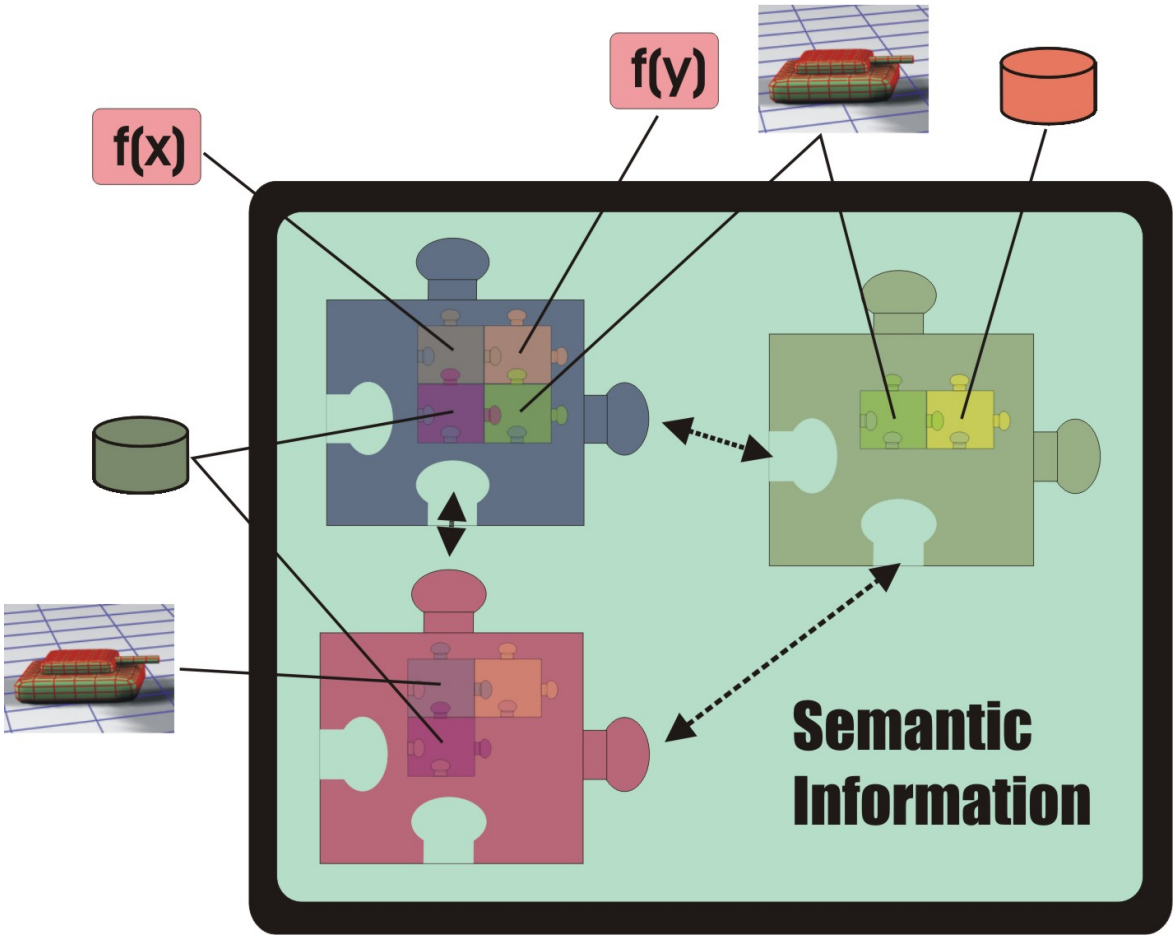
Infrastruktur för informationshantering inom logistikdomänen

Martin Gülich, Martin Eklöf



FOI-R-- 1647 --SE
ISSN 1650-1942

Systemteknik
Metodrapport

April 2005

FOI är en huvudsakligen uppdragsfinansierad myndighet under Försvarsdepartementet. Kärnverksamheten är forskning, metod- och teknikutveckling till nytta för försvar och säkerhet. Organisationen har cirka 1350 anställda varav ungefär 950 är forskare. Detta gör organisationen till Sveriges största forskningsinstitut. FOI ger kunderna tillgång till ledande expertis inom ett stort antal tillämpningsområden såsom säkerhetspolitiska studier och analyser inom försvar och säkerhet, bedömningen av olika typer av hot, system för ledning och hantering av kriser, skydd mot hantering av farliga ämnen, IT-säkerhet och nya sensorers möjligheter.

FOI
Totalförsvarets forskningsinstitut
Systemteknik
164 90 Stockholm

Tel: 08-5550 3000

www.foi.se

Infrastruktur för informationshantering inom logistikdomänen

FOI-R--1647--SE
ISSN 1650-1942

Systemteknik
Metodrapport

April 2005

Utgivare	Rapportnummer, ISRN	Klassificering
FOI - Totalförsvarets Forskningsinstitut Systemteknik 164 90 Stockholm	FOI-R--1647--SE	Metodrapport
	Forskningsområde 2. Operationsanalys, modellering och simulering	
	Månad, år April 2005	**Projektnummer** E6872
	Delområde 23 Logistik	
	Delområde 2	

Författare/redaktör	Projektledare
Martin Gülich Martin Eklöf	Martin Eklöf
	Godkänd av Monica Dahlen
	Uppdragsgivare/kundbeteckning FMV
	Tekniskt och/eller vetenskapligt ansvarig Farshad Moradi

Rapportens titel

Infrastruktur för informationshantering inom logistikdomänen

Sammanfattning (högst 200 ord)

Syftet med detta projekt är att belysa hur semantiska beskrivningar kan främja interoperabilitet, samt beskriva en övergripande infrastruktur för automatiserad informationshantering till stöd för olika logistiska aktiviteter. Informationsmodeller (ontologier) beskriver kunskap på ett formellt sätt, och kan användas för att möjliggöra interoperabilitet mellan applikationer på en semantisk nivå. OWL, *Web Ontology Language*, är ett språk för att skapa sådana informationsmodeller. Semantiska beskrivningar av webbtjänster möjliggör automatisk upptäckt, komposition och exekvering av dessa. OWL-S, *OWL-Services*, är en vokabulär för att skapa sådana beskrivningar. För automatisk upptäck behövs också ett register för webbtjänster som kan utföra sökningar baserade på semantiska beskrivningar, istället för enbart nyckelord. Genom automatisk komposition kan komplexa informationsbehov uppfyllas utan att processer av webbtjänster har skapats i förväg. Automatisk upptäckt och komposition av webbtjänster är två mycket svåra problem, och olika ansatser för att lösa dem har undersökts i denna studie. En infrastruktur för automatiserad informationshämtning från heterogena datakällor har också föreslagits. Datakällorna är däri tillgängliga som semantiskt beskrivna webbtjänster. Kärnan i infrastrukturen utgörs av ett register och en kompositör som automatiskt kan hitta respektive komponera ihop webbtjänster. Ett scenario med en internationell insats har tagits fram, och utifrån detta har en prototyp av infrastrukturen implementerats.

Nyckelord

Semantik, informationshantering, OWL, RDF, webbtjänst, interoperabilitet

Övriga bibliografiska uppgifter	Språk Svenska
ISSN 1650-1942	**Antal sidor:** 56 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Issuing organization	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency Systems Technology SE-164 90 Stockholm	FOI-R--1647--SE	Methodology report
	Programme Areas 2. Operational Research, Modelling and Simulation	
	Month year April 2005	**Project no.** E6872
	Subcategories 23 Logistics	
	Subcategories 2	

Author/s (editor/s)	Project manager
Martin Gülich Martin Eklöf	Martin Eklöf
	Approved by Monica Dahlen
	Sponsoring agency FMV
	Scientifically and technically responsible Farshad Moradi

Report title (In translation)

An infrastructure for information management in the logistics domain

Abstract (not more than 200 words)

The aim of this study is to investigate how semantic descriptions can facilitate interoperability, and to suggest an overall infrastructure for automatic information management in support of different logistic activities. Information models (ontologies) describe knowledge in a formal way, and can be used to enable interoperability between applications on a semantic level. OWL, *Web Ontology Language*, is a language in which such information models can be created. Semantic descriptions of web services enable automatic discovery, composition and execution. OWL-S, *OWL-Services*, is a vocabulary in which semantic descriptions of web services can be created. To enable automatic discovery a registry for web services that can perform searches based on semantic descriptions, and not only key words, is also needed. Complex information needs can be met through automatic composition, without having to pre-define processes of web services. Automatic discovery and composition are two very difficult problems, and different approaches to solve them have been investigated in this study. An infrastructure for automatic information gathering from heterogeneous data sources has also been suggested. The data sources are available as semantic web services in this. The core consists of a registry and a composer that can automatically discover and compose web services. A scenario with an international operation has been developed, and based on this a prototype of the infrastructure has been developed.

Keywords

Semantics, information management, OWL, RDF, web service, interoperability

Further bibliographic information	Language Swedish
ISSN 1650-1942	**Pages** 56 p.
	Price acc. to pricelist

Sammanfattning

Inom det nätverksbaserade försvaret (NBF) kommer bl.a. modellering och simulering (M&S) att vara av avgörande betydelse. M&S ställer ofta krav på tillgång till data av varierande slag. Syftet med detta projekt är att beskriva en övergripande infrastruktur för automatiserad informationshantering till stöd för olika logistiska aktiviteter, däribland M&S. Denna infrastruktur bör bygga på IT-baserade tjänster med metadatabeskrivningar, något som också förespråkas i projektet IRM Vision [22]. Arbetet ska också belysa hur semantiska beskrivningar kan främja interoperabilitet och automatiserad informationshantering.

Vid informationshantering kommer i framtiden en tjänstebaserad infrastruktur, till vilken olika system kan anslutas, att behövas för att kunna hantera en ständigt föränderlig omvärld, där anpassningsförmåga krävs. En sådan infrastruktur bör främja interoperabilitet och samtidigt stödja en hög grad av automatisering. IT-baserade tjänster, t.ex. webbtjänster, åstadkommer en viss interoperabilitet genom att vara plattformsoberoende. Dock krävs även interoperabilitet på semantisk nivå för att olika system till fullo ska kunna utnyttja och återanvända komponenter i andra system. Semantiska metadatabeskrivningar kan användas för att åstadkomma interoperabilitet även på denna nivå. Dessa beskrivningar kan också beskriva tjänster utifrån ett visst sammanhang så att maskiner automatiskt kan tolka hur en tjänst är tänkt att användas, inte bara vad den gör. Detta möjliggör en hög grad av automatisering.

Ett förslag till en infrastruktur där automatisk informationshämtning från heterogena datakällor kan ske, har tagits fram i denna studie. Denna infrastruktur baseras på semantiskt beskrivna webbtjänster. Datakällorna kan utgöras av exempelvis lagersystem eller databaser. Genom att göra datakällorna åtkomliga via semantiska webbtjänster kan ett godtyckligt system lokalisera en datakälla och interagera med den, och på så sätt automatiskt hämta information därifrån. Nödvändiga komponenter för att skapa en sådan infrastruktur har identifierats, och de aktuella forskningsområdena har överblickats för att se hur dessa komponenter kan skapas.

Ett scenario med en internationell operation, baserat på projekt VSHMOD-UH-2010, Fas 2 Förnödenhetsförsörjning [23], har också tagits fram för att visa på användbarheten av den föreslagna infrastrukturen. Scenariot beskriver en FN-insats, där Sverige bidrar med en samordnande logistiktrupp. Till sin hjälp har denna trupp ett avancerat logistiksystem som automatiskt kan interagera med de övriga ländernas logistiksystem, och därigenom få en bild av alla tillgängliga förnödenhetstillgångar, oavsett vilket land de tillhör. Utifrån scenariot har också en prototyp av infrastrukturen implementerats.

Innehållsförteckning

1 Introduktion

1.1 Syfte

Visionen av ett framtida logistiskt system inom det nätverksbaserade försvaret (NBF) omfattar tjänster som underlättar verkställandet av strategiska, operativa och taktiska mål. Vid verkställandet av dessa mål kommer modellering och simulering (M&S), resursledningssystem (RLS), TAV-system (Total Asset Visibility) etc., att vara av avgörande betydelse. M&S och andra system inom den logistiska domänen ställer ofta krav på tillgång till data av varierande slag. Det finns idag en stor mängd data att tillgå, men skilda lagringssätt, inkonsekvent semantik och syntax etc., gör det svårt att utnyttja denna. För att kunna tillfredsställa kraven på data och information, som informationskonsumerande komponenter och processer har, krävs således att data kan integreras automatiskt från heterogena datakällor.

Syftet med detta projekt är att beskriva en övergripande infrastruktur för automatiserad informationshantering till stöd för olika logistiska aktiviteter, t.ex. M&S. Arbetet ska också belysa hur semantiska beskrivningar och användandet av informationsmodeller inom en specifik domän kan främja interoperabilitet och automatiserad informationshantering. Mer konkret ska det undersökas hur dynamisk och automatiserad informationshämtning från heterogena datakällor kan implementeras för att tillgodose informationsbehov hos t.ex. simuleringar, samt vilken betydelse semantiska beskrivningar har vid detta förfarande. En infrastruktur baserad på tjänster och semantiska informationsmodeller, i vilken sådan informationshämtning är möjlig, ska beskrivas och nödvändiga komponenter identifieras. Aktuella forskningsområden ska sedan ses över för att undersöka hur dessa komponenter kan skapas. En prototyp av infrastrukturen ska också implementeras för att visa användbarheten och för att erhålla mer konkreta erfarenheter.

1.2 Begreppsdefinitioner

- *Tjänst,* någon form av funktionalitet som erbjuds. En tjänst kan exempelvis vara tillgänglig elektroniskt över ett nätverk, eller manuellt i en reception.
- *Webbtjänst*, en återanvändbar programmodul, tillgänglig i ett nätverk, som realiserar en tjänst. I avsnitt som handlar om webbtjänster kommer *tjänst* och *webbtjänst* att användas synonymt.
- *Data,* information ifrån en datakälla, t.ex. dokument ur en databas. *Information* används här ibland synonymt med *data*.
- *Domän,* ett avskilt användnings- eller tillämpningsområde, t.ex. logistik.
- *Informationsmodell,* även kallad *ontologi*, en formell och explicit definition av olika begrepps innebörd, och relationer mellan olika begrepp.

1.3 Metod

Detta projekt har den förstudie som genomfördes under 2003 [1] som utgångspunkt. I förstudien inventerades nya metoder och möjligheter för integrering av information från heterogena datakällor. Studien fokuserade på en tjänstebaserad arkitektur där IT-baserade tjänster tillhandahåller information och annan funktionalitet. Främst undersöktes standarder

och ramverk från forskningsområdet ”den semantiska webben”. Inom detta område är tjänstebegreppet centralt. Tjänster förespråkas också i rapporten *IRM VISION* [2], där IT-baserade tjänster med metadatabeskrivningar anses vara en kritisk komponent i det framtida försvaret. En tjänstebaserad arkitektur motiveras av den ständigt föränderliga omvärld i vilken det framtida försvaret kommer att verka. Flexibilitet och anpassningsförmåga möjliggörs av en dynamisk infrastruktur för informationshantering där ny funktionalitet enkelt kan tillföras, upptäckas och användas. Minimalt beroende av en specifik teknologi är också önskvärt. Webbtjänster kan utgöra stommen i en sådan infrastruktur, eftersom de tillhandahåller funktionalitet på ett enhetligt vis, oberoende av plattform, samtidigt som de kan uppdateras och tillföras eller tas bort kontinuerligt.

Baserat på förstudien undersöktes först olika språk för att skapa formella informationsmodeller. Sedan undersöktes hur vanliga webbtjänster kan utökas med semantiska beskrivningar, kopplade till begrepp i globala informationsmodeller, för att möjliggöra interoperabilitet på den semantiska nivån. Nödvändiga komponenter i en infrastruktur för information baserad på semantiskt beskrivna webbtjänster undersöktes också, och ett förslag på en sådan infrastruktur togs fram. Till sist beskrevs och realiserades ett scenario där semantiska webbtjänster användes. En teoretisk jämförelse med en realisering av samma scenario med vanliga webbtjänster utfördes också för att visa användbarheten av semantiska webbtjänster.

1.4 Rapportens struktur

I avsnitt 2, *Bakgrund*, ges först en genomgång av språk för semantiska beskrivningar. Sedan ges en genomgång av webbtjänster och framförallt semantiska webbtjänster. Det mesta gällande semantiska webbtjänster gås igenom; integrering av informationsmodeller, publicering och lokalisering, komposition samt exekvering. Aktuella forskningsområden för dessa frågor överblickas också. Detta avsnitt bör läsas av den som vill skaffa sig kunskaper om semantiska beskrivningar, informationsmodeller och semantiska webbtjänster, samt av den som vill se vilka problem som måste lösas för att semantiska webbtjänster ska komma till praktisk användning.

I avsnitt 3, *Förslag på infrastruktur*, ges ett förslag på en infrastruktur för att möjliggöra automatisk inhämtning av information från heterogena datakällor, och nödvändiga komponenter beskrivs. Till sist beskrivs ett scenario som tillämpning av infrastrukturen. Detta avsnitt bör läsas för att se hur en infrastruktur för informationshantering med hjälp av semantiska webbtjänster kan se ut, samt hur den kan användas.

I avsnitt 4, *Realisering av scenario*, beskrivs hur scenariot realiserats genom att en prototyp av infrastrukturen med nödvändiga komponenter har implementerats.

I avsnitt 5, *Diskussion*, förs en diskussion kring nyttan av semantiska beskrivningar vid informationshantering.

Avsnitt 2 och 4 innehåller förhållandevis detaljerad information och är därför uppdelade i två separata delar. För att få en överblick över ämnet som tas upp rekommenderas att endast läsa den första delen. Om mer detaljerade kunskaper eftersträvas bör även den andra delen läsas.

2 Bakgrund

2.1 Språk för informationsmodeller

Encyclopedia Britannica (www.brittanica.com) definierar interoperabilitet som *”ability of a system (as a weapons system) to use the parts or equipment of another system”*. Detta innebär att de två systemen måste kunna förstå varandra på syntaktisk, semantisk och pragmatisk nivå. På samma sätt som grammatik skapar en gemensam syntax, skapar informationsmodeller en gemensam semantik mellan system. Exempelvis så kan datasträngen ’sten’ betyda olika saker i ett geologisystem och ett namnsdagssystem. Genom att ha en informationsmodell som beskriver begreppet ’Namn’, och genom att koppla datasträngen ’sten’ till detta begrepp skapas en gemensam förståelse för vad ’sten’ representerar. En informationsmodell kan med andra ord ses som ett gemensamt schema som systemen måste följa. Väldefinierade och brett accepterade sätt att beskriva data syntaktiskt finns redan, t.ex. XML. Några olika språk för att beskriva data semantiskt genom att skapa formella informationsmodeller beskrivs nedan. Interoperabilitet på den pragmatiska nivån tas inte upp här.

2.2 RDF / RDFS

RDF [29], *Resource Description Framework*, tillhandahåller ett gemensamt ramverk för att uttrycka metadata om resurser, så att den kan utbytas mellan applikationer utan att betydelsen går förlorad. Det fundamentala konceptet bakom RDF är att alla resurser kan beskrivas med s.k. triplar. Varje trippel är en utsaga baserad på tre delar av information; objekt, predikat och subjekt. Ett objekt kan relateras till ett subjekt via ett predikat och på så sätt uttrycks information om objektet, t.ex. *Kalle ålder 25*. Det är viktigt att poängtera att RDF endast beskriver resurser. Det är upp till enskilda applikationer att använda den metadata som uttrycks.

RDFS, [30] *RDF Schema*, är en utökning av RDF med möjlighet att uttrycka t.ex. arv mellan klasser och på så sätt beskriva hierarkier. En utförligare beskrivning av både RDF och RDFS ges i förstudien [1].

