Institutionen för Informatik
Handelshögskolan vid Göteborgs universitet
Magisteruppsats HT-02

# WEB SERVICES

## Har Teknologin En Framtid?

ABSTRAKT

*Ett nytt begrepp som dykt upp på senare tid är Web services. Denna teknologi består av tre olika standarder som tillsammans underlättar kommunikationen mellan olika system. Teknologin är avsedd både för intern användning, inom det egna nätverket och för extern användning, över Internet. Eftersom teknologin är relativt ny är syftet med detta arbete att undersöka vad Web service teknologin är och om den har en framtid. Uppsatsen baseras på litteraturstudier för att få en förståelse av teknologin och dess uppbyggnad samt av intervjuer med personer från olika företag som utvecklar Web services, där målsättningen var att få reda på deras åsikter om teknologin. Resultatet av intervjuerna och litteraturstudien visade att både utvecklarna och företagen som använder Web services är positiva till teknologin. Dock finns det ett antal hinder som måste överkommas för att garantera en riktig framgång.*

Författare: Roxanna Gustafson
Handledare: Kari Wahll

# Innehållsförteckning

# 1 Inledning

En nytt begrepp som uppkommit på senare år är Web services. Trots att begreppet är ganska nytt, bygger Web services på en gammal trend som pågått de senaste åren, där målsättnigen är att företagen ska få möjlighet att bryta ner stora och oöverblickbara system till mindre delar för att kunna hantera det lättare. Web services bygger på tre olika standarder, *Simple Object Access Protocol* (SOAP), *Web Services Description Language* (WSDL) och *Universal Discovery, Description and Integration* (UDDI). Det är via deras interaktion som företag idag kan vara glada över en enklare hantering av sina system. Grunden i Web services är *eXtensible Markup Language* (XML). XML är ett regelverk för att skapa märkspråk, ett så kallat metaspråk som gör det möjligt att utbyta information mellan olika applikationer oberoende av det dataformat som används internt.[1] Genom att använda XML i Web services bidrar det till att teknologin även kan använda sig av Internet för att erbjuda kommunikation mellan olika företag och system.

## *1.1 Bakgrund*

När Internet först skapades var det av militära skäl. Det var USA som upptäckte att det fanns ett stort kommunikationsproblem under det kalla kriget. Viljan att kunna ha ett kommunikationssätt som skulle fungera även om telefonnätet slogs ut gjorde att man lyckades 1969 utveckla ett datanätverk, *ARPANET* som inte hade någon central auktoritet.
Detta datanätverk skulle kunna ta emot och sända information även om en länk i kedjan bröts, genom att ta en annan väg. ARPANET blev grundstenen till dagens Internet. [2] 1991 släpptes *World Wide Web* (WWW) av forskningsgruppen *le Conseil Européen pour la Recherche Nucléaire* (CERN*)* och utvecklaren hette Tim Berners-Lee. Ingen hade kunnat förutse att det skulle leda till en sådan förändring i samhället. Tjänsten var liten från början men ökade explosionsartat i användning. [3] Internet har idag genomgått många förändringar. I den första fasen fanns det bara statiska dokument som var skrivna i *HyperText Markup Language* (HTML), där företag enkelt kunde lägga ut information om sig själva. Denna fas kom att kallas *Dokumentwebben*. Efterhand att tiden gick ökade företagens önskan att integrera sig med kunder genom transaktioner, vilket ledde till att mer dynamiska HTML-dokument skapades. Bland dessa återfinns teknikerna *Active Server Pages* (ASP), *Common Gateway Interface* (CGI), *Java Server Pages* (JSP) med flera. Denna fas kom att kallas *Applikationswebben*. När företagen insåg vinsten med att kunna göra elektroniska affärer med varandra behövdes något mer än både Dokumentwebben och Applikationswebben, vilket ledde till den tredje fasen, *Tjänstewebben*. Här används idag tekniker som XML och Web services. Denna fas är fortfarande i utvecklingsstadiet men går mot en alltmer vanlig och mogen Internetstruktur. [4]

---

[1] Brett McLaughlin. *JAVA and XML.* (Sebastopol: O´Reilly, 2000), 2.

[2] Bruce Sterling (1992). *Short History of the Internet* [online].
Tillgänglig: http://w3.aces.uiuc.edu/AIM/scale/nethistory.html. [2002-09-16]

[3] CERN (1997). *History and growth* [online]. Tillgänglig:
http://public.web.cern.ch/Public/ACHIEVEMENTS/WEB/history.html . [2002-09-16]

[4] Arthur Ryman (2000). *Understanding Web Services.* [online]. Tillgänglig:
http://www7.software.ibm.com/vad.nsf/Data/Document4362?OpenDocument&p=1&BCT=#&Footer=1
[2002-12-02]

## *1.2 Problemområde*

Eftersom Web service teknologin är relativt ny finns det många frågor kring den både för utvecklarna och för företagen. Det finns ett antal utmaningar som Web service måste komma igenom innan en verklig stor framgång kan uppnås. En av dessa utmaningar som måste övervinnas är att ena alla leverantörerna runt en standard.[5] Leverantörerna Microsoft, IBM och Sun måste komma fram till en kompromiss samtidigt som de i så hög utsträckning som möjligt vill få igenom sina egna specifikationer. En annan utmaning är att få en ökad säkerhet hos systemen. Här går dock utvecklingen framåt hela tiden och det erbjuds redan idag ett antal lösningar. Något som också är viktigt för framgången är att uppfylla löftena om affärsnytta. Eftersom affärsnytta hänger tätt samman med kostnaden är det viktigt att övertyga företagen om nyttan i Web services. Enligt analysföretaget Forrester kommer företagen att använda Web services från och med år 2004 i alla IT-projekt. De tror att kostnaden för integration kommer att falla men att kostnaden för implementering kommer att vara fortsatt hög.[6]

Många analysföretag spår en strålande framtid för Web service teknologin, anledningen till detta tros vara en ökad investeringsvilja hos företagen.[7] Företagen anser att Web services har en enorm potential, samtidigt som det råder en gemensam medvetenhet om teknologins omognad. Trots detta är intresset för Web services är stort och växer mer och mer hela tiden, men det är viktigt att mitt i detta flöde få en balans mellan förväntningar och verklighet. Risken är annars att företagen blir besvikna och tappar tilltron till teknologin.[8]

