er moet bijgevolg gezocht worden naar de services die elk pakket zou moeten voorzien. Ambler beschrijft hiervoor drie stappen die doorlopen moeten worden.

In een eerste stap wordt het object design vereenvoudigd. Dit houdt in dat er abstractie gemaakt wordt van persistente klassen (klassen die instaan voor de toegang tot persistente opslag) en systeemklassen (klassen die technische elementen bevatten) om zodoende meer te concentreren op de zakelijke kant van het ontwerp. Daarenboven moeten ook hiërarchieën vereenvoudigd worden.

In de tweede stap wordt het design geherorganiseerd in pakketten met grote cohesie. Hiertoe worden eerst alle operaties geïdentificeerd die opgeroepen kunnen worden door andere klassen. Vervolgens wordt voor elke klasse bekeken of het een server-klasse, een client-klasse of een client&server-klasse is. Daarna worden de domein-pakketten samengesteld. Deze pakketten hebben als doel te reageren op berichten. Ze lijken (net als componenten, zie supra) van de buitenkant eenvoudig, maar kunnen intern heel complex zijn. Een vuistregel bij het samenstellen van de pakketten is dat elke server-klasse een potentieel domein-pakket is. Daarenboven is het vaak verantwoord twee pakketten te combineren indien een domein-pakket slechts een server is voor één ander pakket. Een tweede vuistregel is dat client-klassen niet thuishoren in pakketten, aangezien zij berichten genereren in plaats van ze te ontvangen. Bij het samenstellen van pakketten is het ook belangrijk oog te hebben voor efficiëntie. Zo kunnen klassen die vaak samenwerken best samengebracht worden in één pakket aangezien dit de overhead bij het oproepen van web-services voorkomt.

In de derde stap worden de web-services geïdentificeerd die elk pakket aanbiedt. Dit houdt in dat de operaties genoteerd worden die een domein-pakket aanbiedt aan andere pakketten. Om het aantal aangeboden web-services te beperken vindt Ambler het aangeraden te zoeken naar operaties die samengevoegd kunnen worden tot één geheel met grotere functionaliteit. Eens de web-services geïdentificeerd zijn, moet bepaald worden welke informatie *minimum* nodig is opdat de web-service zijn doel zou kunnen vervullen. Deze informatiebehoefte is de basis voor het vastleggen van parameters en return-values van de web-service.

Uit het voorgaande stappenplan blijkt duidelijk dat er bij de bottom-up benadering weinig aandacht geschonken wordt aan de services zelf en aan de vraag welke services er eigenlijk nodig zijn. Bij de top-down benadering is dit echter het startpunt van het hele ontwikkelingsproces. Ook voor de top-down benadering

⁴ Een domein pakket is een verzameling van klassen die samenwerken om een samenhangende (cohesive) set van contracten te ondersteunen.

kan UML gebruikt worden (zie bijvoorbeeld [54]). We gaan niet dieper in op de top-down benadering vermits dit ons zou leiden tot het bespreken van klassiekere analyse- en designmethoden, wat weinig vernieuwends inhoudt.

Uit de voorgaande sectie blijkt dat het gebruik van web-services technologie voor de integratie van systemen heel beloftevol is, doch dat de technologie nog volop in ontwikkeling is. De ontwikkeling van (open) standaarden is van primair belang, wil de technologie de automatische integratie van systemen mogelijk maken..

3 Het semantisch web

De integratie van computersystemen beperkt zich niet tot het realiseren van een fysieke connectie, maar behelst ook een logische stap die gericht is op semantiek [56]. De informatie die gedeeld wordt tussen bedrijven is immers nutteloos indien beide partijen niet weten wat de informatie eigenlijk wil zeggen. Terwijl het eerste punt, de fysieke integratie, meer en meer op punt gesteld wordt (bijvoorbeeld door het gebruik van SOAP, zie 2.3.1), lijkt de logische integratie – het effectief *verstaanbaar* maken van de gedeelde bronnen – nog maar in een eerste ontwikkelingsfase [56]. Het moge duidelijk wezen dat het toevoegen van XML-tags aan data nog niet duidelijk maakt wat de data eigenlijk betekenen. Immers, de betekenis van het woord ‘klantcode’ kan fel verschillen van partij tot partij. Weinig organisaties lijken de ernst van dit probleem goed in te schatten. Pollock [56] aanziet het onderschatten van het belang van semantiek bij integratievraagstukken dan ook als één van de belangrijkste oorzaken van het falen van integratiepogingen. Het is een misvatting te denken dat het gebruik van web-services dit probleem automatisch oplost. Web-services maken de betekenis van data immers evenmin duidelijk. Grand Central en Bowstreet, twee vernieuwers in het web-services domein, hechten wel veel belang aan semantiek bij het gebruik van web-services [56,57,58]. We komen terug op de relatie tussen semantiek en web-services in 3.3.2. Eerst (in 3.1) verduidelijken we echter wat we verstaan onder de term ‘semantisch web’ en schetsen we een beeld van de totale problematiek. Vervolgens bespreken we (in 3.2 en 3.3) een aantal standaarden die als basis voor het semantisch web dienst zouden kunnen doen. ‘Zouden’, want het semantisch web is eerlang nog geen realiteit.

3.1 Wat is het semantisch web?

Om de betekenis van data op een *flexibele* manier globaal verstaanbaar te maken - niet enkel voor mensen maar ook voor computers - beoogt men de realisatie van een semantisch web. Bemerkt dat we streven naar een *flexibele* manier om betekenissen duidelijk te maken. Mechanismes, zoals EDI (Electronic Data Interchange) bijvoorbeeld, waarbij lange negotiaties nodig zijn tussen twee partijen om duidelijk te stellen wat de uitgewisselde data willen zeggen, zijn

immers uit den boze als een adaptief systeem gecreëerd moet worden. Merk eveneens op dat het irrealistisch is te veronderstellen dat eenieder na afspraak eenzelfde standaardwoord zal gebruiken voor een bepaald concept. Het semantisch web moet dus toelaten dat elke partij zijn eigen vocabularium blijft gebruiken, doch dat het voor een andere partij duidelijk is wat bedoeld wordt met bepaalde termen.

Het semantisch web staat nog maar in de kinderschoenen, wat zich uit in heel vage definities van het concept. Bij het W3C (met aan het hoofd Tim Berners-Lee, een groot protagonist van het semantisch web) vinden we bijvoorbeeld volgende definitie [59]:

“The Semantic Web is the abstract representation of data on the World Wide Web, based on the RDF standards and other standards to be defined.”

De volgende beschrijving, eveneens van het W3C [60], verduidelijkt meer het doel van het semantisch web:

“The semantic Web is a vision: the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications.”