2.3 OWL

2.3.1 Översikt

OWL [31], *Web Ontology Language*, är ett språk för att publicera och dela informationsmodeller (ontologier), och ingår i W3Cs[1] program för den semantiska webben. Språket är tänkt att användas av applikationer som behöver kunna tolka innehållet av information istället för att bara presentera det. OWL är kraftfullt och ger större möjlighet till maskinell tolkning av information än XML, RDF och RDFS, genom att tillhandahålla en större vokabulär med formell semantik. Det finns tre subtyper av OWL med ökande uttrycksfullhet; OWL Lite, OWL DL och OWL Full. Varje beskrivning i OWL Lite är en giltig beskrivning i OWL DL, men inte tvärtom, och varje beskrivning i OWL DL är en giltig beskrivning i OWL Full, men inte tvärtom.

[1] Den semantiska webben (The Semantic Web) är tänkt att tillhandahålla ett gemensamt ramwerk för att dela och återanvända data mellan applikationer och organisationer. Det är en satsning som leds av W3C, *World Wide Web Consortium*, där både forskare och industripartners ingår.

OWL har DAML+OIL [34] som föregångare och bygger på RDF och RDFS. Eftersom det bygger på RDF är informationsmodeller skapade i OWL skalbara, kompatibla med webbstandarder, öppna och möjliga att utöka, samtidigt som de kan vara distribuerade över flera system. Den större vokabulären möjliggör rikare och mer uttrycksfulla beskrivningar av klasser, relationer och egenskaper. I språket ingår *klasser*, *individer* och *relationer*. Individer motsvarar enstaka instanser, t.ex. en bil eller ett äpple. Klasser består av mängder av individer. Relationer kan finnas mellan antingen klasser, individer eller båda. Restriktioner på hur dessa språkstrukturer får användas i semantiska beskrivningar beror på vilket subspråk av OWL som används. Nedan följer några exempel på formella beskrivningar som går att uttrycka i OWL, men inte i RDF/RDFS:

- Ange att två klasser är disjunkta, dvs. att de inte har några gemensamma individer.
- Ange kardinalitet för relationer, dvs. hur många olika relationer av samma typ som en klass eller en individ får ingå i.
- Ange att relationer är symmetriska, transitiva eller inverser av varandra. *brorTill* är ett exempel på en transitiv relation, eftersom en brors bror även är bror med den första.
- Ange att två olika klasser är ekvivalenta, dvs. att de har samma individer.

Eftersom OWL bygger på RDF brukar OWL-beskrivningar skrivas med hjälp av RDF/XML-notationen. Nedan följer ett exempel på denna notation. I figur 2-1 definieras en OWL-individ som hör till klassen *Country*. Individen får namnet *Karundia*, och anges vara en region i *Africa*. Med OWL är det nu alltså beskrivit att landet *Karundia* ligger i *Africa*.

```
<Country ID="Karundia">
   <subRegionOf resource="#Africa"/>
</Country>
```

Figur 2-1
Exempel på OWL med RDF/XML-notation.

OWL ger också möjlighet till kraftfull *resonering* (reasoning). Med resonering menas här processen att dra logiska slutsatser utifrån den information som finns i formella beskrivningar, och därmed härleda mer information än vad som explicit är uttryckt. Ett exempel på detta kan vara en informationsmodell för släktforskning i vilken det står att Kalle är pappa till Pelle, och att Pelle är pappa till Olle. Om det också är formellt uttryckt att en pappas pappa är en farfar, kan det med resonering härledas att Kalle är Olles farfar, trots att detta inte uttryckligen finns beskrivet någonstans.

2.3.2 Detaljer

Subspråket OWL DL har en väldefinierad koppling till en viss typ av formell logik, *Description Logic (DL)*. Eftersom det är en formell logik kan resonering ske automatiskt med en s.k. resonerare (reasoner). Det är för detta ändamål som OWL DL och OWL Full är två skilda subspråk. De innehåller samma språkliga strukturer, men i OWL DL finns det restriktioner på hur de får användas, vilket gör att OWL DL är mindre uttrycksfullt än OWL Full. Anledningen är att resoneringen ska bli enklare och möjlig att beräkna, så att automatiska resonerare kan användas.

En ny klass kan definieras genom att villkor som måste uppfyllas för att en individ ska anses tillhöra klassen anges. Villkoren utgörs av restriktioner på relationer till andra klasser eller

individer. Ett villkor som definierar klassen *Förälder* kan till exempel vara en restriktion på relationen *harBarn*, som säger att den ska ha kardinalitet större än ett, > 1. Kardinalitet större än ett innebär att en individ måste ingå i minst en relation av typen *harBarn* för att anses tillhöra klassen *Förälder*.

Fundamentalt för resonering i OWL är möjligheten att beskriva både *primitiva* klasser och *definierade* klasser. Detta är fundamentalt eftersom OWL tillämpar ett antagande om en *öppen värld*. Antagandet om en öppen värld innebär att bara för att något inte beskrivs i de tillgängliga beskrivningarna så behöver det inte vara falskt. Till exempel kan en mamma mycket väl också vara en pappa, så länge det inte står någonstans att detta inte är fallet. Med enbart primitiva klasser kan klassen som en individ tillhör inte automatiskt härledas, dvs. exakt vilken typ av äpple som finns i fruktkorgen kan inte tas reda på om äppelklassen är en primitiv klass. En definierad klass tillåter däremot sådan klassificering. Detta gör OWL mycket flexibelt och kraftfullt eftersom nya klasser då kan definieras genom att enbart ange villkor. Relationerna mellan en ny klass och de redan existerande klasserna behöver alltså inte beskrivas vilket gör skapandet av nya klasser enklare. Med hjälp av resonering kan sedan alla individer klassificeras automatiskt och de som tillhör den nya klassen kan sorteras ut. Detta förfarande möjliggör även dynamiska och kraftfulla sökningar bland de beskrivna individerna. I t.ex. UML [35] och många andra modelleringsspråk kan enbart primitiva klasser beskrivas. Dessa språk erbjuder därför inte samma möjligheter till utökning och sökning med hjälp av resonering.

Resonering är inte bara användbart utan också nödvändigt vid skapandet av informationsmodeller. En informationsmodell kan enbart utökas, genom att nya beskrivningar läggs till då begrepp förfinas etc. Eftersom inga beskrivningar tas bort måste informationsmodellen kontrolleras allteftersom nya beskrivningar tillkommer så att inga motsägelser förekommer. För om så är fallet går det inte längre att tolka beskrivningarna entydigt, och maskiner kan då inte tolka informationen automatiskt med hjälp av informationsmodellen. Dessa kontroller blir snabbt för komplexa för att i praktiken kunna utföras för hand.

2.4 OWL-Services

2.4.1 Översikt

OWL-S [32], *OWL-Services*, är en OWL-baserad informationsmodell som tillhandahåller en formell vokabulär för att skapa semantiska beskrivningar av enskilda webbtjänster. OWL-S skapades som en vidareutveckling av DAML-S [28], och är tänkt att möjliggöra följande fyra uppgifter, utan att föreskriva något specifikt sätt att lösa dem:

1- Automatisk lokalisering av webbtjänster.

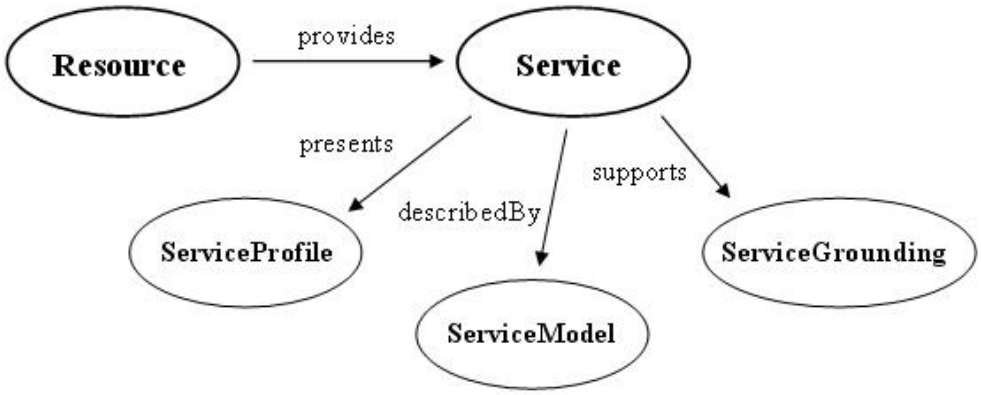
2- Automatisk komposition av och interaktion mellan webbtjänster.

3- Automatisk exekvering av webbtjänster.

4- Automatisk kontroll av exekvering av webbtjänster.

OWL-S består av tre delar, se figur 2-2. En nätbaserad resurs tillhandahåller en tjänst med en semantisk beskrivning som består av en *tjänsteprofil*, en *tjänstemodell* och en *tjänsteförankring (*grounding). Profilen beskriver vad tjänsten kräver och vad den tillhandahåller. Modellen beskriver hur tjänsten fungerar. Förankringen beskriver hur tjänsten

ska användas, genom att förankra den till ett språk för exekvering av webbtjänster, oftast WDSL [36]. Det får finnas max en tjänstemodell, men det kan finnas flera profiler och förankringar. Detta eftersom tjänsten kan presenteras på olika sätt, och förankras till olika språk för exekvering av webbtjänster, men den fungerar hela tiden likadant.



Figur 2-2

Översta nivån i OWL-S. En resurs tillhandahåller en webbtjänst som beskrivs av en tjänsteprofil, en tjänstemodell och en tjänsteförankring. Se figur i [32].

Tjänsteprofilen beskriver en tjänst utifrån vem som tillhandahåller den, vad den utför eller beräknar, samt statiska egenskaper som karaktäriserar den. Den funktionella beskrivningen av vad en tjänst utför består av indata och utdata, samt startvillkor och effekter. Exempelvis kan en försäljningstjänst ha kortnummer och utgångsdatum som indata samt att kortnumret ska vara giltigt som startvillkor. Utdata kan vara ett kvitto och effekten kan vara att priset debiteras kortet. Icke-funktionella egenskaper som karaktäriserar en tjänst kan vara vilken QoS tjänsten har, hur lång tid den tar att exekvera etc.

Tjänstemodellen ger detaljerad information om hur en tjänst fungerar. En webbtjänst kan vara mer eller mindre komplex. Komplexa webbtjänster kan beskrivas som processer av andra enklare webbtjänster. Följande tre typer av processer finns,

1- *AtomicProcess*, dessa är atomära och representerar en enkel operation som kan exekveras direkt och i ett enda steg. Varje atomär process har en egen förankring till en implementation av en webbtjänst.

2- *CompositeProcess*, dessa är sammansatta av flera atomära och/eller komposita processer och kan alltså inte exekveras direkt, utan varje ingående atomär process måste exekveras var för sig.

3- *SimpleProcess*, dessa ser utifrån ut som atomära processer, men de har ingen förankring. Syftet är istället att användas som en abstrakt vy av en mer eller mindre komplicerad tjänst. På detta sätt är det möjligt att beskriva hur tjänster fungerar med olika upplösning, och att gömma detaljer.

Förankringar beskriver hur en tjänst kan realiseras genom att beskriva detaljer kring protokoll, meddelandeformat och transport av meddelanden. Varje förankring kan ses som en mappning mellan en abstrakt beskrivning och en konkret realisering. Varje atomär process är en sådan abstrakt beskrivning, och motsvaras av en konkret realisering i en implementering av webbtjänsten**.**

2.4.1 Detaljer

En tjänsteprofil föreskriver inte någon slutgiltig representation av tjänster, eftersom det alltid går att skapa mer specialiserade profiler som subklasser på vanligt vis i OWL. En möjlig representation tillhandahålls dock av OWL-S 1.1 genom klassen *Profile*, som är en subklass till *ServiceProfile*. Denna klass beskriver en tjänst utifrån vem som tillhandahåller den, vad den utför eller beräknar, samt statiska egenskaper som karaktäriserar den. Den funktionella beskrivningen av vad en tjänst utför består av indata och utdata, samt startvillkor och effekter (IOPE – Inputs, Outputs, Preconditions and Effects). Indata och utdata beskriver vad tjänsten beräknar, och effekter beskriver vilken förändring i omvärlden tjänsten åstadkommer. I figur 2-3 visas ett exempel på en profil för en enkel webbtjänst som adderar två tal:

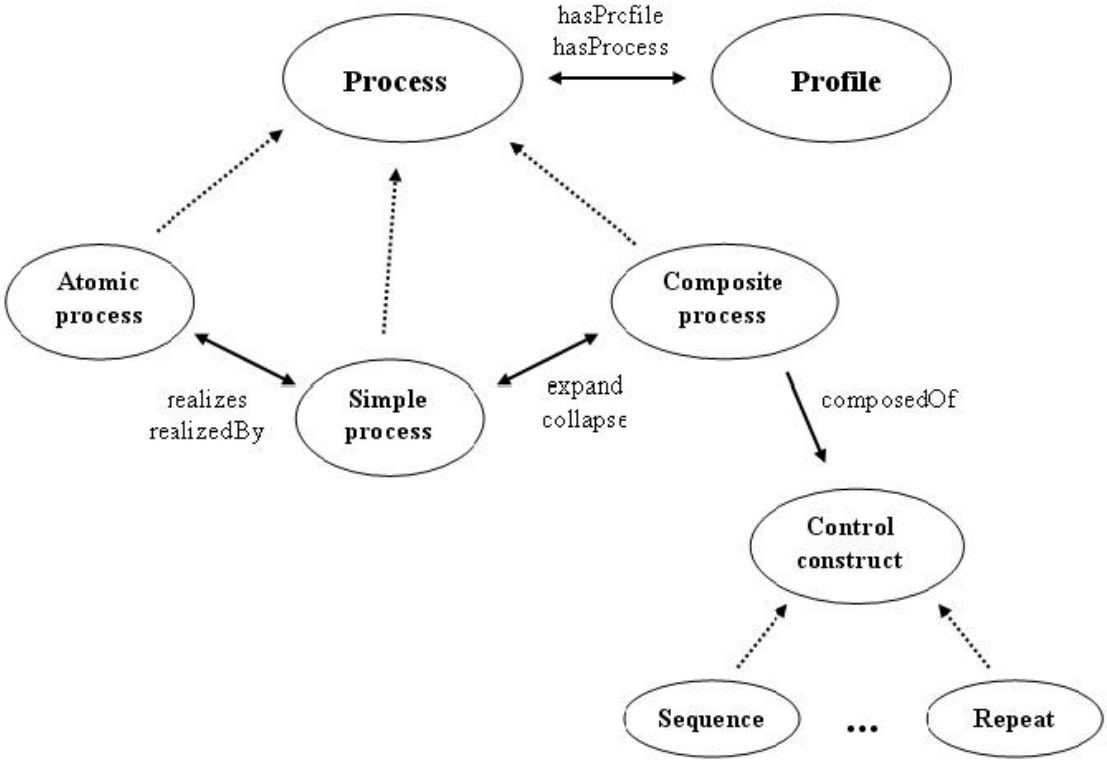
```
<Profile ID="AddProfile">
   <presentedBy resource="#AddService"/>
   <serviceName>
      Adder2000
   </serviceName>
   <textDescription>
      Adds two terms into a sum.
   </textDescription>
   <contactInformation>
      #An_Organization
   </contactInformation>
   <hasInput resource="#Term1"/>
   <hasInput resource="#Term2"/>
   <hasOutput resource="#Sum"/>
</Profile>
```

Figur 2-3

En tjänsteprofil för en webbtjänst som adderar två tal.

En tjänstemodell ger detaljerad information om hur en tjänst fungerar, och kan ses som en process. Klassen *ProcessModel* i OWL-S 1.1 är en subklass till *ServiceModel*, och används för att representera en tjänstemodell. Alla processtyperna beskrivs med indata, utdata, startvillkor och effekter. Till skillnad från tjänsteprofilen, där dessa endast används för att presentera tjänsten, används de här för att beskriva logiken i tjänsten. Komposita processer beskrivs därför med kontrollstrukturer som beskriver exekveringsflödet i processen, se figur 2-4. Exempel på kontrollstrukturer är:

- *Sequence*, en lista på processer som ska utföra i ordning.
- *Split*, en lista på processer som kan utföras parallellt.
- *If-Then-Else*, väljer olika processer beroende på om villkor är sanna eller falska.
- *Repeat*, upprepar en eller flera processer.



Figur 2-4

Processmodell i OWL-S. En process kan vara atomär, enkel eller komposit. En komposit process består av kontrollstrukturer, t.ex. Sequence och Repeat. Se figur i [32].

Kontrollstrukturer kan användas dels för att beskriva ett dataflöde i processen, och dels för att sköta exekveringen av en komplicerad tjänst. Atomära processer beskrivs inte med kontrollstrukturer, eftersom de endast består av ett steg. I figur 2-5 visas ett exempel på en komposit process för en enkel webbtjänst som adderar två tal:

```
<CompositeProcess ID="AddCompositeProcess">
   <composedOf>
      <Sequence>
         <components>
            <AtomicProcess about=
                  "#AddAtomicProcess"/>
            <AtomicProcess about=
                  "#PrintAtomicProcess"/>
         </components>
      </Sequence>
   </composedOf>
   <hasInput resource="#Term1"/>
   <hasInput resource="#Term2"/>
   <hasOutput resource="#Sum"/>
</CompositeProcess>
```

Figur 2-5

En komposit process för en webbtjänst som adderar två tal och skriver ut summan.

Klassen *WsdlAtomicProcessGrounding* i OWL-S 1.1, som är en subklass till *ServiceGrounding*, används för att förankra en tjänstebeskrivning i OWL-S till WSDL. Varje atomär process måste förankras till en konkret implementation av webbtjänsten**.** I figur 2-6 visas ett förenklat exempel på en förankring av en enkel webbtjänst som adderar två tal från

OWL-S till WSDL. Varje in- eller utdata i OWL-S mappas till motsvarande in- eller utdata i WSDL:

```
<WsdlAtomicProcessGrounding ID="AddAtomicProcessGrounding">
    <owlsProcess resource="#AddAtomicProcess"/>
    <wsdlDocument>
        &wdsl_doc;          // Förkortn. av t.ex. http://mitt_WSDL_dokument.xml
    </wsdlDocument>
    <wsdlInputMessageParts>
        <wsdlMessageMap>
            <owlsParameter resource="#Term1"/>
    <wsdlMessagePart>
                &wsdl_doc;#realInput1
                </wsdlMessagePart>
        </wsdlMessageMap>
        <wsdlMessageMap>
            <owlsParameter resource="#Term2"/>
            <wsdlMessagePart>
                &wsdl_doc;#realInput2
            </wsdlMessagePart>
        </wsdlMessageMap>
    </wsdlInputMessageParts>
    <wsdlOutputMessageParts>
        <wsdlMessageMap>
            <owlsParameter resource="#Sum"/>
            <wsdlMessagePart>
                &wsdl_doc;#realOutput
            </wsdlMessagePart>
        </wsdlMessageMap>
    </wsdlOutputMessageParts>
</WsdlAtomicProcessGrounding>
```

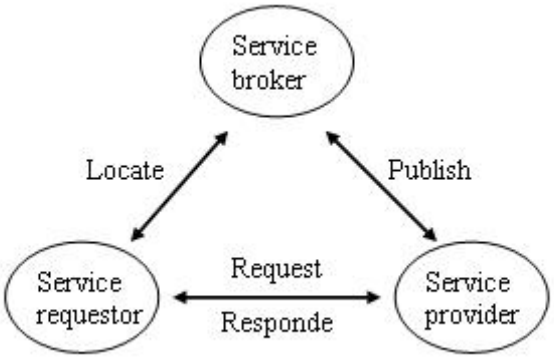
Figur 2-6

Del av förankring till WSDL av en atomär process för en webbtjänst som adderar två tal.