Företaget *Wilson Logistics Group* använder sig av Web services i sitt företag. De anser att den största fördelen med Web services är att den ger möjlighet till återanvändning. De kan återanvända sina gamla funktioner och system. Företaget utnyttjar redan tekniken för extern kommunikation. Eftersom Web services är lite osäker använder det flesta företagen tekniken i sina interna system, men de ser den externa möjligheten för kommunikation som en potentialen hos tekniken.[9]

Även *myndigheterna* i Sverige kommer att använda Web services för att knyta ihop alla myndigheterna och sudda gränserna. På detta sätt kan de skapa ett standardiserat system för att kunna utbyta information och tjänster. Anledningen till att de valde Web services är att tekniken erbjuder återanvändning av investeringar hos kommunerna, vilket leder till bättre kommunikation mellan dem. På detta sätt ska en medborgare endast behöva gå in på en webbsida och därifrån komma åt allt den behöver. Förhandlingarna har redan börjat och enligt IT-cheferna och ansvariga för arbetet är det inte omöjligt att genomföra arbetet. De är positiva och tror att säkerhets- och identifieringsproblemen kan överkommas. Vidare är tanken att genom att ena alla myndigheterna blir kostnaden lägre och myndighetskrånglet kan underlättas för medborgarna.[10]

---

[5] Ethan Cerami, *Web Services Essentials.* (O´Reilly & Associates: Sebastopol, 2002), 24.

[6] Martin Wallström, ”Tre tunga utmaningar”, *Computer Sweden*, 23 oktober 2002, 24-26.

[7] Martin Wallström, ”Snabb expension lockar många aktörer” *Computer Sweden*, 9 oktober 2002, 27

[8] Kent Olofsson, ”Web services vid brytpunkten”, *Computer Sweden*, 16 oktober 2002, 31.

[9] Lars Danielsson(2002). *Web services förändra utvecklingen i grunden.* [online]. Tillgänglig: http://computersweden.idg.se/ArticlePages/200209/26/20020926155442_CS465/20020926155442_CS465.dbp.asp [2002-12-17]

[10] Henrik Svidén, ”Myndihetssverige suddar gränserna-Web services knyter ihop alla myndigheter”, *Computer Sweden*, 14 oktober 2002, s.4

Ett annat företag som använder sig av Web services idag är *Storebrand*, Norges största försäkringsbolag. Företaget hjälper 6500 kundföretag med hanteringen av de anställdas pensioner och försäkringar. Dessa kundföretag har i sin tur 390 000 anställda sammanlagt. Arbetet som Storebrand gör är att varje gång en anställd slutar, nyanställs eller får nya lönevillkor måste personakterna uppdateras i Storebrand pensionssystem. Dessa kan vara förändringar som är upp till 15000 per dag. Detta arbete krävde förr 50 personer som fick ta emot fax, e-post eller filer med uppgifter. Utmaningen var att automatisera informationsflödet mellan kunderna och Storebrands pensionssystem för att kunna minska kostnaden för den manuella hanteringen och eliminera risken för felaktiga inmatningar. Anledningen till att valet föll på Web services är att den möjliggör återanvändning och extern kommunikation. De såg möjligheten att återanvända sina interna funktioner och göra dem tillgängliga på webben för extern användning. Lösningen som används idag fungerar på det sättet att varje förändring i någon av Storebrands kunders lönesystem uppdateras automatisk hos Storebrand pensionssystem utan handpåläggning. Företaget är mycket nöjda med resultatet.[11]

Det är inget fel på intresset för Web services och många analytiker tror på framgången för Web service teknologin. Anledningen till detta är att teknologin erbjuder möjligheter att skapa sådana applikationer som kopplar ihop data som aldrig hade kunnat kopplats ihop tidigare. Teknologin erbjuder inte endast utveckling av applikationer utan en snabb sådan, vilket leder till minskade kostnader för företag. Web services tillhandahåller ett enkelt och accepterat ramverk för utveckling.[12] En annan anledning till analytikers tro på teknologin är att det finns många stora leverantörer bakom den, bland dessa återfinns Microsoft, IBM, Sun. Företag som tidigare aldrig arbetat mot samma mål gör detta nu och de menar att detta endast kan leda till ett resultat.[13]

Eftersom Web service teknologin är ganska ny står Företag och utvecklare inför ett svårt val. De kan antigen börja tillämpa denna nya teknologi eller vänta tills en annan, ersättande teknologi dyker upp.

## 1.3 Problemformulering

Efter att ha angett problemområdet som beskrivits ovan kan problemformuleringen för denna uppsats beskrivas vara: Har Web services teknologin en framtid?

## 1.4 Syfte

Syftet med denna uppsats har varit att få en förståelse i hur Web services teknologi fungerar. I detta arbete ville jag undersöka mer ingående de olika Web service standarderna för att förstå

---

[11] Ingemar Eriksson (2002). *Web services i praktiken: Storebrand har gått från hajp till handling.* [online]. Tillgänglig: http://cio.idg.se/artikelarkiv/artikel.asp?ID=373 [2002-12-17]

[12] Tom Clement (2001) *Web services: Why all the buzz?* [online]. Tillgänglig: http://news.com.com/2010-1078-281503.html?legacy=cnet [2002-12-17]

[13] Andrew Binstock. *Staking New Territory, Breaking New Ground.* [online]. Tillgänglig: http://www.devx.com/javaSR/articles/binstock/binstockp.asp

hur dessa arbetar tillsammans med varandra för att bygga upp en Web Service. Jag ville också få en inblick i hur personer som utvecklar denna teknologi känner inför de olika utmaningar som ställts upp inför dess framtid.

## *1.5 Avgränsning*

Arbetet har avgränsats till att behandla ämnet teoretiskt, vilket innebär att det inte har gjorts någon studie av praktiska tillämpningar.

# 2 Teoretisk bakgrund

I detta kapitel beskrivs Web service teknologin kortfattat. Läsaren hänvisas till Bilaga 1 - Ordlista för att lättare förstå alla begrepp som dyker upp i uppsatsen och till Bilaga 3 - Web Service Arkitekturen för en mer ingående beskrivning av Web service arkitekturen.