Berners-Lee et al. [62] stellen dat er voor de werking van het semantisch web twee zaken nodig zijn. Ten eerste moeten computers toegang krijgen tot gestructureerde verzamelingen van informatie (ontologieën, zie infra) en ten tweede moeten computers over sets met *inference rules* beschikken die ze kunnen gebruiken om automatisch te redeneren. Deze zaken worden binnen het AI (Artificiële Intelligentie)-domein al lange tijd onderzocht onder de noemer ‘Knowledge Representation’(KR), dit echter los van de evolutie van het Internet. Levesque [63] et al. definiëren KR als

“... the field of study within AI concerned with using formal symbols to represent a collection of propositions believed by some agent.”

Sowa [64] beschrijft KR als:

“... the application of logic and ontology to the task of constructing computable models for some domain.”

De *logic* (de *inference rules*) waarover Sowa het heeft, zorgt ervoor dat er effectief met de gestructureerde informatie (bijvoorbeeld een verzameling RDF-propositions, zie infra) omgegaan kan worden. Merk op dat *propositions* de *beliefs* van een agent weergeven (bijvoorbeeld het geloof dat *A de zoon is van B*),

doch dat niet *alle* beliefs van een agent noodzakelijk geëxpliciteerd zijn in propositions. Immers als

- (1) geweten is dat een zoon van een zoon een kleinzoon is, en
- (2) dat A de zoon is van B en B de zoon is van C,

dan is hierin impliciet vervat dat A de kleinzoon is van C. Levesque [63] stelt dat *“It is the role of reasoning to bridge the gap between what is represented and the full set of propositions believed.”*

De overeenkomst van Sowa’s definitie met de visie van Berners-Lee is duidelijk. Sowa’s *logic* doelt immers op het gebruik van inference rules en Sowa’s *ontology* komt overeen met gestructureerde verzamelingen van informatie. Merk op dat traditionele KR toch in enige mate los gezien moet worden van de realisatie van het semantisch web. Traditionele KR-systemen beperken immers doorgaans de vragen die gesteld kunnen worden aan het systeem, opdat er steeds een betrouwbaar antwoord gegeven zou kunnen worden. Bij het semantisch web wordt er echter vanuit gegaan dat niet alle vragen beantwoord kunnen worden en dat er paradoxen kunnen optreden [62]. Daarenboven zijn traditionele KR-systemen gecentraliseerd (dit in tegenstelling tot het Internet) en vereisen zij bijgevolg dat eenieder exact dezelfde definitie van concepten hanteert.

Zoals gezegd is het in een B2B context irrealistisch te veronderstellen dat iedereen een gemeenschappelijk, eenduidig vocabularium zal gebruiken. Het is eerder zo dat ieder een eigen vocabularium zal ontwikkelen, en dat de verschillende vocabularia met elkaar verbonden zullen worden om duidelijk te stellen hoe termen geïnterpreteerd moeten worden. Immers, de betekenis van uitgewisselde boodschappen moet voor alle betrokken partijen automatisch duidelijk zijn. Nauw gerelateerd aan het concept van (individuele) vocabularia, is het idee van ‘ontologieën’.

Ontologieën worden gebruikt om vast te leggen welke relatie er bestaat tussen *concrete* concepten. Een ontologie kan bijvoorbeeld duidelijk maken dat een bank een zitmeubel is en dat zitmeubels op hun beurt meubels zijn. Om te verduidelijken wat verstaan wordt onder een ontologie kunnen we best te rade gaan bij Gruber [65]. Hij beschrijft een ontologie als *“a specification of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents.”* Hermans [66] omschrijft ontologieën als *“a shared set of terms describing the application domain with common understanding... not only the terms but also their relationships”*. Door op webpagina’s pointers te plaatsen naar ontologieën wordt de *betekenis* van de inhoud van de pagina duidelijk. Op deze manier kan het zoeken naar informatie op het web effectiever verlopen. Het zoeken naar banken (als financiële

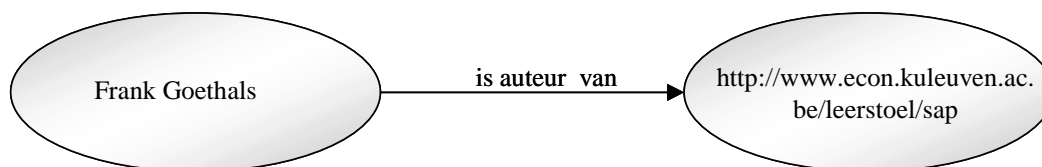
instellingen) bijvoorbeeld zal dus niet langer resultaten opleveren waarbij het woord bank gebruikt wordt om een zitmeubel aan te duiden.

Het combineren van ontologieën zoals eerder voorgesteld, is onderhevig aan verscheidene problemen. Klein [67] ziet problemen op twee niveaus. Ten eerste kunnen er op het niveau van het meta-model (het niveau van de ontologietaal) moeilijkheden rijzen doordat verschillende partijen verschillende ontologietalen gebruiken die zaken op verschillende manieren uitdrukken (zie infra). De ene taal biedt overigens ook meer mogelijkheden dan de andere. Ten tweede kunnen er zich problemen voordoen op het niveau van het model zelf (de ontologie zelf). Deze kunnen ook optreden wanneer er eenzelfde ontologietaal gebruikt wordt. We kunnen op dit niveau een onderscheid maken tussen ‘conceptualization mismatches’ (door een verschil in scope en granularity) en ‘explication mismatches’ (bijvoorbeeld doordat concepten op andere niveaus geplaatst worden in de ontologie of dat er synoniemen en homonymen kunnen voorkomen).

3.2 RDF als basisstandaard voor het creëren van gestructureerde verzamelingen van informatie

De *propositions* uit de definitie van Levesque moeten op één of andere manier weergegeven kunnen worden. Hier komen o.a. XML en RDF ten tonele. Zoals uit de definitie van het semantisch web blijkt, speelt RDF (Resource Description Framework, een W3C Recommendation sinds februari 1999) een belangrijke rol bij de realisatie van het semantisch web. Een andere basistechnologie is XML (eXtensible Markup Language). XML biedt de mogelijkheid een willekeurige structuur toe te voegen aan documenten, maar zegt niets over de betekenis van de structuur⁵. Dit laatste wordt gerealiseerd door RDF [62]⁶.

RDF codeert betekenissen door *resources*, *properties* te geven met een bepaalde *value*. Zo kan bijvoorbeeld de resource ‘Frank Goethals’ de *property* ‘is auteur van’ krijgen, met de value “<http://www.econ.kuleuven.ac.be/leerstoel/sap>”. Dit wordt weergegeven in Figuur 3:



Figuur 3. Voorbeeld van een RDF statement

⁵ Appendices B t.e.m. F kunnen bekeken worden als voorbeelden van XML-documenten.