2.5 Webbtjänster

Webbtjänster (Web Services) är väldefinierade och återanvändbara mjukvarukomponenter som utför specifika uppgifter och är tillgängliga via standardiserade nätverksmekanismer. Enkla webbtjänster kan utgöras av korta program eller funktioner, medan mer komplexa webbtjänster kan utgöras av en process av flera enklare webbtjänster.

Det grundläggande konceptet för hur en webbtjänst fungerar visas i figur 2-7. En *service provider* är ansvarig för utveckling, spridning och tillhandahållande av en tjänst. En s*ervice broker* förmedlar kontakten mellan en webbtjänst och en användare genom att ansvara för registrering och lokalisering av tjänsten. Varje service provider registrerar sin tjänst hos en service broker. En användare, *service requestor*, lokaliserar till sist en tjänst via en service broker, men exekverar den hos dess service provider.



Figur 2-7

En service provider registrerar en webbtjänst hos en service broker. En service requestor kontaktar sedan denna för att lokalisera webbtjänsten. Brokern talar om för requestorn var den finns. Requestorn kommunicerar sedan direkt med providern för att exekvera webbtjänsten.

Web Service Description Language, WSDL [36], är ett språk för tekniska metadata-beskrivningar av webbtjänster. Sådana beskrivningar är tillräckligt utförliga för att en användare ska kunna komma åt en tjänst och exekvera den.

Simple Object Access Protocol, SOAP [37], är det grundläggande meddelandeprotokoll som används vid kommunikation mellan bl.a. en service requestor och en service provider.

Universal Description, Discovery and Integration protocol, UDDI [38], är en standardiserad metod för att publicera och lokalisera webbtjänster i register. Service brokern består därför bland annat av ett UDDI-register.

En utförligare beskrivning av webbtjänster finns i förstudien [1].

2.6 Semantiska webbtjänster

Vanliga webbtjänster kan samarbeta och förstå strukturen på meddelanden som utbyts, men de kan inte förstå innehållet. Det finns i sådana tjänster statiska beskrivningar av vilka andra tjänster de kan samarbeta med och på vilket sätt. För att konfigurera om en webbtjänst krävs dessutom en manuell insats av en programmerare. Resultatet är statiska webbtjänster som kan vara svåra att anpassa till förändringar. Semantiska webbtjänster kan tack vare sina metadatabeskrivningar användas mer dynamiskt, med möjlighet att interagera autonomt och att konfigurera om sig själva utan manuellt ingripande, exempelvis vid återhämtning efter fel. Semantiska webbtjänster kan därmed automatiskt anpassa sig till förändringar. Om till exempel den partnertjänst som en tjänst samarbetar med blir otillgänglig kan en ny partner automatiskt lokaliseras, eller om en ny och bättre partner blir tillgänglig kan denna användas istället.

Semantiska beskrivningar av tjänster är det som möjliggör autonomt interagerande. För att automatiskt kunna avgöra vad som är en passande partner, eller avgöra hur bra en annan tjänst är, krävs en formell och explicit beskrivning av dess förmågor, samt att beskrivningen görs tillgänglig för andra tjänster. Den semantiska webben [14] är ett nätverk där information har en väldefinierad betydelse. Betydelsen beskrivs på semantisk nivå av ett formellt språk. Därmed kan information automatiskt tolkas och användas på ett enhetligt sätt. Det krävs också en infrastruktur för att förmedla beskrivningar av tjänster och information. Denna

möjliggör också för maskiner att med hjälp av inferens dra egna slutsatser kring den information som finns tillgänglig.

I det här avsnittet undersöks vad som krävs för att skapa ett semantiskt nätverk. En del förslag till lösningar inom nödvändiga områden överblickas också.

2.6.1 Semantiska beskrivningar

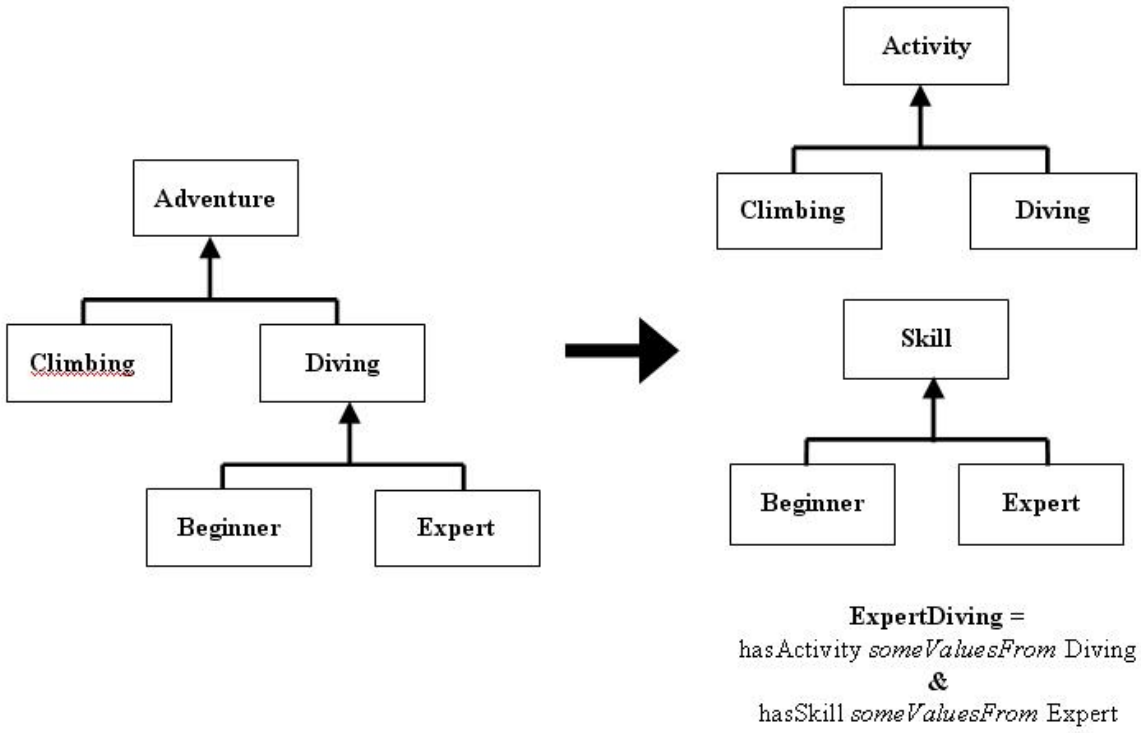
Vanliga webbtjänster beskrivs av metadata uttryckt i WSDL. Sådan metadata är dock inte tillräckligt uttrycksfull för de beskrivningar som behövs till en semantisk webbtjänst. För dessa behövs ett mer kraftfullt språk, i vilket informationsmodeller för beskrivning av själva tjänsten och för beskrivning av de domänspecifika begrepp som används, kan skapas. För webbtjänster behövs också ett språk som kan beskriva vilka egenskaper de har och vilka funktioner de utför. Det måste också gå att specificera på hög nivå hur en tjänst interagerar med andra tjänster, vilka konsekvenser varje interaktion har, och hur de kan exekveras på olika plattformar.

Eftersom en webbtjänst kan ha flera semantiska beskrivningar, t.ex. med olika upplösning eller detaljdjup, är publiceringen av själva webbtjänsten separerad från publiceringen av dess beskrivningar. Tjänsten finns typiskt tillgänglig på en server någonstans, medan dess beskrivningar publiceras i register som är tillgängliga via nätet. OWL-S möjliggör sätt att söka i registren baserat på krav gällande funktionalitet och andra egenskaper, istället för endast på nyckelord som med vanliga webbtjänster. Olika förslag på sådana register beskrivs senare.

2.6.2 Integrering av informationsmodeller

2.6.2.1 Översikt av problemet

Informationsmodeller, ontologier, är formaliseringar av kunskap inom en viss domän. Huvudsyftet med en informationsmodell är att sprida denna kunskap och göra den allmänt tillgänglig för att möjliggöra interoperabilitet mellan maskiner på den semantiska nivån. Detta kräver att modellen är återanvändbar, samt att den är lätt att underhålla och utveckla. Det är också nödvändigt att kunna integrera en informationsmodell över en domän med andra, mer specifika, modeller som beskriver delområden i domänen. Tyvärr är begreppen i en informationsmodell ofta starkt sammankopplade, vilket gör hela modellen svår att återanvända, underhålla och utveckla. Ett användbart knep för att underlätta detta är att normalisera modellen och dela upp den i moduler [4], se figur 2-8. Begrepp i olika moduler kan sedan återrelateras till varandra med hjälp enkla logiska villkor. Fördelen med att ha en modulariserad informationsmodell är som sagt att det underlättar återanvändning och underhåll, men ofta så blir också implicit kunskap i modellen explicit uttryckt. Det blir också enklare att kontrollera så att modellen inte innehåller motsägelser efter uppdateringar. Att ha moduler underlättar också återanvändning eftersom kompletta informationsmodeller inte behöver importeras och behandlas.



Figur 2-8

En informationsmodell delas upp i moduler, som sedan återrelateras för att tydligare beskriva samma information, se figur 1 och 2 i [4].

2.6.2.2 Detaljer och ansatser till lösning

Till grund för normaliseringen av informationsmodellen används dess *primitiva skelett*, vilket är en arvshierarki av alla primitiva begrepp i modellen. Skelettet utgör ett träd, dvs. varje begrepp ärver från högst ett annat primitivt begrepp. Under normaliseringen delas det primitiva skelettet upp i moduler, se figur 2-8. Dessa återrelateras sedan till varandra med hjälp enkla logiska villkor. I [6] beskrivs ett elegant sätt att skapa informationsmodeller baserat på ett primitivt skelett av begrepp med OWL. Där beskrivs bland annat hur en klass kan skapas i OWL och sedan automatiskt placeras in på rätt plats i informationsmodellens arvshierarki med hjälp av resonering.

Syftet med att integrera olika informationsmodeller är att återanvända den kunskap som finns i enskilda domäner, vid skapandet av en gemensam modell för flera domäner. Även om integrering underlättas av att ha informationsmodeller uppdelade i moduler så uppstår problem där de olika modellerna har begrepp som överlappar varandra. För att lösa dessa problem måste modellerna matchas mot varandra för att hitta semantiskt lika begrepp. Begrepp kan skilja sig från varandra på både språklig och ontologisk nivå [4]. Problem på den språkliga nivån kan utgöras av olika syntax, olika betydelser av logiska och andra språkliga strukturer, eller av olika expressivitet i språken som informationsmodellerna är uttryckta i. Problem på den ontologiska nivån kan utgöras av skillnader i hur domänen uppfattas konceptuellt av den som gjort informationsmodellen, eller av skillnader i hur begrepp är definierade. Språkliga skillnader kan rättas till genom enkla omskrivningsregler då syntaxen skiljer, eller genom mappningsregler mellan kända ontologispråk. Olikheter i hur domänen uppfattats konceptuellt är i princip omöjliga att rätta till automatiskt, eftersom det behövs sådan kunskap som endast domänexperter besitter. Vissa ansatser med artificiell intelligens har dock prövats men hittills utan sådan framgång att det är tillämpbart i praktiken. För

närvarande är det troligen mest användbart med metoder och verktyg som erbjuder halvautomatisk matchning där en användare erbjuds förslag och får välja de som passar bäst för situationen. Detta är acceptabelt om matchning inte sker så ofta. Ett problem som kvarstår är dock att i princip alla verktyg endast föreslår 1-1 mappningar mellan begrepp, och inte komplexa mappningar där ett begrepp i en informationsmodell kan motsvara snittet av flera begrepp i en annan. Att hitta komplexa mappningar är ett svårt problem.

En av de mest lovande ansatserna till integrering av informationsmodeller är GLUE [6]. Här kombineras maskininlärning med användning av heuristiker och domänrestriktioner för att hitta mappningar mellan modeller. *Joint probability distributions* används för att avgöra om två begrepp matchar varandra. Samma teknik används i t.ex. neuronnät för att avgöra liknande igenkänningsproblem. För att få fram dessa sannolikhetsfördelningar används maskininlärningsteknik där en *learner* tränas på att känna igen begrepp med den ena informationsmodellen som indata. Sedan får samma learner försöka identifiera motsvarande begrepp i den andra. Igenkänningen sker med multistrategi-inlärning baserat dels på begreppens namn och dels på begreppens textinnehåll. Kärnan i GLUE är alltså ordigenkänning, om än avancerad sådan, och inte begreppigenkänning. Till sist används *relaxation labeling* med olika heuristiker som tar hänsyn till domänrestriktioner. GLUE är testat på tre verkliga fall där automatiska matchningar jämfördes med sådana gjorda för hand. Det visade sig att GLUE var korrekt i 66-97 % av fallen. Resultatet är alltså lite osäkert och GLUE lämpar sig än så länge bäst till halv-automatisk matchning där en användare slutligen får avgöra om matchningarna är korrekta.

2.6.3 Publicering och lokalisering

2.6.3.1 Översikt av problemet

För att webbtjänster ska kunna användas effektivt är det absolut nödvändigt att kunna lokalisera passande webbtjänster. Vanliga beskrivningar av webbtjänster, t.ex. WSDL, kan visserligen användas för att hitta och ta reda på hur tjänsterna används, men de innehåller ingen explicit semantisk information. Därmed måste önskade webbtjänster vara utvalda i förväg eller så måste de uppsökas manuellt med någon slags sökmotor. Med semantisk information möjliggörs mer kraftfulla och exakta sökningar. Vid sökning kan då speciella eftersökta egenskaper hos en tjänst anges istället för vilka nyckelord som ska ingå i dess namn eller beskrivning.

WSDL-dokument publiceras vanligtvis i UDDI-register. UDDI är en framväxande industristandard för register för webbtjänster, men två stora brister förhindrar publicering av semantiska beskrivningar av webbtjänster. Den första bristen är sökmekanismen. Det är i princip endast möjligt att göra sökningar efter nyckelord. Det går visserligen att söka enligt kategoritillhörighet, men detta ger mycket grova sökningar med många orelevanta träffar. Ett exempel är sökning enligt kategorin *”Tjänster med WSDL-beskrivning”*, vilket kan ge en hel del träffar. Så om inte omöjligt är det i alla fall svårt att hitta en ”Bilförsäljare” som annonserar sina tjänster som ”Automobilmånglare” etc. Det krävs också manuellt arbete vilket strider mot ett av huvudsyftena med semantiska webbtjänster, att de ska kunna väljas automatiskt av maskiner. Den andra bristen, som UDDI delar med bl.a. WSDL, är att XML används för att beskriva dess datamodell och innehåll. Detta garanterar syntaktiskt interoperabilitet, men misslyckas med att ge en enhetlig semantisk beskrivning av innehållet. Två identiska beskrivningar kan därmed ha helt olika innebörd, och tvärtom, beroende på hur de används. För att råda bot på detta måste ett register klara av att representera semantiska beskrivningar av webbtjänster. Detta kräver i sin tur en intern datamodell som är tillräckligt

uttrycksfull för att kunna representera tjänstebeskrivningar utan att dess betydelse förloras eller förvrängs. Det måste också vara möjligt att utföra sökningar baserade på egenskaper istället för nyckelord. Till exempel så ska en bilförsäljare kunna hittas genom att egenskapen ”försäljare av bilar” efterfrågas, oavsett vilket namn som står i registerannonsen. Sökningar av denna typ bygger på semantisk matchning, dvs. matchning av semantiskt beskrivna förmågor och egenskaper (capability-based matching).

Generell semantisk matchning av begrepp ur olika informationsmodeller är som beskrevs i avsnittet om integrering av informationsmodeller ett svårt problem, utan någon praktiskt användbar lösning ännu. Några ansatser beskrivs nedan.

2.6.3.2 Ansatser till lösning

För att kunna tillämpa semantisk matchning i praktiken antas det i alla nedan beskrivna ansatser att endast begrepp ur en och samma informationsmodell används i tjänsteannonser och sökfrågor. Detta antagande förenklar matchningsprocessen till att avgöra om ett begrepp i sökfrågan är samma som ett begrepp i en annons, eller om det är ett besläktat begrepp till detta osv.

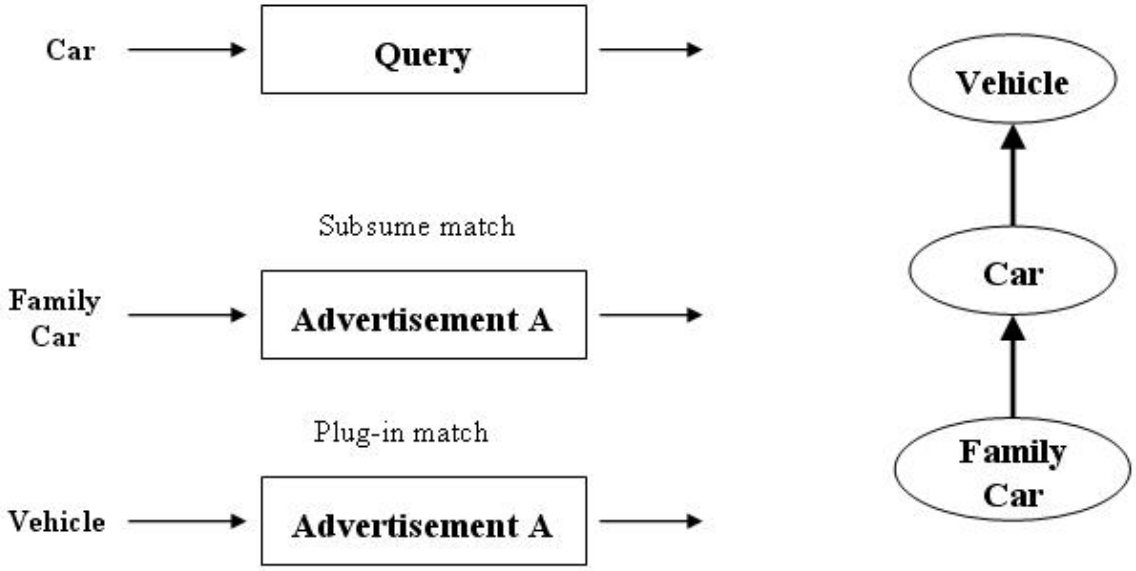
En av de mest kända ansatserna till förmågebaserade sökningar efter semantiska webbtjänster som beskrivs med OWL-S, återfinns i [10, 11, 12]. Där beskrivs ett register med en matchningsenhet, kallad *The OWL-S/UDDI Matchmaker*, som möjliggör publicering och matchning av semantisk information. Ett UDDI-register utökas här med en kommunikationsmodul, en matchningsmotor som matchar OWL-S-beskrivningar och en översättare som översätter OWL-S-beskrivningar till UDDIs datamodell. Registret klarar därigenom av att utföra begränsade förmågobaserade sökningar. I [10] beskrivs hur en tjänsteprofil i OWL-S kan översättas så att den kan lagras i ett vanligt UDDI-register. Förutom de nyckelordsbaserade sökningar som kan utföras som vanligt i ett UDDI-register, kan också förmågobaserade sökningar göras.