## *2.1 Web service teknologin*

Ordet ”Web services” står inte för tjänster till en slutanvändare utan till ett system, vilket innebär att Web service teknologin är transparent för en slutanvändare. Detta betyder att allt som sker mellan två system är osynligt för slutanvändaren. Web services kan genomföra nästan vilken uppgift som helst. Dess funktionalitet kan vara så enkel som att räkna samman två tal eller komplext genom att hantera beräkningar på hela system och dess relationer.[14]

En Web service kan skapas i vilket programmeringsspråk som helts, eftersom de är operativsystem-, plattforms- och programmeringsspråkoberoende. En av anledningarna till att alla Web services kan kommunicera med varandra är att de ”pratar” samma språk, som är XML. De använder XML för att beskriva sitt gränssnitt och för att koda sina meddelanden.[15]

En Web service bör bestå av två egenskaper; den bör vara självbeskrivande och upptäckbar. Självbeskrivande innebär att tjänsten bör innehålla en läslig dokumentation som beskriver tjänsten, detta för att underlätta integrationen av tjänsten för andra. WSDL-protokollet används för detta ändamål. Upptäckbar innebär att när en tjänst skapats bör den vara lätt att publicera, detta för att andra intresserade parter ska kunna hitta tjänsten, UDDI erbjuder denna egenskap.[16] Figur 1 visar de olika delar som ingår i en Web service.

Som redan nämnts tidigare, bygger Web services på en gammal trend. Med trend menas att det finns ett antal olika teknologier som kommit upp och som erbjudit liknande lösningar som Web service teknologin erbjuder idag.[17] Det finns några teknologier för att anropa tjänster över ett nätverk, tjänster som är plattforms- och språkoberoende och som man inte behöver veta hur de är implementerade utan endast deras gränssnitt. Det finns även teknologier för att få en kommunikation mellan olika system, och som ger möjlighet att dela upp stora system till mindre delar för lättare hantering. Med hjälp av några teknologier kan man återanvända sina gamla system, när ett nytt skapas. Alla dessa trender har funnits länge och tidigare teknologier som erbjudit dessa lösningar är exempelvis *Common Object Request Broker Architechture* (CORBA)*, Distributed Component Object Model* (DCOM)*, Remote Procedure Call (*RPC*)* med flera.[18] Den stora skillnaden dock mellan dessa och Web services är att dessa inte riktigt har nått upp till de krav som ställts på dem och de har heller inte lyckats hålla vad det har utlovat. Det är här Web services har en stor utmaning som den måste klara av.

---

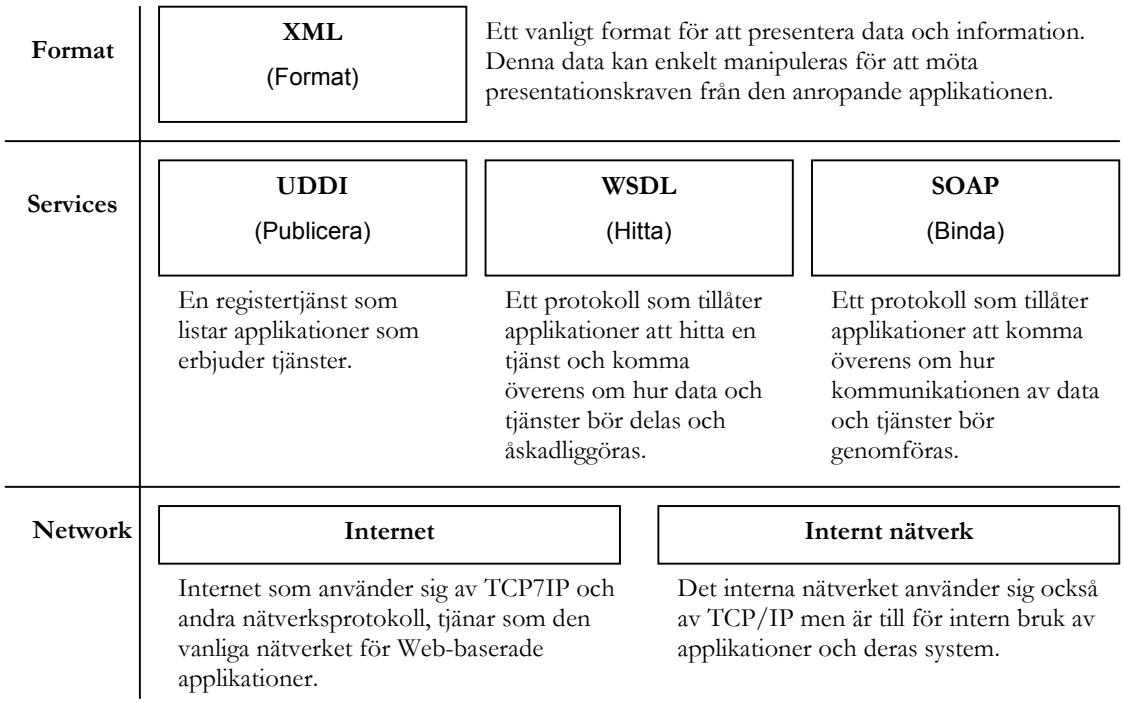
[14] Harvey M Deitel, Paul. J Deitel. (2002). *Web Services A Technical Introduction.* [pdf]. Tillgänglig: http://www.deitel.com/newsletter/20020523/articles/WebServices_SOAP-WSDL-UDDI.pdf [2002-10-01]

[15] Systinet (2002). *Introduction to Web Services.* [white paper]. Tillgänglig: http://www.systinet.com/download/d3f2d622be58c0647c6cb125e4514983/wp_Introduction_to_Web_Services.pdf [2002-10-29]

[16] Cerami, 4-5.

[17] Magnus Höij, ”Web services - Drömmen som blev sann”, *Computer Sweden*, 2 oktober 2002, 22.

[18] Henrik Olsson, ”Web services bra men inte nytt”, *Computer Sweden*, 28 oktober 2002, 15.




*Figur 1 En enkel beskrivning över de olika delarna som ingår i en Web service.*[19]

## 2.2 Definitioner

Eftersom Web service teknologin är ganska ny, finns det ett antal olika definitioner på vad det är, nedan nämns några av dem:

> *" Standardiserade system för utbyte av information och tjänster mellan webbplatser "*
> *(*Kent Olofsson, 2002. *Computer Sweden, Nr 99 s. 23).*

> *"A web service is any service that is available over the Internet, uses a standardized XML messaging system, and is not tied to any one operating system or programming language "*
> *(*Ethan Cerami, 2002. *Web Services Essentials s. 3).*

> *" Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. Any piece of code can be transformed into a Web service. Services can be legacy functions or new software. Also, all components in a system can become services "*
> *(*IBM, 2002 *)*

[19] Joe Clabby, *Web Services Explained*, (New Jersey: Prentice Hall, 2002), 26.

