FOI-R--1128--SE
December 2003
ISSN 1650-1942
Metodrapport

Martin Eklöf

Informationshantering & -modellering till stöd för logistisk M&S - en förstudie

Systemteknik
172 90 Stockholm

TOTALFÖRSVARETS FORSKNINGSINSTITUT
Systemteknik
172 90 Stockholm

FOI-R--1128--SE
December 2003
ISSN 1650-1942
Metodrapport

Martin Eklöf

Informationshantering & -modellering till stöd för logistisk M&S - en förstudie

Utgivare Totalförsvarets Forskningsinstitut - FOI Systemteknik 172 90 Stockholm	**Rapportnummer, ISRN** FOI-R--1128--SE	**Klassificering** Metodrapport
	Forskningsområde 2. Operationsanalys, modellering och simulering	
	Månad, år December 2003	**Projektnummer** E6872
	Verksamhetsgren 5. Uppdragsfinansierad verksamhet	
	Delområde 23 Logistik	
Författare/redaktör Martin Eklöf	**Projektledare** Martin Eklöf	
	Godkänd av Monica Dahlén	
	Uppdragsgivare/kundbeteckning FMV	
	Tekniskt och/eller vetenskapligt ansvarig Farshad Moradi	

Rapportens titel

Informationshantering & -modellering till stöd för logistisk M&S - en förstudie

Sammanfattning (högst 200 ord)

Visionen av ett framtida logistiskt system, inom det nätverksbaserade försvaret (NBF), omfattas av tjänster (verktyg) som underlättar verkställande av uppsatta strategiska, operativa och taktiska mål. Inom detta system kommer LRS (resursledningssystem), TAV (*Total Asset Visibility*) och Modellering och simulering (M&S) att spela en avgörande roll, tex. vid prognostisering av krav på framtida försvarssystem eller vid planering av kommande militära operationer. Alla dessa områden kräver tillgång till data av skilda slag. Utifrån de sätt med vilka dessa data hanteras idag kommer det dock att bli komplicerat för framtida logistiska system/förmågor/verktyg att utnyttja informationsmängden på ett effektivt sätt. Vad som krävs av ett framtida logistiskt system är att utifrån krav från informationskonsumerande system automatiskt kunna integrera den eftersökta informationsmängden från heterogena datakällor inom nätverket.

Denna förstudie syftar till att studera nya möjligheter för integrering av information från heterogena datakällor. Med integrering avses i detta fall automatisk lokalisering, urval, sammansättning och exekvering av tjänster som kan förväntas leverera en efterfrågad informationsmängd. Arbetet tar sin utgångspunkt i det arbete som bedrivits inom ”the Semantic Web” och beskriver teknologier och standarder som förväntas bidra till ett tjänsteutnyttjande baserat på ontologisk grund. I detta fall avses teknologier som RDF (”Resource Description Framework”), DAML-S (”Darpa Agent Mark-up Language – Services”) och ”Web Services”, som förväntas kunna automatisera och förenkla utnyttjande av tjänster på nätet, samt lägga grunden till interoperabilitet mellan heterogena system.

Nyckelord

Logistik, M&S, Informationshantering, Informationsmodellering, Semantiska Webben, RDF, Ontologi

Övriga bibliografiska uppgifter	**Språk** Svenska
ISSN 1650-1942	**Antal sidor:** 45
Distribution enligt missiv	**Pris: Enligt prislista**

Issuing organization	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency Systems Technology SE-172 90 Stockholm	FOI-R--1128--SE	Methodology report
	Programme Areas 2. Operational Research, Modelling and Simulation	
	Month year December 2003	**Project no.** E6872
	General Research Areas 5. Commissioned Research	
	Subcategories 23 Logistics	

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

Report title (In translation)

Management and modeling of information in support of logistic M&S

Abstract (not more than 200 words)

The future vision of a logistic system, within the network centric defence, comprises a number of services that will support the realisation of strategic, operative and tactical goals. Within the framework of this system, LRS, TAV and M&S will play an important part, for example when planning for future combat systems or upcoming military operations. All these areas will require data of various types. Considering the ways of managing this kind of logistic data today, the prospective for successful use of the information by future logistic system is limited. The requirements on a future logistic system include the capability to automatically integrate information from multiple, heterogeneous information sources, to support consumers of information.

This study aims at exploring the possibilities for integration of information from multiple, heterogeneous sources. The integration process should support automatic localisation, selection, composition and execution of services that could produce the information required by an information consuming system. The study considers the research carried out for the Semantic Web as a starting point for further explorations and describes technologies derived from this context, such as RDF (Resource Description Framework) and DAML-S (Darpa Agent Mark-up Language - Services). These technologies are expected to support ontology based Web Services that will automate and simplify the use of services on a network and help bringing heterogeneous systems together, i.e. increased interoperability.

Keywords

Logistics, M&S, Information Management, Information Modelling, Semantic Web, Ontology

Further bibliographic information	Language Swedish
ISSN 1650-1942	**Pages** 45 p.
	Price acc. to pricelist

Innehållsförteckning

Sammanfattning

Visionen av ett framtida logistiskt system, inom det nätverksbaserade försvaret (NBF), omfattas av tjänster (verktyg) som underlättar verkställande av uppsatta strategiska, operativa och taktiska mål. Inom detta system kommer LRS (resursledningssystem), TAV (*Total Asset Visibility*) modellering och simulering (M&S) att spela en avgörande roll, tex. vid prognostisering av krav på framtida försvarssystem eller vid planering av kommande militära operationer. Alla dessa områden kräver tillgång till data av skilda slag. Utifrån de sätt med vilka dessa data hanteras idag kommer det dock att bli komplicerat för framtida logistiska system/förmågor/verktyg att utnyttja informationsmängden på ett effektivt sätt. Vad som krävs av ett framtida logistiskt system är att utifrån krav från informationskonsumerande system automatiskt kunna integrera den eftersökta informationsmängden från heterogena datakällor inom nätverket. Logistikledningen med dess sammanhörande informationssystem skall stödja möjligheten att sätta samman och samordna system och förband. För att kunna utnyttja tillgängliga resurser på ett effektivt sätt och sätta samman dem, krävs information om egna system och förband.

Den grundläggande förutsättningen för att system sömnlöst skall kunna producera och konsumera information är en gemensam begreppsvärld för aktuell domän, tex. logistikområdet. Det är av största vikt att interoperabilitet, både på den syntaktiska och semantiska nivån, tillgodoses. Detta kan liknas vid förmågan att kunna läsa en text (syntax) och samtidigt även förstå innebörden av texten (semantik). Det krävs således någon form av mekanism som möjliggör att information får en entydig innebörd mellan de system som konsumerar den, samt att information representeras på ett plattformsoberoende vis. Genom att modellera en delmängd av den information, eller kunskap, som en domän omfattas av, skapas grunden till en gemensam begreppsvärld för ett område. Den gemensamma uppfattningen av aktuell domän kan representeras i form av en så kallad ontologi. En ontologi representerar bland annat klasser (flygfarkost, jaktflyg etc.), relationer mellan klasser (tex. att jaktflyg är en subklass till flygfarkost), egenskaper för klasser (tex. flottiljtillhörighet för ett jaktflyg) samt begränsningar för relationer mellan klasser och egenskaper för klasser (tex. att egenskapen flottiljtillhörighet kan anta värdena: F21, F7 etc.).

En ontologi för den logistiska domänen bör inkludera information om materielsystem i form av indelning i klasser, egenskaper för dessa klasser, samt även relationer mellan klasser. Denna kunskap bygger upp grundläggande modeller som kan utnyttjas för att kommunicera information om materielsystem mellan system. Ontologin utgör den ”globala” modell som olika typer av system måste förstå för att vara interoperabla med andra system. Tanken med detta är inte bygga om system som representerar begrepp på ett sätt som skiljer sig från det globala synsättet. Dessa förses istället med ett ”lager” som kan översätta information uttryckt enligt ontologin till lokala representationer av begrepp.

Utöver representation av kunskap krävs någon form av tjänstarkitektur som möjliggör att informationskonsumerande system kan avkräva information från informationsproducenter (i detta fall avses tjänster ur ett IT-perspektiv). Denna arkitektur bör baseras på en teknologi som utnyttjar plattformsoberoende format för hantering av meddelanden. Vidare bör stöd för effektiv lokalisering, sammansättning och exekvering av tjänster finnas tillgängligt. I detta sammanhang är även ontologier av stor betydelse för entydig beskrivning av tjänster. Med detta avses beskrivning av vilken typ av information en tjänst kan leverera, hur en tjänst skall utnyttjas etc.

Denna förstudie utgår från det arbete som har bedrivits inom *the Semantic Web* (den semantiska webben). Målet med den semantiska webben är att införa beskrivningar av webbaserade resurser som kan tolkas av maskiner och som utgår från en gemensam uppfattning av en domän – en ontologi. Den semantiska webben skall ge en miljö inom vilken ”intelligenta” programvaror kan resonera kring ontologiska beskrivningar av resurser i syfte att automatisera lokalisering, urval, sammansättning och användning av tjänster. Arbetet beskriver några grundläggande teknologier/standarder som utvecklats inom detta område, samt exemplifierar en delmängd av teknologin med en begränsad implementering. Observera att arbetet inte fokuseras på vilken information/kunskap som bör modelleras, utan snarare på hur information/kunskap kan representeras på ett effektivt sätt i en heterogen nätverksmiljö.

Förstudien visar att ett system för integrering av information från multipla, heterogena datakällor, till stöd för logistiskområdet, bör baseras på en ontologisk grund. Ontologier skapar här förutsättningar för interoperabilitet mellan system på flera plan, genom att representera den information som kan utbytas mellan system, information som krävs för klassificering av tjänster, samt de generella informationsmodeller som utnyttjas inom logistik domänen. I detta sammanhang är det viktigt att poängtera att den kunskap som en ontologi representerar kan delas av alla ingående parter (verkanssystem, logistiska system och användare av dessa). Detta innebär att kunskapen bör deklareras på en övergripande, global nivå och inte kapslas in på applikationsnivå. Eftersom logistisk information är en viktig delmängd i det totala ledningssystemet krävs att framtagning av dessa ontologier samordnas på central nivå inom försvarsmakten.

Inom den semantiska webben framstår RDF (*Resource Description Framework*) och teknologier som baseras på RDF, som kraftfulla verktyg för att representera kunskap (ontologier) i dynamiska nätverksmiljöer. Utöver denna standardisering bör tjänster utformas efter en väletablerad och erkänd standard. I detta sammanhang utgör *Web Services* ett bra alternativ vars integrering med koncept från ”den semantiska webben” (RDF-relaterade teknologier) av många anses lyfta tjänstebegreppet till en ny nivå. Slutligen krävs även ”intelligenta” verktyg som kan utnyttja den infrastruktur som *Web Services* och ontologier bildar.

Akronymer

A2A – Application to Application

API – Application Programming Interface

B2B – Business to Business

CORBA – Common Object Request Broker Architecture

DAML – Darpa Agent Mark-up Language

DAML+OIL – Darpa Agent Mark-up Language + Ontology Inference Language

DAML-S – Darpa Agent Mark-up Language – Services

DCOM – Distributed Component Object Model

DBMS – Database Management System

IEEE – Institute of Electrical and Electronics Engineers

JDBC – Java Database Connectivity

OWL – Web Ontology Language

MAS – Multi Agent System

M&S – Modellering och Simulering

NBF – Nätverksbaserat Försvar

P2P – Peer-to-Peer

RAL – Repository Abstraction Layer

RDF – Resource Description Language

RDQL – RDF Data Query Language

RMI – Remote Method Invocation

RQL – RDF Query Language

SAIL – Storage And Inference Layer

SeRQL – Sesame RDF Query Language

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

UDDI – Universal Description, Discovery and Integration

URI – Uniform Resource Identifier

W3C – World Wide Web Consortium

WS – Web Services

WSDL – Web Service Description Language

XML – Extensible Mark-up Language

QoS – Quality of Service

1. Introduktion

1.1 Bakgrund

Visionen av ett framtida logistiskt system, inom det nätverksbaserade försvaret (NBF), omfattas av tjänster (verktyg) som underlättar verkställande av uppsatta strategiska, operativa och taktiska mål. På strategisk nivå kan dessa tjänster utnyttjas för att prognostisera krav på framtida försvarssystem, samt till att specificera krav vid materialanskaffningsprocessen. Vid operativ/taktisk nivå kan tjänsterna användas som planerings-, dimensionerings- och verifieringsverktyg för kommande operationer.

Modellering och simulering (M&S) kommer att vara av avgörande betydelse för ovan beskrivna system. M&S inom den logistiska domänen kräver ofta tillgång till data av skilda slag. Det finns idag en stor mängd data att tillgå, men skilda lagringssätt, inkonsekvent semantik och syntax etc., gör det svårt att utnyttja denna till fullo. I ett framtida system för logistisk M&S krävs att data kan integreras automatisk från heterogena datakällor för att tillfredställa krav från simuleringsmodeller eller andra informationskonsumerande komponenter.

Internet och nätverkskommunikation i allmänhet har ökat tillgången på information dramatiskt under senare år. Teknologin möjliggör att information och tjänster kan delas på ett mycket effektivt sätt inom och mellan organisationer av skilda slag. Allteftersom denna utveckling eskalerat har behovet av att standardisera sätten att kommunicera och informationen som utbyts framgått tydligt. Problemen inom detta område är många. Hur kan intressanta resurser inom denna gigantiska uppsjö av information lokaliseras på ett effektivt sätt? Hur kan system som bygger på olika tekniker interagera på ett smidigt sätt? Förutsatt att system kan interagera, hur kan korrektheten i denna kommunikation säkerställas på en semantisk och syntaktisk nivå?