Matchningsalgoritmen som används i registret beskrivs i [12]. Den går ut på att det finns fyra grader av matchning mellan en förfrågan och annonserade tjänster. Först matchas utdata från förfrågan mot alla annonser i registret, en i taget. Efter detta matchas indata också. Startvillkor och effekter betraktas visserligen inte i Matchmakern, men vid tiden för implementeringen fanns ingen standard för hur dessa ska uttryckas i OWL-S. De in- och utdatabegrepp som stämmer överens rankas på följande vis med hjälp av fyra matchningsgrader:

- En *exakt* matchning erhålls om någon parameter i in- eller utdata beskrivs av samma begrepp i både förfrågan och annonsen, eller om begreppet i förfrågan är ett direkt subbegrepp till det i annonsen. Skälet till detta är att en tjänst som beskrivs med ett begrepp, t.ex. *Bil*, också förväntas tillhandahålla alla direkta subbegrepp, t.ex. *Sportbil* och *Familjebil*.
- En *plug-in*-matchning erhålls om ett begrepp i förfrågan kan inordnas under motsvarande i annonsen, dvs. om begreppet i förfrågan är ett subbegrepp (ej direkt) till det i annonsen.
- En *inordnings*-matchning (subsume match) erhålls om begreppet i annonsen istället kan inordnas under begreppet i förfrågan.
- Ett *misslyckande* erhålls om ingen av de ovan angivna matchningarna hittas.

De fyra graderna är här angivna i fallande önskvärdhetsordning, dvs. exakt matchning är bäst.

Ett exempel på semantisk matchning ges nedan i figur 2-9. En webbtjänst med indata *Car* eftersöks med hjälp av sökfrågan överst i figuren. Annons A har indata *Family Car*, vilket är ett underordnat begrepp till *Car*, och matchningen mellan sökfrågan och denna annons är därmed en inordningsmatchning. Annons B har indata *Vehicle*, vilket begreppet *Car* är ett subbegrepp till, och denna matchning är därmed en plug-in-matchning. Annons B passar matchar således sökfrågan bäst vad gäller indata.



Figur 2-9

Exempel på semantisk matchning. Annons A är en inordningsmatchning till sökfrågran, och Annons B är en plug-in-matchning.

För att spara tid vid förfrågningar görs alla beräkningar för matchning i Matchmakern redan vid publiceringen av en tjänst. Resultatet sparas sedan så att berörda informationsmodeller inte behöver laddas in på nytt och beräkningarna göras om vid varje förfrågan. Vid förfrågningar görs alltså i stort sett bara uppslagningar. När Matchmakern tar emot en förfrågan matchar den först mot de sparade beräkningar som berör de semantiska begrepp som förekommer i förfrågan. Träffarna, dvs. de annonser i registret som matchar förfrågan, rankas sedan och visas upp. En fördel med att ha beräknat matchningen av begrepp i förväg är förstås att det sparar tid, men det finns ett par allvarliga nackdelar också. För det första är matchningen ofullständig eftersom endast begreppen i de informationsmodeller som fanns tillgängliga vid de olika tjänsternas publicering tas med i sökningen. Ofullständigheten beror på att förfrågningar till Matchmakern kan innehålla nya begrepp som är besläktade med begrepp som förekommer bland gamla annonser. Detta kan exempelvis vara fallet om begreppet B*il* får *Rymdbil* som ett nytt subbegrepp. Eftersom detta nya begrepp inte fanns med när annonserna publiceras så finns det inte med i Matchmakerns sparade matchningsberäkningar, och passande annonser kommer därför inte att hittas trots att sådana finns. Forskningsgruppen bakom Matchmakern anser dock att sannolikheten för denna händelse är liten och att det hela därför är acceptabelt för att spara tid. En annan nackdel är att Matchmakern kan komma att kräva mycket minne för att hantera de begrepp som förekommer bland in- och utdata i annonserna.

Ovanstående algoritm, tillsammans med andra, matchar endast in- och utdata från en annons och en sökfråga med hjälp av deras OWL-S-profiler. Även om semantisk information används begränsas effektiviteten och korrektheten hos matchningen av att endast OWL-S-profilen för en webbtjänst används. Processmodellen innehåller information som är nödvändig att ta med i vissa fall. Ett exempel är då kontrollstrukturen *choice* används i processmodellen. Denna kontrollstruktur kan användas för att slumpvis välja ut vilka atomära processer som exekveras. Profilen för tjänsten kan då deklarera att *både* utdata *U1* och *U2* ges, medan endast *en* av de två verkligen fås då tjänsten exekveras. Matchning mot enbart profilen skulle i detta fall ge felaktiga träffar eftersom den felaktigt anger att två utdataparametrar fås. Processmodellen kan däremot användas för att reda ut situationen eftersom innebörden av kontrollstrukturen *choice* i exemplet ovan då kan tolkas och tas med i matchningen, så att matchning endast sker mot utdata *U1* eller *U2*, istället för båda. I [19] beskrivs en ansats till matchning av tjänster med algoritmer som matchar in- och utdata hos tjänster genom att också använda processmodellen.

Förutom att undersöka hur ett enda register ska handha semantiska webbtjänster bör det också undersökas hur registret i sig ska vara organiserat för att vara lättillgängligt och robust etc. En intressant lösning på ett distribuerat register används i *METEOR-S Web Service Discovery Infrastructure (MWSDI)* [13]. Detta är en implementation av ett distribuerat UDDI-register. Det består av ett P2P-nätverk (peer-to-peer) av många UDDI-register som har transparent tillgång till varandra. I UDDI-specifikationen finns inga speciella funktioner för att hitta andra register, så varje webbtjänst antas i förväg känna till vilka register som finns och var de finns någonstans. Därför behövs en infrastruktur som MWSDI för att ge register tillgång till varandra. Även om endast vanliga webbtjänster kan publiceras med METEOR-S är lösningen intressant att studera, eftersom ett distribuerat semantiskt register kan organiseras på ett liknande sätt. Semantisk information om själva registren, inte innehållet, lagras i en informationsmodell som kallas *XTRO (Extended Registries Ontology)*. XTRO innehåller till exempel information om vilka register som innehåller tjänster som rör en viss domän etc. Den är uttryckt i OWL och används för att klassificera och lokalisera de olika registren automatiskt. Utifrån de olika typer av funktionalitet som behövs i nätverket har olika typer av plattformar (peers) definierats:

- *Gateway-peer*. Dessa peers fungerar som ingångspunkt till det distribuerade registret. De är ansvariga för att uppdatera XTRO med relevant information varje gång ett nytt register ansluter sig till nätverket.
- *Operator-peer*. Dessa peers tillhandahåller varsitt av alla de UDDI-register som tillsammans tillhandahåller innehållet i det distribuerade registret. Varje peer är ansvarig för att sköta publicering och sökning i dess lokala UDDI-register.
- *Auxiliary-peer*. Dessa peers tillhandahåller tillsammans informationsmodellen XTRO och gör den tillgängligt i hela nätverket, så att information om vad ett register innehåller etc. finns tillgängligt.
- *Client-peer*. Dessa peers är tillfälliga medlemmar av nätverket och instansieras endast för att tillåta användare att komma åt det distribuerade registret.

De semantiska beskrivningarna av registren gör att federationer, bestående av delmängder av alla ingående register, kan definieras. En federation består typiskt av register där innehåll och tjänster inom en viss domän finns publicerade. De semantiska beskrivningarna tillåter dessutom maskiner att automatiskt resonera kring registren. Frågor om register R stödjer domän D, eller om vad register R1 har för relation till register R2 etc., kan alltså besvaras.

Federationer av register underlättar handhavandet av en distribuerad infrastruktur för information eftersom olika enheter har kontroll över sin egen information och sina egna tjänstebeskrivningar. Ansvaret för underhåll kan också delegeras till de olika enheterna, och det är mer skalbart än ett stort centraliserat register.

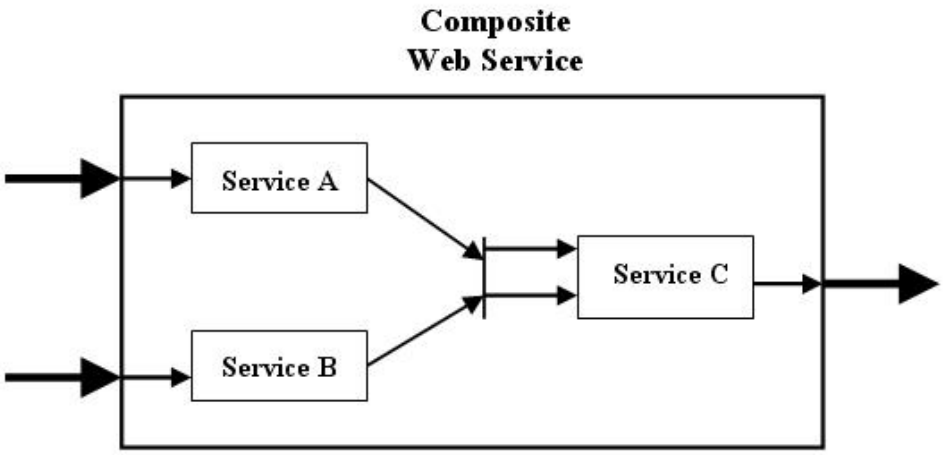
MWSDI kan som sagt endast registrera vanliga webbtjänster, men vi såg tidigare hur ett vanligt UDDI-register kan utökas med hjälp av UDDI/OWL-S-Matchmaker för att även klara av semantiska webbtjänster. Detta blir dock svårare med ett distribuerat register. För det första måste datastrukturen i Matchmakern, som användes vid matchning, utökas genom att referera till tjänsteannonser i andra register, dvs. en distribuerad Matchmaker behövs. För det andra kommer flera informationsmodeller att användas, vilket ökar chansen för att någon använd informationsmodell förändras eller utökas. När detta sker måste en ny inferens göras, vilket kan ta lång tid. Tiden en inferensomgång tar i anspråk beror på hur många informationsmodeller som används för att beskriva allt innehåll i hela P2P-nätverket. Alla informationsmodellerna måste nämligen undersökas och samköras för att upptäcka logiska inkonsekvenser.

2.6.4 Komposition

2.6.4.1 Översikt av problemet

Genom att ha semantiskt beskrivna webbtjänster och möjlighet att lokalisera dem baserat på funktionalitet, kan de göras mer flexibla och anpassningsbara. För att få en uppgift utförd krävs dock att en passande tjänst som kan utföra hela uppgiften existerar. Om en sådan inte existerar kanske uppgiften kan utföras ändå genom att kombinera ett antal av de tillgängliga tjänsterna, dvs. skapa en sammansatt (komposit) webbtjänst. En sammansatt webbtjänst består av ett antal andra webbtjänster och kan beskrivas som en process med ett exekveringsflöde och ett dataflöde, se figur 2-10. Exekveringsflödet beskriver i vilken ordning de ingående tjänsterna exekveras, och dataflödet hur information överförs mellan tjänsterna under exekveringen, dvs. hur utdata från en tjänst blir indata till nästa osv. I figur 2-10 visas en sammansatt webbtjänst som består av tre andra, där dataflödet visas av pilarna. Exekveringsflödet visas av den vertikala linjen som anger att webbtjänsterna A och B måste exekveras innan C kan exekvera.

Språk för beskrivning av processer av webbtjänster är t.ex. BPEL4WS (Business Process Execution Language For Web Services) och processmodellen i OWL-S. Med webbtjänster som saknar semantiska beskrivningar är det endast möjligt att skapa sammansättningar manuellt, antingen när behovet uppstår eller i förväg. Sammansatta webbtjänster som skapats manuellt publiceras ofta som en ny webbtjänst så att sammansättningen kan återanvändas. Att manuellt behöva skapa sammansättningar begränsar webbtjänsternas flexibilitet, speciellt om det sker i förväg eftersom olika behov kan vara svåra att förutse. Även om sammansättningar definieras manuellt när behovet uppstår, så kan det vara ogenomförbart för en människa att analysera massor av webbtjänster, för att sedan välja ut ett fåtal och sätta samman dem på ett korrekt sätt. Maskiner är bättre på att gå igenom stora mängder information.



Figur 2-10

En sammansatt webbtjänst som består av tre andra. Pilarna visar dataflödet. Den vertikala linjen visar exekveringsflödet, dvs. att webbtjänsterna A och B måste exekvera klart innan C kan exekvera

Semantiska beskrivningar av webbtjänster möjliggör en lösning på problemen med att manuellt skapa sammansatta webbtjänster. Genom att låta maskiner analysera alla tillgängliga tjänster utifrån funktionalitet, och sedan automatiskt skapa sammansättningar, reduceras tiden avsevärt, samtidigt som fler tjänster kan behandlas. Sammansättningar behöver därmed inte heller definieras i förväg, utan kan skapas när behovet uppstår. Sådana dynamiska sammansättningar möjliggör mer flexibla webbtjänster och tillåter (i teorin) att godtyckligt komplexa uppgifter löses automatiskt.

För att åstadkomma dynamiska sammansättningar av webbtjänster måste en användares behov mappas till en delmängd av alla tillgängliga tjänster som tillsammans kan uppfylla dem, och en exekveringsföljd måste tas fram automatiskt. Först måste ett komplext behov brytas ner i deluppgifter som ska lösas. Varje deluppgift kan sedan mappas till en passande tjänst som lokaliseras genom sökning i ett register, som beskrivet i föregående avsnitt. När tjänster som kan utföra alla deluppgifter lokaliserats måste en exekveringsordning fastställas. Semantiska beskrivningar i OWL-S möjliggör detta, men tillhandahåller inget sätt att skapa kompositionen. Profiler kan användas till att lokalisera tjänster, och processmodeller till att sammanfoga de lokaliserade tjänsterna till en process. Att välja webbtjänster och fastställa en exekveringsordning är ett svårt problem. Nedan följer beskrivningar av några förslag på hur det kan gå till.

2.6.4.2 Ansatser till lösning

En enkel halv-automatisk ansats till komposition beskrivs i [15]. Här används semantiska beskrivningar av webbtjänster i DAML-S, en föregångare till OWL-S, för att avgöra vilka webbtjänster som kan kopplas ihop. Denna kontroll görs genom att typen av utdata från en tjänst matchas mot typen indata till nästa tjänst i processen. Om utdata från en tjänst kan vara indata till en annan är de två möjliga att koppla ihop. En träff är exakt då två parametrar beskrivs av exakt samma OWL-begrepp, eller allmän (generic) då den ena är ett subbegrepp till den andra. Alla träffar kan sedan filtreras med hjälp av krav på icke-funktionella parameterar, t.ex. typ av tjänst, geografisk plats etc. Ansatsen består av en kompositör (composer) och en inferensmotor. Användaren får med hjälp av kompositören själv börja att komponera ihop en sammansättning genom att först välja en tjänst ur en lista över alla tillgängliga. Sedan används inferensmotorn för att ta fram alla webbtjänster som kan kopplas ihop med den valda tjänsten. Användaren väljer sedan nästa tjänst i sammansättningen ur en

reducerad lista osv. En brist i denna ansats är att manuellt beslutsfattande krävs, men detta gör å andra sidan att denna ansats idag är praktiskt genomförbar, till skillnad från en del andra förslag.

En enkel, men helautomatisk, ansats beskrivs i [16]. Här ses komposition som ett planeringsproblem. Varje webbservice representeras av en nod i en graf och indata/utdata-matchingar länkar samman noderna. En kortaste stig genom grafen från användarens önskade indata till utdata hittas med en modifierad Bellman-Ford-algoritm. Tiden det tar är dock $O(n^3)$, vilket kan vara ohållbart då många tjänster finns tillgängliga. Eftersom hänsyn till startvillkor och effekter också bör tas vid matchningen mellan noder, kommer kanske tiden att bli ännu längre.

En annan mer lovande ansats, som också ser komposition som ett planeringsproblem, beskrivs i [17]. Här används SHOP2, ett domänoberoende HTN-planeringssystem (Hierarchical Task Network). HTN-planering är en teknik inom AI-området för att skapa en plan genom att dela upp en uppgift i atomära deluppgifter och planera en utförandeordning för dessa. Metoden är därför lämplig att använda för att skapa processer av webbtjänster. SHOP2 skiljer sig från de flesta andra HTN-planeringssystem genom att deluppgifter planeras i samma följd som de sedan kommer att utföras. Det innebär att SHOP2 kan hålla reda på det aktuella tillståndet i exekveringen av en process. Därför kan inferensmotorer användas och externa program kan även anropas under processexekveringen. Vid utförandet av uppgiften behöver SHOP2 kunskap om den aktuella domänen. Denna finns i en kunskapsbas med *operatorer* och *metoder*. Operatorer beskriver hur en atomär deluppgift utförs, och metoder beskriver hur en sammansatt uppgift kan delas upp i deluppgifter. Processmodeller i OWL-S kan översättas till SHOP2-domäner. En komplex uppgift kan alltså beskrivas som en processmodell i OWL-S. Sedan kan tjänster som tillsammans bygger upp en likvärdig processmodell lokaliseras och exekveras. Det finns dock en del begränsningar än så länge. Det går t.ex. inte att direkt översätta OWL-DL-semantik till SHOP2-axiom, vilket gör att inferensmotorer för OWL-DL inte kan användas fullt ut. För att det ska gå att översätta en processmodell antas också att:

1- Alla atomära processer kan ha effekter *eller* utdata, inte både och.

2- Det finns inga komposita processer med kontrollstrukturerna *Split* eller *Split+Join*. *Split* innebär att två atomära processer kan utföras parallellt, och *Split+Join* innebär att exekveringen av den sammansatt tjänsten inte kan fortsätta förrän båda är färdiga.

Dessa begränsningar hindrar än så länge denna ansats från praktisk användning.

Ytterliggare en ansats, lika lovande som den förra, men med lite annorlunda inriktning, beskrivs i [18]. Här skapas sammansättningar av atomära webbtjänster genom att bevis i linjär logik (LL) genereras. Linär logik kan användas till att hålla reda på resurser, samt är tillräckligt uttrycksfullt för att beskriva både funktionella och icke-funktionella parametrar. Externt används profiler i DAML-S, dvs. en DAML-S-profil av önskad tjänst används som uppgiftsspecifikation. Denna översätts sedan till ett logiskt påstående, ett LL-teorem, eftersom linjär logik används internt i bevisgeneratorn för att beskriva såväl tjänster som informationsmodeller. Tillgängliga tjänsters atomära processer representeras internt av LL-axiom. Ett bevis för LL-teoremet genereras sedan automatiskt utifrån dessa axiom, om ett sådant existerar. Det genererade beviset representerar en sammansatt webbtjänst som kan utföra den specificerade uppgiften. Till sin hjälp har bevisgeneratorn en semantisk inferensmotor för att analysera tjänstebeskrivningar och informationsmodeller. Varje steg i beviset motsvarar en av de atomära processer som ingår i sammansättningen. Om inget bevis

kan genereras finns det ingen sammansättning som uppfyller uppgiftsspecifikationen. Ur beviset extraheras till sist ett exekveringsflöde uttryckt antingen i BPEL4WS, eller som en processmodell i DAML-S. Linjär logik är sunt och komplett, vilket innebär att alla slutsatser är sanna och om en lösning finns så kommer den att hittas. Detta innebär för kompositionen att en genererad komposition garanterat fungerar, samt att om det finns en fungerande komposition så hittas den. En brist är att villkor och effekter inte finns med i den interna representationen av processbeskrivningar. Det är kanske möjligt att införa dessa, men det tas inte upp i [18].

En viktig skillnad mellan ansatserna till komposition med SHOP2 och LL-bevisgeneratorn, är det som utgör uppgiftsspecifikationen, utifrån vilken en sammansättning av webbtjänster definieras. SHOP2 utgår från en processmodell till en önskad komplex tjänst i OWL-S, och hittar sedan tjänster som utför uppgiften. LL-bevisgeneratorn utgår endast från en profil av en önskad komplex tjänst i DAML-S och hittar sedan en sammansättning som utgör en processmodell till den tjänsten. En processmodell innehåller mer information om en tjänst än vad en profil gör, så det finns mer utrymme för felaktiga sammansättningar med LL-bevisgeneratorn än med SHOP2.