⁶ Een ander initiatief op dit vlak, dat we hier niet verder behandelen, is XTM (XML Topic Maps). Meer informatie hierover is te vinden op <http://www.topicmaps.org/xtm/>.

Dergelijke RDF-beschrijvingen kunnen in XML uitgedrukt worden. Het bovenstaande voorbeeld wordt in RDF/XML:

```
<?xml version="1.0"?>
<RDF
  xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
  <Description
    about="http://www.econ.kuleuven.ac.be/leerstoel/sap">
    <s:auteur>Frank Goethals</s:auteur>
  </Description>
</RDF>
```

De lezer kan zich nu afvragen wat het nut is van RDF. RDF-statements kunnen immers gewoon in XML uitgedrukt worden. Het punt is dat RDF voorziet in een standaard manier om statements te schrijven, zodat het duidelijk is welke zaken *resource*, *property* en *value* zijn van een statement. Terwijl er verschillende XML-bomen mogelijk zijn om een statement te beschrijven⁷, kunnen al deze bomen omgevormd worden tot één RDF-boom voor het beschrijven van het statement [68].

De combinatie van de drie elementen <resource, property, value> blijkt een natuurlijke manier te zijn om een groot deel van de informatie te beschrijven die door machines verwerkt wordt [62]. Merk op dat RDF een generiek model is: RDF voorziet een mechanisme om informatie te beschrijven, los van het domein waaruit deze informatie komt. Naast RDF bestaat er wel een RDFS (RDF Schema) taal die basisstructuren voorziet zoals klassen, subklassen en eigenschappen. Met RDFS kan bijvoorbeeld een klasse ‘Product’ gedefinieerd worden, of een eigenschap ‘Productnummer’. Hoe dit gebeurt, en hoe in RDF specifieke producten beschreven worden, wordt geïllustreerd in Appendix F⁸. Elk van de elementen in een RDF-statement wordt geïdentificeerd door een Universal Resource Identifier (URI), welke doorgaans de vorm aanneemt van een URL (Uniform Resource Locator). De URIs zorgen ervoor dat elk concept een éénduidige betekenis heeft. Immers, een woord (bv. ‘bank’) kan meerdere betekenissen hebben (bv. meubel of financiële instelling). Voor elke verschillende betekenis moet bijgevolg een URI bestaan.

⁷ We zouden het voorbeeld-statement bijvoorbeeld ook als volgt kunnen schrijven in XML:

```
<auteur>
  <uri> http://www.econ.kuleuven.ac.be/leerstoel/sap </uri>
  <naam>Frank Goethals</naam>
</auteur>
```

⁸ Merk op dat de relatie tussen RDFS en RDF niet dezelfde is als die tussen XML Schema en XML. Immers, door XML Schema wordt een ordening en een combinatie van tags in een XML document *opgelegd*. RDFS daarentegen voegt enkel semantiek toe aan de beschrijving. Een RDFS kan natuurlijk wel vertaald worden in een XML Schema om een documenttype voor te schrijven waarvan de *instances* zullen voldoen aan de RDFS specificaties [78].

3.3 Andere standaarden

Levesque [69] stelt dat er een afweging moet worden gemaakt tussen de efficiëntie waarmee inference rules toegepast kunnen worden enerzijds en de uitdrukingskracht en de leesbaarheid van een KR anderzijds. Omwille hiervan is het nodig op zoek te gaan naar een goed representatie-formalisme. Eén poging in die richting is DAML+OIL (DARPA Agent Markup Language + Ontology Interchange Language, ook wel Ontology Inference Language) [70]. DAML+OIL bouwt verder op RDF(S). We verdiepen ons verder in DAML+OIL in 3.3.1. In 3.3.2 bespreken we DAML-S (DAML-Services), een standaard gericht op het beschrijven van web-services.

3.3.1 Een stap verder met DAML+OIL

Net als bij RDF, is het doel van DAML+OIL constructen te voorzien om ontologieën te creëren en om een markup toe te voegen aan informatie zodat die verstaanbaar wordt voor machines [71]. Wegens de grote eenvoud van RDF(S) bleek deze standaard ontoereikend op bepaalde vlakken. Zo was het bijvoorbeeld wenselijk een consistente expressie te bepalen voor opsommingen. DAML+OIL is een Note van het W3C.

DAML+OIL laat toe meer gesofisticeerde klassen en eigenschappen uit te drukken dan mogelijk was in RDFS. Zo kunnen de waarden van eigenschappen bijvoorbeeld beperkt worden tot datatypes die gedefinieerd zijn in XSDL (XML Schema Definition Language) of die door de gebruiker zelf gedefinieerd zijn. Verder kan voor een bepaalde eigenschap (bv. productnummer) bepaald worden dat de waarde ervan uniek moet zijn over alle instances van een bepaalde klasse (bv. product). Deze twee zaken zijn geïllustreerd in Appendix G.

Merk op dat de DAML+OIL primitieven gedefinieerd kunnen worden in RDF Schema, zodat elke ontologie die geschreven is in DAML+OIL een geldig RDF document vormt. De grote sterkte van DAML+OIL zit in de uitgebreide mogelijkheden om klassen te beschrijven. Zo is het bijvoorbeeld mogelijk te bepalen dat twee klassen strikt gescheiden moeten zijn.

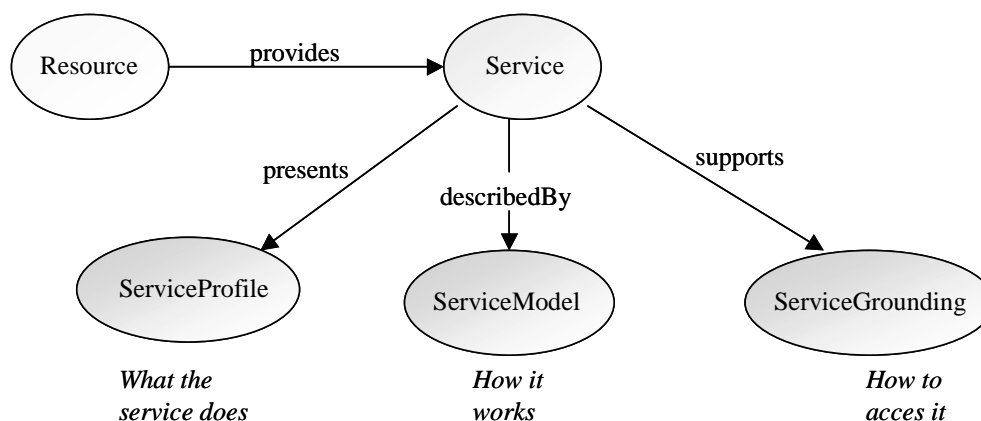
Noteer eveneens dat DAML+OIL de 'samePropertyAs'-constructie voorziet, waarmee aangegeven kan worden dat twee eigenschappen gelijk zijn aan elkaar. Dit is interessant, aangezien het iedereen toelaat z'n eigen vocabularium te ontwikkelen, en later de definities van het eigen vocabularium uit te breiden met de samePropertyAs eigenschap. Een onderneming die in zijn ontologie de eigenschap 'productnr' definieert, kan later aangeven dat deze eigenschap overeenkomt met de eigenschap 'productID' in een andere ontologie. Deze mogelijkheid vergemakkelijkt het combineren van ontologieën, zoals beschreven in 3.1. Zoals gezegd, kunnen er zich echter nog vele andere problemen voordoen wanneer ontologieën gecombineerd moeten worden.