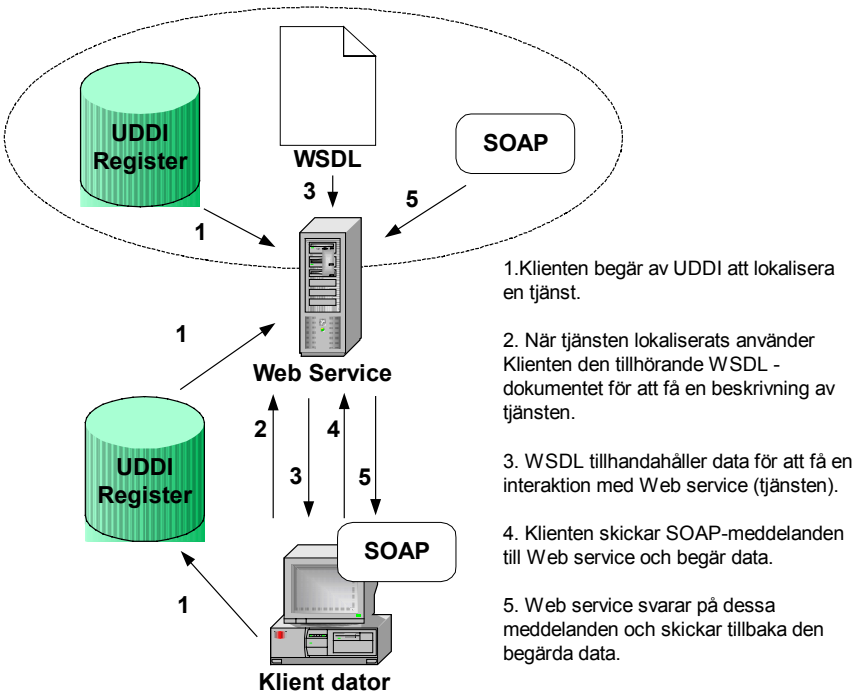
Innan vi går närmare in på Web service teknologin måste ett antal begrepp klargöras som jag använder i uppsatsen. Dessa är *Web service teknologin, Web service, tjänst, system* och *applikation*.

*Web service teknologin* använder jag som det samlande begreppet för standarderna tillsammans med de regler och krav som finns. Med *Web Service* menas en *tjänst* som erbjuds antingen intern inom det egna nätet eller extern över Internet. Denna tjänst (Web service) finns i ett *system* som en *applikation*. Tjänsten kan anropas från en annan applikation på ett annat system över Internet och blir då en extern Web service, eller agera som en ”mellanhand” mellan två interna system. Detta kommer att klargöras bättre senare.

## 2.3 Arkitekturen

För en mer ingående beskrivning av arkitekturen hänvisas läsaren till *Bilaga 3 - Web Service Arkitekturen.*

Web Service arkitekturen består av tre standarder som tillsammans skapar en Web service. Dessa är SOAP, WSDL och UDDI. Dessa tre standarder möjliggör en kommunikation mellan en Web service och en anropande applikation på ett system, där kommunikationen är oberoende av programmeringsspråk, operativsystem och hårdvaruplattform.[20] Standarden SOAP tillhandahåller en kommunikationsmekanism mellan en Web service och en applikation, standarden WSDL erbjuder en gemensam metod för att beskriva en Web service för andra program och standarden UDDI gör det möjligt att söka efter en specifik Web service. När dessa tre standarder slåss samman kan en utvecklare skapa en applikation som erbjuder en tjänst och sedan kan denna tjänst publiceras på webben.[21] Figur 2 visar relationen mellan de olika standarderna. En enkel beskrivning av dessa tre standarder följer.




*Figur 2 SOAP, UDDI och WSDL i interaktion med varandra för att nå en Web service.*

[20] Cerami, 3-5.

### 2.3.1 SOAP

För en mer ingående beskrivning av SOAP hänvisas läsaren till rubriken SOAP (Simple Object Access Protocol) under Bilaga 3 - Web Service Arkitekturen.

SOAP är ett XML-baserad protokoll för utbyte av data mellan system som finns distribuerade över ett nätverk. Den första versionen av protokollet blev tillgänglig för allmänheten 1999 och var ett resultat av ett samarbete mellan utvecklare på *Microsoft*, *DevelopMentor* och *UserLand Software*. Deras arbete ledde till att den första versionen SOAP 1.1 vilket överlämandes till W3C för standardisering, släpptes för allmänheten den 8 maj 2000.
W3C har senare lagt till ändringar och släppt ytterligare en version, SOAP 1.2 i december 2001 som ännu inte antagits som en rekommendation.[22]

Eftersom SOAP använder sig av XML bidrar det till att protokollet är plattforms- och språkoberoende. SOAP består av tre delar: ett *envelope* som specificerar ett antal regler för inkapslad överföring mellan datorer, *encoding rules* (kodningsregler) som används för omvandling av datatyper och *RPC conventions* som är regler för meddelandesystemet.[23]

### 2.3.2 WSDL

För en mer ingående beskrivning av WSDL hänvisas läsaren till rubriken WSDL (Web services Description Language) under Bilaga 3 - Web Service Arkitekturen.

WSDL är en specifikation som definierar hur en Web service beskrivs med hjälp av XML. Ett WSDL-dokument beskriver vilka funktioner en Web service erbjuder, hur den kommunicerar och var den kan nås. WSDL tillhandahåller en strukturerad mekanism för att beskriva funktionerna som en Web service kan genomföra, formatet på meddelanden som den kan hantera, protokollen som den stödjer och åtkomsten av en instans i en Web service.[24]

WSDL utvecklades av *IBM* och *Microsoft*. WSDL 1.1 specifikationen överlämnades till W3C i mars 2001 för standardisering. Den har dock ännu inte nått standardnivå.[25] Trots att teknologin fortfarande befinner sig under utveckling tillhandahåller nästan alla produkterna stöd för WSDL.

WSDL är oberoende av plattform och programmeringsspråk, den används främst för att beskriva tjänster. Varje Web service som publiceras på Internet medföljer ett WSDL-dokument som listar dess tjänster och instruktioner för dess användning. Ett WSDL-dokument fastställer typen på meddelanden som Web servicen kan skicka och ta emot samt specificerar vilken data den anropande applikationen måste tillhandahålla för att Web servicen ska kunna genomföra sin uppgift. WSDL språket och dokumentet är endast avsedd för applikationer och inte för människor, därför är det bara applikationer som förstår dess innehåll.[26]

---

[21] Ibid., 15-18.

[22] Vivek Chopra, Zaev Zoran, Gary Damschen, Chris Dix et al. (2001). *Professional XML Web Services* [online]. Tillgänglig: http://www.vbip.com/books/1861005091/chapter_5091_01.asp [2002-10-01]

[23] Cerami, 50.