Även om det idag finns tillräckliga tekniska möjligheter att göra system skapade under skilda teknologiska epoker tillgängliga via ett nätverk, saknas kunskap kring hur information som extraheras från dessa källor kan göras interoperabel. Ofta krävs inblandning från användare av systemen för att lösa inkonsekvens i begreppsdefinitioner och syntax. Utöver detta finns små möjligheter att automatisera lokalisering, matchning och sammansättning av resurser utifrån krav på information som användare eller system har. Resurser, bestående av flera underordnade tjänster, representeras ofta av statiska beskrivningar. Denna metodik stödjer inte automatisk och dynamisk komposition av tjänster som för tillfället uppfyller uppställda krav bäst.

Problemet är i praktiken tredelat. För det första krävs någon form av gemensam uppfattning av den domän som är av intresse där begrepp och relationer mellan begrepp definieras explicit. Till denna begreppsvärld kan ”lokala” representationer av omvärlden hos existerande system relateras, vilket lägger grunden till interoperabel information. För det andra krävs beskrivningar av resurser på en övergripande nivå. Resurser måste beskrivas ur ett tjänsteperspektiv med specifikation av vilken typ av tjänster en resurs kan leverera, på vilket sätt dessa tjänster levereras, relationer till andra tjänster etc. (i detta fall avses tjänster på webben). Båda dessa punkter kräver ett formellt språk för representation av kunskap inom domänen som avses. Slutligen krävs även funktionalitet som kan utnyttja beskrivningar av resurser och resonera kring dessa i syfte att utföra en specifik uppgift på uppdrag från användare eller system.

1.2 Mål

Denna rapport redovisar resultaten från en förstudie med syftet att se på nya möjligheter att integrera information från heterogena informationskällor. Detta för att uppfylla krav från informationskonsumerande system i form av simuleringsmodeller, simulatorer etc. inom den logistiska domänen. Med integrering avses i detta fall automatisk lokalisering, urval, sammansättning och exekvering av tjänster som kan förväntas leverera den efterfrågade informationsmängden. Studien tar sin utgångspunkt i det arbete som har bedrivits inom *the Semantic Web* och dess relation till *Web Services*. Arbetet beskriver de grundläggande teknologier/standarder som utvecklats inom detta område, samt exemplifierar en delmängd av teknologin med en begränsad implementering. Inom ett framtida logistiskt system kommer LRS (resursledningssystem), TAV (Total Asset Visibility) och simuleringar att kräva integrering av information från heterogena datakällor. Detta arbete begränsas dock till att diskutera detta problemområde ur ett M&S-perspektiv.

1.3 Rapportens struktur

Kapitel 2 ger en övergripande bild av de problem som uppstår vid integrering av heterogena datorsystem. Vidare beskrivs teknologier som har potentialen att utgöra en grund för informationshantering i heterogena, nätverksbaserade datormiljöer, nämligen *Web Services* (WS), *Resource Description Framework* (RDF) och *Darpa Agent Mark-up Language – Services* (DAML-S). Tyngdpunkten läggs i detta sammanhang på RDF då detta utgör grunden till många teknologier inom den semantiska webben. I kapitel 3 presenteras ett kortfattat scenario som belyser den potentiella nyttan av RDF. Vidare presenteras även en förhållandevis enkel implementering utifrån ovan nämnda scenario. I kapitel 4 diskuteras skillnaden mellan XML Schema och RDF Schema och behovet av explicit definierad semantik i heterogena datormiljöer. Utöver detta diskuteras övergripande vilken funktionalitet som krävs av ett system för hantering av information till stöd för logistisk M&S och hur idéer och visioner inom den semantiska webben kan appliceras i denna kontext. Kapitel 5 presenterar slutsatser från arbetet, medan kapitel 6 föreslår framtida inriktning på forskning inom området.

Observera att vissa delar av dessa kapitel är relativt tekniska till sin natur. För att få en övergripande bild av arbetet, utan detaljerade tekniska aspekter, hänvisas läsaren till introduktionsavsnitten under respektive kapitel, samt den avslutande diskussionen.

2. Bakgrund

2.1 Interoperabilitet

Sammanlänkning av system, utvecklade för olika ändamål, under skilda teknologiska tidsåldrar och för olika plattformar, kan medföra flera svårigheter. Ett avgörande hinder att överkomma i detta sammanhang är frågan om interoperabillitet. Enligt IEEE[1] [1] kan interoperabilitet definieras enligt:

> ”*the ability of two or more systems or components to exchange information and to use the information that has been exchanged*”

Den grundläggande förutsättningen enligt denna definition är att kommunikation mellan system är möjlig. Vad som i sammanhanget är mer komplext att hantera är den andra delen av definitionen, nämligen att system kan använda den information som har utbytts. Detta säger dock inget om att informationen även bör utnyttjas på ett korrekt sätt, vilket medför ytterligare en nivå av komplexitet. För att system i en dynamisk miljö skall kunna interagera korrekt krävs i praktiken interoperabillitet på tre nivåer; syntaktisk, semantisk och pragmatisk interoperabillitet. Skiljelinjen mellan den semantiska och pragmatiska nivån kan diskuteras. Generellt sett avser semantik betydelsen av ett uttryck, medan pragmatik förklarar hur skillnader mellan intentioner hos skaparen av ett uttryck och betydelsen av ett uttryck kan överbryggas. För att uppnå interoperabilitet på dessa nivåer krävs det att den kunskap som en domän omfattar kan representeras på ett formellt och konsekvent sätt.

Allteftersom webbaserade/nätverksbaserade lösningar blir mer vanliga kommer interoperabilitet att framstå som en avgörande fråga. I dagsläget är, med hjälp av Internet och andra nätverk, distribution och tillgång till information inte ett avgörande problem. Problematiken ligger snarare i hur erhållen information ska behandlas och tolkas.

2.2 Web Services

2.2.1 Introduktion

Konceptet *Web Services* (WS) baseras i grunden på tankar och idéer som har utvecklats inom området distribuerade datorsystem under åtskilliga år. Ett distribuerat datorsystem definieras enligt [2] som:

> *”Ett distribuerat datorsystem är en samling oberoende datorer som från en användares perspektiv upplevs som ett enkelt, sammanhängande system”*

Denna definition beskriver två viktiga aspekter, nämligen att datorerna inom systemet är autonoma och att användaren upplever systemet som en lokal installerad programvara. Fördelarna med ett distribuerat system är många, bland annat [3]:

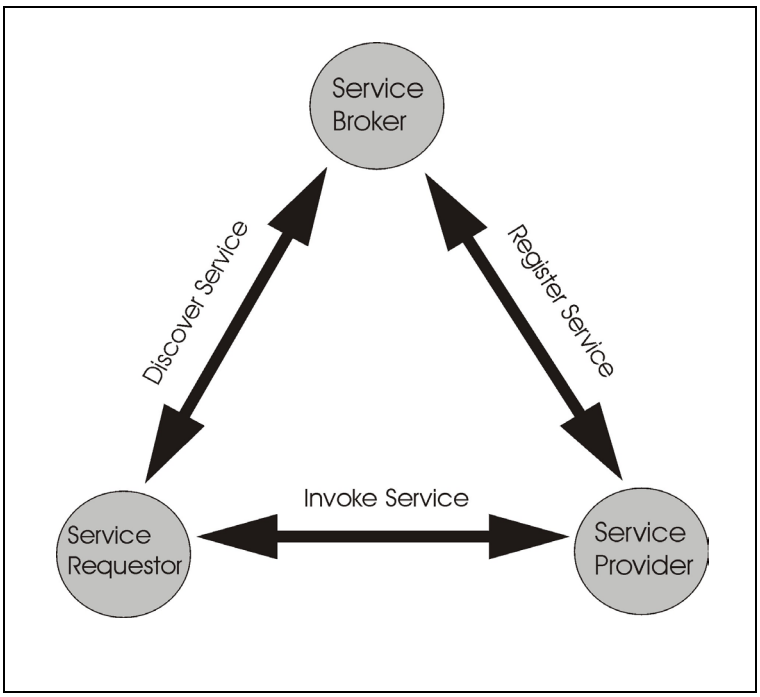
- En enskild användare kan utföra processorintensiva uppgifter genom att applikationer kan exekveras parallellt och distribuera ett arbete till ett flertal maskiner. Genom att dela på maskinvara kan en användare få tillgång till beräkningskraft som är flera gånger större en den egna datorns kapacitet

[1] IEEE – Institute of Electrical and Electronics Engineers

- Genom att distribuera ett system kan en högre grad av robusthet och tillgänglighet uppnås. Målet är att undvika en *single-point of failure* i system och därmed alltid vidmakthålla en viss servicenivå
- En komponent som inkluderar funktionalitet som är generell och som därmed kan utnyttjas av flertalet olika applikationer kan återanvändas på ett effektivt sätt
- Distribuering av system kan även ses som en kostnadsbesparande åtgärd då resurser av skilda slag kan delas mellan flera användare. Resurser kan i detta fall vara beräkningskraft, programvara, data etc.

Ett flertal teknologier för utveckling av distribuerade system existerar idag, bland annat CORBA (*Common Object Request Broker Architecture*), Java RMI (*Java Remote Method Invocation*) och Microsoft DCOM (*Distributed Component Object Model*). I och med Internets snabba utveckling har behovet av serviceorienterade arkitekturer som stödjer B2B (*Business-to-Business*) och A2A (*Application-to-Application*) kommunikation, inom och mellan skilda domäner, framgått allt tydligare. Genom standardiserade webbteknologier kan idag heterogena applikationer göras tillgängliga via Internet i form av tjänster – *XML Web Services* – som kommunicerar genom standardiserade meddelandeformat – XML-meddelanden (*Extensible Mark-up Language*). WS standardiserar kommunikations-mekanismen mellan applikationer som har utvecklats i skilda programmeringsspråk och för skilda plattformar, vilket skapar en grund för interoperabilitet.

Det grundläggande konceptet för WS illustreras av figur 2.1. I denna schematiska bild finns tre huvudsakliga parter representerade; *Service Requestor*, *Service Broker* och *Service Provider*. Denna konfiguration är typisk för transaktioner som baseras på WS-konceptet. En *Service Provider* är ansvarig för utveckling och spridning av en tjänst. Detta omfattar även registrering av tjänsten hos en *Service Broker*. En *Service Broker* ansvarar för registrering och lokalisering av tjänster. Detta omfattar tillhandahållande av en lista över typer av tjänster och dess beskrivningar, samt var dessa finns lokaliserade. Slutligen finns en användare av tjänster, benämnd *Service Requestor*, som lokaliserar tillgängliga tjänster via en *Service Broker* och exekverar önskvärda tjänster hos en *Service Provider* [3].



Figur 2.1. Grundläggande koncept för Web Services.

Det finns en lång rad teknologier tillgängliga som stödjer WS-konceptet. Nedan presenteras några av de mest framträdande teknologierna översiktligt.

2.2.2 Simple Object Access Protocol – SOAP

I kärnan av WS-modellen används SOAP (*Simple Object Access Protocol*) som grundläggande meddelandeprotokoll. SOAP är ett enkelt protokoll för utbyte av strukturerad information i en decentraliserad, distribuerad miljö och baseras på XML. Ramverket som SOAP definierar kan enkelt utvidgas och stödjer utbyte av meddelanden baserat på flertalet grundläggande nätverksprotokoll. Vidare är ramverket oberoende av programmeringsspråk och plattform, vilket kan tillskrivas det interoperabla meddelandeformat som XML utgör. En av de fundamentala tankarna bakom SOAP är att göra specifikationen av standarden så enkel som möjligt, utan att för den delen hämma utökningar som specifika användningsområden kräver. Utökningar av den befintliga standarden inkluderar lager för säkerhet, *routing* av meddelanden och feltolerans [4]. Specifikationen av standarden för SOAP hanteras av W3C (*World Wide Web Consortium*). Den senaste specifikationen av SOAP, version 1.2, går att finna under [5]. Figur 2.2 visar hur ett förhållandevis enkelt SOAP-meddelande kan utformas.

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <x:MeterFeetConverter xmlns:x="urn:examples-org:geography">
   <unit>meter</unit>
   <value>100</value>
  </x:MeterFeetConvereter>
 </soap:Body>
</soap:Envelope>
```

Figur 2.2. SOAP-meddelande för begäran om konvertering från meter till fot.

Rot-elementet i ett SOAP-meddelande består alltid av ett *Envelope*-element. Detta gör det enkelt för applikationer att identifiera SOAP-meddelanden genom att inspektera rot-elementet. *Envelope*-elementet innehåller i sin tur ett *Header*-element som egentligen är valfritt, samt ett obligatoriskt *Body*-element. *Body*-elementet inkluderar själva substansen i meddelandet och kan omfatta ett godtyckligt antal subelement. *Header*-elementet inkluderar information som kontrollerar hur informationen i *Body*-elementet skall bearbetas. SOAP-specifikation definierar inga standardiserade *Header*-block, utan detta lämnas öppet för eventuella utökningar av protokollet. I exemplet (figur 2.2) inkluderar *Body*-elementet en förfrågan angående konvertering mellan enheterna meter och fot. I detta fall gäller förfrågan en konvertering av sträckan 100 meter till motsvarande sträcka uttryckt i fot (anges av elementen <unit> och <value>). Svaret på denna förfrågan skulle kunna utformas enligt meddelandet i figur 2.3. Subelement till *Body*-elementet anger att det handlar om ett svar på en *MeterToFeetConverter*-förfrågan och att 100 meter motsvaras av 328 fot.

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <x:MeterFeetConverterResponse xmlns:x="urn:examples-org:geography">
   <unit>feet</unit>
   <value>328</value>
  </x:MeterFeetConvereterResponse>
 </soap:Body>
</soap:Envelope>
```

Figur 2.3. SOAP-meddelande för svar på begäran om konvertering från meter till fot.