2.6.5 Exekvering

Processmodellen i en semantisk beskrivning av en webbtjänst, uttryckt i OWL-S, består av atomära och komposita processer. Endast atomära processer kan exekveras direkt. Komposita processer består av flera atomära processer som exekveras en i taget. Processmodellen måste också analyseras för att se vilka atomära processer som kräver att andra atomära processer ska vara exekverade innan de själva kan exekveras, vilka som kan exekveras parallellt etc. En enhet som exekverar komposita webbtjänster måste alltså både analysera och exekvera.

Att exekvera en webbtjänst utifrån en metadatabeskrivning på nivå med t.ex. WSDL är inga problem eftersom datatyper etc. beskrivs med XML-Schema i både WSDL och i det protokoll som används för att anropa en webbtjänst, oftast SOAP. Problemet här är att OWL-S beskriver en tjänst på en för hög semantiskt nivå för att den ska gå att exekvera direkt. Istället måste all nödvändig information omvandlas till XML-nivå utifrån den betydligt kraftfullare semantiska beskrivningen i OWL-S, på ett sätt så att all semantik bibehålls. Detta innebär att översätta explicit uttryckta semantiska beskrivningar till implicit representation på den plattform där webbtjänsten är implementerad. Förankringen i en OWL-S-beskrivning är denna brygga ifrån OWL-S till ett mer teknisk språk på lägre nivå, oftast WSDL. Med OWL-S kan således helt vanliga webbtjänster utan ändring utökas med semantisk information, samtidigt som de standarder som finns till vanliga webbtjänster utnyttjas. Översättningen är möjlig genom att OWL-S och WSDL överlappar varandra på tre områden [32]:

1- En atomär process i OWL-S motsvarar en operation i WSDL.

2- In- och utdata till en atomär process i OWL-S motsvarar in- och utdelarna av ett meddelande i WSDL.

3- Typerna av in- och utdata i OWL-S motsvarar abstrakta typer i WSDL.

En atomär process med in- och utdata i OWL-S kan således mappas till en operation i WSDL och sedan exekveras. Ett problem som uppstår härmed är mappningen av datatyper mellan OWL-S och WSDL, eftersom datatyper uttrycks i OWL i OWL-S och i XML-Schema i WSDL. För att översätta OWL till XML-Schema används XSLT [27]. XSLT identifierar syntaktiska strukturer i serialiseringen av OWL-S-beskrivningen, vilket är XML, och

översätter till ett annat XML-format, nämligen WSDL. Eftersom översättningen därmed sker baserat på syntax och inte semantik, kan problem uppstå eftersom olika OWL-beskrivningar kan vara skrivna på RDF/XML-notation på syntaxmässigt olika sätt. För enkelhetens skull bortses från detta i denna studie, men ett alternativt sätt att översätta datatyper beskrivs i [3].

2.6.6 Kontroll

Individuella tjänster kan ta en viss tid att exekvera helt och hållet, vilket gör att en användare kan vilja övervaka exekveringen av en webbtjänst. Under exekveringen av en sammansatt webbtjänst kan till exempel planerna ha ändrats så att exekveringen måste anpassas till detta. Semantisk information i OWL-S möjliggör övervakning av exekvering genom att processmodellen beskriver de ingående webbtjänsterna och hur långt exekveringen har kommit. Själva övervakningen måste ske av intelligent mjukvara (agenter) eftersom OWL-S endast möjliggör kontroll utan att tillhandahålla någon specifik metod för hur det kan gå till. Kontroll och övervakning av exekvering har inte tagits upp i denna studie.

3 Infrastruktur för informationshantering

Vid framtagningen av en infrastruktur för automatiserad informationshämtning från heterogena datakällor koncentrerade vi oss främst på framväxande standardteknologier inom den semantiska webben, t.ex. OWL. Den nedan föreslagna infrastrukturen använder således informationsmodeller uttryckta i OWL och webbtjänster med semantiska beskrivningar uttryckta i OWL-S. Andra infrastrukturer med liknande syften som berörts ytligt är Internet Reasoning Service [20] och Edutella [21].

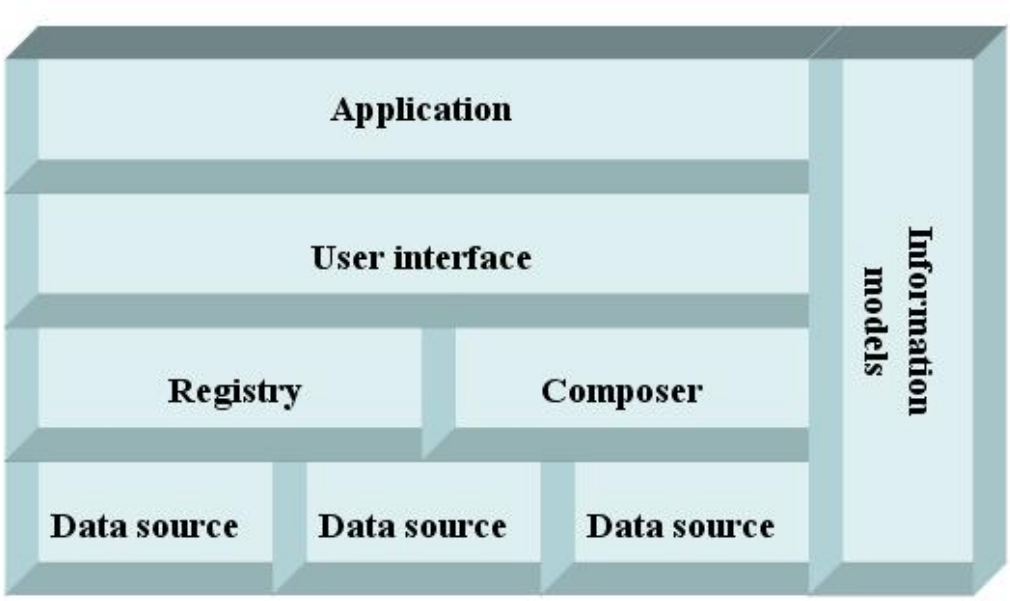
3.1 Semantiska webbtjänster inom logistik

Effektivt och dynamiskt utnyttjande av tjänster i ett nätverk är beroende av semantisk modellering av tjänster. Betydelsen av detta påpekas bland annat i IRM Vision [22] för Försvarsmakten. Där förespråkas IT-baserade tjänster med metadatabeskrivningar, vilka anses vara en kritisk komponent i det framtida försvaret. I IRM Vision påpekas också att stabila standarder är viktigt för att framtidssäkra modelleringsarbetet. RDF och OWL är standarder godkända av W3C, och lämpar sig därför väl för semantisk modellering av webbtjänster.

Projekt VSHMOD-UH-2010 som slutfördes under 2003 hade syftet att ta fram krav på det informationsstöd som Försvarsmaktens logistikverksamhet kräver i NBF bortom år 2010. I fas 2 av projektet, Förnödenhetsförsörjning [23], togs exempelvis krav gällande försörjning av förnödenheter fram. Processmodellering visade sig passa väl för modellering av logistikverksamheten, och ett antal processer togs fram i varje fas. Varje sådan process har ett informationsbehov. I det långa perspektivet ska inte användaren behöva bekymra sig om vilket specifikt informationssystem som tar fram önskad information, utan en enhetlig bild av information från olika specifika system ska presenteras. Detta kräver att skilda system kan interagera sömlöst även på den semantiska nivån. Här passar principerna bakom den semantiska webben in perfekt. All önskad funktionalitet från olika informationssystem kan tillhandahållas av semantiskt beskriva webbtjänster, vilket ger möjlighet att presentera en enhetlig bild över samtlig tillgänglig information inom en hel domän för en användare. Vi har därför föreslagit en infrastruktur för information baserat på den semantiska webben. Infrastrukturen kan i en förlängning fungera som en grund för informationshantering inom logistik i det framtida försvaret.

3.2 Komponenter

Infrastrukturen vi föreslagit är i princip en realisering av en semantisk webb. De komponenter som ingår beskrivs i figur 3-1. De två understa lagren, och i viss mån det näst översta, utgör komponenterna i den infrastruktur som vi föreslagit. I det understa lagret finns datakällorna som information ska hämtas ifrån. Varje datakälla är tillgänglig med hjälp av semantiska webbtjänster. Därför finns på nästa lager ett register där webbtjänster kan publiceras med semantisk information, samt en kompositör som kan skapa dynamiska sammansättningar av webbtjänster. Till höger i figuren finns ett vertikalt lager med informationsmodeller. Dessa används i varje lager för att tillhandahålla begrepp för semantiska beskrivningar. Informationsmodellerna kan visserligen också vara uppdelade i olika nivåer, men detta tas ej upp här. De två översta lagren utgörs av ett gränssnitt mot infrastrukturen samt applikationer som använder infrastrukturen, men endast gränssnitt tas delvis upp i denna studie. De finns endast med i figuren för att visa var de passar in gentemot den infrastruktur vi föreslagit.



Figur 3-1

Komponenterna i den föreslagna infrastrukturen utgörs av de två understa lagren, och i viss mån det näst översta. I varje lager används informationsmodeller för att tillhandahålla begrepp för semantiska beskrivningar. De två översta lagren utgörs av gränssnitt och applikationer, och visar hur de passar in gentemot den föreslagna infrastrukturen.

3.2.1 Datakällor och informationsmodeller

Varje datakälla är tillgänglig via nätverket i form av en semantisk webbtjänst. Datakällor kan utgöras av till exempel lagersystem eller databaser med felrapporter etc. En datakälla tillhandahåller information, men semantiska webbtjänster som tillhandahåller annan typ av funktionalitet förekommer också. Olika informationssystem som ska interagera publicerar nödvändig funktionalitet som semantiskt beskrivna webbtjänster. Med hjälp av webbtjänsterna fungerar alltså infrastrukturen som ett gränssnitt mellan olika informationssystem.

Varje datakälla beskrivs med hjälp av OWL-S precis som en vanlig semantisk webbtjänst. Beskrivningar av datakällor publiceras i ett register som är åtkomligt överallt i nätverket. De semantiska begrepp som behövs för att beskriva en tjänst, men som inte specifikt har med domänen att göra, finns tillgängliga i en global informationsmodell för tjänster (tjänsteontologi). De semantiska begrepp som beskriver domänen finns tillgängliga i en global och domänspecifik informationsmodell (domänontologi). Dessa informationsmodeller är uttryckta i OWL, och begrepp från dessa kan därför användas direkt i beskrivningar av datakällorna. Ett förenklande antagande som gjorts vid prototypimplementeringen är att det endast finns en informationsmodell för domänen och en för webbtjänster, men dessa modeller kan förändras och utökas kontinuerligt genom att integreras med andra informationsmodeller.

Ett potentiellt problem med semantiska beskrivningar av webbtjänster är att typer av in- och utdata är polymorfiska, dvs. en in- eller utdataparameter som anges vara av typen *Fordon* kan i själva verket vara av mer specifika typer av fordon, till exempel *Bil* och *Båt*. Detta får betydelse eftersom själva datakällorna är beskrivna med hjälp av OWL-S, medan innehållet troligen inte är det. Om inte det faktiska innehållet är semantiskt beskrivet kan mottagaren bara tolka värdet av en parameter som att den är av exakt den typ beskrivningen av en webbtjänst anger. Ibland kan detta bli fel. Ett exempel är en parameter som anges vara av

typen *Fordon*, men vars verkliga värde endast kommer att vara av typen *Bil* eller *Motorcykel*. Om en sådan datakälla anropas med ett parametervärde av typen *Båt* accepteras anropet eftersom en båt är ett fordon, men resultatet blir felaktigt. För att undvika problem av denna typ måste antingen alla mer specifika typer än den som anges kunna hanteras, eller så måste en parameter anges vara av *någon* av ett antal specifika typer, dvs. generaliseringar bör undvikas om inte alla typer som omfattas tillåts.

3.2.2 Register

Den ena delen av kärnan i infrastrukturen utgörs av ett register som kan publicera semantiska webbtjänster och utföra sökningar baserat på förmågor och egenskaper hos dessa tjänster genom semantisk matchning. Eftersom tjänsterna är beskrivna med hjälp av OWL-S måste registrets interna datamodell klara av att representera semantik på denna nivå. Ett alternativ för att åstadkomma detta är att utgå från ett UDDI-register och använda en enhet liknande den i OWL-S/UDDI-Matchmaker för att kunna representera tjänstebeskrivningarna. Annars kan en egen datamodell utvecklas och användas. Det finns för närvarande ingen standard för sådana register, men för att framtidssäkra infrastrukturen bör en standard användas när en sådan godkänns.

För att få riktigt bra träffar vid sökning kan både profiler och processmodeller lagras i registret och användas vid matchning. Förutom in- och utdata bör också startvillkor och effekter, samt icke-funktionella parametrar för en tjänst lagras och matchas mot. Någon exakt metod för matchning av OWL-S-beskrivningar har vi inte tagit fram eftersom detta är en alltför stor uppgift för detta projekt. Det är dock troligen enklare att matcha beskrivningar i OWL-S än generella OWL-beskrivningar eftersom de har ett ganska bestämt format. Matchningen av in- och utdata som OWL-S/UDDI-Matchmaker gör [12] verkar vara en bra utgångspunkt.

3.2.3 Kompositör

Den andra delen i kärnan av infrastrukturen är kompositören, dvs. den enhet som dynamiskt skapar sammansatta webbtjänster för att besvara komplexa sökfrågor. En sökfråga som ska besvaras med hjälp av infrastrukturen kommer först till kompositören. Denna delegerar först sökfrågan till registret. Om ingen existerande tjänst som kan besvara sökfrågan hittas, så försöker kompositören att skapa en sammansatt webbtjänst som klarar av det. Om en lyckad sammansättning hittas kan den sedan lagras som en ny webbtjänst i registret. Den dynamiska sammansättningen har då blivit statisk. Om detta sker ofta så kanske registret fylls av mer eller mindre specifika sammansättningar. Någon slags mekanism för att inte registret ska bli översvämmat behövs i så fall. Till exempel så kan sammansatta webbtjänster som inte använts på länge för att besvara sökfrågor tas bort, förutsatt att de inte publicerats av någon användare, utan kom till som dynamiska sammansättningar.

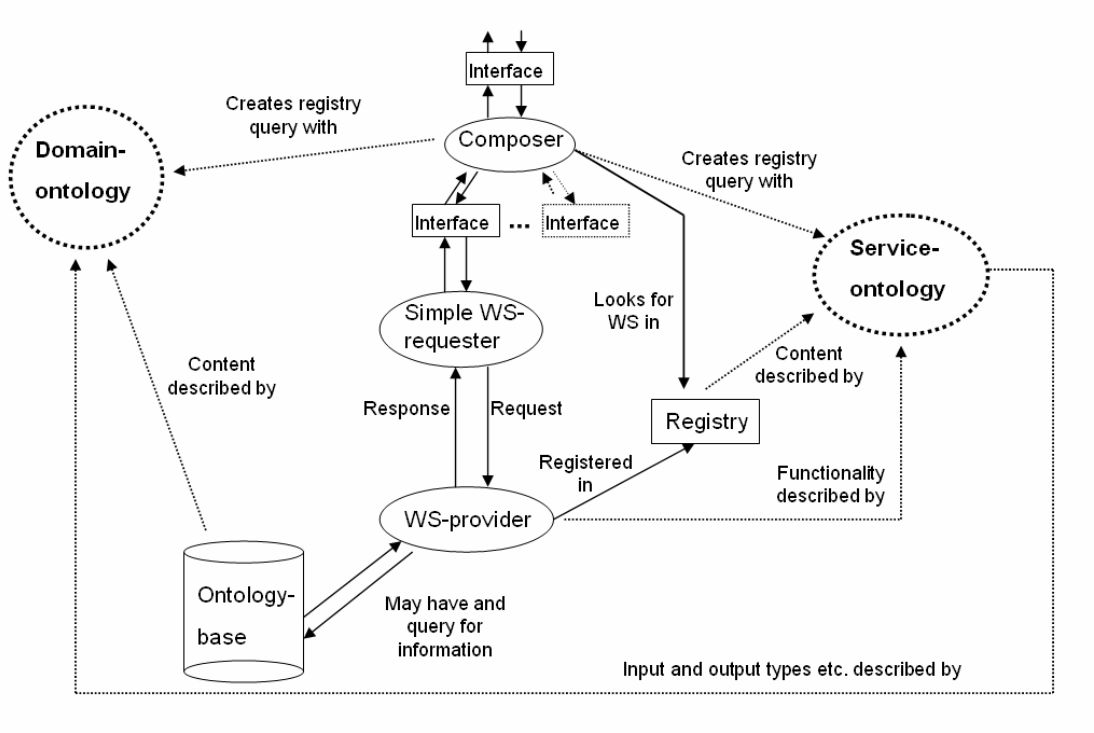
Någon metod för automatisk komposition har vi inte tagit fram eftersom det, precis som semantisk matchning, är en alltför stor uppgift för detta projekt. SHOP2 [17] eller komposition genom bevisgenerering [18] verkar lovande och är troligen en bra utgångspunkt.

3.2.4 Arkitektur

Arkitekturen för vårt förslag på en infrastruktur för automatisk informationshämtning från heterogena datakällor visas i figur 3-2. Den är i stort sett en implementering av ett semantiskt nätverk. Datakällorna är åtkomliga via webbtjänster, *WS-providers* i figur 3-2, som är semantiskt beskrivna med OWL-S, där begreppen tas ur en tjänsteontologi (service ontology)

uttryckt i OWL. Innehållet i en datakälla är semantiskt beskrivet med begrepp kopplade till en domänontologi, också den uttryckt i OWL.

WS-providers finns i utkanterna av nätverket. Kärnan i infrastrukturen är ett semantiskt register och en kompositör. Registret lagrar annonser för semantiska webbtjänster. Kompositören skapar sammansatta webbtjänster ad hoc, genom att hitta lämpliga tjänster och sätta samman dem till en process som tillhandahåller mer komplex funktionalitet än vad någon enskild webbtjänst erbjuder. En *WS-requester* i figur 3-2 är en enkel klient som exekverar enskilda webbtjänster. För att exekvera en hel process krävs en WS-requester för varje ingående atomär webbtjänst. En begäran om funktionalitet, eller information, går alltså först till kompositören, som antingen hittar en tjänst direkt eller skapar en sammansättning av tjänster. Uppgiften att hitta tjänster delegerar kompositören till registret. Till sist exekveras varje enskild webbtjänst av varsin WS-requester. Förutom att skapa sammansättningen har kompositören också till uppgift att koordinera in- och utdata, till och från dessa.



Figur 3-2

Den föreslagna infrastrukturen. Webbtjänster (WS-providers) publiceras med semantisk beskrivning i ett register, där de sedan kan lokaliseras baserat på funktionalitet och inte enbart nyckelord. En domänontologi och en tjänsteontologi tillhandahåller semantiska begrepp. Kompositören (Composer) skapar komplexa webbtjänster ad hoc genom att sätta samman enskilda webbtjänster till en process. Varje enskild webbtjänst exekveras av en enkel klient (WS-requester).

3.3 Realisering av infrastruktur

För att kunna implementera en prototyp, och visa användbarheten av infrastrukturen, togs ett scenario fram. Scenariot omfattar en internationell insats, där Sverige deltar med logistikexperter. Dessa har till sin hjälp ett avancerat logistiksystem, baserat på den föreslagna infrastrukturen.

3.3.1 Scenario

Det är år 2011 och relationerna mellan republikerna Larendien och Karundia i Afrika som gränsar mot varandra har under de senaste åren blivit alltmer spända och aggressiva. Under den senaste tiden har en milis understödd av den Larendiska armén med våld försökt fördriva etniska Karundier från gränsområdet. Omvärldens fruktan för ytterliggare våldsamheter har lett till en FN-resolution om att skicka in en fredsbevarande styrka på 5000 man. Styrkan har redan anlänt och kommer att stanna tills läget är förbättrat, det handlar om uppskattningsvis 2 år. Ett antal länder är inblandade, däribland Sverige, Storbritannien och USA. Sverige bidrar inte med någon stridande trupp, men har istället ansvar för all logistik.