[24] Systinet, http://www.systinet.com/download/332fd7be0e8021ff85e15a95d73a7d96/wp_Introduction_to_Web_Services.pdf

[25] Ibid.

[26] Deitel & Deitel, http://www.deitel.com/newsletter/20020523/articles/WebServices_SOAP-WSDL-UDDI.pdf

### 2.3.3 UDDI

För en mer ingående beskrivning av UDDI hänvisas läsaren till rubriken UDDI (Universal Discovery Description and Integration) under Bilaga 3 - Web Service Arkitekturen.

UDDI 1.0 uppkom i september 2000 som ett resultat av samarbetet mellan *Microsoft*, *IBM* och *Ariba*. Sedan den offentliggjordes har den växt till att idag omfatta mer än 280 företag. I juni 2001 publicerades UDDI 2.0 av samma företag och den 19 juli 2002 publicerades UDDI 3.0.[27]

UDDI är tänkt att vara en allmän katalog- och integrationstjänst som lagrar och ger information om tillgängliga Web services. Den tillåter användare att publicera och söka efter Web services på Internet. När en förfrågan sänds från en applikation svarar UDDI-registret med att ge information som omfattar upptäckta tjänster, deras status och tillgängligheten av andra tjänster som stämmer överens med förfrågan. UDDI ger därmed företag en världsomspännande webbaserad distribuerad katalog som de kan publicera sina tjänster i.[28]

Företag kan lagra sin information antingen i en privat UDDI-katalog som endast är tillgänglig för godkända företag eller i allmänna UDDI-kataloger där alla får tillgång till informationen. Strukturen för UDDI kan liknas vid en telefonbok, data som samlas inuti UDDI är indelad i tre kategorier; *Vita sidor* som innehåller allmän information om ett företag, exempelvis dess namn, adress, telefon och annat som kan vara av intresse. *Gula sidor* som innehåller information om verksamheten där data klassificeras för varje företag eller tjänst. *Gröna sidor* innehåller tekniskt information om en tjänst vilket kan vara programmeringsspråk, plattform med mera. Den innehåller också en adress för att kunna anropa en tjänst.[29] För att kunna kommunicera med UDDI används SOAP-meddelanden.[30]

## *2.4 Interna Web services*

Interna Web services ägs av samma företag som använder dem. De kan kopplas till olika delar av ett företags interna informationssystem, som försäljning, produktion, finans med mera. En finansapplikation kan till exempel anropa en Web service applikation som är en Euro till USD omvandlare inom företaget. En annan Web service kan användas för att erhålla särskilda data om kunder från en intern databas.[31]

Ett företag kan använda sig av ett antal olika interna databaser exempelvis *Oracle, SQLServer* och *MySQL*. Dessa kan i vanliga fall inte kommunicera med varandra, men genom att använda sig av Web services kan dessa databaser hitta ett sätt att arbeta med varandra. Ett företag kan skapa en *Enterprise Information Portal* (EIP) baserad på SOAP. Denna portal kan bestå av komponenter som en Web service har behov av och eftersom portalen använder sig av SOAP kan även vanliga programmeringsspråk och standard problem undvikas.[32]

Eftersom Web service teknologin använder sig av standarden SOAP har en sådan applikation också förmågan att erbjuda återanvändning av applikationer i ett företag under förutsättningen

---

[27] Ibid.

[28] Cerami, 157-159.

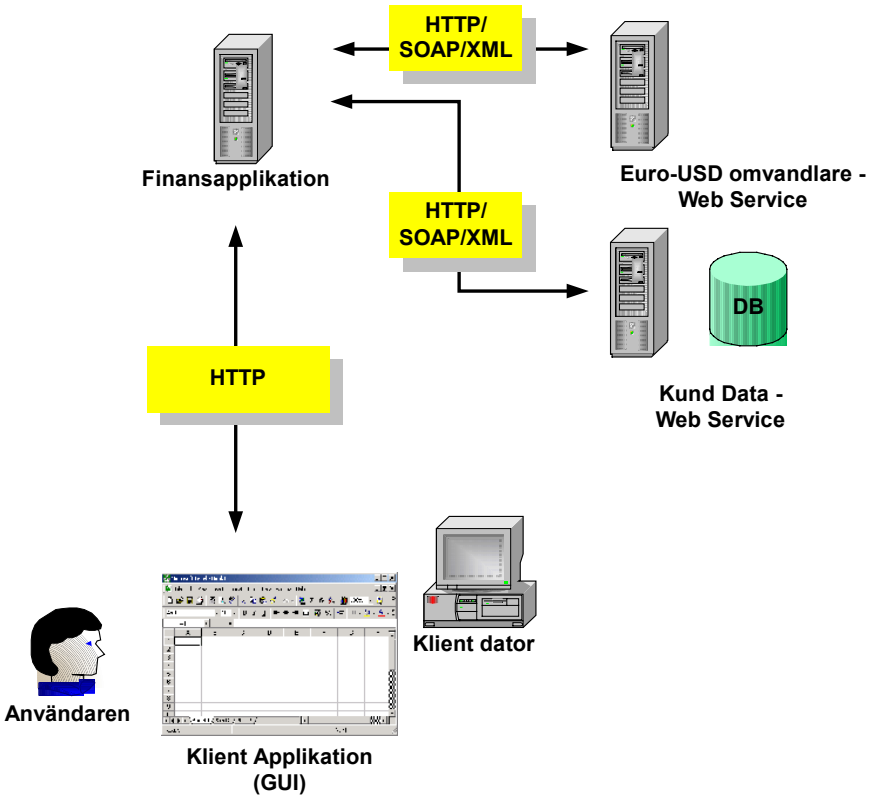
[29] UDDI Organisation (2000). *UDDI Technical White Paper.*[pdf]. Tillgänglig: http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf [2002-10-15]

[30] Cerami, 161.

[31] Johann Dumser (2001). *Internal or External Web Services? - Which One Will Dominate?* [online]. Tillgänglig: http://www.webservicesarchitect.com/content/articles/dumser03.asp [2002-12-01]

[32] Ibid.

att modularitet är möjlig. Detta bidrar till att istället för att skapa en helt ny version av en applikation från början igen i ett annat programmeringsspråk eller plattform, räcker det med att en utvecklare möjliggör applikationen för SOAP. Denna applikation kan sedan integreras till andra applikationer och på så sätt minska antalet steg i en utveckling och samtidigt underlätta applikationshanteringen genom att centralisera dess underhåll.
Om skapandet av en Web service är enkel kommer utvecklare att använda sig av teknologin. Web service teknologin förbättrar möjligheterna för applikationsarkitekturen genom att erbjuda återanvändning och koddelning.[33] Figur 3 visar ett exempel på en intern Web service lösning.