2.2.3 Web Service Description Language – WSDL

WSDL (*Web Service Description Language*) är en central teknologi inom WS-konceptet som används för att beskriva tjänster ur ett metadataperspektiv. WSDL baseras på XML och beskriver vilken funktionalitet som en specifik tjänst har, var den är lokaliserad, samt hur den kan utnyttjas. Mer specifikt inkluderar en WSDL-beskrivning följande information om en tjänst [6]:

- Gränssnitt – publika funktioner som en tjänst omfattar
- Datatyper – definition av datatyper som förfrågningar (*requests*) och svar (*respones*) kan inkludera (I/O-parametrar för funktioner)
- Protokollbindning – information om protokoll som används för att utnyttja en tjänst
- Lokalisering – adress till en specifik tjänst

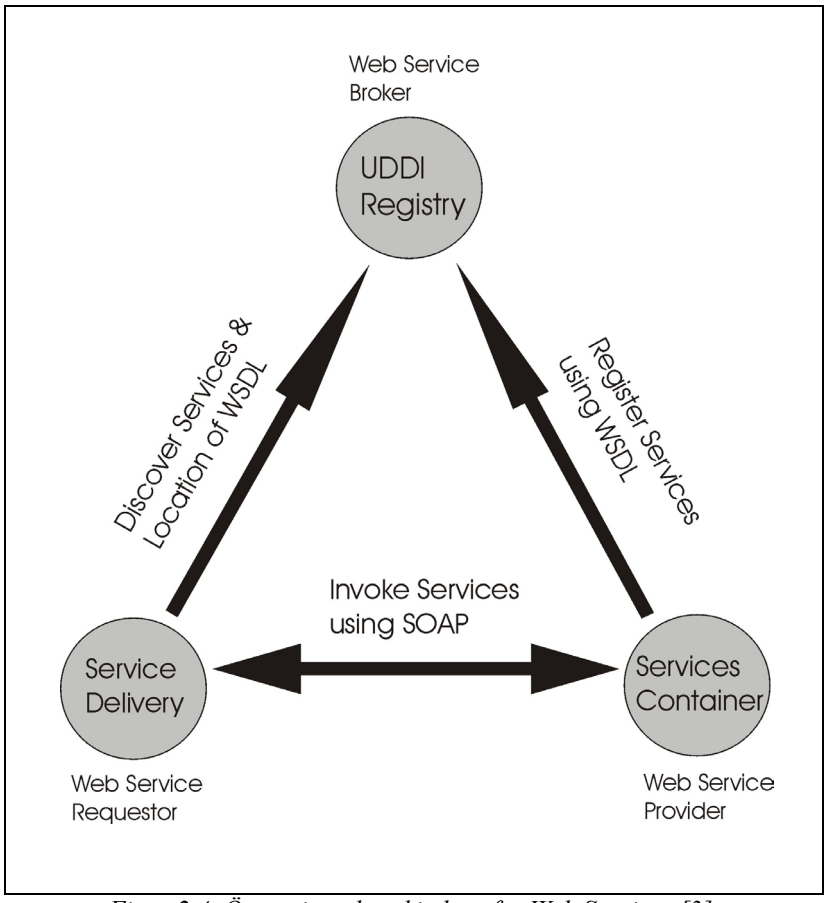
2.2.4 Universal Description, Discovery and Integration – UDDI

För att en *Resource Requestor* skall kunna använda distribuerade resurser inom ett nätverk krävs en mekanism för lokalisering av dessa. UDDI (*Universal Description, Discovery and Integration*) omfattar en standardiserad metod för publicering och lokalisering av information rörande tjänster. Ur ett konceptuellt perspektiv kan ett UDDI-baserat register inkludera tre skilda grupper av information kring tillgängliga tjänster [7]:

- Kontaktinformation till den organisation som tillhandahåller tjänsten. Detta medför att tjänster kan lokaliseras baserat på organisationstillhörighet
- Information baserad på någon form av klassificeringssystem. Medför att tjänster kan lokaliseras baserat på innehåll
- Teknisk information om en tjänsts egenskaper och uppträdande

2.2.5 Arkitektur

Genom att nyttja ovan beskrivna standarder kan en övergripande arkitektur för WS beskrivas, se figur 2.4. En *Service Provider* publicerar information om en tjänst, specificerad enligt WSDL-standarden, till ett UDDI-register. UDDI-registret låter en *Service Requestor* söka efter lämpliga tjänster. Identifierade tjänster kan därefter utnyttjas, enligt WSDL-beskrivningen, genom utbyte av SOAP-meddelanden.



Figur 2.4. Övergripande arkitektur för Web Services [3].

2.3 Den semantiska webben

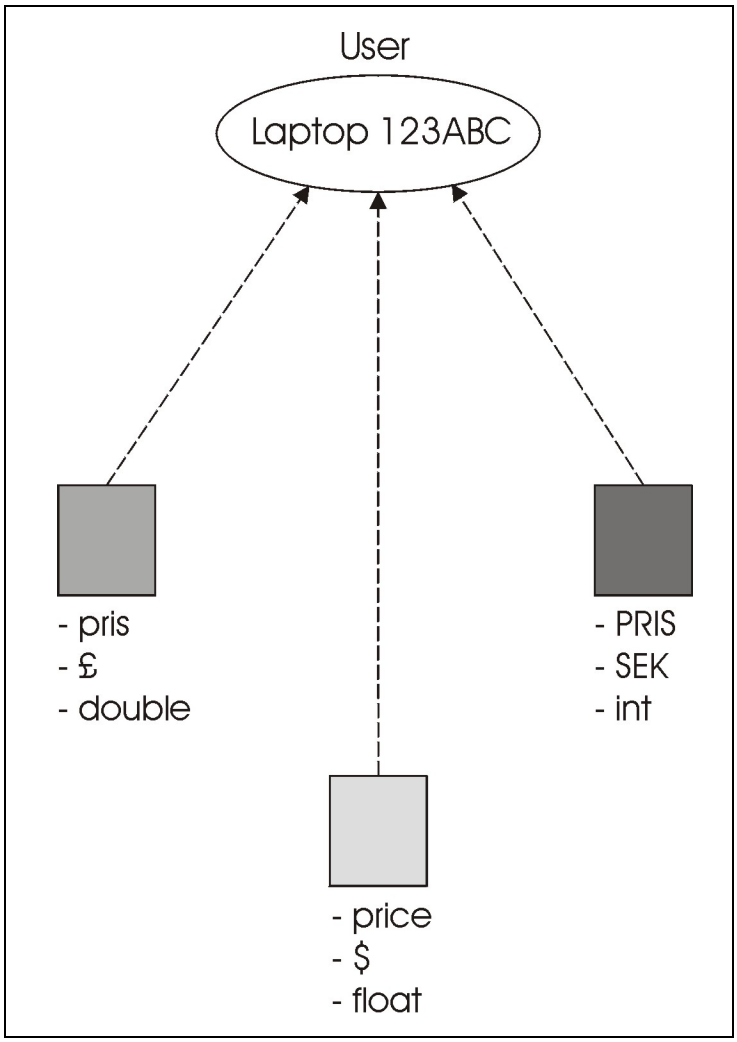
2.3.1 Introduktion

The Semantic Web, eller den semantiska webben som den kommer att benämnas i följande text, var en del av den ursprungliga webben som introducerades av Tim Berners-Lee 1989. Denna del av konceptet fick aldrig något större genomslag allteftersom webben fick till största uppgift att generera *human-readable* media. Under senare år har dock detta spår fått förnyad kraft då behovet av strukturering av innehåll har framgått med klar tydlighet, allteftersom webbens informationsinnehåll fullständigt exploderat. Utnyttjande av dokument och tjänster inom dagens webb sker i största utsträckning med webbsidan som medium. Detta tillvägagångssätt kräver mänsklig intelligens för att tolka informationsmängden och agera utifrån detta. Den grundläggande tanken med den semantiska webben är att integrera formaliserad kunskap med dagens webb, vilket medför att även maskiner kan tolka innehållet. Den semantiska webben kan därmed ses som en infrastruktur för att införa formaliserad kunskap i en annars informell informationsmängd [8]. Enligt Berners-Lee [9] kommer den semantiska webben att medföra en strukturering av innehåll på webben, vilket ger en miljö inom vilken ”intelligenta” applikationer/system kan utföra komplicerade uppgifter till stöd för användare [10]. Konceptet möjliggör effektiv identifiering, automatisering, integrering och återanvändning av tjänster/system. Databaser, applikationer, sensorer etc. kommer sömlöst att både konsumera och producera information.

Nedan följer ett exempel som illustrerar hur den semantiska webben kan möjliggöra effektiv och automatiserad integrering av information från distribuerade datakällor inom ett nätverk.

En användare, process eller ett system är intresserad av en prisuppgift för en viss typ av artikel, i detta fall en laptop med modellbeteckningen 123ABC. Figur 2.5 till 2.7 visar tre exempel på hur detta kan ske med varierad grad av automatisering.

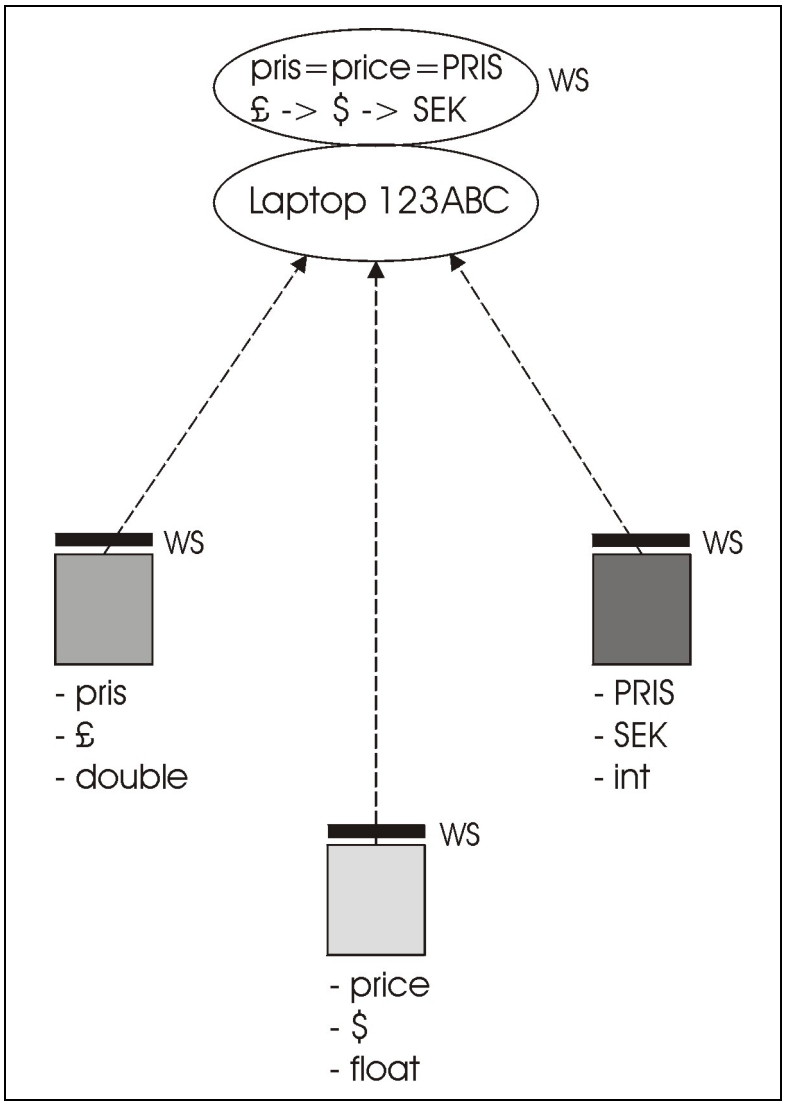
Figur 2.5 presenterar hur operationen kan fortgå på den absolut enklaste nivån. En användare söker efter webbsidor som saluför den specifika produkten av intresse. Sökningen är främst baserad på nyckelord via någon form av söktjänst (Google, Altavista etc.). Relevant information inhämtas genom att besöka identifierade webbsidor och via någon typ av formulär begära en prisuppgift. Observera att representationen av begreppet ”pris” skiljer sig mellan de olika sidorna. Detta har dock mindre betydelse i detta fall då en människa förstår att ”PRIS”, ”price” och ”pris” betyder samma sak, samt förmodligen har förmågan att konvertera mellan valutor för en prisjämförelse. Detta är ingen automatiserad process och skulle inte fungera om en maskin vore intresserad av samma data.



Figur 2.5. Inhämtning av information från heterogena, webbaserade datakällor.