Till sin hjälp har svenskarna ett avancerat logistiksystem. Systemet är baserat på de krav som framkom i projektet *VSHMOD-UH-2010* [7], och är till stor hjälp vid till exempel materialunderhåll och förnödenhetsförsörjning, genom att tillhandahålla stöd för processer som nyttjandebevakning och förvaringsplanering. Nyttjandebevakning av all använd utrustning, oavsett nationalitet, görs enkelt genom att systemet samarbetar med de andra deltagande ländernas logistiksystem. Användarna av systemet (svenskarna) har därmed en fullständig bild över vilka förnödenhetsresurser som finns tillgängliga när förnödenhetsförsörjning utförs, dvs. när något går sönder och måste bytas ut. Om exempelvis en GPS-sändare i en amerikansk helikopter gör sönder finns information om möjliga ersättningar, till exempel en kompatibel brittisk komponent med samma funktion. När något sådant händer registreras en felrapport i det amerikanska logistiksystemet, som automatiskt delegerar åtgärdandet till det svenska. Varje land använder således sitt eget logistiksystem som vanligt, men det svenska fungerar som en spindel i nätet och den svenska personalen som sköter logistiken beordrar åtgärdandet av felet.

Systemet för informationshantering som logistiksystemet använder har en tjänstebaserad arkitektur uppbyggd av semantiskt beskrivna webbtjänster. Informationssystemet är alltså ett lager, eller infrastruktur, som logistiksystemet bygger på. Varje lands eget logistiksystem kan användas av det svenska logistiksystemet, och tvärtom, genom infrastrukturen. Interoperabilitet möjliggörs av att (1) funktionalitet publiceras som webbtjänster, (2) informationsmodeller används för att beskriva domänspecifik information, och (3) informationsmodeller används även för att beskriva tjänster semantiskt. I dessa specificeras t.ex. vad en förnödenhet är. Genom att referera till informationsmodeller kan alla länders respektive logistiksystem tolka information på samma sätt, och tolkningsfel såsom förväxling av måtten meter och fot undviks enkelt.

3.3.2 Typfall

Det första typfallet som visar hur informationssystemet i logistiksystemet fungerar, hämtas från ”Nyttjandebevakning” [7]. Detta är en subprocess inom ”Materialunderhåll”, och innehåller i sin tur bland annat aktiviteten ”Planera nyttjande per individ”, där en individ är en fysisk sak av en viss förnödenhetstyp, t.ex. ett däck till ett patrullfordon. Denna aktivitet består bland annat av arbetssteget ”Välj individ för respektive uppdrag”. Detta arbetssteg har ett informationsbehov om vilka individer som finns tillgängliga. Detta informationsbehov tillfredsställs genom att det svenska logistiksystemet sammanställer alla tillgängliga individer från de olika länderna. Informationen erhålls genom att det svenska logistiksystemet anropar webbtjänster publicerade av andra länder, som tillhandahåller önskad information. Informationen tolkas på rätt sätt med hjälp av de använda informationsmodellerna.

Det andra typfallet, [9], hämtas från ”Beredning av förnödenhetsbehov” som är en subprocess inom ”Förnödenhetsförsörning” [8]. Denna subprocess innehåller i sin tur aktiviteten ”Detaljera förnödenhetsbehov”, som bland annat innehåller arbetssteget ”Kontrollera befintliga tillgångar”. På liknande sätt som i det första typfallet tillfredsställs detta informationsbehov genom att det svenska logistiksystemet kontrollerar befintliga tillgångar i lager hos de andra länderna.

Dessa två typfall visar hur det svenska logistiksystemet hämtar information från andra länders logistiksystem, men det motsatta sker också till exempel när ett logistikuppdrag inte kan fullföljas. Då underrättar det svenska logistiksystemet ett annat lands logistiksystem genom att anropa en webbtjänst som registrerar en felrapport i detta.

4 Realisering av scenario

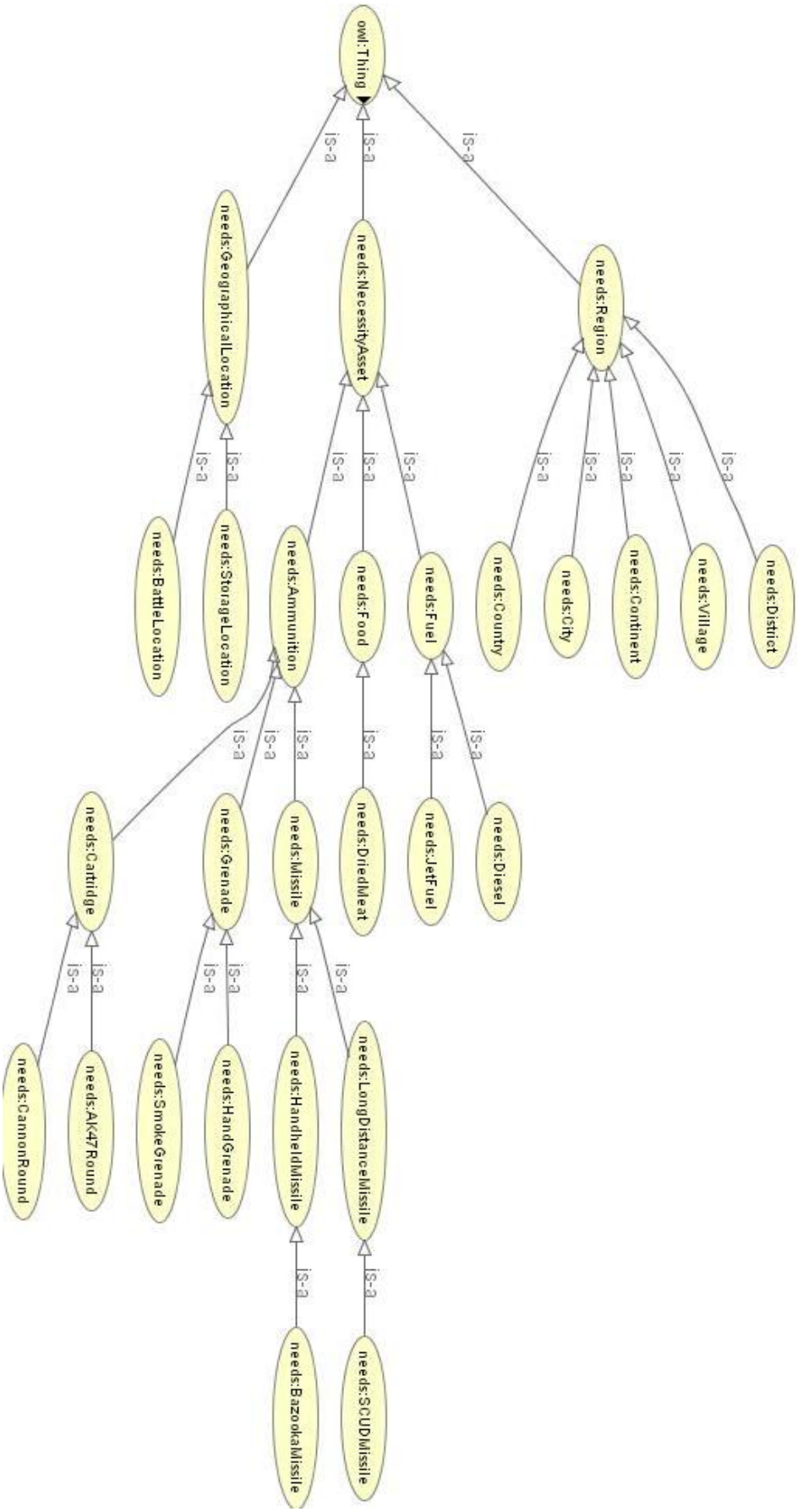
En prototyp av infrastrukturen har implementerats och scenariot har realiserats genom att tre olika fiktiva logistiksystem har skapats. Ett tillhör USA, ett tillhör Storbritannien, och det sista tillhör Sverige. Det svenska logistiksystemet fungerar som spindeln i nätet och styr all logistikverksamhet, medan de amerikanska och engelska systemen endast fungerar som agenter åt det svenska. Systemen tillhandahåller enbart funktionalitet för att kunna sammanställa alla tillgängliga förnödenhetstillgångar inom valfria regioner. Detta scenario representerar lite av ett extremfall, där interoperabilitet mellan tre helt olika system tillhörande tre olika nationer, ska realiseras. Logistiksystemen kunde likväl tillhöra tre olika domäner inom det svenska försvaret, t.ex. mark, luft och vatten.

4.1 Informationsmodeller

För att realisera scenariot har följande informationsmodeller utvecklats:

- *NecessityNeeds*, en global domänmodell (domänontologi) med begrepp för bl.a. olika typer av förnödenhetstillgångar och regioner, se figur 4-1.
- *LogisticServices*, en global tjänstemodell (tjänsteontologi) med beskrivningar av in- och utdata till de semantiska webbtjänsterna i scenariot. Här finns bland annat definitioner av listor med förnödenhetstillgångar, eftersom domänmodellen endast innehåller definitioner av enskilda instanser. Beskrivningar av innehållet i dessa listor refererar till den domänspecifika modellen.
- *ServiceParameters*, en global informationsmodell som egentligen tillhör tjänstemodellen. Innehåller beskrivningar av icke-funktionella parametrar.
- *USA*, en specifik informationsmodell för beskrivning av de tillgångar USA förfogar över i scenariot. Denna modell tillhör det amerikanska logistiksystemet, men det svenska systemet har åtkomst eftersom det kan styra de beskrivna tillgångarna.
- *UK*, en specifik informationsmodell för beskrivning av de tillgångar som Storbritannien förfogar över i scenariot. Det svenska systemet har åtkomst även till denna.

Ovanstående informationsmodeller är uttryckta i OWL DL, och har modellerats med hjälp av Protégé-2000 [40] och den tillhörande plugin-applikation för OWL som finns tillgänglig. Protégé är en ontologieditor utvecklad i ett projekt med öppen källkod.



Figur 4-1
De viktigaste OWL-klasserna i informationsmodellen NecessityNeeds.

4.2 Tjänster

Varje lands logistiksystem gör utvald funktionalitet tillgänglig för andra genom att publicera ett antal webbtjänster. Dessa är implementerade och publicerade med Apache Axis, ett Java-baserat ramverk för webbtjänster. Varje tjänst har både en WSDL-beskrivning och en OWL-S-beskrivning.

Logistiksystemen tillhandahåller funktionalitet för att kunna få fram vilka tillgångar respektive land har inom olika områden där den fredsbevarande operationen i scenariot pågår. Det amerikanska systemet publicerar följande två semantiska webbtjänster för detta ändamål:

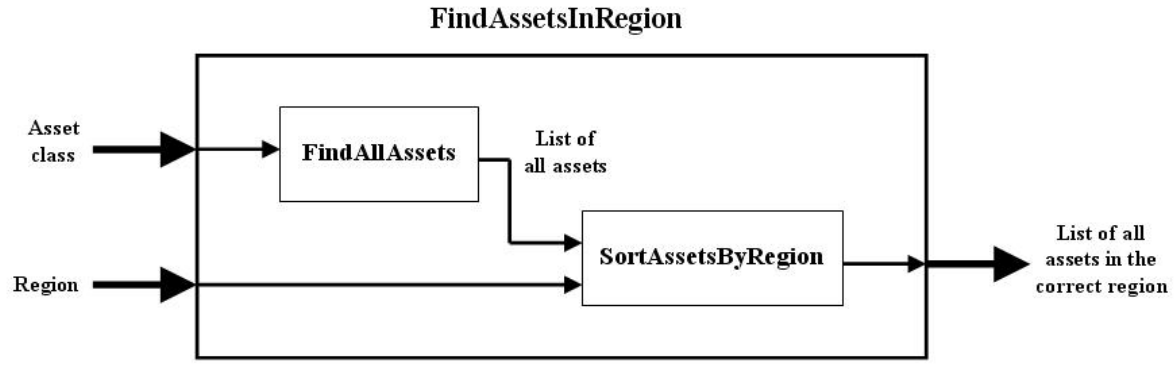
1- *FindAllAssets*, hittar samtliga tillgängliga förnödenhetstillgångar, t.ex. ammunition och bränsle, som USA förfogar över. Indata är en typ av tillgång och utdata är en lista av alla faktiska tillgångar av denna typ.

2- *SortByRegion*, sorterar ut de tillgångar som finns tillgängliga i ett visst område från en större mängd tillgångar. Indata är en lista av tillgångar och den region som är av intresse. Utdata är en delmängd av indatalistan med de tillgångar som finns i den angivna regionen.

Var och en av dessa har en semantisk beskrivning i OWL-S vars förankring är kopplad till respektive WSDL-beskrivning. Endast dessa två webbtjänster finns implementerade på den underliggande plattformen, men följande sammansatta webbtjänst finns också tillgänglig:

3- *FindAssetsInRegion*, se figur 4-2, är sammansatt av (1) och (2) och hittar alla tillgängliga förnödenhetstillgångar av en viss typ inom ett visst område. Indata är en typ av tillgång och en region. Utdata är alla tillgångar av den angivna typen som finns i den angivna regionen.

En OWL-S-beskrivning av denna tjänst finns, med en processmodell där (1) *FindAllAssets* och (2) *SortByRegion* ingår som atomära processer. Utdata från (1) utgör indatalistan till (2), och utdata från (2) är processens slutgiltiga utdata. *FindAssetsInRegion* existerar alltså enbart på den semantiska nivån.



Figur 4-2

Processmodellen för den sammansatta webbtjänsten FindAssetsInRegion, består av två atomära processer som realiseras av webbtjänsterna FindAllAssets och SortAssetByRegion.

Det amerikanska logistiksystemet har en centraliserad bild av alla tillgängliga förnödenhetstillgångar. De amerikanska webbtjänsterna kan därför användas av det svenska logistiksystemet med hjälp av tunna klienter eftersom mycket av funktionaliteten finns implementerad i webbtjänsterna. Det brittiska logistiksystemet är uppbyggt på ett annorlunda

sätt. Detta system har en mer distribuerad lösning med tillgångarna uppdelade i depåer. Varje depå innehåller endast en viss typ av tillgångar. Exempelvis innehåller *GrenadeDepot* endast granater. Istället för att ha en webbtjänst som kan returnera alla tillgångar ett land har, så har istället varje depå en egen webbtjänst som endast returnerar tillgångarna som finns i denna. Varje depå finns i en viss region, som anges i beskrivningen av tjänsten. Följande tio tjänster har implementerats inom det brittiska logistiksystemet:

- *AK47RoundDepot*, returnerar de tillgångar av typ *AK47Round* som finns tillgängliga i denna depå.
- *BazookaDepot1,* tillgångar av typ *Bazooka* på samma sätt som ovan.
- *BazookaDepot2*, som ovan osv.
- *CartridgeDepot*
- *FoodDepot1*
- *FoodDepot*
- *FuelDepot*
- *GrenadeDepot*
- *HandGrenadeDepot*
- *TomahawkMissileDepot*

Var och en av ovanstående tjänster är alltså publicerad av en depå som innehåller dessa tillgångar. Alla tar samma typ av input, en enkel invokeringssträng, och returnerar en lista av tillgångar som output. Alla har också en viss region som icke-funktionell serviceparameter. Depåer som innehåller flera typer av tillgångar publiceras som tillhandahållare av tillgångar av den mest specifika gemensamma typen. Exempelvis så innehåller *CartridgeDepot* patroner till både AK47or och kanoner, *Cartridge* (patron) anges därför som typen av tillgång i depån.

Lösningarna i det amerikanska och brittiska logistiksystemet har båda för- och nackdelar. Det amerikanska tillåter mer kontroll och hemlighållande eftersom systemet är centralt organiserat. Det tillåter också tunnare klienter, men är mindre flexibelt. Det brittiska systemet är mer distribuerat och flexibelt, men kräver mer av sina användare. Som tur är kan infrastrukturen i det semantiska nätverket mellan systemen avlasta dessa. För att skapa en fullständig bild över de amerikanska tillgångarna med traditionella webbtjänster krävs ett enda anrop, men för att se samtliga brittiska tillgångar behövs ganska tjocka klienter med en hel del funktionalitet. En infrastruktur av semantiska webbtjänster kan här avlasta klienterna genom att ta över en stor del av funktionalitetsbördan. Genom en sökfråga efter en sammansatt semantisk webbtjänst, som returnerar en fullständig bild över de brittiska tillgångarna, krävs även här endast ett enda tjänsteanrop. De tjocka klienterna kan göras tunnare eftersom hela arbetet med att hitta och fråga enstaka webbtjänster utförs automatiskt av registret och kompositören i infrastrukturen.

Det svenska logistiksystemet ska kunna utföra avancerade tjänster som nyttjandebevakning och förnödenhetsförsörjning för alla trupper som ingår i FN-styrkan. Detta har dock inte implementerats, utan endast funktionalitet för att sammanställa alla tillgängliga tillgångar, såväl amerikanska som brittiska. Med hjälp av ett grafiskt gränssnitt kan en användare i det svenska systemet söka efter tjänster och exekvera dem för att se vilka tillgångar som finns var (se nedan).

4.3 Register

Ett semantiskt register har implementerats. En egen datamodell för att representera tjänstebeskrivningar i OWL-S antas för enkelhetens skull endast lagra profiler av tjänster. Detta är dock fullt tillräckligt för att utföra meningsfull semantisk matchning mellan tjänster i denna prototyp. När tjänster publiceras registreras alltså deras profiler och används senare för att matcha en tjänsteförfrågan mot. Det är därför naturligt att representera en tjänsteförfrågan med en ihopsatt OWL-S-profil som beskriver den funktionalitet som önskas. Registret är utrustat med en algoritm som utför generell semantisk matchning av denna profil mot alla de profiler som finns registrerade i registret. De träffar som konstateras rankas sedan efter hur väl de stämmer överens med förfrågan. Matchningsförfarandet beskrivs i nästa avsnitt.

4.3.1 Semantisk matchning

4.3.1.1 Översikt

För att matcha en sökfråga mot en tjänsteannons används en algoritm som matchar in- och utdata var för sig. Varje tjänst måste därför passera algoritmen två gånger. Resultatet från matchningen av in- respektive utdata kan sedan kombineras eller redovisas var för sig.

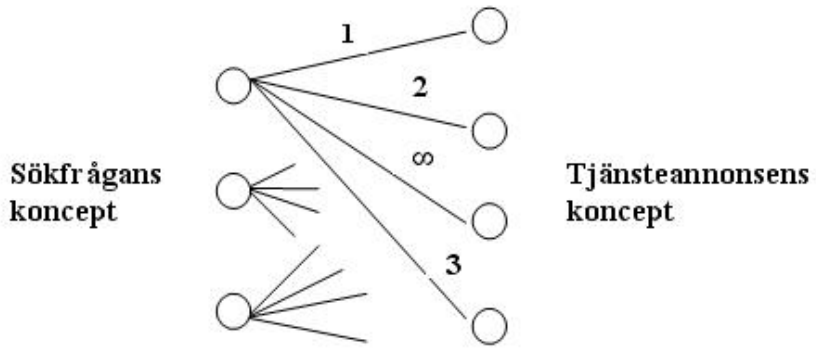
Den semantiska matchning som utförs här är kraftigt förenklad till att endast ta hänsyn till in- och utdata, samt icke-funktionella parametrar. Trots det var det en relativt stor uppgift att implementera ett register med semantisk matchning. Att verkligen implementera en generell, fullständig och effektiv semantisk matchning som tar hänsyn till alla delar av OWL-S, även processmodeller, är därför enligt våra erfarenheter varken en liten eller trivial uppgift.