*Figur 3 Användaren vill få reda på företagets totala inkomst under det gångna året i Euro. För att kunna få denna information är det nödvändigt att få reda på köpsumman från alla kunderna och sedan göra om värdet till Euro. Det enda användaren gör är att ställa en förfrågan till finansapplikationen som i sin tur hämtar information via SOAP från de två olika tjänsterna.*

Interna Web services används främst av stora företag som redan har en etablerad webbinfrastruktur. Små företag använder sig oftast av ett enda system och det kan därför ta längre tid för dem att hitta en anledning till att börja använda Web services. Det är också dyrt för små företag att nyttja Web service teknologin. Många tror dock att efterhand som Web service teknologin blir mer allmän och sjunker i pris kommer även små företag att välja Web service alternativet.[34]

[33] Ibid.

[34] Ibid.

## 2.5 Externa Web services

Externa Web services gör det möjligt för olika företag att byta tjänster över Internet. Ett företag som letar efter en specifik tjänst kan gå till ett Web service register, som UDDI och leta efter en särskild tjänst. Företagen når sedan ett överenskommelse angående tjänsten innan tjänsten kan användas.[35]

Web services är skapade och publicerade av ett företag och används av ett annat. En anledning till att utveckla Web services är att målgruppen inte har några begränsningar, leverantörer, klienter och partners kan alla använda sig av samma applikation. Web services kan också anpassas för olika enheter eftersom webbapplikationen, mobiltelefonen och handdatorn kan idag ta emot Web services.[36] Eftersom Web services förväntas bli en revolution.[37] är det möjligt för människor att bilda nya företag som antingen skapar eller hyr ut Web services.[38]

För att en Web service ska vara användningsbar och konkurrenskraftig måste den täcka ett av dessa krav:

- Tillhandahålla ett innehåll – Den måste erbjuda en meningsfull tjänst.
- Tillhandahålla en specifik och tillförlitlig tjänst – Den måste erbjuda en tjänst inom ett område och den måste vara riktig. Exempelvis om den ska tala om när en aktie förändras och dess nya värde.

Även för företag som inte utvecklar egna Web services ger användningen av tjänsterna som andra företag erbjuder många fördelar. Detta genom att information som är specialiserad för ett område kan erhållas.[39]

De funktioner som en Web service tillhandahåller kan vara specifika arbetsuppgifter som genomförs av en applikation som lämnar ifrån sig ett resultat till en annan applikation i ett annat system. Detta innebär att ett system kan skicka en begäran till en applikation på ett annat system över Internet och få ett svar.[40] Tabellen ger ett exempel på en extern Web service för att enkelt beskriva vad det är och hur den kan fungera.

*Tabell 2:1 En beskrivning över hur en extern Web service kan se ut.*

| |
|---|
| *En utvecklare ska skapa en chatklient och skulle vilja att den kunde översätta inmatad text till ett annat språk. Han vet att det finns en sådan tjänst tillgänglig på Internet och istället för att skapa en sådan söker han efter den via UDDI. Han hittar tjänsten och via WSDL som beskriver denna Web service, får han reda på de olika metoderna med dess parametrar, det vill säga hur anropet skall se ut. Han kan sedan via SOAP anropa Web servicen och börja nyttja denna i sin nya applikation. Resultatet av detta blir att en användare sedan kan skriva in en mening på engelska och få den översatt till spanska.* |

[35] Ibid.

[36] Ibid.

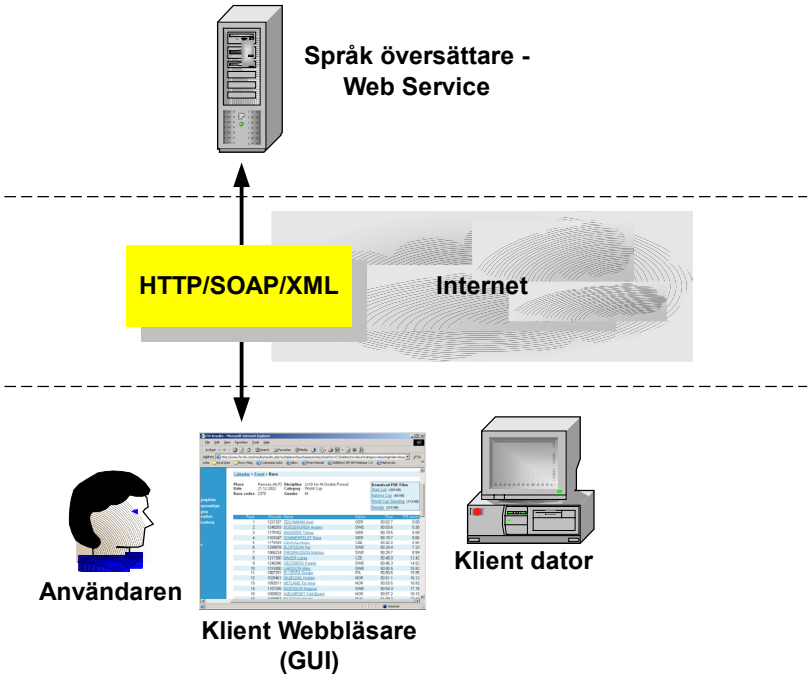
[37] Olofsson, 31.

[38] Dumser, http://www.webservicesarchitect.com/content/articles/dumser03.asp

[39] Ibid.

[40] Deitel & Deitel. http://www.deitel.com/newsletter/20020523/articles/WebServices_SOAP-WSDL-UDDI.pdf

Tjänsten gör det möjligt att mata in en mening på engelska och få den översatt till ett antal språk. Själva Web servicen är transparent för användaren. Det som användaren inte ser är att meningen som han/hon matar in, skickas vidare till en Web service, vars uppgift är att ta emot en mening, översätta den till valt språk och sedan skicka den översatta meningen tillbaka. Allt detta går så snabbt att användaren aldrig hinner märka att meningen gått iväg till en annan plats. Figur 4 visar en bild över hur den externa tjänsten kan se ut.