Inhämtningen av information av detta slag skulle utan större svårigheter kunna automatiseras i större utsträckning. Figur 2.6 visar en schematisk bild av detta. De tre databaserna har i detta fall försetts med ett gränssnitt till nätverket genom utnyttjande av WS. Utöver dessa komponenter finns en överordnad tjänst som kan samla in prisuppgifter på uppdrag från system eller användare. När en användare eller ett system begär in en prisuppgift på en laptop

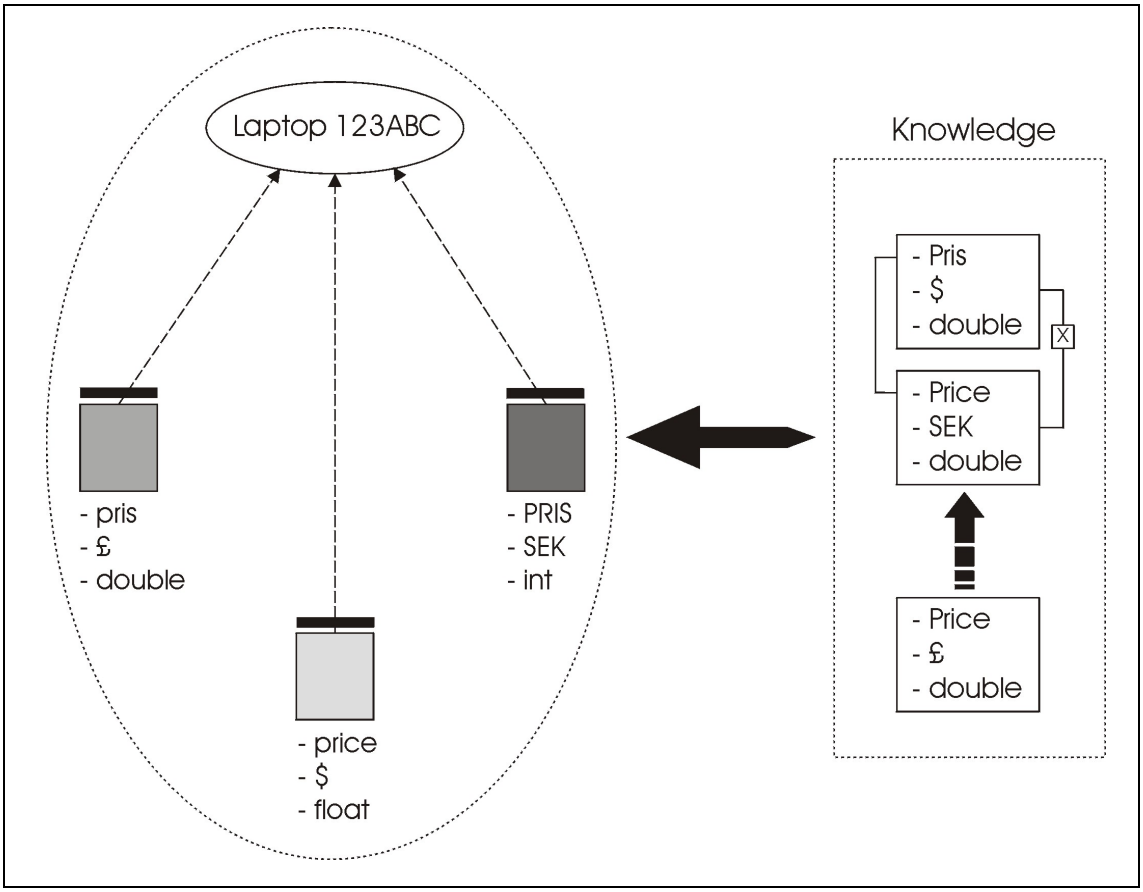
med modellbeteckningen 123ABC, inhämtar den överordnade tjänsten informationen från de enskilda databaserna via WS-gränssnitten. Problemet i detta fall är att den överordnade tjänsten måste inkludera logik som specificerar att ”pris”, ”price” och ”PRIS” är samma sak, samt hur valutorna som priset uttrycks i kan konverteras till ett enhetligt format. Om ytterligare en underordnad tjänst ansluts till systemet, som definierar ett pris enligt; ”Price”, ”NOK” och ”int”, måste den interna logiken förändras för att spegla detta. I ett verkligt system, där antalet parametrar och antalet möjliga kombinationer är långt större, kan dessa förändringar bli ohanterliga.



Figur 2.6. Inhämtning av information från heterogena, webbaserade datakällor med hänsyn tagen till semantiska och syntaktiska skillnader. Kunskapen är dock deklarerad på applikationsnivå.

För att komma till rätta med detta problem bör kunskap inom en domän deklareras på en global nivå. I detta fall global för en specifik domän, inte för hela Internet. Figur 2.7 visar en schematisk bild av ett system där begrepp, koncept, relationer etc. har deklarerats på en global nivå och delas av alla ingående komponenter (gemensam semantik och syntax). Enligt deklarationen kan begreppet ”pris” uttryckas enligt; ”Pris”, ”$”, ”double” eller ”Price”,

”SEK”, ”double”. Den gemensamma begreppsvärlden anger även att ”Pris” är ekvivalent med ”Price” och att processen ”X” kan användas för att konvertera mellan valutorna SEK och $. Utifrån denna kunskap beskriver de underordnande tjänsterna sina förmågor för den överordnade tjänsten direkt eller via någon form av infrastruktur för metadata. De underordnade tjänsternas interna logik måste även mappa den interna representationen av begreppet ”pris” mot det globala begrepp som resurserna beskrivs enligt. När den överordnade tjänsten söker efter pris för en laptop med modellbeteckningen 123ABC kan denna, utifrån den gemensamma begreppsvärlden, dra slutsatser kring vilka databaser som kan leverera prisuppgifter (kontrollera vilka begrepp som är ekvivalenta med ”pris”), samt hur dessa kan konverteras till önskvärd valuta. I detta fall representeras inte denna kunskap internt i applikationen, vilket vi ville komma bort ifrån. Om det som i fallet med figur 2.7 tillkommer ytterligare en representation av begreppet ”pris” (i detta fall för att inkludera priser i £) krävs ingen förändring av varken den överordnade eller de underordnande tjänsternas logik. Det enda som krävs är en utökning av den globala, gemensamma begreppsvärlden.



Figur 2.6. Inhämtning av information från heterogena, webbaserade datakällor med hänsyn tagen till semantiska och syntaktiska skillnader. Kunskapen är i detta fall deklarerad på en ”global” nivå och delas av ingående komponenter.

Den semantiska webben skapar en grund för maskintolkbara beskrivningar av webbaserade resurser, enligt en globalt deklarerad kunskap (ontologi), vars förändringar inte påverkar applikationers interna logik. Utifrån den semantiska webben kan system resonera kring tjänsters meta-beskrivningar och den begreppsvärld som dessa härstammar från, för automatiserad lokalisering, urval, sammansättning och användning av webbaserade resurser.

2.3.2 Resource Description Framework - RDF

För att skapa förutsättningar för en semantisk webb har en rad ”språk” utvecklats i syfte att införa ”intelligenta” beskrivningar av webbaserade resurser. I huvudsak omfattar ett språk för den semantiska webben två aspekter: en formell semantik och syntax som möjliggör automatiserad bearbetning samt ett standardiserat vokabulär, ontologi, som medför att kunskap kan delas mellan involverade parter (såväl datoriserade som mänskliga). Den fundamentala arkitektur som har fått störst genomslag är *Resource Description Framework* (RDF) vars första specifikation producerades 1997. Sedan dess har specifikationen undergått en mognadsprocess genom W3Cs försorg. RDF specifikationen består i praktiken av sex olika dokument som återfinns under [11]:

- *RDF Concepts and Abstract Syntax*
- *RDF Semantics*
- *RDF/XML Syntax Specification*
- *RDF Schema*, *RDF Primer*
- *RDF Test Cases*

Det fundamentala konceptet bakom RDF är att alla resurser kan beskrivas utifrån utsagor som baseras på tre delar av information – subjekt, predikat och objekt. Ur meningen:

Artikelns namn är ”Den Semantiska Webben”

kan dessa beståndsdelar identifieras som artikel (subjekt), namn (predikat) och ”Den Semantiska Webben” (objekt) på sedvanligt vis. Subjektet motsvarar saken som beskrivs, i detta fall en webbaserad resurs av något slag. Predikatet motsvaras av attribut, relation eller karaktäristik för resursen som beskrivs. Slutligen motsvarar objektet värdet av predikatet. En utsaga om en webbaserad resurs, tex. meningen: Artikeln, som nås via http://www.exempel.org/SemWeb.pdf, har titeln ”Den Semantiska Webben” kan enligt ovanstående mönster representeras av:

{http://www.exempel.org/SemWeb.pdf, titel, ”Den Semantiska Webben”}

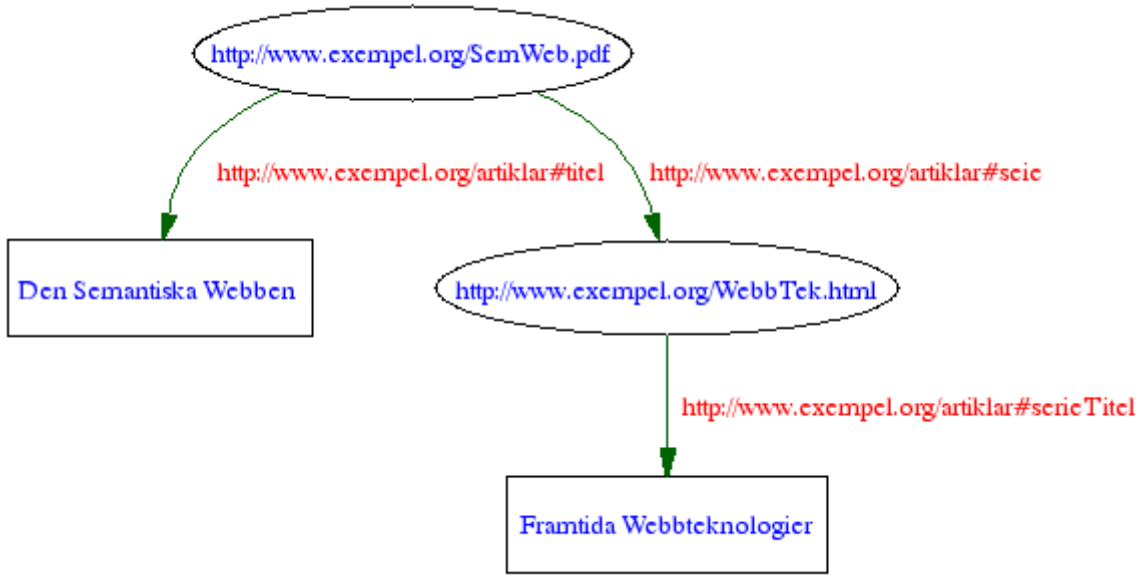
Det finns en mängd metoder för att representera (serialisera) RDF-data; N-Triples, RDF/XML eller *directed graph*, varav *directed graph* är det formella sättet. I grafmodellen illustreras en resurs av en ellips innefattande resursens URI. Predikatet representeras av en riktad ”båge”, betecknad med predikatets namn. Objektet kan anta två olika former; antingen som ytterligare en resurs eller som en litteral. Om objektet är en litteral representeras denna av en rektangel innefattande det aktuella värdet. Grafrepresentationen av ovanstående exempel redovisas i figur 2.7.

Figur 2.7. RDF-exempel (grafnotation).

För att klargöra grafnotationen ytterligare, beakta följande utsagor om en resurs:

- **[http://www.exempel.org/SemWeb.pdf, titel, ”Den Semantiska Webben”]**
- **[http://www.exempel.org/SemWeb.pdf, serie, http://www.exempel.org/WebbTek.html]**
- **[http://www.exempel.org/Semantik.html, serieTitel, ”Framtida Webbteknologier”]**

Dessa utsagor säger att artikeln som finns lokaliserad vid http://www.exempel.org/SemWeb.pdf har titeln ”Den Semantiska Webben”, samt att denna artikel ingår i en serie som nås från http://www.exempel.org/WebbTek.html med titeln ”Framtida Webbteknologier”. Den grafiska RDF-representationen av dessa utsagor presenteras i figur 2.8.



Figur 2.8. RDF-exempel (grafnotation).

Grafrepresentationen är dock inte praktiskt användbar vid lagring av och åtkomst till RDF-data, utan nyttjas främst för att visualisera datamängden på ett lättförståeligt sätt. Den mest frekvent använda metoden för att serialisera RDF-data är genom RDF/XML-notation. RDF/XML-representationen av exemplet i figur 2.7 redovisas i figur 2.9.

```
<?xml version=”1.0”?>
<rdf:RDF
        xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
        xmlns:ex=”http://www.exempel.org/artiklar#”
        <rdf:Description rdf:about=”http://www.exempel.org/SemWeb.pdf”>
                <ex:titel>Den Semantiska Webben</ex:titel>
        </rdf:Description>
</rdf:RDF>
```

Figur 2.9. RDF/XML-serialisering av RDF-data i figur 2.7.

Rad 3 och 4 i figur 2.9 anger de namnrymder som används inom RDF/XML dokumentet. På rad 3 definieras namnrymden för det vokabulär som är ”inbyggt” i RDF, medan rad 4 anger namnrymden för det vokabulär som är specifik för den domän som modelleras (se avsnitt 3.2.1 för beskrivning av hur ett domänspecifikt vokabulär kan utvecklas). Deklarationen av namnrymder innebär att elementnamn kan skrivas på kortform, tex. ex:titel istället för http://www.exempel.org/artiklar#titel, samt kanske än viktigare, att alla elementnamn förses med ett unikt namn. Från rad 5-7 (i elementet rdf:Description) beskrivs artikeln som finns lokaliserad vid http://www.exempel.org/SemWeb.pdf med attributet ex:titel som har värdet ”Den Semantiska Webben”.