De Web-Ontology Working Group van het W3C bouwt momenteel verder op DAML+OIL om een nieuwe ontologie-taal te creëren, OWL (Ontology Web Language) genaamd [72].

3.3.2 Het krachtiger beschrijven van web-services met DAML-S

RDF en DAML+OIL blijken ook mogelijkheden te bieden voor het beschrijven van web-services. DAML+OIL werd zelfs verder ontwikkeld tot DAML-S (DAML-Services). DAML-S voorziet web-service providers van een aantal markup constructies om de eigenschappen en mogelijkheden van web-services te beschrijven in een eenduidige vorm die door een computer geïnterpreteerd kan worden. Het probleem dat DAML-S probeert aan te pakken zit hierin dat de huidige stand van zaken op het vlak van web-services nog steeds een menselijke inbreng vereist om benodigde web-services te zoeken (bijvoorbeeld via UDDI) en op te roepen. Op lange termijn is het echter de bedoeling de menselijke tussenstap hier overbodig te maken, en software agents⁹ autonoom te laten beslissen welke services opgeroepen moeten worden in bepaalde omstandigheden. Deze agents zouden (dankzij een semantische markup van de gegevens) in staat moeten zijn services te ontdekken, te verstaan en samen te stellen tot transacties en krachtigere web-services. Merk op dat UDDI geen ondersteuning biedt voor het beschrijven van procesmodellen, en dus geen automatische samenstelling van services toelaat. Zoals we zullen zien, pakt DAML-S dit probleem wel aan.

Het basisidee achter DAML-S wordt afgebeeld in Figuur 4:



Figuur 4. De basis van DAML-S [73]

⁹ Een eenduidige definitie van de term 'software agent' bestaat er niet. Bij Wooldridge [13] vinden we de volgende beschrijving: "An agent is a computer system capable of autonomous action in some environment."

DAML-S is gebouwd op drie vragen:

- Wat vereist de service van de gebruiker(s), of van andere agenten, en wat levert de service hen? Dit is een voor mensen begrijpbare beschrijving van de service, de input en output types, de precondities en postcondities, etc.
- Wat gebeurt er bij de uitvoering van de service?
- Hoe kan de service opgeroepen worden (poortnummers, communicatie-protocol, etc.)?

Om deze vragen te beantwoorden worden drie eigenschappen gedefinieerd van web-services. Voor de bovenstaande vragen zijn deze respectievelijk 'presents', 'describedBy' en 'supports'. Voor elk van deze eigenschappen wordt een bereik gedefinieerd, respectievelijk 'ServiceProfile', 'ServiceModel' en 'ServiceGrounding'. In het algemeen kunnen we stellen dat een ServiceProfile de benodigde informatie bevat opdat een agent de service zou kunnen ontdekken. Het ServiceModel en de ServiceGrounding voorzien de nodige informatie om een service-oproep te kunnen implementeren en de service te kunnen gebruiken.

De 'Service' uit Figuur 4 is in feite een klasse, waarvan subclasses afgeleid kunnen worden (bijvoorbeeld de subklasse 'B2B-transaction'). Van elk van deze subclasses wordt verwacht dat ze gekoppeld worden aan subclasses van ServiceProfile, ServiceModel en ServiceGrounding. Merk op dat de DAML-S specificatie geen cardinaliteiten oplegt voor de eigenschappen. De enige beperking is dat er minstens één ondersteunende ServiceGrounding gespecificeerd moet zijn. Ankolekar et al. [25] stelt dat de grounding kritisch is voor het succesvol implementeren van DAML-S, aangezien die zorgt voor de koppeling tussen de semantische beschrijving enerzijds en de beschrijving op lagere niveaus (voornamelijk de beschrijving van de communicatie-faciliteiten door WSDL) anderzijds. DAML-S kan dan ook bekeken worden als een complement van WSDL. Er wordt verwacht dat de bij DAML-S gebruikte groundings zich voornamelijk tot een kleine set van groundings zullen beperken. Als dit effectief zo blijkt te zijn, zullen deze groundings wellicht gespecificeerd worden op welgekende URIs [73].

Eén van de sterke punten van DAML-S zit in de aanwezigheid van een ServiceModel, wat niet terug te vinden is bij andere initiatieven (zoals WSDL bijvoorbeeld). Het ServiceModel biedt mogelijkheden om web-services samen te laten werken en samen te stellen. Daarenboven laat het toe de uitvoering van web-services op te volgen [74]. DAML-S definieert een specifieke subklasse van ServiceModel, namelijk het ProcessModel. Dit ProcessModel omvat twee grote componenten:

- Een *proces* dat de service beschrijft in termen van inputs, outputs, precondities en effecten, en eventueel de subprocessen van het proces. Dit wordt de ‘Process Ontology’ genoemd.
- Een *proces controle model* dat agenten toelaat de uitvoering van een serviceaanvraag op te volgen. Dit wordt de ‘Process Control Ontology’ genoemd.

Zoals eerder gezegd zou het zoeken naar services op basis van de *semantische gelijkheid* tussen een gezochte service enerzijds en aangeboden services anderzijds een krachtige zoekmachine kunnen opleveren. Dit idee wordt ook uitgedragen door onderzoekers van Carnegie Mellon University [41]. Zij baseren zich op DAML-S als taal om web services te beschrijven, als aanvulling voor WSDL. Daarnaast ontwikkelen de onderzoekers ook een *matching engine* om vraag en aanbod te vergelijken in termen van inputs en outputs (de precondities en de effecten worden dus buiten beschouwing gelaten). Bemerkt dat de onderzoekers ervan uitgaan dat het irrealistisch is een perfecte match te vinden: “...*when they describe exactly the same service. This definition is too restrictive, because advertisers and requesters have no prior agreement on how a service is represented; furthermore, they have very different objectives.*” Er wordt bijgevolg gebruik gemaakt van een soort maatstaf die weergeeft in welke mate vraag en aanbod overeenkomen. De onderzoekers zoeken daarnaast naar een mechanisme om te voorkomen dat gewiekste service providers te generieke web-services zouden adverteren, die met veel te veel requests zouden overeenkomen.