4.3.1.2 Detaljer

Grundtanken bakom algoritmen är att semantisk matchning av en tjänsts in- eller utdata kan reduceras till en bipartit matchning med viktade kanter. En *kant* förbinder två noder i en graf, och en *vikt* på denna kan ses som en kostnad för att använda den. I en *bipartit* graf kan alla noder delas upp i två kolumner, så att alla kanter går ifrån den ena kolumnen till den andra. I figur 4-3 visas en bipartit graf. Den vänstra kolumnen innehåller begreppen från en sökfrågas in- eller utdata, och den högra kolumnen innehåller motsvarande från en tjänsteannons. Varje begrepp till vänster kan antas vara sammankopplat med alla begrepp till höger, där vikten på kanten mellan två begrepp avspeglar hur pass väl de matchar varandra. Denna vikt fås genom att matchningsalgoritmen i [12] används enligt följande:

- En exakt matchning får vikt 1.
- En plugin-matchning får vikt 2.
- En inordnings-matchning får vikt 3.
- Ett misslyckande får en oändlig vikt.

Målet är att hitta en bipartit matchning med minimal vikt, i vilken varje begrepp till vänster (sökfrågan) ingår. Omatchade begrepp till höger tillåts eftersom all in- eller utdata från en passande tjänsteannons inte behöver användas. Vid matchning av utdata kan omatchade begrepp ses som överflödig funktionalitet, och vid matchning av indata antas att omatchade begrepp antingen ignoreras eller kompletteras.



Figur 4-3

En bipartit graf mellan begrepp. Det översta begreppet till vänster respektive höger matchar varandra exakt.

Algoritmen som åstadkommer en bipartit matchning av minimal vikt beskrivs i [26], och är en blandning av Ford-Fulkersons algoritm [24] och Dijkstras algoritm [25]. Ford-Fulkersons algoritm används för att hitta en bipartit matchning, och Dijkstras algoritm till att hitta den kortaste och billigaste stigen från vänster till höger i figur 4-3. Algoritmen består av tre steg och är som följer:

1- Hitta den billigaste stigen från vänster till höger med Dijkstras algoritm. Alla vikter antas vara positiva. Det ”kortaste avståndet” (minsta kantsumma) till varje nod, F(v), beräknas samtidigt.

2- Modifiera vikterna på alla kanter enligt $w'_{i,j} = w_{i,j} + F(v_i) - F(v_j)$, där $w_{i,j}$ är vikten på kanten från nod i till nod j, samt där $F(v_i)$ och $F(v_j)$ är det kortaste avståndet till nod i respektive nod j. Därmed gäller fortfarande $F(v_j) \leq F(v_i) + w_{i,j}$ så att inga vikter blir negativa efter justering.

3- Lägg till den billigaste stigen som just tagits fram i matchningen som vi håller på att bygga upp, genom att vända riktningen på de kanter som ingår i den billigaste stigen. Gå tillbaka och upprepa steg 1-3 tills ingen stig från vänster till höger finns längre.

När ingen stig från vänster till höger hittas längre har en maximal matchning, dvs. en matchning med så många kanter som möjligt, och med minimal kantsumma hittats. Bevis för att algoritmen åstadkommer detta finns i [26]. Denna matchning utgör alltså den bästa möjliga matchningen mellan in- eller utdata i sökfrågan och i den aktuella annonsen. För att accepteras måste förstås denna matchning innehålla alla begreppen som angavs i sökfrågan, dvs. alla begreppen i den vänstra kolumnen i figur 4-3. Algoritmen matchar som sagt varje gång en sökfrågas in- *eller* utdata mot motsvarande i *en* tjänsteannons. För att hitta den totalt bäst passande annonsen måste alltså algoritmen köras två gånger för varje annons i registret. För att kunna ranka hela annonser efter hur pass bra de matchar sökfrågan, måste resultatet från båda körningarna av algoritmen för en och samma annons kombineras. Någon slags policy för hur detta ska gå till behövs, eftersom annonsen med den bästa matchningen av utdata, mycket väl också kan ha den sämsta matchningen av indata. Enligt [12] bör utdata prioriteras, eftersom bäst matchning av utdata innebär att den hittade tjänsten är den som bäst uppfyller användarens krav på funktionalitet. Indata kan i värsta fall kompletteras eller delvis ignoreras. När en policy för hur matchningen av hela annonser, inte bara in- eller utdata, fastställts kan de sorteras och presenteras.

4.4 Kompositör

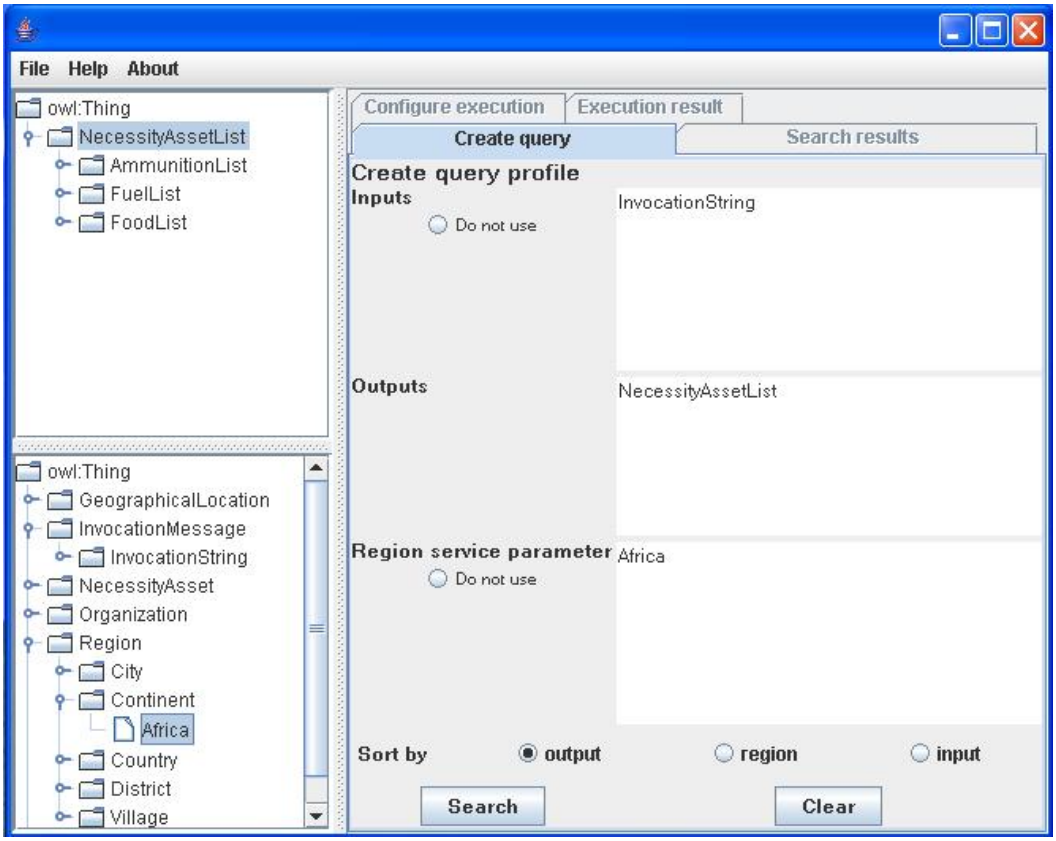
Någon komplett kompositör har inte implementerats eftersom det är alldeles för tidskrävande. Algoritmen för matchning som beskrevs i det förra avsnittet påminner om kompositionsalgoritmen i [16]. Eventuellt skulle därför en kompositör till denna prototyp ha kunnat implementeras utifrån den algoritmen, men eftersom syftet med prototypen endast är att demonstrera användbarheten lades ingen tid på detta. Den kompositör som används i prototypen skapar alltså inte sammansättningar dynamiskt, utan är för enkelhetens skull istället försedd med sammansättningar skapade i förväg. Kompositören kan därför endast svara på de komplexa sökfrågor för vilka det finns en färdig sammansatt webbtjänst.

4.5 Grafiskt gränssnitt

Ett grafiskt gränssnitt har implementerats för det svenska logistiksystemet. Med detta kan en användare söka efter tjänster och exekvera dem för att se vilka tillgångar som finns tillgängliga i olika regioner. Gränssnittet är kopplat till en kompositör. Kompositören i sin tur använder ett semantiskt register där alla amerikanska och brittiska webbtjänster finns publicerade. Gränssnittet kan ses som ett användargränssnitt till det svenska logistiksystemet. Genom gränssnittet kan användare få en överblick över alla tillgängliga tillgångar inom olika regioner, såväl brittiska som amerikanska. Exempelvis kan alla tillgångar i hela Afrika tas fram, eller bara de i en av huvudstäderna till länderna i scenariot.

Gränssnittet erbjuder möjlighet att:

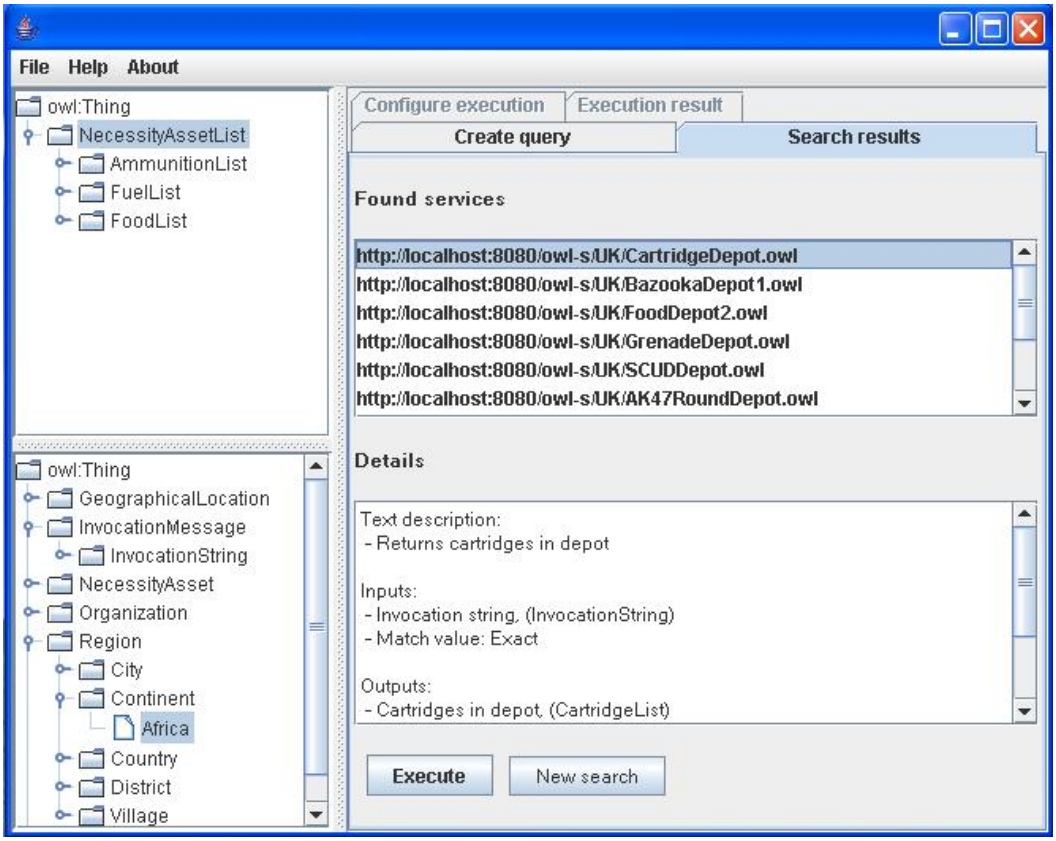
1- Sätta ihop en OWL-S-profil som utgör en sökfråga till det semantiska registret, och söka efter semantiska webbtjänster, se figur 4-4. Till vänster i figuren ses två träd. Det övre innehåller de begrepp och individer som finns i informationsmodellen för tjänster, *LogisticServices*, och det undre innehåller de som finns i den domänspecifika informationsmodellen, *NecessityNeeds*. Genom att välja i träden kan typer av in- och utdata i sökfrågan specificeras. Valda begrepp eller individer dyker då upp i ett av de vita fälten till höger. Det brittiska logistiksystemets tjänster beskrivs också med en serviceparameter, *Region*, som anger i vilken region den depå som tjänsten tillhör finns. Värdet på denna parameter kan väljas ur det undre trädet, se figur 4-4 där *Africa* valts. Indata och serviceparameter behöver ej anges, utan matchning av tjänster kan ske utan att ta hänsyn till endera av dessa, genom att användaren markerar *'Do not use'*. Hur de lokaliserade tjänsterna ska sorteras kan väljas genom att markera ett av alternativen ovanför sökknappen. *Output* innebär en sortering där den tjänst vars utdata matchade sökfrågan bäst hamnar överst i listan osv. När sökfrågan är färdig klickar användaren på knappen *'Search'* för att starta sökningen.



Figur 4-4

En användare kan skapa en OWL-S-profil och använda denna som sökfråga vid sökning efter semantiska webbtjänster. Begrepp och individer väljs ur träden till vänster, och dyker upp i fälten till höger. Användaren kan välja att inte matcha tjänster mot indata eller serviceparameter genom att markera 'Do not use'. Hur resultaten av sökningen ska sorteras kan väljas genom att markera ett av alternativen ovanför sökknappen. När sökfrågan är färdig startas sökningen genom att klicka på 'Search'.

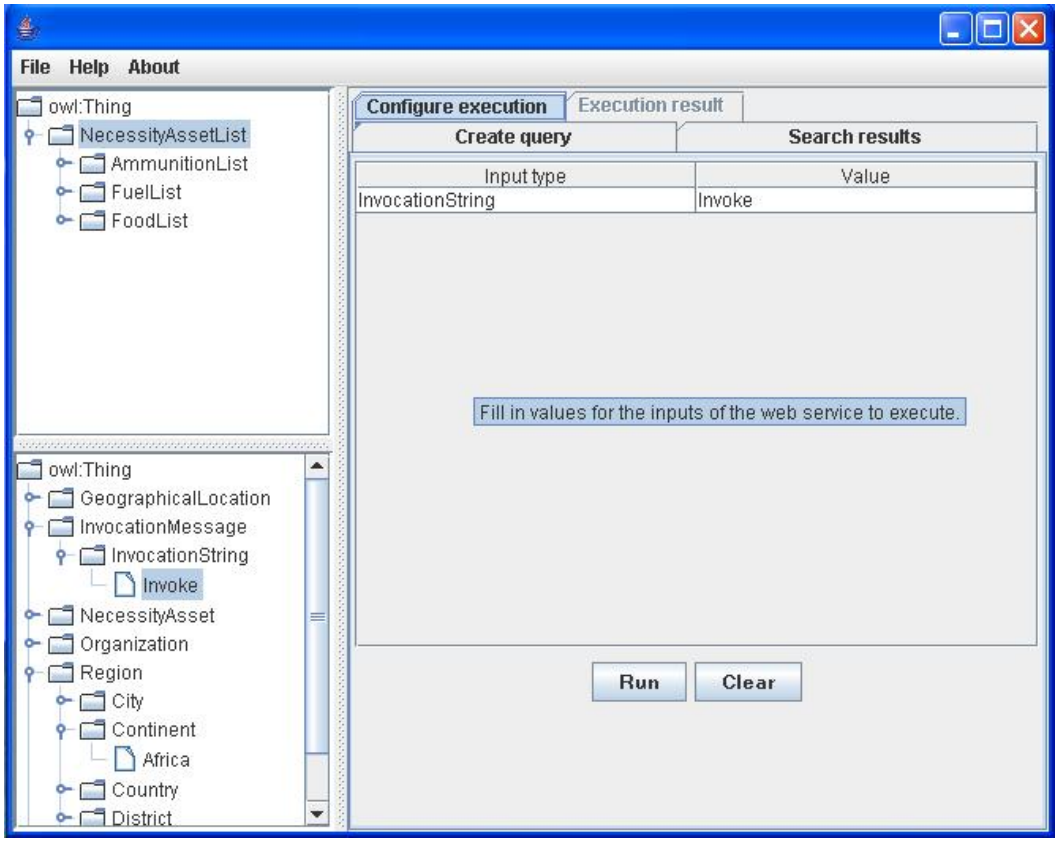
2- Se de tjänster som påträffas vid en sökning samt detaljerad information om dessa, se figur 4-5. Denna flik blir tillgänglig när användaren har fyllt i en sökfråga och utfört en sökning. Träden till vänster är synliga även nu, men används inte här. I det övre fönstret till höger syns alla hittade webbtjänster. Tjänsterna är sorterade efter hur väl de matchade sökfrågan, enligt det sätt som valdes i den första fliken. Detaljerad information kan ses i det undre fönstret till höger genom att markera en av tjänsterna i det övre fönstret. Denna information omfattar en textbeskrivning, vilken typ av in- och utdata tjänsten tar, och eventuell serviceparameter. Användaren kan nu välja en av de påträffade tjänsterna och exekvera den, genom att klicka på *'Execute'*, eller utföra en ny sökning genom att klicka på *'New search'*.



Figur 4-5

Resultaten efter en sökning efter tjänster. De tjänster som påträffades visas i det övre fönstret till höger, och mer detaljerad information om en markerad tjänst visas i det undre. En vald tjänst kan exekveras genom att klicka på 'Execute', eller så kan en ny sökning utföras genom att klicka på 'New search'.

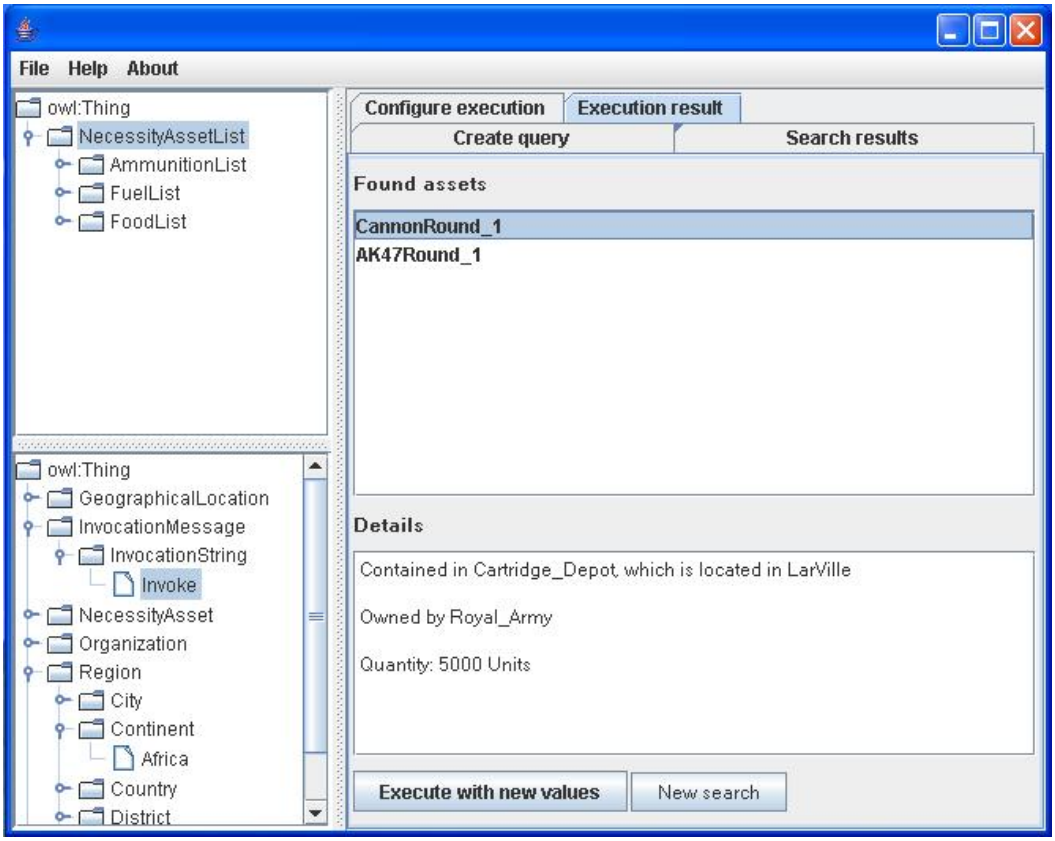
3- Välja en av de tjänster som hittades och exekvera den, se figur 4-6. Denna flik blir tillgänglig när användaren valt att exekvera en hittad tjänst i den föregående fliken. Här måste användaren konfigurera exekveringen av tjänsten genom att välja *värden* på tjänstens indata. Vid sökningen anges som regel endast en *typ* av indata. Tabellen till höger visar alla indataparametrar den valda tjänsten behöver. Typen av indata visas till vänster i tabellen. I figuren är denna av typ *InvocationString*, och kan utgöras av en enkel textsträng. Användaren måste fylla i ett värde av denna typ. Detta kan göras genom att välja ur träden till vänster, precis som i den första fliken. Skillnaden mot den första fliken är att där väljs som regel begrepp, vilka beskriver en typ av parameter, medan här väljs som regel individer, vilka beskriver verkliga värden. Det är dock en skillnad som inte alltid gäller, och därför kan användaren ange såväl begrepp som individer här. När alla nödvändiga värden fyllts i klickar användaren på *'Run'* för att utföra själva exekveringen.