Figur 2.8 presenterar en lite mer komplicerad RDF-graf. RDF/XML representationen av denna graf visas i figur 2.10. På samma sätt som i föregående exempel definieras de namnrymder som utnyttjas inom dokumentet. På rad 5 börjar beskrivningen av den resurs som modelleras, dokumentet SemWebb.pdf. Först specificeras namnet på artikeln, precis som i föregående exempel, följt av attributet ex:serie som anger resursen för den serie som artikeln ingår i. På rad 9 beskrivs i sin tur resursen http://www.exempel.org/Webbtek.html (serien) som har attributet ex:serieTitel med värdet ”Framtida Webbteknologier”.

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://www.exempel.org/artiklar#">
        <rdf:Description rdf:about="http://www.exempel.org/SemWeb.pdf">
                <ex:titel>Den Semantiska Webben</ex:titel>
                <ex:seie rdf:resource="http://www.exempel.org/WebbTek.html"/>
        </rdf:Description>
        <rdf:Description rdf:about="http://www.exempel.org/WebbTek.html">
                <ex:serieTitel>Framtida Webbteknologier</ex:serieTitel>
        </rdf:Description>
</rdf:RDF>
```

Figur 2.10. RDF/XML-serialisering av RDF-data i figur 2.8.

2.4 The Semantic Web och Web Services

2.4.1 Introduktion

I en dynamisk nätverksmiljö, där tillgång, sammansättning och lokalisering av tjänster varierar över tiden, krävs en utökning av dagens koncept för WS. För att möjliggöra automatiserad lokalisering, sammansättning och exekvering av WS krävs ett stöd för beskrivning av tjänster utöver det som erbjuds av WSDL idag. Det krävs en WS-ontologi som tjänster kan beskrivas utifrån, samt en maskintolkbar representation av dessa beskrivningar.

Följande exempel är hämtade från DAML-S specifikationen [12] och redogör för hur integrationen mellan koncept från den semantiska webben och WS kan förändra tjänsteutnyttjandet på nätet.

- **Automatisk lokalisering av tjänster:** En användare vill lokalisera en tjänst för bokning av en flygbiljett mellan två städer som accepterar en viss typ av kreditkort. På dagens Internet måste denna typ av uppgift utföras manuellt genom att en användare använder en sökmotor för att hitta en ”resesajt”, läser innehållet på erhållen webbsida och exekverar därefter tjänsten via någon form av formulär.

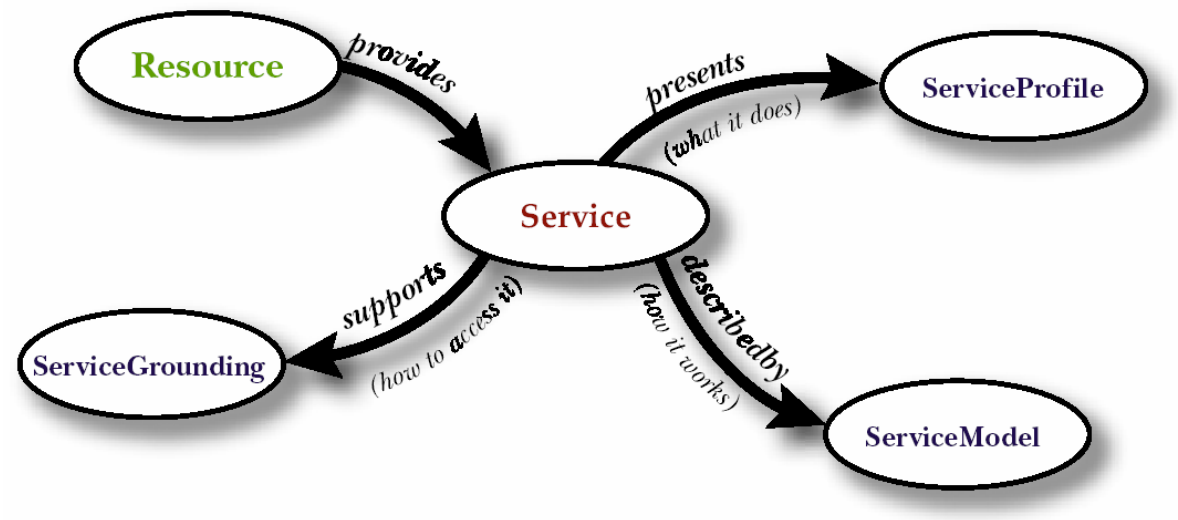
Genom att använda maskintolkbara beskrivningar, utifrån en etablerad ontologi, som publiceras i någon form av ”registry” eller görs tillgängliga via en ontologibaserad sökmotor, skulle rätt tjänst kunna lokaliseras utan att manuellt gå igenom ett antal länkar som eventuellt uppfyller sökkriteriet

- **Automatisk sammansättning av tjänster:** En användare vill utföra samtliga resebeställningar inför en konferensresa. På dagens Internet krävs att användaren manuellt definierar sammansättningen av tjänster som utför detta arbete, samt ser till att dessa tjänster är interoperabla. Semantiska WS inkluderar beskrivningar av enskilda tjänster, tillräckligt extensiva för att programvaror (agenter) kan sammanlänka tjänster på ett sätt som uppfyller överordnade krav (producera en sammansatt tjänst automatiskt)
- **Automatisk exekvering av tjänster:** En användare vill köpa en flygbiljett från en specifik webbplats. På dagens Internet måste användaren gå till webbplatsen som erbjuder önskad flygbiljett, fylla i ett formulär, samt slutligen exekvera en tjänst som utför begäran. Exekvering av tjänster kan liknas vid en serie funktionsanrop. Semantiska WS omfattas av maskintolkbara gränssnitt för exekvering av dessa funktionsanrop. Ett system (agent) skall ha möjligheten att tolka beskrivningar av tjänster för att avgöra vilka indata som krävs av ett funktionsanrop samt vilket resultat som funktionsanropet genererar
- **Kontroll av exekverande tjänster:** Exekvering av en tjänst, framförallt sammansatta och komplexa tjänster, kan kräva viss tid för att slutföras. Under denna period kan det vara önskvärt att kunna övervaka status för exekveringsförloppet eller interagera med exekveringen. En ontologi för WS bör därför inkludera konstruktioner även för denna typ av beskrivning (ett förlopp)

Integration av den semantiska webben med WS kommer att påverka utnyttjandet av webbaserade applikationer och data på ett markant sätt. Lokalisering av tjänster kommer att kunna automatiseras i större utsträckning jämfört med idag. Genom semantisk och maskintolkbar märkning av tjänster kommer de steg som idag kräver mänsklig inblandning vid exempelvis sökning att begränsas. Detta kommer medföra ett mer automatiserat och effektivt sätt att integrera information från heterogena datakällor som kan utnyttjas av informationskonsumerande system, tex. simuleringsmodeller.

2.4.2 DAML-S

Inom ramen för DAML (Darpa Agent Mark-up Language) har en ontologi för beskrivning av tjänster utvecklats, nämligen DAML-S. Med tjänster avses i detta sammanhang webbaserade resurser som inte enbart erbjuder statisk information (tex. en enkel html-sida), utan som även tillåter någon form av interaktion. Motivet bakom denna satsning är att möjliggöra automatisk lokalisering, selektering, sammansättning, exekvering och övervakning av webbaserade tjänster i den semantiska webbens anda. Även om specifikationen för DAML-S inte strikt begränsar dess användning till beskrivning av WS har det i praktiken fått störst spridning inom just detta område, eftersom WS är en spridd och välaccepterad standard. Konceptet bygger likt andra satsningar på att förse webbaserat material/innehåll med maskintolkbara beskrivningar utifrån en etablerad och gemensam uppfattning av omvärlden – en ontologi. Utifrån dessa beskrivningar kan ”agenter” agera på uppdrag från användare/system som vill utföra en specifik uppgift. För närvarande bygger DAML-S på *DAML+Ontology Interface Language* (DAML+OIL), men planer finns på att basera tekniken på *Ontology Web Language* (OWL) när denna har kompletterats ytterligare. DAML+OIL och OWL bygger i grunden på RDF [12].



Figur 2.11. Beskrivning av en tjänst på högsta nivå; Service-Ontologin

DAML-S består i praktiken av fyra olika ontologier: *Service*, *Profile*, *Process* och *Grounding*. *Service*-ontologin definierar en tjänst på högsta nivå, se figur 2.11, omfattande *Service*-klassens relation till andra klasser inom DAML-S, närmare bestämt relationer till klasserna *ServiceProfile*, *ServiceModel* och *ServiceGrounding*. *ServiceProfile* beskriver tjänsters egenskaper utifrån ett lokaliseringsperspektiv, medan *ServiceModel* definierar tjänsters interna mekanismer, tex. förväntad I/O. Slutligen specificerar *ServiceGrounding* länken mellan den abstrakta representationen i *ServiceProfile* till faktiska implementeringsdetaljer, dvs. specifikation av nätverksprotokoll och format för meddelanden. *Profile*-ontologin omfattar vokabulären som används för *ServiceProfile*. Centralt inom denna ontologi är klassen *Profile* (subklass till *ServiceProfile*), som beskriver den organisation som tillhandahåller tjänsten, specifikation av tjänstens funktion (*input*, *output*, *preconditions* och *effect*), samt slutligen tjänstens karaktäristik (QoS, klassificering). Process-ontologin definierar vokabulären för klassen *ServiceModel* som i sin tur har den specialiserade subklassen *ProcessModell*. Klassen *ProcessModell* omfattar en klass av typen *Process* som definierar hur tjänsten är logiskt konstruerad, dvs. om det är en enkel eller sammansatt tjänst, samt om en tjänst är sammansatt, hur de skilda komponenterna skall exekveras (parallellt, seriellt etc.). *Grounding*-ontologin omfattar det vokabulär som uttrycker implementeringsspecifika egenskaper hos tjänster. Detta kan liknas vid att mappa den abstrakta definitionen av en tjänst som ges i *ServiceModell* och *ServiceProfile* till en konkret representation som anger exakt hur en tjänst kan utnyttjas. För närvarande är klassen *WSDLGrounding* den som utnyttjas mest för detta specifika ändamål, då den binder samman DAML-S med *Web Services* [12].

3. RDF-baserad informationsintegrering

3.1 Introduktion

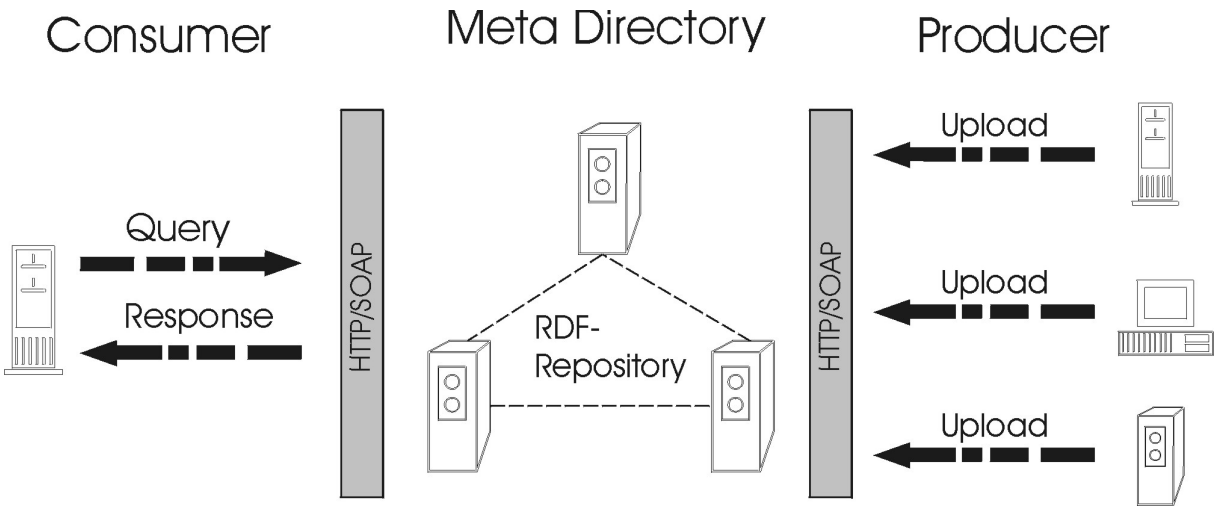
För att påvisa hur RDF och *RDF-Schema*, samt RDF-baserade verktyg kan utnyttjas för sökning och insamling av distribuerade dokument skapades ett enkelt scenario som grund. I detta scenario förutsätts att en viss typ av dokument finns distribuerade över nätverket. Samtliga av dessa dokument skall göras tillgängliga för sökning och insamling från intressenter av skilda slag. Intressenter kan vara godtycklig informationskonsumerande part, i form av en användare eller ett system. Den grundläggande förutsättningen är dock att sökning och insamling inte sker från en central databas innehållande samliga dokument. Dokumenten är distribuerade i flera separata databaser som administreras av skilda organisationer. I detta specifika fall utgörs de distribuerade dokumenten av rapporter som beskriver fel hos olika system inom flygdomänen. I följande text benämns dessa dokument som ”felrapporter”. Ett uppträtt fel som genererar en felrapport kan representeras i mer än en databas samtidigt. När denna typ av konflikt inträffar vid sökning och insamling, väljs den felrapport från den organisation som rankas högst enligt en prioriteringslista.

3.2 Modellering av resurser - RDF-Schema

I syfte att tillgängliggöra dokument, distribuerade i nätverket, för sökning och insamling krävs att metainformation kring dessa genereras och lagras i någon form av sökbart register. Det krävs ett uniformt sätt att uttrycka vilka egenskaper ett visst dokument har, dvs. i vårt scenario krävs ett vokabulär för felrapporter (en ontologi). RDF stödjer i grunden enkla utsagor om resurser, tex. klassen personbil är subklass till klassen fordon. Det krävs dock en mekanism för att uttrycka specifika klasser av resurser, samt specifika attribut för dessa klasser. Ett transportföretag skulle exempelvis ha behov av att uttrycka egenskaper som maximal lastkapacitet, hastighet etc. för olika resurser (transportmedel). Genom att uttrycka dessa attribut och klasser i ett RDF-vokabulär kan domänspecifika koncept fångas. RDF-vokabulär definieras genom mekanismer i *RDF-Schema* [13], vilket kan liknas vid ett typ-system (*type system*) för RDF. I appendix A redovisas översiktlig hur ett förenklat RDF-vokabulär (ontologi) för felrapporter kan skapas med *RDF-Schema*.

3.3 Systemöversikt

Utifrån det scenario och RDF-vokabulär (ontologi) som beskrivits ovan implementerades ett enkelt system. Systemet möjliggör insamling av distribuerade dokument (felrapporter) utifrån sökvillkor som specificeras i ett grafiskt användargränssnitt. Den funktionalitet som användargränssnittet erbjuder en användare skulle lika gärna kunna göras tillgänglig för olika typer av resurskonsumerande system, tex. agenter som samlar in information till stöd för logistisk M&S. Gränssnittet illustrerar på ett enkelt sätt hur RDF kan utnyttjas för ontologibaserad sökning och insamling av information. Applikationen som inkluderar användargränssnittet kan exekveras från en godtycklig dator med tillgänglig Internet-uppkoppling och insamling av dokument kan ske från godtycklig webbserver.



Figur 3.6. Strukturering av ett RDF-baserat informationssystem.

Figur 3.6 redovisar hur ett RDF-baserat informationssystem kan struktureras översiktligt. Systemet utgörs av tre lager, som har tre syften eller funktioner, dessa är konsument-, producent- och metadatalager. Informationsproducerande enheter inom nätverket konstruerar metadatabeskrivningar av dess resurser efter fördefinierade mallar (baserade på gemensamma koncept – ontologier). Metabeskrivningarna publiceras därefter i metadatabasen, i figur 3.6 utgörs denna av en RDF-databas. Utifrån metabeskrivningarna resonerar applikationer/system (konsumentlagret) kring vilka datakällor som kan utnyttjas för att lösa en specifik uppgift. När en konsument har identifierat en lämplig producent av information, upprättas direktkontakt mellan dessa, varefter informationsöverföringen sker. En producent av information kan i sin tur integrera information från en eller flera underordnade producenter. Nedan diskuteras dessa tre lager mer ingående utifrån den exempelimplementering som har gjorts, samt utifrån potentiella vidareutvecklingar och möjligheter med RDF.

3.3.1 Metadatabas

I den nuvarande implementeringen av systemet baseras ontologin på RDF, se avsnitt 3.2.1. Vad som därtill krävs är ett sätt att hantera lagring och åtkomst till RDF över ett nätverk. Det finns för närvarande ett mindre antal databaser som stödjer direktlagring av RDF-modeller och frågespråk för RDF-data, däribland Sesame som den nuvarande implementeringen baseras på [14]. Utvecklingen av Sesame har utförts av Aidministrator, Holland, till en början som ett led i EU-finansierad forskning kring den semantiska webben (EU-IST-1999-10132) [15], men för närvarande genom forskningsmedel från NLnet Foundation [16]. Sesame är *open-source* och kan fritt laddas ner från Sourceforge [17].

Sesame inkluderar i sig ingen DBMS (*Database Management System*), utan kopplas till en extern databashanterare eller läser in data direkt till RAM-minnet. För tillfället stödjer Sesame kopplingar till MySQL [18], PostgreSQL [19] och Oracle9i [20], varav MySQL är den databashanterare som för tillfället rekommenderas. I praktiken är lagringssättet, databashanteraren, skild från RDF-logiken via Sesame SAIL (*Storage And Inference Layer*), vars API möjliggör kopplingar till godtyckligt lagringsmedia. Utöver Sesame och databashanteraren krävs bryggor mellan dessa i form av JDBC-drivrutiner (*Java Database Connectivity*). Dessa kan erhållas fritt från [18] och [19] för MySQL och PostgreSQL respektive.

En avgörande del av en RDF-databas är de frågespråk (*Query Languages*) som kan utnyttjas i syfte att exploatera informationen. Sesame stödjer tre SQL-liknande frågespråk; RQL (*RDF Query Language*), RDQL (*RDF Data Query Language*) och SeRQL (*Sesame RDF Query Language*). Den fundamentala skillnaden mellan dessa är huruvida de tar hänsyn till *RDF-Schema* eller inte. Det är ofta önskvärt att kunna besvara frågor som ”alla subklasser till X” eller ”alla resurser med egenskapen Y av klassen Z ”. För att kunna besvara dessa frågor korrekt måste systemet ta hänsyn till schemat som definierar begreppsvärlden, eftersom de instanser som skapas utifrån schemat inte inkluderar denna information. Av de frågespråk som stöds av Sesame är det enbart RQL som omfattas av denna typ av funktionalitet. På grund av detta valdes RQL som huvudsaklig mekanism för extrahering av data i exempelimplementeringen. En formell specifikation av RQL återfinns under [21]. Denna skiljer sig dock på vissa punkter från Sesames implementering av RQL.

Utifrån det schema och instanser av schemat som existerar för vårt fiktiva exempel, kan en RQL-fråga definieras utifrån följande ansats:

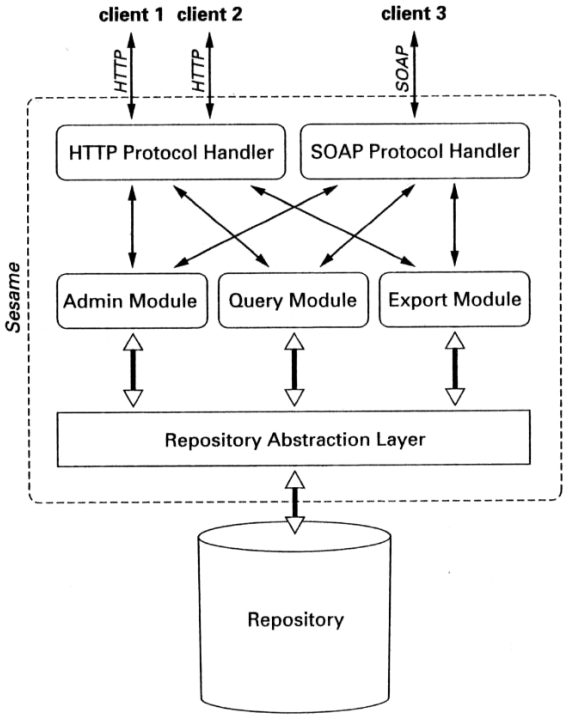
> *”Välj alla resurser (X), som i detta fall motsvaras av felrapporter, som har skapats av en organisation med namnet FOI”*

Denna förhållandevis enkla fråga får följande motsvarighet i RQL:

```
select X from {X} log:skapare {Y : log:Organisation} . log:namn {Z}
where Z like ”FOI”
using namespace log = http://www.foi.se/schemas/logistics#
```

I denna konstruktion återfinns de begrepp som definierades i schemat för vårt fiktiva exempel tidigare. Frågan inkluderar egenskapen ”skapare” som har klassen ”Organisation” som värde, som i sin tur har egenskapen ”namn” med en litteral av typen ”String” som värde. Alla dessa begrepp definieras i schemat ”log” som mappas till namnrymden ”http://www.foi.se/schemas/logistics#”. Svaret på denna fråga ges i form av URLer för de felrapporter som har behandlats av FOI. För en mer utförlig beskrivning av RQL och dess möjligheter se [22].

Figur 3.7 presenterar den övergripande arkitekturen för Sesame. Sesame är utvecklat med målet att vara oberoende av DBMS, vilket innebär att DBMS-specifik kod har koncentrerats till ett separat lager – *Repository Abstraction Layer* (RAL). RAL översätter metodanrop från funktionella moduler (*Admin Module*, *Query Module* och *Export Module*) till DBMS-specifika anrop. Fördelen med detta angreppssätt är att stöd för olika databashanterare kan implementeras utan förändring av övriga komponenter i Sesame. *Query Module* hanterar förfrågningar från användaren, i vårt fall RQL-frågor. *Admin Module* hanterar uppladdning och borttagning av RDF-data, medan *Export Module* hanterar export av en komplett databas med RDF-data. Beroende på vilken miljö som Sesame är aktiv inom krävs skilda kommunikationssätt. För hantering av kommunikation finns *protocol handlers* som implementerar skilda protokoll (http, SOAP etc.) och som fungerar som ett lager mellan klienter och funktionella moduler. En intressant aspekt av Sesame är dess P2P-funktionalitet (Peer-to-Peer). I praktiken innebär detta att ett antal Sesame-databaser från användarens perspektiv upplevs som en enkel databas. Förfrågningar till en databas propageras vidare till andra anslutna databaser med automatik [23].



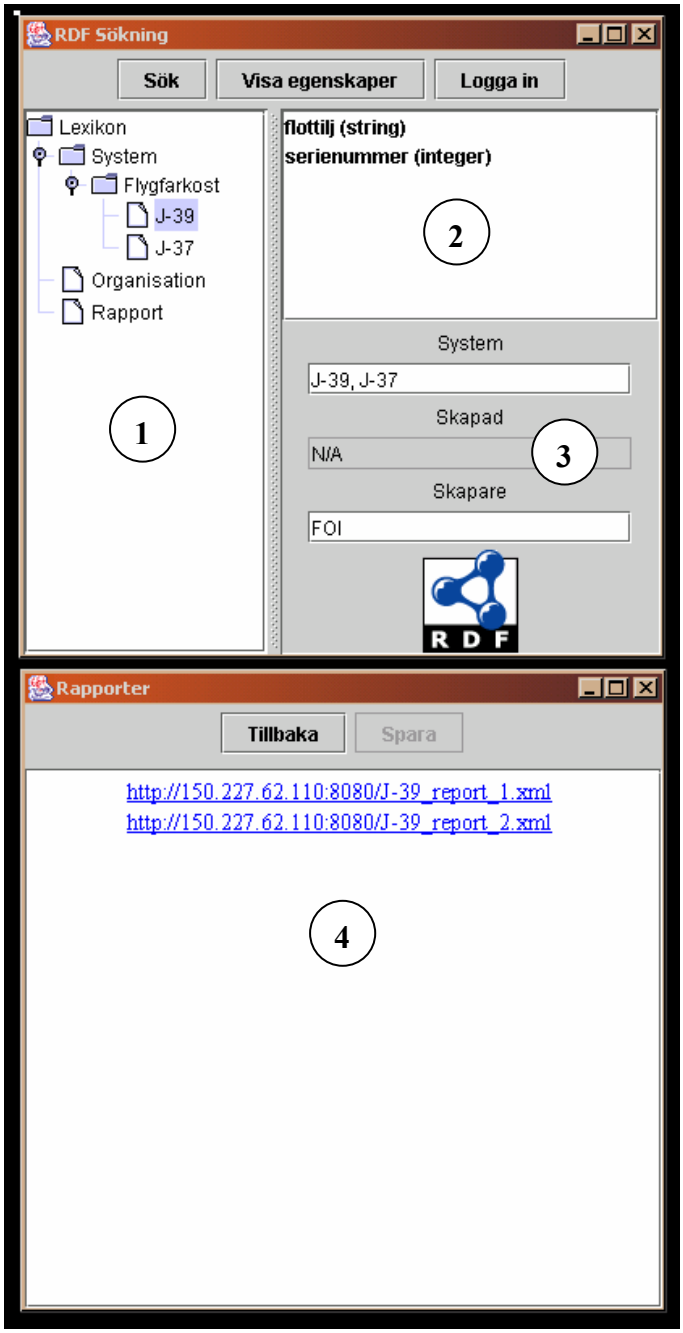
Figur 3.7. Övergripande arkitektur för Sesame [23].

3.3.2 Informationskonsumenter (tjänstekonsumenter)

För att exemplifiera hur en databas för lagring av RDF-data kan utnyttjas av informationskonsumerande system skapades ett enkelt gränssnitt för sökning och visualisering av distribuerade dokument. I figur 3.8 presenteras hur detta gränssnitt ser ut vid sökning efter felrapporter enligt ovan beskrivna scenario. Nedan följer en översiktlig beskrivning av ingående delar av gränssnittet (siffrorna nedan refererar till motsvarande siffror i figur 3.8).

1. Denna vy presenterar den ontologi (*RDF-Schema*) som metabeskrivningar av de distribuerade dokumenten baseras på. I detta fall presenteras de klasser som definierades i appendix A i en trädstruktur som reflekterar sub/superklassförhållanden
2. I detta fält presenteras egenskaper för klasser som selekteras i vy 1. När en klass väljs extraheras aktuella egenskaper ur ontologin och presenteras tillsammans med den datatyp som egenskapen utgörs av. I exemplet i figur 3.8 väljs klassen J-39 och egenskaperna flottilj och serienummer, samt dess respektive datatyp (”string” och ”integer”), redovisas i fältet
3. Dessa fält används för att söka efter dokument. Med ledning av de begrepp som ontologin omfattas av, och som presenteras i vy 1, kan databasfrågor konstrueras som exekveras mot metadatabasen. Fälten medför en förenkling av sökprocessen då användaren slipper skriva databasfrågor i RQL. I exemplet i figur 3.8 genomförs en sökning efter felrapporter för J-37 respektive J-39 som är skapade av organisationen FOI. Det är även möjligt att söka på mer abstrakta klasser som tillexempel ”Flygfarkost”, vilket motsvaras av en sökning efter alla subklasser till ”Flygfarkost”
4. I denna vy presenteras de felrapporter som överensstämmer med användarens sökning. För närvarande presenteras en URL till varje enskild felrapport. När användaren följer en av dessa länkar hämtas den aktuella filen genom ett *http-request* och presenteras i fönstret. I exemplet i figur 3.8 matchade två felrapporter användarens sökning, vilka presenteras med länkar i vyn

I den nuvarande versionen av systemet förutsätts det att felrapporter lagras i någon form av webbserver som kan nås via http. Detta skulle kunna kompletteras med åtkomst till WS som tillhandahåller felrapporter, genom att införa stöd för SOAP i applikationen. Vidare finns idag inget stöd för att hantera de rapporter som har identifierats, dvs. lagra dem lokalt.



Figur 3.8. Gränssnitt för sökning och inhämtning av distribuerade dokument

3.3.3 Informationsproducenter (tjänsteproducenter)

Utifrån scenariot, redovisat under avsnitt 3.1, förutsätts att ett antal heterogena databaser innehållande felrapporter existerar. Detta har förenklats något i exempelimplementeringen där kopplingen mellan databas och server som tillhandahåller felrapporter har utelämnats. I implementeringen förutsätts att felrapporter kan nås genom en URL. Detta har dock ingen betydelse för hur metadata kring felrapporten kan modelleras och hanteras. Denna kunskap

skall, som nämndes tidigare, ändå existera på en ”global” nivå och inte definieras på applikationsnivå. Vad som krävs av en fullständig implementering är funktionalitet hos servern som kan översätta anrop från ett konsumerande system till en giltig databassökning som kan extrahera eftersökta rapporter. Detta är något som måste lösas individuellt, beroende på vilken typ av databas som ska göras tillgänglig via nätverket och behandlas inte i denna studie.

Vad som är viktigt att poängtera är att alla rapporter som görs tillgängliga i systemet beskrivs utifrån samma ontologi. Detta är något som i ett verkligt system måste kunna hanteras på ett konsekvent sätt. Det måste finnas mallar som beskriver hur metadata kring en rapport skall konstrueras. För detta syfte krävs en utökning av den ontologi som har beskrivits ovan, till att inkludera restriktioner för vilka värden ett attribut kan anta etc.

3.3.4 Framtida utökningar

Det finns naturligtvis olika nivåer där RDF kan utnyttjas. I exempelimplementeringen används RDF för att modellera resurser (dokument) som utgörs av enskilda felrapporter. Det är naturligtvis möjligt att använda RDF, eller RDF-relaterade teknologier, på andra sätt också. En sådan möjlighet är att beskriva datakällor ur ett tjänsteperspektiv, tex. enligt den modell som DAML-S föreskriver. Den enkla ontologi som utvecklades för felrapporter, under avsnitt 3.2.1, fyller fortfarande en funktion, då den utgör den semantiska modell utifrån vilken resurskonsumerande system kan lokalisera intressanta rapporter. Databaser innehållande felrapporter förses med ett WS-gränssnitt, dvs. en server som kan hantera SOAP-meddelanden (exempelvis en Sesame-databas), och beskrivs utifrån de ontologier som DAML-S inkluderar. Giltiga sökningar kan därefter konstrueras utifrån ontologin för felrapporten, då denna beskriver vilken metadata som finns tillgänglig för enskilda dokument. Användning av DAML-S är kanske inte berättigad i detta relativt enkla fall. I en dynamisk miljö med flertalet heterogena tjänster, vars sammansättning förändras över tiden, blir dock inkorporering av semantik i beskrivningar av tjänster en nödvändighet.

Det kan även vara fördelaktigt att använda RDF på ytterligare en nivå i systemet. Om integrering av information från multipla datakällor omfattar insamling av enskilda värden hos datakällor kan det vara av betydelse att modellera en komplett datastruktur (semantisk modell) för en felrapport. Givet att enskilda databaser har interna representationer av begrepp inom felrapportsdomänen som skiljer sig åt måste en för databaserna globalt definierad semantisk modell utnyttjas. Vad som i praktiken oftast används för denna typ av informationsintegrering är *mediators* och *wrappers*. En klient (användare eller system) som är intresserad av viss typ av data deklarerar sina intressen hos en *mediator* som har till uppgift att distribuera sökningen till aktuella datakällor. Svaret från datakällorna integreras av *mediatorn* och levereras till klienten. Detta angreppssätt exemplifieras nedan.

Antag att följande semantiska modell existerar för en felrapport:

- Felrapport
  - System — klass System
    - System_ID — int
  - Komponent — klass Komponent
    - Feltyp — int
  - Datum — yy-mm-dd
  - ID — int

Två databaser existerar som kan leverera felrapporter men dessa skiljer sig med avseende på etablerad begreppsvärld. Databas A har följande struktur:

- Felrapport
  - System — klass System
    - ID — double
  - Komponent — klass Komponent
    - Feltyp — double
  - Datum — yyyy-mm-dd
  - Rapport_ID — double

Databas B följer den struktur som den semantiska modellen beskriver ovan. Vad som krävs vid sökning i dessa databaser (A och B) via någon form av *mediator* är att den interna representationen av en felrapport i databas A mappas mot den globalt definierade begreppsvärlden. I detta specifika fall finns både semantiska och syntaktiska skillnader mellan dessa två representationer. Klassen System har i den globala representationen ett attribut benämnd System_ID, medan representationen av klassen System i databas A har ett attribut benämnt ID. Detta är den semantiska skillnaden. Vidare skiljer sig de båda representationerna med avseende på datatyp för ett flertal attribut, int kontra double, samt syntaxen för hur ett datum skall representeras. Detta är den syntaktiska skillnaden. Vad som krävs är ett lager (*wrapper*) till databas A som omvandlar frågor från en *mediator*, som baseras på den globalt definierade begreppsvärlden, till databasens interna representation av begreppen. Detta gäller den semantiska såväl som den syntaktiska nivån. Vidare krävs att svaret från databasen formateras enligt den modell som har deklarerats på global nivå.

4. Diskussion

Nedan följer en diskussion kring tre centrala frågeställningar som är nödvändiga att behandla vid utveckling av metoder för informationsintegrering. Med informationsintegrering avses i detta fall sökning, inhämtning och integrering av information från heterogena datakällor i en dynamisk nätverksmiljö. Under avsnitt 4.1 diskuteras explicit definierad semantik i relation till XML- och RDF-Schema. I avsnitt 4.2 diskuteras semantiska nätverkstjänster, dvs. på vilka sätt som RDF och relaterade tekniker kan förbättra utnyttjandet av tjänster inom ett nätverk. Slutligen diskuteras under avsnitt 4.3 teknologier som kan utnyttja semantiska beskrivningar av tjänster och genom detta, utföra komplicerade uppgifter på uppdrag från användare eller system.

4.1 Explicit semantik

Det finns en del meningsskiljaktigheter vad gäller användning av RDF (RDF/XML) kontra XML (XML Schema). Det är dock viktigt att påpeka att dessa teknologiers användbarhet skiljer sig med avseende på tillämpning. Följande text beskriver några av de fundamentala skillnader som föreligger mellan de båda.

XML är främst inriktat mot hur dokument är strukturerade och mindre på hur data inom ett dokument skall tolkas. Detta medför att XML är bra som format då interagerande system känner till innehållet i dokumenten, men mindre bra i en dynamisk miljö då nya system ansluter sporadiskt och som inte känner till det exakta innehållet i dokumenten [24]. XML är kraftfullt för att beskriva hierarkiska strukturer, men medför ingen djupare innebörd av förekomsten av ett specifikt element eller relationer mellan element (semantik). Semantiken är implicit formulerad och kan tolkas av människor som läser dokumentet, men inte av maskiner som kräver en mer explicit formulering.