3.4 Eindbemerkingen

Terwijl het semantisch web vaak bekeken wordt als ‘*a web of data, in some ways like a global database*’ [76] lijken er ook heel wat mogelijkheden te zijn voor een semantisch web van services. Web-services, eventueel geleverd door onbekende partijen, zullen effectief gezocht en gebruikt kunnen worden via het Internet. De realisatie van een semantisch web van data en diensten zal echter nog veel voeten in de aarde hebben en zal nog heel wat onderzoek vereisen.

Het is duidelijk (en logisch) dat webmasters bestaande zoekmethodes proberen uit te buiten. Zoekmachines die het aantal keer tellen dat een zoekterm op een website voorkomt, kunnen bijvoorbeeld misleid worden door bepaalde woorden honderden keren onzichtbaar op een pagina te plaatsen. Ons inziens kan het semantisch web eveneens te lijden krijgen onder malafide figuren, die ongepaste tags zullen gebruiken op hun website, bijvoorbeeld als marketingmiddel.

Wanneer we deze lijn doortrekken naar een semantisch web van services, is het duidelijk dat er op één of andere manier een waarborg gecreëerd moet worden dat partijen betrouwbaar zijn.

4 De basis voor verder onderzoek

Web-services en het semantisch web beloven veel moois. Momenteel is er nog veel plaats voor grote ontwikkelingen in dit domein. Het ziet er zelfs naar uit dat grote vlakken over het hoofd gezien worden. Eén van die vlakken willen wij verder aankaarten: de realisatie van vraag-gebaseerde web-services.

Enkele constataties uit voorgaande discussie verduidelijken dit:

- We stevenen af op een economie waarin alles zo goed mogelijk gepersonaliseerd wordt. Opvallend hieraan is dat we dit willen bereiken door een web-services technologie, waarbij elk bedrijf zomaar web-services aanbiedt (publiceert in een UDDI-registry), zonder te kijken wat de vraag naar web-services eigenlijk is.
- Het is frappant op te merken dat een aanbieder van web-services geen feedback krijgt over welke services gezocht worden en in welke mate en in welke zin zijn services van gezochte services verschillen. Op de goederenmarkt is men eindelijk tot het inzicht gekomen dat er heel wat kennis over de wensen van de consument verscholen zit in de Supply Chain. In de web-services markt worden wensen van vragers genegeerd. Wellicht kunnen we de les van de goederenmarkt echter doortrekken naar die van de web-services, en kunnen we stellen dat het bedrijf dat de web-services levert die het meest voldoen aan de eisen van de vrager, een groter marktaandeel zal behalen.
- Er mag dan al gesproken worden over een semantisch web van services, nog nergens zien we pogingen opduiken om ook de *inference rules* hier toe te passen. Echter, voor een semantisch web is er niet enkel gestructureerde informatie nodig, maar moet ook geredeneerd kunnen worden. Stel dat een web-service aangeboden wordt waarmee een klant op kan vragen of er nog voorraad is van een specifiek product. Een klant die op zoek is naar alle producten waarvan er voorraad is, is met zo'n service niet geholpen. Echter, het systeem zou zelf moeten kunnen inzien dat het de gevraagde service wel kan leveren, door de aangeboden service te manipuleren.

De creatie van adaptieve systemen kan op deze manier een nieuwe wending en een nieuwe impuls krijgen. Om deze vraag-gebaseerde¹⁰ web-services te realiseren, voorzien we onder andere de volgende noden:

¹⁰ Merk op dat we de term 'vraag-gebaseerde' gebruiken i.p.v. 'klant-gerichte'. De laatste term draagt immers nog steeds de aanbiederende partij in zich als startpunt, terwijl de eerste term duidelijk stelt dat de service gebaseerd is op expliciete eisen van een vragende partij.

- Er moet een taal ontwikkeld worden om web-services-vragen voldoende goed te kunnen formuleren, en om aangeboden services voldoende goed te kunnen beschrijven. Merk op dat aangeboden services eventueel heel generiek kunnen zijn en dus op een heel generiek niveau beschreven kunnen worden.
- Er moet goede feedback gegeven kunnen worden aan service-providers, zodat de personalisatie (automatisch) doorgevoerd kan worden in de systemen van de provider. Bemerkt dat een web-service soms opgebouwd is uit de combinatie van een aantal kleinere web-services, die dan alle in de juiste volgorde opgeroepen moeten worden en correct gepersonaliseerd moeten worden. Een engine kan hier via inference-rules misschien een uitkomst bieden.
- Automatische code-generatie kan nuttig blijken om snel in te spelen op de vraag naar services. Voor dit laatste is het misschien interessant XML te bekijken als een programmeertaal (zoals bijvoorbeeld gebeurt bij de beschrijving van een choreografie van web-services). Programma's in deze taal zouden bijvoorbeeld via XSLT en dergelijke gewijzigd kunnen worden.

5 Conclusie

Een flexibele, goedkope integratietechnologie kan voor vele partijen voordelen opleveren. In dit artikel werden twee recente ICT-ontwikkelingen besproken, web-services en het semantisch web, die samen de nodige technologie zouden kunnen vormen in de toekomst.

In de web-services gemeenschap heerst de mening dat er gestreefd moet worden naar een flexibele manier om web-services te zoeken en op te roepen. Deze web-services moeten daarenboven gemakkelijk gecombineerd kunnen worden tot grotere, sterkere services. De web-services beloftes zijn vooralsnog verre van gerealiseerd. De industrie vertoont evenwel een grote inzet inzake onderzoek en standaardisering. Het ziet er dan ook naar uit dat de web-services droom effectief gerealiseerd zal worden.

De grootste beperking op het gebruik van web-services zit hierin dat de huidige infrastructuur nog steeds een menselijke tussenkomst vergt om web-services te vinden, op te roepen en samen te stellen tot krachtigere services. Willen we tot flexibele geïntegreerde systemen komen, dan moeten software agents deze zaken kunnen regelen.

De integratie van systemen vereist niet enkel een fysieke integratie maar ook een semantische integratie. Het semantisch web is het hoogst nastreefbare inzake de realisatie van semantische interoperabiliteit. De combinatie van het semantisch web en web-services technologie kan een krachtige technologie opleveren voor de flexibele integratie van computersystemen.

Dankbetuiging

Dit artikel werd geschreven in het kader van de SAP-leerstoel van de K.U.Leuven, gefinancierd door SAP-Belgium.

Dank aan Wilfried Lemahieu en Maurice Verhelst voor hun feedback op een eerdere versie van dit document.