*Figur 4 Användaren skriver in en mening i sin klient. Meningen skickas vidare via SOAP över Internet till en Web service. Meningen översätts och skickas tillbaka till användaren. Användaren får upp meningen översatt i sin klient.*

En Web service kan använda sig av olika Internet protokoll för att överföra data. Det vanligaste protokollet är dock *Hypertext Transform Protocol* (HTTP).[41] Anledningen till att detta protokoll används är att den är enkel, stabil och allmänt accepterad. De flesta brandväggarna tillåter HTTP trafik, vilket gör att SOAP-meddelanden lätt kan ta sig förbi. En nackdel är dock säkerheten. Web services har inte kommit fram till någon säkerhetsstandard och det finns idag tre viktiga säkerhetsåtgärder som Web services måste vidta; konfidentiell kommunikation, autentisering och nätverkssäkerhet.[42]

### 2.5.1 Konfidentiell kommunikation

När en klient skickar en begäran via ett SOAP-meddelande till en Web service vill klienten vara säker på att kommunikationen är konfidentiell. På vilket sätt är det möjligt för klienten att vara säker på att ingen avlyssnar kommunikationen? Eftersom SOAP-meddelanden använder sig av XML kan de gå över HTTP. XML gör det möjligt att kryptera kommunikationen genom att använda sig av *Secure Socket Layer* (SSL). SSL är en accepterad och tillförlitlig krypteringsteknik som används av många. Emellertid är Web service teknologin tänkt att möjliggöra att en enda

[41] Ibid.

[42] Cerami, 19-22.

Web service består av en kedja av applikationer, med detta menas att en stor Web service kan sammanbinda tjänster från tre andra applikationer. I ett sådant fall är SSL inte lämplig eftersom meddelandena måste krypteras vid varje nod på sin väg och varje nod representerar en möjlig svag länk i kedjan. För tillfället finns det inte någon gemensam lösning på problemet.[43]

### 2.5.2 Autentisering

När en klient kopplar sig till en Web service är det viktigt att kunna identifiera användaren. Det är också viktig att veta om klienten är auktoriserad att använda tjänsten. En lösning på detta problem är att lägga till HTTP autentisering som stödjer grundläggande och förkortad autentisering. Detta innebär att tjänsterna kan skyddas på samma sätt som HTML-dokument skyddas. Många säkerhetsexperter anser dock att denna lösning inte är godtagbar eftersom de anser HTTP autentisering är tillräcklig bra. Det finns ingen gemensam lösning på detta problem heller men det finns många ramverk under utveckling. Några av dessa lösningar är *SOAP Security Extensions:Digital Signature* (SOAP-DSIG), *Security Assertion Markup Language* (SAML) och *XML Key Management Services* (XKMS).[44]

SOAP-DSIG erbjuder kryptering med hjälp av publika nycklar som bidrar till att ett SOAP-meddelande signeras digitalt. Med hjälp av denna signatur kan klienten eller servern validera identiteten hos den andra parten. Denna förslag har lämnats åt W3C för standard. SAML är designad för att underlätta utbytet av autentisering och auktoriseringsinformation mellan företagspartners. XKMS definierar en serie av tjänster för distribuering och hantering av publika nycklar och certifikat. Protokollet är byggd på SOAP och WSDL.[45]

### 2.5.3 Nätverkssäkerhet

SOAP-meddelanden går över HTTP och port 80. Det leder till att de lätt kan ta sig igenom brandväggar. Det är dock här problemet ligger. Företagen öppnar sina brandväggar för att låta utomstående system prata med varandra. Detta bidrar till att virus, maskar och annan farlig kod lätt kan ta sig in i systemen. Det är idag svårt för brandväggar att filtrera meddelandena.[46]

Några tror att de kan lösa problemet genom att använda sig av något annat transportprotokoll istället för HTTP samtidigt som försäljare av brandväggar håller på att utveckla verktyg som är designade för att filtrera Web service trafik.[47]

## *2.6 Definition av framgång*

Begreppet framgång består av ett antal olika definitioner. Det är viktigt att nå en klar bild av vad som menas med ordet framgång för att i nästa steg kunna fastställa en teknologis framtidsutsikt. Enligt *Bonniers Synonym Lexikon* är framgång definierad som välgång, gott resultat och goda framsteg. Enligt *Markus Uvell*, vd på omvärldsanalysföretaget *United Minds* är ett tecken på en framgångsrik teknologi, en teknologi som vänder sig till rätt målgrupp och anpassar sig efter den. Användarfokus är väldigt viktigt för en teknologi, användarna av teknologin måste känna att den är skapad för deras skull och behov. Om detta uppnås är

[43] Ibid., 21.

[44] Ibid., 22.

[45] Ibid.

[46] Wallström, 25-26.

[47] Cerami, 22.

framgången ett faktum. Endast genom att teknologin vänder sig till slutanvändaren och erbjuder den en enklare vardag, kan dess framgång försäkras.[48]

## *2.7 Framgångfaktorer*

Efter att ha fått en klarare bild av framgång, övergår nyfikenheten till hur vägen dit kan nås. Det är viktigt att fastslå vilka faktorer som påverkar chansen till framgång. Inom många andra teknikområden finns det flera författare som delgivit sina råd i artiklar, ofta kallade ”*Critical Success Factors*”. Däremot har jag inte hittat någon sådan artikel för att fastställa en teknologis framtid. Eftersom jag inte hittade några framtagna framgångsfaktorer har en undersökning av litteraturen gjorts för att hitta och sammanställa ett antal framgångsfaktorer. Dessa har tagits fram genom att samla ihop olika åsikter från olika artiklar om vilka faktorer som anses vara viktiga för att en ny teknologi, vilken som helst ska kunna slå igenom.

### 2.7.1 Framgångsfaktorer

Antalet framgångsfaktorer begränsades till sex. Dessa är standard, säkerhet, affärsnytta, enkelhet, tillförlitlighet och återanvändning. Nedan ges en utförligare presentation av nämnda framgångsfaktorer.