```
<person>
   <name>Martin<name>
</person>
<iso8601date>
   <W3Cprofiledate>2003-09-30</W3Cprofiledate>
</iso8601date>
```

Figur 4.1.Ett enkelt XML-dokument.

En människa som läser XML-dokumentet i figur 4.1 förstår att <name> kan relateras till <person> som namn på en person och att W3Cprofiledate är en specifik typ av iso8601date. Strukturellt och syntaktiskt ser dessa två ”utsagor” lika ut, fast typen av relation mellan elementparen är egentligen helt skilda. I RDF kan denna semantik uttryckas explicit och medför att även maskiner kan bearbeta informationen utifrån det sätt som skaparen av informationen avsåg. Därmed minskar behovet av att på förhand specificera exakt hur den information som skall utbytas mellan system måste se ut. I XML Schema finns inte något naturligt sätt att definiera klasser och klassers attribut. Det finns enbart element och subelement som möjligen skulle kunna tolkas som klass/subklass eller klass/attribut. I RDF Schema definieras klasser och dess attribut genom konstruktionerna rdfs:Class och rdfs:Property. Vad gäller ärvning finns inte heller där något verkligt stöd i XML Schema. I XML Schema är det möjligt att utöka eller begränsa existerande *element types*, men dessa nya typer blir inte automatiskt subklasser till de typer som de baseras på. I RDF Schema däremot stöds subklasser och subattribut genom konstruktionerna rdfs:subClassOf och

rdfs:subPropertyOf. En utförlig jämförelse mellan XML Schema, RDF Schema och DAML+OIL återfinns i [25].

Utöver dessa skillnader finns mer tekniska aspekter avseende behandling av dokument baserade på respektive teknologi. XML-dokument är till sin natur strikt hierarkiskt uppbyggda. Element som kan relateras till ett överordnat element måste nästlas inom detta. RDF-dokument följer däremot ett mer platt, trippel-baserat mönster, vilket kan göra bearbetning av dessa effektivare vid exempelvis sökning efter information. Skillnaden mellan sökning i XML respektive RDF-dokument kan liknas vid skillnaden mellan sökning i relationsdatabaser och hierarkiskt uppbyggda databaser [24].

4.2 Semantiska nätverkstjänster

Att hitta rätt information på Internet är ingen trivial uppgift med tanke på den enorma mängd information som finns tillgänglig idag. Svårigheten att precist lokalisera rätt information utgår från det faktum att en majoritet av informationen på Internet är ostrukturerad till sin natur [26]. Det har skett omfattande forskning kring metoder för sökning på Internet och en rad tekniker för detta har utvecklats; sökmotorer, meta-sökmotorer och metoder för efterbearbetning av inrapporterade länkar från en sökning (*Web content mining*) bland annat. Även om dessa metoder under vissa förutsättningar är fullt tillräckliga, kvarstår ändå den fundamentala problematiken med dagens Internet; nämligen att data ofta är ostrukturerad till sin natur och att information oftast är avsedd för konsumtion av människor och inte av maskiner. Den alltmer spridda användningen av XML, som separerar data från presentationsform, kan komma att förändra denna bild till viss del. Vad som dock saknas i detta sammanhang är förmågan att kunna uttrycka explicit semantik i XML, se avsnitt 4.1. För att möjliggöra effektiv sökning, bearbetning och integrering av information krävs strukturering av innehåll som inkorporerar formaliserad kunskap till stöd för system av skilda slag. Vad som är avgörande i detta sammanhang är att kunskap representeras på ett standardiserat och plattformsoberoende sätt som medför att heterogena system erhåller samma uppfattning av en domän. Detta är självklart ingen trivial uppgift att genomföra på Internet och flera tvivlar på betydelsen av ontologier i Internetsammanhang, se exempelvis [27]. I en avgränsad domän, som den logistiska eller hela NBF för den delen, bör utsikterna för framgångsrik implementering av gemensamma begreppsvärldar vara bättre. I själva verket bör gemensamma begreppsvärldar vara ett funktionellt krav i dessa sammanhang då interoperabilitet mellan system är en grundläggande förutsättning.

I en nätverksmiljö med många heterogena datakällor, vars information försörjer informationskonsumerande processer (exempelvis simuleringsmodeller), kommer ontologier att få stor betydelse. För att automatisera sökning, sammansättning och exekvering av tjänster, krävs det att tjänster beskrivs på ett konsistent vis. Genom att modellera de begrepp som beskrivningar av tjänster omfattas av, exempelvis klasser, egenskaper för klasser, samt begränsningar och relationer för klasser och dess egenskaper, kan enskilda tjänster förses med beskrivningar som är konsekventa och som inkluderar semantik (jämför med DAML-S). Utifrån tjänstebeskrivningar och dess relation till den modell som ontologin representerar, kan system resonera kring den bäst lämpade sammansättningen av tjänster för att lösa en specifik uppgift. Detta scenario kräver dock att fullständig interoperabilitet råder mellan de tjänster som skall interagera. I detta sammanhang framträder ytterligare en fördel med ontologier. Om målet utgörs av att integrera information från multipla datakällor krävs ett uniformt schema som kan appliceras för dessa. Det är inte realistiskt att anta att alla datakällor följer en gemensam semantik och syntax. I denna kontext kan en ontologi uttrycka ett ”globalt” schema som enskilda datakällor är skyldiga att mappa interna representationer mot (mha en

wrapper). Slutligen kan en ontologi användas för att standardisera kommunikation inom ett system. Genom att modellera den information som potentiellt kan utbytas mellan heterogena system kan interoperabel kommunikation säkerställas.

4.3 Utnyttjande av semantiska nätverkstjänster

En teknologi som anses avgörande i samband med den semantiska webben är så kallade agenter. Agenter utgörs i dessa fall av ”intelligenta” programvaror som kan utföra komplicerade uppgifter på uppdrag från användare och system, baserat på formaliserad kunskap, tex. tjänstebeskrivningar grundade på RDF. Det är trots allt inte meningsfullt att ha tillgång till maskintolkbara beskrivningar av resurser om det inte finns system som är tillräckligt ”intelligenta” för att utnyttja dem på ett effektivt sätt. Det finns flertalet arbeten som pekar på betydelsen av agentsystem som konsumenter av RDF-baserade resursbeskrivningar, se tex. [28]. Agenter utgör i dessa sammanhang ett lager mellan informationsproducerande och informationskonsumerande system genom att utföra lokalisering och integrering av distribuerad information. En delmängd av agentsystem är system baserade på mobila agenter. Mobila agenter karaktäriseras av en hög grad av autonomi och förmågan att transporteras mellan skilda värdmiljöer. Att en agent är autonom har flera fördelar [29]:

- Utnyttjande av datorresurser kan begränsas hos den som initierar ett uppdrag. Detta är av särskild betydelse för olika typer av mobila klienter med begränsade datorresurser, tex. handdatorer eller mobiltelefoner
- Nätverkskommunikation kan begränsas genom att en agent överförs till den lokala domänen för en resurs av intresse. Tex. vid sökning efter information kan agenten utföra sina uppgifter lokalt och inte över nätverket
- I dynamiska miljöer med sporadisk kontakt med nätverket kan en agent transporteras till en resurs av intresse när uppkopplingen lever, utföra sitt arbete lokalt och slutligen transporteras tillbaka när första bästa tillfälle ges. På detta sätt stör inte en sporadisk nätverksuppkoppling det arbete som måste utföras

Det finns stor potential i integrationen mellan WS och tekniker som tagits fram inom ramen för den semantiska webben. WS tillhandahåller en grundläggande arkitektur för utveckling och driftsättning av tjänster som potentiellt kan utnyttjas oberoende av plattform och programmeringsspråk hos klienter. Genom att införa ytterligare en nivå till detta koncept, i form av en formell semantik, kan utnyttjandet av tjänster inom ett nätverk effektiviseras. Detta gäller framförallt automatiserad sökning, sammansättning och exekvering av tjänster på uppdrag från användare eller system. Ett steg i denna riktning har tagits genom utvecklingen av DAML-S, vilket omfattar en mängd ontologier till stöd för dynamisk sammansättning av tjänster. DAML-S är inte begränsat till beskrivning av WS utan kan användas för godtycklig tjänst inom ett nätverk. En huvuddel av arbetet har dock inriktats mot just WS. Vad som krävs för att uppfylla visionen om den semantiska webben är verktyg som kan utnyttja semantiska beskrivningar av resurser på ett ”intelligent” sätt. Inom denna domän framstår agentsystem som en möjlig framtida inriktning.

5. Slutsatser

Ett system för integrering av information från multipla, heterogena datakällor, till stöd för logistisk M&S, bör baseras på en ontologisk grund. Ontologier skapar här förutsättningar för interoperabilitet mellan system på flera plan, genom att representera den information som kan utbytas mellan system, information som krävs för klassificering av tjänster, samt de generella informationsmodeller som utnyttjas inom logistik domänen. I detta sammanhang är det viktigt att poängtera att den kunskap som ontologin representerar delas av alla ingående parter (logistiska system och användare av dessa). Detta innebär att kunskapen bör deklareras på en för domänen global nivå och inte kapslas in på applikationsnivå. Inom detta fält framstår RDF (*Resource Description Framework*) och teknologier som baseras på RDF, som ett kraftfullt verktyg för att representera kunskap (ontologier) i dynamiska nätverksmiljöer. Utöver denna standardisering bör tjänster utformas efter en väletablerad och erkänd standard. I detta sammanhang utgör *Web Services* ett bra alternativ vars integrering med koncept från ”den semantiska webben” (RDF-relaterade teknologier) av många anses lyfta tjänstebegreppet till en ny nivå. Slutligen krävs även ”intelligenta” verktyg som kan utnyttja den infrastruktur som *Web Services* och ontologier bildar.

6. Fortsatt arbete

Eftersom ontologier anses ha en central roll i integrering av information från heterogena datakällor bör denna typ av kunskapsrepresentation studeras närmare. I detta sammanhang kan det vara värdefullt att inventera mer kraftfulla språk för kunskapsrepresentation som DAML och OWL. Vidare bör DAML-S utvärderas och testas för att utröna dess lämplighet som grund för automatisering av nätverksbaserat tjänsteutnyttjande. I detta sammanhang är det även av vikt att studera och testa tekniker och metoder som på ett effektivt sätt kan utnyttja semantiska beskrivningar av nätverksbaserade resurser, exempelvis agentsystem. Ovan nämnda inriktningar bör appliceras på ett mer fullständigt scenario, jämfört med det som redovisas i denna rapport. Scenariot bör inkludera såväl informationskonsumerande som informationsproducerande parter för att belysa hur ett krav på information kan fullgöras genom inhämtning och integrering av information från heterogena datakällor

7. Referenser

[1] Institute of Electrical and Electronics Engineers (IEEE), *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*, New York, 1990.
[2] A.S. Tanenbaum, M. van Steen, *Distributed Systems, Principles and Paradigms*, Prentice Hall, Upper Saddle River, New Jersey, 2002.
[3] R. Nagappan, R. Skoczyklas, R.P. Sriganesh, *Developing Java Web Services*, Wiley Publishing Inc., Indianapolis, Indiana, 2003.
[4] A. Skonnard, *Understanding SOAP*, http://msdn.microsoft.com/library/enus/dnsoap/html/understandsoap.asp, 2003.
[5] SOAP Specifications, http://www.w3.org/2000/xp/Group/.
[6] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *Web Services Description Language (WSDL) 1.1*, http://www.w3.org/TR/wsdl, 2001.
[7] Universal Description, Discovery and Integration of Web Services, http://www.uddi.org.
[8] J. Euzenat, *Research Challenges and Perspectives of the Semantic Web*, IEEE Intelligent Systems, http://www.computer.org/intelligent, 2002.
[9] T. Berners-Lee, J. Hendler, O. Lassila, *The Semantic web*, Scientific American, http://www.scientificamerican.com, May 2001 issue.
[10] S. Powers, *Practical RDF*, O'Reilly & Associates, Sebastopol, California, 2003.
[11] RDF Specifications, http://www.w3c.org/RDF/.
[12] The DAML Services Coalition, *DAML-S: Semantic Markup for Web Services*, http://www.daml.org/services/daml-s/0.9/daml-s.pdf, 2003.
[13] D. Brickley, *RDF Vocabulary Description Language 1.0: RDF Schema*, http://www.w3c.org/TR/rdf-schema/, 2003.
[14] Sesame RDF Repository, http://sesame.aidministrator.nl/.
[15] On-To-Knowledge: Content-driven Knowledge-Management through Evolving Ontologies, http://www.ontoknowledge.org/.
[16] NLnet Foundation, http://www.nlnet.nl/.
[17] Sesame at Sourceforge, http://sourceforge.net/projects/sesame/.
[18] mySQL, http://www.mySQL.org.
[19] postgreSQL, http://www.postgreSQL.org.
[20] Oracle9i, http://www.oracle.com.
[21] Specifikation av RQL, http://139.91.183.30:9090/RDF/RQL/.
[22] RQL Tutorial, http://sesame.aidministrator.nl/publications/rql-tutorial.html.
[23] J. Broekstra, A. Kampman, F. Van Harmelen, *Sesame: An Architecture for Storing and Querying RDF Data and Schema Information*, In: *Spinning the Semantic Web*, The MIT Press, London, 2003.
[24] S. Decker, F. van Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein, S. Melnik, *The Semantic Web – on the respective Roles of XML and RDF*, IEEE Internet Computing, 2000.
[25] Y. Gil, V. Ratnakar, *A Comparision of (Semantic) Markup Languages*, American Association for Artificial Intelligence, 2000.
[26] M. Chau, D. Zeng, H. Chen, M. Huang, D. Hendriawan, *Design and evaluation of a multi-agent collaborative Web mining system*, Decision Support Systems 998, 2002.
[27] J.L. Arjona, R. Corchuelo, M. Toro, *Automatic Extraction of Information from the Web*, http://citeseer.nj.nec.com/arjona02automatic.html.
[28] K. Hui, S. Chalmers, P. Gray, A. Preece, *Experience in using RDF in Agent- mediated Knowledge Architectures*, American Association for Artificial Intelligence, 2003.
[29] S. Bergamaschi, G. Cabri, F. Guerra, L. Leonardi, M. Vincini, F. Zambonelli, *Mobile Agents for Information Integration*, http://citeseer.nj.nec.com/bergamaschi01mobile.html.

Appendix A – RDF-Schema

Från vårt fiktiva scenario, se avsnitt 3.1, har ett antal egenskaper för felrapporter valts ut för att illustrera hur *RDF-Schema* kan utnyttjas. Dessa utgörs av:

- Vilket system som felrapporten avser, tex. J-39, J-37 osv.
- Serienummer för systemet (ID-nummer)
- Flottiljtillhörighet
- Datum för det inträffade
- Namn på skapare av felrapporten

Först skapas en hierarki för de sammansatta system som felrapporterna behandlar. Detta måste genomföras för att senare kunna söka på mer generella begrepp av typen ”flygfarkost”, istället för att strikt definiera ett intresse för J-39, J-37 osv. I detta enkla exempel kommer attributet ”serienummer” att knytas till toppklassen ”System” och attributet ”flottiljtillhörighet” till klassen ”Flygfarkost”. För att definiera klasser och subklasser används konstruktionerna i figur 1. Den första satsen talar om att “System” är en klass. Den andra satsen deklarerar att ”Flygfarkost” är en klass, samt en subklass till klassen ”System”. För att förse klasserna med attribut används konstruktionerna i figur 2.