Referenties

- [1] Lambrecht Marc (1998), Productie- en Voorraadbeheer, Wolters-Kluwer.
- [2] Sanjay Gosain (2002), EAI: The Business Drivers and Technical Challenges, http://eai.ebizq.net/str/gosain_1a.html.
- [3] Ericson Jim (2001), What The Heck is ERP II?, <http://www.line56.com/articles/default.asp?NewsID=2851>.
- [4] Parker Kevin (2001), Collaborative Manufacturing in the e-Business era, <http://www.manufacturingsystems.com/custompub/0701qad.pdf>.
- [5] Hayes Marc, Partner Relationship Management - The Next Generation of the eCRM Extended Enterprise, http://www.crmproject.com/login.asp?s_id=221&d_ID=783&login=true&mode=print.
- [6] Simchi-Levi Edith, Kaminsky Philip and Simchi-Levi David (1999), Designing and Managing the Supply Chain, pp 321, McGraw-Hill/Irwin.
- [7] Huggins-Chan Sean, EDI, <http://ksi.cpsc.ucalgary.ca/courses/547-95/seanh/edi.html>.
- [8] Kotok Alan (2000); EDI, Warts and All; <http://www.xml.com/lpt/a/1999/08/edi/index2.html>.
- [10] White Andrew (2001), Return on Relationship versus ROI: Relationship Life Cycles and Collaboration, <http://www.cpfr.org/WhitePapers/ReturnonRelationshipvsROI.pdf>.
- [11] Norris Mark (2001), eBusiness essentials - Technology and Network requirements for mobile and online markets, pp 352 Wiley.
- [12] Don Tapscot, David Ticoll, Alex Lowy (2000) Digital Capital: Harnessing the Power of Business Webs, pp 320, Harvard Business School Press.
- [13] Slides 4th European Agent Systems Summer School (Bologna), 8-12 July 2002.
- [14] Ambler Scott (2002), Deriving Web services from UML models, <http://www-106.ibm.com/developerworks/webservices/library/ws-uml1/>.
- [15] DiGenova Ron (2001), Value-chain Optimization: Finding Real Profits Through Collaboration, <http://www.supplychainebusiness.com/archives/12.01.opinion2.htm?adcode=30>.
- [16] Gunjan Samtani (2001), EAI and web-services, <http://www.webservicesarchitect.com/content/articles/samtani01print.asp>.
- [17] Linthicum, David (2000), B2B Application Integration: e-Business-Enable Your Enterprise, pp 464, Addison Wesley.
- [18] Gunjan Samtani (2002), Integration Brokers and Web Services, <http://www.webservicesarchitect.com/content/articles/samtani03print.asp>.

- [19] Patricia Seybold Group (2002), An executive's Guide to Web Services, How to Optimize Web Services Investments to Improve Your Customer Experience, www.psgroup.com.
- [20] W3C (2002), Web Services Architecture Requirements, <http://www.w3.org/TR/2002/WD-wsa-reqs-20020429>.
- [21] Tarak Modi (2001), Clean up your wire protocol with SOAP, http://www.itworld.com/nl/xml_prac/04262001.
- [22] Leymann Frank, Roller Dieter (2002), Business processes in a Web services world, <http://www-106.ibm.com/developerworks/webservices/library/ws-bpelwp/>.
- [23] Fensel Dieter, Bussler C. (2002), The web service modeling framework WSMF, <http://www.cs.vu.nl/~dieter/ftp/paper/wsmf.pdf>.
- [24] Glass Graham (2000), The web services (r)evolution, part 1 - Applying web services, <http://www-106.ibm.com/developerworks/webservices/library/ws-peer1.html>.
- [25] The DAML Services Coalition (2002), DAML-S: web service description for the semantic web, <http://www.daml.org/services/ISWC2002-DAMLS.pdf>.
- [26] Roxburgh Ulrich (2001), BizTalk Orchestration: Transactions, Exceptions, and Debugging, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbiz/html/btsorch.asp>.
- [27] Guy Crets (2002), SAI-workshop Methoden voor Applicatie Integratie.
- [28] Jean-jacques dubray, e-mail communicatie 5 september 2002.
- [29] Jean-jacques dubray, BPML 1.0 Analysis, <http://www.ebpml.org/bpml.htm>.
- [30] Sys-Con Media (2002), UBL and Web Services, <http://www.sys-con.com/xml/articleprint.cfm?id=415>
- [31] ebXML.org, ebXML Industry Support, <http://www.ebxml.org/endorsements.htm>.
- [32] Leymann Frank, Roller Dieter (2002), Business processes in a Web services world, <http://www-106.ibm.com/developerworks/webservices/library/ws-bpelwp/>.
- [33] Rick van der Lans (2002), SAI-workshop, Webservices Hét nieuwe webparadigma.
- [34] W3C (2001) Web Services Description Language WSDL, <http://www.w3.org/TR/wsdl>.
- [35] Martin Gudgin, Ewald Timothy (2002), The IDL that isn't, <http://www.xml.com/lpt/a/2002/01/16/endpoints.html>.
- [36] IBM, <http://www.idealliance.org/papers/xml2001/papers/html/images/03-02-03/Web%20Services%20Stack.jpg>
- [37] http://www.webgain.com/pics/communities/developers/expose_web_services/web_services_framework.gif
- [38] Judith Myerson (2002), Web services architecture - how they stack up, <http://www.webservicesarchitect.com/content/articles/myerson01.asp>.
- [39] Ballinger Keith, Nagy William (2001), The WS-Inspection and UDDI Relationship, <http://www-106.ibm.com/developerworks/webservices/library/ws-wsiluddi.html#resource3>.
- [40] Ballinger Keith, Brittenham Peter, Malhotra Ashok, Nagy William, Pharies Stefan, (2001), Web Services Inspection Language (WS-Inspection) 1.0, <http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html>.
- [41] Paolucci Massimo et al. (2002), Semantic Matching of Web Services Capabilities, <http://www.daml.org/services/ISWC2002-Matchmaker.pdf>.