Figur 4-6

En användare konfigurerar här exekveringen av en tjänst, genom att fylla i verkliga värden på de indataparametrar tjänsten kräver. Värden väljs ur träden till vänster, på samma sätt som i den första fliken. När alla värden fyllt i klickar användaren på 'Run' för att utföra själva exekveringen.

4- Se vilka förnödenhetstillgångar den exekverade tjänsten returnerar, se figur 4-7. Denna flik blir tillgänglig när en tjänst exekverats via den föregående fliken. Resultaten av exekveringen, dvs. de förnödenhetstillgångar som tjänsten returnerar, visas sedan i det övre fönstret till höger. Mer detaljerad information om varje tillgång kan ses i det undre fönstret till höger genom att markera en tillgång i det övre fönstret. Denna information beskriver var tillgången finns, vilken organisation den tillhör, samt vilka kvantiteter som är tillgängliga. Varje tillgång är här en mängd av enheter för att spara arbete med att lägga till och beskriva varje tillgång. För att exekvera samma tjänst igen, men med nya värden, kan användaren klicka på *'Execute with new values'*, eller så kan en ny sökning utföras genom att klicka på *'New search'*.



Figur 4-7

Resultaten av att exekvera en tjänst, dvs. de förnödenhetstillgångar som returnerades, visas i det övre fönstret till höger. Mer detaljerad information kan ses i det undre fönstret till höger genom att markera en tillgång i det övre fönstret. Samma tjänst kan exekveras igen, men med nya värden, genom att klicka på 'Execute with new values'. En ny sökning kan också utföras genom att klicka på 'New search'.

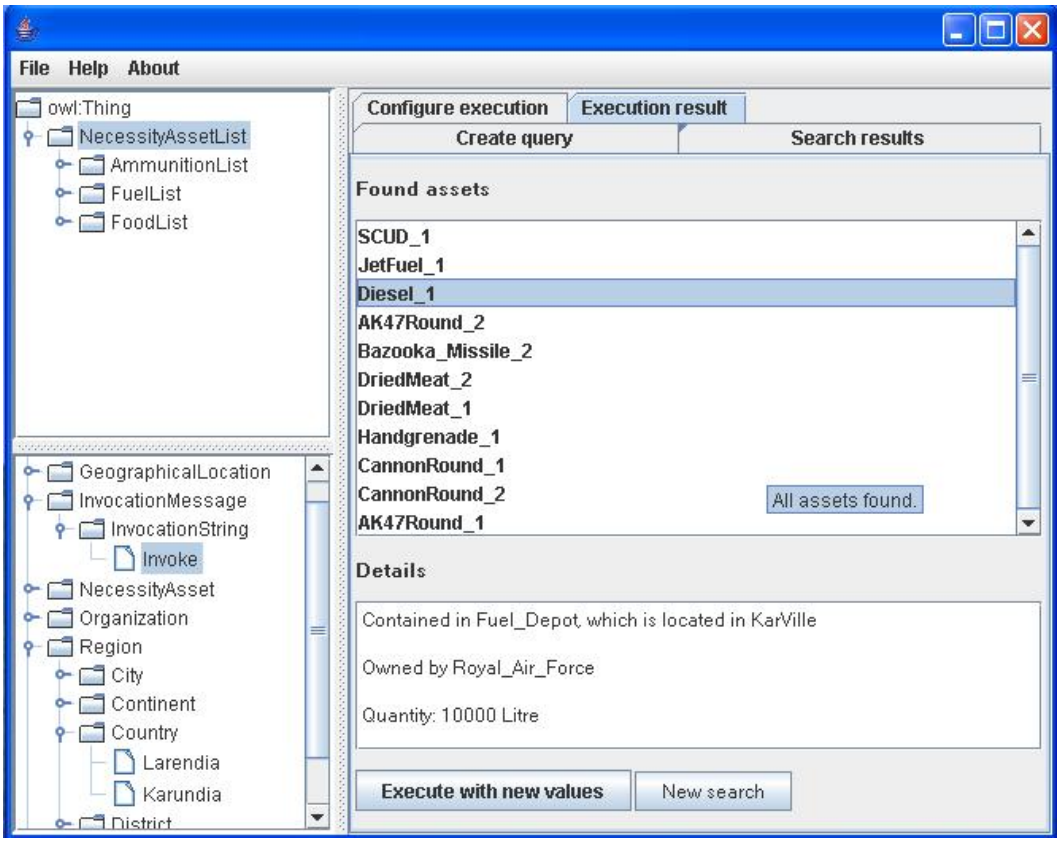
Gränssnittet som implementerats här utför alltså sökningar och returnerar en sorterad lista av passande tjänster, i vilken en användare sedan får välja en passande tjänst och till sist exekvera den. Sökningen och exekveringen av en webbtjänst är därmed endast halvautomatisk. Det är dock viktigt att notera att denna begränsning endast beror på utformningen av själva gränssnittet, inte infrastrukturen. Den underliggande infrastrukturen möjliggör helt automatiska urval och exekvering av tjänster. Ansvaret för att detta sker ligger dock hos de applikationer som använder den. Ett annat gränssnitt skulle till exempel kunna fungera så att en användare skapar önskad profil samtidigt som värden anges. En sökning efter tjänster kan då utföras, den bästa väljas och exekveras, allt i ett steg. En användare upplever då endast ett knapptryck jämfört med den flerstegsprocessen som upplevs med detta gränssnitt. Syftet med gränssnittets utformning här är att bättre kunna demonstrera vad som sker inuti infrastrukturen.

4.6 Resultat

Realiseringen av scenariot ledde till en prototyp av en infrastruktur för informationshantering där information kan hämtas automatiskt från heterogena datakällor, baserat på semantiska sökningar. Datakällorna är implementerade och tillgängliga som semantiska webbtjänster. Kärnan i infrastrukturen består av ett semantiskt register och en kompositör som tillsammans klarar av att besvara komplexa sökfrågor på en semantisk nivå. Det semantiska registret kan sedan utföra semantisk sökning utifrån denna sökfråga, och hitta tjänster baserat på deras funktionalitet istället för enbart nyckelord. Om befintliga tjänster inte kan besvara en

sökfråga, undersöker kompositören ifall en sammansättning av flera webbtjänster kan besvara frågan. I så fall skapas denna sammansättning dynamiskt. Vald tjänst eller sammansättning kan sedan exekveras och resultatet visas upp.

Användbarheten av infrastrukturen framgår bland annat av möjligheten att avlasta klienter till webbtjänster, så att dessa kan göras tunnare. Till exempel kan en sökning efter samtliga brittiska tillgångar i scenariot, om den görs med en infrastruktur som den beskriven här, förenklas till en enda sökfråga. Om samma sökning gjorts i en infrastruktur uppbyggd av traditionella webbtjänster, så skulle en sökning efter varje tänkbar typ av tillgång behövts göras, och sedan sammanställas. Att automatisera denna process ökar användbarheten mycket, eftersom sådan hantering av stora mängder data kan utföras mycket effektivare av maskiner. Figur 4-8 visar resultatet av att exekvera en sammansättning av webbtjänster som kan returnera alla tillgängliga tillgångar i landet Karundia (se scenariot). Tillgångarna är sammanställda ifrån fem olika brittiska förnödenhetsdepåer och två olika amerikanska förnödenhetslager, vilka finns i olika delar av Karundia.



Figur 4-8

Resultatet av att exekvera en sammansättning av webbtjänster som returnerar alla tillgångar i landet Karundia, (se scenariot).

I sökfrågan för att skapa sammansättningen ovan angavs *NecessityAsset*, dvs. den mest generella typen av tillgång, som den typ av tillgång som efterfrågades. Trots det hittades alla typer av tillgångar som fanns tillgängliga. Här märks därför hur den semantiska matchningen fungerar. De olika tillgångarna refererar inte till *NecessityAsset* på något annat sätt än att deklarera att de alla är av typer som beskrivs av dess subbegrepp. Något gemensamt nyckelord finns alltså inte, utan tillgångarna hittas med hjälp av semantisk information om vilken typ av tillgång de tillhör.

Prototypen som utvecklats klarar av att göra det som utlovas i realiseringen av scenariot, med reservation för att kompositören egentligen inte skapar sammansatta webbtjänster dynamiskt, och därför endast kan svara på de komplexa sökfrågor för vilka en komposition gjorts i förväg.

5 Diskussion

5.1 Nyttan av semantisk information

Vid informationshantering kommer i framtiden en tjänstebaserad arkitektur att vara nödvändig för att kunna hantera en ständigt föränderlig omvärld, där anpassningsförmåga krävs. Precis som i rapporten IRM Vision [2] förespråkas här IT-baserade tjänster med metadatabeskrivningar för att skapa en sådan. Det främsta skälet är den grundläggande interoperabilitet som är möjlig dels av att sådana tjänster är plattformsoberoende, och dels av att de kan sättas in i ett sammanhang och därför återanvändas i helt olika system utan ändringar. Möjligheten för maskiner att tolka information på semantisk nivå, gör den semantiska informationen än mer värdefull, eftersom automatisering då möjliggörs i högre grad. Automatisering innebär att betydligt större mängder information kan hanteras, på kortare tid och på ett konsekvent sätt.

Ett exempel på nyttan av semantisk information kan tas ur scenariot beskrivet tidigare, där uppgiften kan vara att hitta alla tillgångar av typen *Ammunition* i landet *Karundia*. Med webbtjänster försedda med semantiska beskrivningar räcker det med att ställa en enda fråga. Frågan kan sedan besvaras genom att alla tillgängliga webbtjänster betraktas, matchas och eventuellt används. Därigenom fås ett fullständigt svar, dvs. inga tillgångar utelämnas i svaret. För att åstadkomma samma sak utan semantiska beskrivningar skulle alla webbtjänster behöva behandlas manuellt, vilket skulle kunna ta mycket lång tid.

Ytterligare ett exempel på nyttan av semantisk information, också det taget ur scenariot beskrivet tidigare, kan vara att hitta en depå med *Diesel*. Enligt scenariot publicerar varje depå en webbtjänst som talar om vilka tillgångar depån innehåller. Uppgiften kan lösas genom att registret manuellt söks igenom med en vanlig nyckelordsbaserad sökning efter ordet ’diesel’. En stor nackdel är då förstås att om tjänsten inte är publicerad med exakt det ordet, så kommer motsvarande depå inte att hittas. Exempelvis kanske orden ’vanligaste lastbilsbränslet’ användes istället. Samma problem leder också till att en depå som publicerats med ordet ’bränsle’, men som innehåller både diesel och bensin, helt missas. För att hitta sådana måste en användare explicit söka efter just ordet ’bränsle’. I fall där en tjänst kan ha publicerats med något av många alternativa ord, kommer användaren att behöva ha stort ordförråd och göra många sökningar för att inte missa en existerande depå. Med semantiska webbtjänster undviks sådana problem enkelt eftersom en sökning efter depåer baserad på begreppet *Diesel* istället för ordet, faktiskt skulle returnera en depå med *Bränsle* som en träff utan att extra sökningar skulle krävas.

Semantiska beskrivningar är således avgörande för att få webbtjänster att bli de fristående moduler av inkapslad funktionalitet de är tänkta att vara. Utan sådana beskrivningar är visserligen funktionaliteten inkapslad, men kunskap om hur den är avsedd att användas saknas. Det har tidigare framgått att det inte finns någon skillnad mellan vanliga webbtjänster och semantiska webbtjänster gällande funktionaliteten hos själva tjänsten. Skillnaden ligger i hur denna beskrivs och förmedlas. En vanlig webbtjänst kan i detta sammanhang ses som ett stycke funktionalitet som måste kännas till i förväg och manuellt sättas i bruk. Med semantiska beskrivningar kan denna istället hittas när den behövs och automatiskt sättas in i ett sammanhang på ett flexibelt sätt, och därför vara till mycket större nytta.

Webbtjänster med kraftfulla semantiska beskrivningar är som här framgår det bästa sättet att skapa en tjänstebaserad arkitektur för informationshantering, eftersom möjligheterna till automatiserad lokalisering, sammansättning och exekvering är stora.

6 Akronymer

BPEL4WS – Business Process Execution Language For Web Services
DL – Description Logic
HTN – Hierarchical Task Network
IOPE – Inputs, Outputs, Preconditions and Effects
IRM – Information Resource Management
IRS – Internet Reasoning Service
LL – Linear Logic
M&S – Modellering och simulering
MWSDI – METEOR-S Web Service Discovery Infrastructure
NBF – Nätverksbaserade försvaret
OWL – Web Ontology Language
OWL-S – OWL Services
P2P – Peer to Peer
RDF – Resource Description Framework
RDFS – RDF Schema
RLS - Resursledningsystem
SOAP – Simple Object Access Protocol
TAV - Total Asset Visibility
URI – Uniform Resource Identifier
UDDI – Universal Description, Discovery and Integration protocol
UML – Unified Modeling Language
W3C – World Wide Web Consortium
WSDL – Web Service Description Language
XML – Extensible Markup Language
XSD – XML Schema
XSL – Extensible Stylesheet Language
XSLT – Extensible Stylesheet Language Transformations
XTRO - Extended Registries Ontology

7 Referenser

[1] – *"Informationshantering & -modellering till stöd för logistisk M&S – en förstudie"*, M. Eklöf, FOI, ISSN: 1650-1942, 2003.

[2] – *"IRM VISION, Rapport från MS 850 AO 610530"*, J. S Magnusson m.fl., FMV, 2003.

[3] – *"Pitfalls of OWL-S – A Practical Semantic Web Use Case"*, S. Balzer m.fl., ICSOC'04, New York, 2004.

[4] – *"Ontology integration and evolution"*, M. Hall, SE Data & Knowledge Engineering, 2004.

[5] – *"Learning to Match Ontologies on the Semantic Web"*, A. Doan m.fl., The VLDB

Journal Manuscript.

[6] – *"OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns"*, A. Rector m.fl., University of Manchester.

[7] - *"Slutrapport projekt VSHMOD-UH-2010 Fas 1 Materialunderhåll"*, S, Andersson m.fl., FMV, 2003.

[8] – *"Slutrapport projekt VSHMOD-UH-2010 Fas 2 Förnödenhetsförsörjning"*, S, Andersson m.fl., FMV, 2003.

[9] – *Typfall för VSHMOD*, Liselotte Karlow., FMV, 2004.

[10] – "*Importing the Semantic Web in UDDI*", M. Paolucci, m.fl.

[11] – "*Adding OWL-S to UDDI, implementation and throughput*", N. Srinivasan m.fl.

[12] – "*Semantic Matching of Web Services Capabilities*", M. Paolucci m.fl.

[13] – *"Discovery of Web Services in a Federated Registry Environment"*, K. Sivashanmugam m.fl., University of Georgia, USA.

[14] – *"The Semantic Web"*, T. Berners-Lee m.fl., Scientific American, maj 2001.

[15] – "*Semi-Automatic Composition of Web Services using Semantic Descriptions*", E. Sirin m.fl., University of Maryland, USA.

[16] – "*Automatic Composition of Semantic Web Services*", R. Zhang m.fl., University of Georgia, USA.

[17] – "*HTN Planning for Web Services Composition Using SHOP2*", E. Sirin m.fl., University of Maryland, USA.

[18] – "*Logic-based Web Services Composition: from Service Description to Process Model*", J. Rao, P. Küngas, Norwegian University of Science and Tecnology, Norway, M. Matskin, Royal Institute of Technology, Sweden.

[19] – *"Matchmaking of Web Services Based on the DAML-S Service Model"*, S. Bansal, J. M. Vidal, University of South Carolina, USA.

[20] – *"IRS-II: A Framework and Infrastructure for Semantic Web Services"*, E. Motta m.fl., The Open University, Milton Keynes, UK.

[21] – *"EDUTELLA: A P2P Networking Infrastructure Based on RDF"*, W. Nejdl m.fl., University of Hannover, Germany.

[22] – *"IRM VISION, Rapport från MS 850 AO 610530"*, J. S Magnusson m.fl., FMV.

[23] – *"Slutrapport projekt VSHMOD-UH-2010 Fas 2 Förnödenhetsförsörjning"*, S. Andersson m.fl., FMV.

[24] – Ford-Fulkersons algoritm.

[25] – Dijkstra algoritm.

[26] – Kapitel 13, föreläsningsanteckningar, kurs Avancerade Algoritmer, NADA, KTH, Johan Håstad, 2000.

[27] – *" XSL Transformations (XSLT)"*, J. Clark. Se webbsida http://www.w3.org/TR/xslt.

[28] – *" DAML-S 0.7 Draft Release"*, E. Buckley m.fl. Se webbsida http://www.daml.org/services/daml-s/0.7/.

[29] – *"Resource Description Framework (RDF)"*, se webbsida http://www.w3.org/RDF/.

[30] – *"RDF Vocabulary Description Language 1.0: RDF Schema",* se webbsida http://www.w3.org/TR/rdf-schema/.

[31] – *"Web Ontology Language (OWL)",* Eric Miller, Jim Hendler, se webbsida http://www.w3c.org/2004/OWL/.

[32] – *"OWL-S: SemanticMarkup for Web Services"*, D. Martin m.fl. Se webbsida http://www.daml.org/services/owl-s/1.1.

[33] – *"Business Process Execution Language for Web Services Version 1.1"*, T. Andrews Microsoft m.fl., 5 maj 2003, se http://www-128.ibm.com/developerworks/library/ws-bpel/.

[34] – "*DAML+OIL"*, I. Horrocks m.fl. Se webbsida http://www.daml.org/2001/03/daml+oil-index.html.

[35] – *"Unified Modeling Language (UML)"*, Object Management Group, Inc. Se webbsida www.omg.org/uml.

[36] – *"Web Services Description Language (WSDL) 1.1"*, E. Christensen m.fl. Se webbsida http://www.w3.org/TR/wsdl.

[37] – *" Simple Object Access Protocol (SOAP) 1.1"*, D. Box m.fl. Se webbsida http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[38] – *"UDDI Version 3.0.2"*, L. Clement m.fl. Se webbsida http://uddi.org/pubs/uddi_v3.htm.

[39] – *"XML Schema"*, C. M. Sperberg-McQueen, H. Thompson. Se webbsida http://www.w3.org/XML/Schema.

[40] – *Protégé*, En ontologieditor utvecklad inom The Protégé Project, http://protege.stanford.edu/.