```
<rdf:Description rdf:ID="System">
                <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="Flygfarkost">
            <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
            <rdfs:subClassOf rdf:resource="#System"/>
</rdf:Description>
```

Figur 1. Definition av klasser och subklasser i RDF-Schema.

```
<rdf:Description rdf:ID="serienummer">
            <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
            <rdfs:domain rdf:resource="#System"/>
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
</rdf:Description>

<rdf:Description rdf:ID="flottiljtillhörighet">
            <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
            <rdfs:domain rdf:resource="#Flygfarkost"/>
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</rdf:Description>
```

Figur 2. Definition av attribut i RDF-Schema..

Den första satsen i figur 3.2 specificerar att ”serienummer” är ett attribut som tillhör klassen ”System” (*domain*), samt att det är av typen ”integer” (*range*). Den andra konstruktionen anger att ”flottiljtillhörighet” är ett attribut till klassen ”Flygfarkost” (*domain*), samt att det är av typen ”string” (*range*). När detta är avklarat kan felrapportens egenskaper definieras. En felrapport representeras i detta exempel av klassen ”Rapport” som har ett antal attribut, nämligen; ”avsettSystem”, skapare och skapad. Klassen ”Rapport” definieras enligt figur 3. Till klassen ”Rapport” knyts attributen angivna ovan enligt figur 4.

```
<rdf:Description rdf:ID="Rapport">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

Figur 3. Definition av klassen ”Rapport”.

```
<rdf:Description rdf:ID="skapare">
        <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
        <rdfs:domain rdf:resource="#Rapport"/>
</rdf:Description>

<rdf:Description rdf:ID="skapad">
        <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
        <rdfs:domain rdf:resource="#Rapport"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</rdf:Description>

<rdf:Description rdf:ID="avsettSystem">
        <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
        <rdfs:domain rdf:resource="#Rapport"/>
        <rdfs:range rdf:resource="#System"/>
</rdf:Description>
```

Figur 4. Definition av attribut till klassen ”Rapport”.

Notera i figur 4 definitionen av attributet “avsettSystem” som är av typen “System”, som har deklarerats ovan. Om det tillkommer flera olika typer av rapporter som systemet måste hantera, kan dessa enkelt differentieras genom att subklassa klassen ”Rapport” och förse dessa nya klasser med rapportspecifika attribut.

Utifrån ovan beskrivna RDF-schema är det därefter möjligt att skapa instanser av klassen ”Rapport”. I figur 5 nedan beskrivs en felrapport för systemet J-39. I detta fall är rapporten upprättad av organisationen FOI som beskrivs med namn och URL. Vidare framgår av informationen att systemet tillhör flottiljen F6 och att det har serienumret 1234. Notera deklarationen av namnrymden ”log” som ”pekar” på det vokabulär som vi precis har definierat.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                xmlns:log="http://150.227.62.110:8080/logistics#">

<rdf:Description rdf:ID="J-39-1">
                <rdf:type rdf:resource="http://150.227.62.110:8080/logistics#J-39"/>
                <log:serienumme rrdf:datatype=
                  "http://www.w3.org/2001/XMLSchema#integer">1234</log:serienummer>
                <log:flottilj rdf:datatype=
                  "http://www.w3.org/2001/XMLSchema#string">F6</log:flottilj>
</rdf:Description>

<rdf:Description rdf:ID="FOI">
                <rdf:type rdf:resource="http://150.227.62.110:8080/logistics#Organisation"/>
                <log:url rdf:resource="http://www.foi.se"/>
                <log:namn rdf:datatype=
                  "http://www.w3.org/2001/XMLSchema#string">FOI</log:namn>
</rdf:Description>

<rdf:Description rdf:about="http://150.227.62.110:8080/J-39_report_1.xml">
                <rdf:type rdf:resource="http://150.227.62.110:8080/logistics#Rapport"/>
                <log:skapare rdf:resource="#FOI"/>
                <log:skapad rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#string">2003-01-05</log:skapad>
                <log:avsettSystem rdf:resource="#J-39-1"/>
</rdf:Description>
</rdf:RDF>
```

Figur 5. Instans av klassen ”Rapport”.