- [42] Frank Leymann, Roller Dieter (2002), Business processes in a Web services world , <http://www-106.ibm.com/developerworks/webservices/library/ws-bpelwp/>.
- [43] BPMI.org, Business Process Management Initiative, <http://www.bpmi.org/initiative.esp>.
- [44] Arkin Assaf et al. (2002), WSCI Web Service Choreography Interface 1.0, <http://ifr.sap.com/wsci/specification/wsci-spec-10.html#TOC1.3.1>.
- [45] Jean-Jacques Dubray (2002), WSCI, <http://www.ebpml.org/wsci.htm>.
- [46] Vandembulcke Jacques (februari,1998), Met componentensoftware naar de wendbare onderneming, <http://www.sai.be/vbart.htm>, informatie.
- [47] Lindgren Lisa (1998) Application Servers for E-Business, pp 249, Auerbach Publications.
- [48] Merckx Johan – Application Engineers (2002), SAI-cursus, Designing and implementing Components, From Architectural Blueprints to Executable Components.
- [49] Meyer B (2001), Component-based development Gains momentum, <http://www.relativity.com/news/coverage/washpost12-march-01.htm>.
- [50] David Chappell (2001), Presentation PWC-leerstoel K.U.Leuven: Microsoft .NET vs. Java/J2EE: Making the Choice.
- [51] Hanson Jeffrey (2002), .NET versus J2EE Web Services, <http://www.webservicesarchitect.com/content/articles/hanson01.asp>.
- [52] Samtani Gunjan (2002), Web Services and Application Frameworks, <http://www.webservicesarchitect.com/content/articles/samtani04print.asp>.
- [53] Duncan S. (2001), Optimizing Development Productivity Using UML in Oracle9i Jdeveloper, <http://otn.oracle.com/products/jdev/pdf/UMLMdlg.pdf>.
- [54] Merckx Johan (2002), SAI avondconferentie, Van UML Model tot Web Services.
- [55] Ambler Scott (2002), Deriving Web services from UML models, <http://www-106.ibm.com/developerworks/webservices/library/ws-uml1/>.
- [56] Pollock Jeffrey (2002), Dirty Little Secret: It's a Matter of Semantics, http://eai.ebizq.net/str/pollock_2a.html.
- [57] Grand Central Communications, <http://www.grandcentral.com/>
- [58] Bowstreet, <http://www.bowstreet.com/>
- [59] W3C, semantic web, <http://www.w3.org/2001/sw/>
- [60] W3C (2001), Semantic Web Activity Statement, <http://www.w3.org/2001/sw/Activity>.
- [61] Festa Paul (2002), Critics clamor for Web services standards, <http://news.com.com/2100-1023-834990.html>.
- [62] Berners-Lee Tim (2001), The Semantic Web, http://www.scientificamerican.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.
- [63] Levesque H. and Lakemeyer G. (2000), The logic of knowledge bases, The MIT Press, Cambridge.
- [64] Sowa J. (1999), Mathematical background. *Appendix to the book Conceptual structures*. <http://users.bestweb.net/~sowa/misc/mathw.htm>.
- [65] Gruber Tom, What is an Ontology?, <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>.
- [66] Hermans Paul (2002), SAI-avondconferentie, Up to the semantic web.

- [67] Klein M. (2001), Combining and relating ontologies: an analysis of problems and solutions. *Workshop on Ontologies and Information Sharing, IJCAI'01*, Seattle, USA, August 2001. <http://www.cs.vu.nl/~mcaklein/papers/IJCAI01-ws.pdf>.
- [68] Berners-Lee Tim (1998), Why RDF model is different from the XML model, <http://www.w3.org/DesignIssues/RDF-XML.html>.
- [69] Levesque and Brachman (1985), A fundamental tradeoff in knowledge representation and reasoning. In Brachman R. and Levesque H. (Eds), *Readings in knowledge representation*. Los Altos, CA: Morgan Kaufmann.
- [70] Goderis, A. (2002) Reusing problem-solving methods on a pharmacogenomics knowledge base using the IBROW approach. MSc thesis. University of Manchester.
- [71] <http://www.daml.org/>
- [72] Roxane Ouellet, Ogbuji Uche (2002), Introduction to DAML: Part I, <http://www.xml.com/pub/a/2002/01/30/daml1.html>.
- [73] The DAML Services Coalition (2001), DAML-S: Semantic Markup For Web Services, <http://www.daml.org/services/daml-s/2001/10/daml-s.html>.
- [74] NN, DAML related work, <http://www.daml.org/services/daml-s/2001/10/survey-f-release.pdf>.
- [76] Berners-Lee Tim (1998), Semantic Web Road Map, <http://www.w3.org/DesignIssues/Semantic.html>.
- [77] SAP-leerstoel 'Extended Enterprise Infrastructures' <http://www.econ.kuleuven.ac.be/leerstoel/sap>
- [78] Lemahieu Wilfried (2001), Web service description, advertising and discovery: WSDL and beyond; *New Directions in Software Engineering, Liber Amicorum Maurice Verhelst*, Leuven University Press.

Appendix A: Belangrijke Organen

Gezien de snelle evoluties in de web-services wereld is het belangrijk op de hoogte te zijn van de spelers in dit domein.

Het **W3C** (World Wide Web Consortium) is een consortium met 468 leden (waaronder IBM, SUN Microsystems, Microsoft, Oracle e.d.), met aan het hoofd Tim Berners-Lee. Het W3C wil bijdragen tot het realiseren van de mogelijkheden van het Internet door interopererende technologieën te ontwikkelen onder de vorm van specificaties, handleidingen, software en tools. Het W3C houdt zich dan ook niet enkel bezig met web-services, maar eveneens met het semantisch web, XML-gerelateerde zaken, etc.

Momenteel zijn er drie werkgroepen gevormd binnen de web-services activiteiten van het W3C:

- De Web Services Architecture Working Group heeft tot op heden twee Working Drafts gepubliceerd: de Web Services Architecture Requirements en de Web Services Architecture Usage Scenarios.
- De XML Protocol Working Group houdt zich bezig met SOAP specificaties.
- De Web Services Description Working Group concentreert zich op het beschrijven van web-services. Dit beperkt zich in de praktijk tot WSDL specificaties. De door deze werkgroep gespecificeerde talen voor het beschrijven van web-services *moeten* (volgens het charter) echter wel gemapt kunnen worden naar RDF.

Alle gegevens omtrent de web-services activiteiten van het W3C kunnen gevonden worden op <http://www.w3.org/2002/ws/>.

OASIS (Organization for the Advancement of Structured Information Standards) is een consortium dat de ontwikkeling en het gebruik van e-business-standaarden (o.a. ebXML) ondersteunt. De activiteiten van OASIS overschrijden bijgevolg ook de grenzen van het web-services domein. Veel aandacht gaat uit naar XML. OASIS ligt dan ook aan de basis van XML.org (een organisatie die toegang tot informatie omtrent XML gemakkelijker moet maken) en The XML Coverpages (een online referentie voor *interoperable markup language* standaarden). Verdere informatie over OASIS is te vinden op <http://www.oasis-open.org>.

Het **WS-I** (Web Services Interoperability Organization) is een organisatie die heel belangrijk zou kunnen worden in de toekomst. Dit consortium werd in februari 2002 in het leven geroepen door Microsoft, IBM, BEA Systems en Intel om de ontwikkelingen in het web-services domein te versnellen. Publicaties van het WS-I zullen verschijnen op <http://www.ws-i.org>.

Het Business Process Management Initiative (**BPMI**) telt meer dan 120 leden (waaronder IBM, SAP en BEA) en bracht op 26 juni 2002 een public draft uit van de Business Process Modeling Language (BPML) versie 1.0.

DARPA (Defense Advanced Research Projects Agency, <http://www.darpa.mil/>) is een organisatie die onderzoeken coördineert voor het Amerikaanse defensiedepartement. Als dusdanig stond DARPA aan de wieg van het Internet en is DARPA nog steeds op zoek naar nieuwe technologieën. Onder de recente initiatieven vinden we onder meer DAML terug.

Andere interessante initiatieven zijn o.a. te vinden bij EU-IST, webDAV, UDDI.org, WfMC, Rosettanet en EbXML.

Appendix B: Fragment van een BPML-document

Bron: Business Process Modeling Language (BPML), Working Draft 0.4
3/8/2001

```
<process name="TrackTrouble">
  <supports abstract="Customer"/>
  <message name="troubleReportInput" type="request">
    <xsd:element name="service" type="Service"/>
    <xsd:element name="trouble" type="xsd:string"/>
  </message>
  <message name="troubleReportOutput" type="response">
    <xsd:element name="cookie" type="TrackTrouble"/>
  </message>
  <message name="getStatusInput" type="request"/>
  <message name="getStatusOutput" type="response">
    <xsd:element name="status" type="statusCode"/>
  </message>
  <sequence name="reportAndTrack">
    <operation name="reportTrouble">
      <participant name="reportTroubleForm"/>
      <input name="troubleReportInput"/>
      <output name="troubleReportOutput">
        <assign target="cookie" select="TrackTrouble/text()"/>
      </output>
    </operation>
    <operation name="findProvider">
      <participant select="troubleReportInput/service"/>
      <output message="getProviderInput"/>
      <input message="getProviderOutput"/>
    </operation>
    <operation name="createTicket">
      <participant select="getProviderOutput/provider"/>
      <output message="openTicketInput">
        <assign select="troubleReportInput/trouble"/>
        <assign select="TrackTrouble/text()" target="customer"/>
      </output>
      <input message="openTicketOutput"/>
    </operation>
    ...
    <consume name="notifyCustomer">
      <input message="ticketClosed"/>
    </consume>
    ...
  </sequence>
</process>
```

Appendix C: Fragment van een WSFL-document

Bron: <http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>

```
<flowModel name="bookTickets"
  serviceProviderType="airlineFlow">
  <flowSource name="ticketFlowSource">
    <output name="processInstanceData"
      message="tio:receivedTicketOrder"/>
  </flowSource>
  <serviceProvider name="agent" type="agentFlow"/>
  <serviceProvider name="traveler" type="travelerType"/>

  <export lifecycleAction="spawn">
    ...
  </export>
</flowModel>
```

```

</export>
<activity name="reserveSeats">
  <input name="reserveSeatsInput"
    message="tio:receivedTicketOrder"/>
  <output name="reserveSeatsOutput"
    message="tio:reservation"/>
  <performedBy serviceProvider="local"/>
  <implement>
    <internal>
      <!-- .. call to reservation system .. -->
    </internal>
  </implement>
</activity>

<activity name="chargeCreditCard" >
  ...
</activity>

<activity name="confirmFlights" >
  ...
</activity>
<activity name="issueETicket">
  ...
</activity>

<datalink name="gT-rS-data"
  source="ticketFlowSource"
  target="reserveSeats"/>
<controlLink
  name="rS-cC"
  source="reserveSeats"
  target="chargeCreditCard"/>
<dataLink
  name="rS-cCdata"
  source="reserveSeats"
  target="chargeCreditCard"/>
<controlLink
  name="cC-cF"
  source="chargeCreditCard"
  target="confirmFlights"/>
<dataLink
  name="cC-cFdata"
  source="chargeCreditCard"
  target="confirmFlights"/>
  ...
</flowModel>

```

Appendix D: Een SOAP-bericht

Bron: <http://www.w3.org/TR/SOAP/>

```

POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
  ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

```

```

    <SOAP-ENV:Body>
      <m:GetLastTradePrice xmlns:m="Some-URI">
        <symbol>DIS</symbol>
      </m:GetLastTradePrice>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>

```

Appendix E: Een WSDL-document

Bron: http://www.xmethods.net/sd_ibm/CurrencyExchangeService.wsdl

```

<?xml version="1.0" ?>
  <definitions name="CurrencyExchangeService"
    targetNamespace="http://www.xmethods.net/sd/CurrencyExchangeService.wsdl"
    xmlns:tns="http://www.xmethods.net/sd/CurrencyExchangeService.wsdl"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    <message name="getRateRequest">
      <part name="country1" type="xsd:string" />
      <part name="country2" type="xsd:string" />
    </message>
    <message name="getRateResponse">
      <part name="Result" type="xsd:float" />
    </message>
    <portType name="CurrencyExchangePortType">
      <operation name="getRate">
        <input message="getRateRequest" name="getRate" />
        <output message="getRateResponse" name="getRateResponse" />
      </operation>
    </portType>
    <binding name="CurrencyExchangeBinding"
      type="CurrencyExchangePortType">
      <soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http" />
      <operation name="getRate">
        <soap:operation soapAction="" />
      <input>
        <soap:body use="encoded" namespace="urn:xmethods-CurrencyExchange"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output>
        <soap:body use="encoded" namespace="urn:xmethods-CurrencyExchange"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
    </binding>
    <service name="CurrencyExchangeService">
      <documentation>Returns the exchange rate between the two currencies</documentation>
      <port name="CurrencyExchangePort"
        binding="CurrencyExchangeBinding">
        <soap:address location="http://services.xmethods.net:80/soap" />
      </port>
    </service>
  </definitions>

```

Appendix F: Het gebruik van RDF en RDFS

Bron: <http://www.xml.com/lpt/a/2002/01/30/daml1.html>

Met RDF Schema kan een klasse 'Product' als volgt gedefinieerd worden:

```
<rdfs:Class rdf:ID="Product">
  <rdfs:label>Product</rdfs:label>
  <rdfs:comment>An item sold by Super Sports Inc.</rdfs:comment>
</rdfs:Class>
```

De eigenschap 'productNumber' wordt op dezelfde manier gedefinieerd in RDFS:

```
<rdfs:Property rdf:ID="productNumber">
  <rdfs:label>Product Number</rdfs:label>
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Literal"/>
</rdfs:Property>
```

Instances van de gedefinieerde klassen kunnen met bijhorende eigenschappen als volgt weergegeven worden:

```
<Product rdf:ID="WaterBottle">
  <rdfs:label>Water Bottle</rdfs:label>
  <productNumber>38267</productNumber>
</Product>
```

Appendix G: Het gebruik van DAML

Bron: <http://www.xml.com/lpt/a/2002/01/30/daml1.html>

```
<daml:DatatypeProperty rdf:ID="productNumber">
  <rdfs:label>Product Number</rdfs:label>
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger"/>
  <rdf:type
rdf:resource="http://www.w3.org/2001/10/daml+oil#UniqueProperty"/>
</daml:DatatypeProperty>
```