#### 2.7.1.1 Standard

För att en teknologi ska få framgång är det viktigt att den blir en standard, detta eftersom många användare inte vågar använda en teknologi innan den når standardstatus. En standard är enklare och säkrare för en användare att använda. Det innebär en kvalitetsgaranti för användaren och viktigast av allt så erbjuder den kompatibilitet med andra som använder sig av samma standard.[49] När det gäller en teknologi är det viktigt att ena alla leverantörerna kring en enda standard, annars är risken att en konflikt uppkommer mellan leverantörernas vilja att skapa en standard och deras intresse att erbjuda egna lösningar för att på så sätt låsa in kunden i sitt egna arkitektur. Om varje leverantör erbjuder sina egna lösningar blir interoperabiliteten lidande. Detta är inte bra om teknologin är tänkt att vara plattforms- och försäljaroberoende.[50]

#### 2.7.1.2 Säkerhet

I dagens samhälle där information färdas från en sida av jordklotet till den andra är säkerhetsfrågorna av stort intresse. En standard som fungerar men inte är helt säker riskerar att förbises. Problem som trafikavlyssning och hur fel som uppstår ska åtgärdas är några av de viktiga frågorna.[51] Säkerhetsbrister kan ge finansiella förluster och förlust av känslig information till följd. Dessa hål är viktiga att täppa igen innan användarna finner att en standard ens är värd att användas. En teknologi måste lösa sina säkerhetsproblem för att kunna få framgång. Många teknologier har inte slagit igenom på grund av bristande säkerhet.[52]

---

[48] Markus Uvell (2002). *Från affärsmän till tonårstjejer.* [online]. Tillgänglig: http://computersweden.idg.se/ArticlePages/200207/18/20020718164058_CS963/20020718164058_CS963.dbp.asp [2002-12-04]

[49] Techstreet (2002). *The What, Why, and How of Standards.* [online]. Tillgänglig: http://www.cssinfo.com/whystandards.html [2002-12-04]

[50] Wallström, 24.

[51] Ibid.s

[52] Kent Olofsson (2002), *5 heta tekniktrender* [tidning]. Tillgänglig: http://computersweden.idg.se/ArticlePages/200209/10/20020910170703_CS303/20020910170703_CS303.dbp.asp [2002-11-29]

#### 2.7.1.3 Affärsnytta

Om en teknologi ska bli en framtidsteknologi är det viktig att få företagen att inse dess affärsnytta. Det måste finnas en riktig anledning till att satsa på en teknologi. Företagen måste känna behovet av teknologin. Det är mycket lättare för ett företag att våga pröva en ny teknologi om företaget ser nyttan i den. Den nya teknologin måste vara ett bättre alternativ till något som de redan har.[53] Affärsnyttan är oftast nära kopplad med kostnaden. Om företagen inte ser affärsnyttan i en teknologi, ser de inte heller anledningen till att stå för kostnaden som teknologin medför.

#### 2.7.1.4 Enkelhet

En framtidsteknologi måste vara enkel. Teknologin måste vara lättanvänd och lättillgänglig.[54] Komplexa teknologier tenderar att kräva mycket kunskap och tid att lära sig. Detta har användaren oftast varken tid eller lust att lägga ner. Användare vill ha enkla lösningar som lätt kan anpassas. Det är också viktigt att på ett enkelt sätt kunna underhålla sina system och applikationer. Alltför komplexa teknologier brukar inte kunna bli framgångsrika.

#### 2.7.1.5 Tillförlitlighet

Ett annat ord för tillförlitlig är pålitlig. Alla teknologier måste erbjuda detta för att kunna nå framgång. Användaren måste kunna räkna med att deras system och applikationer kommer att starta och att de kommer att klara påfrestningarna som de utsätts för. Detta är krav som en teknologi måste uppfylla. När användare har tillförlitlighet innebär det att de också gör sig beroende av den, därför måste teknologin möta användarnas förväntningar.

#### 2.7.1.6 Återanvändning

Det är viktigt att kunna återanvända den tid och möda som lagts ner i utvecklingen av kod och systemutveckling. Det handlar om att kunna återanvända de investeringar som företag redan lagt ner. På detta sätt känner inte företagen att det lagt ner pengar och tid i onödan. De behöver inte heller börja om från noll.[55]

---

[53] Karin Lidström (2002). *Platt skärm för dyr.* [online]. Tillgänglig: http://computersweden.idg.se/ArticlePages/200211/28/20021128123532_CS508/20021128123532_CS508.dbp.asp

[54] Mikael Ricknäs (2001). *Genombrott för het SIP-protokoll.* [online]. Tillgänglig: http://nyheter.idg.se/display.asp?ID=011123-CS10

[55] Kent Olofsson, ”Snabbare utveckling”, *Computer Sweden*, 16 oktober 2002, s. 30

# 3 Metod

En metod är ett sätt att komma fram till ett resultat. Det finns en rad olika metoder man kan använda sig av för att nå detta mål. Exakt hur man gått tillväga för att nå sitt mål är en viktig del i arbetet. Därför är det också viktigt att ange vilken metod man använt sig av för att nå sitt mål. Tanken med detta är att en annan person ska kunna följa samma tillvägagångssätt och komma fram till exakt samma resultat.[56]

Genom att precisera det problem som en person har för avsikt att undersöka avgörs själva upplägget för undersökningen. Sedan kan man välja mellan olika ansatser för att genomföra sin undersökning. Forskningsansatsen utgörs av ett vetenskapligt angreppssätt, där man kan välja mellan det induktiva eller det deduktiva angreppssättet. Forskningsansatsen utgörs även av den kvalitativa eller den kvantitativa metoden.[57] Nedan följer en beskrivning av de olika forskningsansatserna och ställning tas till de val som är gjorda.

## *3.1 Kvalitativa och kvantitativa metoder*

Forskningen skiljer mellan två olika metodiska angreppssätt när den sker inom samhällsvetenskapen, nämligen den *kvalitativa* och den *kvantitativa*.

Syftet med kvalitativa undersökningar är att försöka förstå och analysera helheter, det vill säga att skaffa ett djupare kunskap inom ett specifikt område. Metoden ger ett förstående syfte av problemet som studeras och det behövs inte ett stort antal respondenter för att ge ett helhetsintryck av en situation. Denna metod kännetecknas av en närhet till forskningsobjektet. Styrkan hos den kvalitativa metoden är att de visar på totalsituationen. Denna helhetsbild gör det möjligt att få en ökad förståelse för sociala processer och sammanhang.[58]

Den kvantitativa metoden används när målet är att jämföra och testa om det förvärvade resultatet är giltigt för samtliga undersökningsenheter. Till skillnad från den kvalitativa metoden, som kännetecknas av en närhet till forskningsobjektet, söker den kvantitativa metoden efter en distans till undersökningsobjektet och undviker att vara en del av det studerade. Ett jag-det-förhållande är önskvärt i undersökningen. Statistiska mätmetoder har stor betydelse i analysen och man använder sig av ett system för att samla in informationen vilket underlättar generaliseringen samt ger en högre tillförlitlighet.[59]

En av anledningarna till att jag valde en kvalitativ metod istället för en kvantitativ berodde på att jag valde att genomföra intervjuer med några personer ur olika företag. Jag valde att ställa samma frågor till alla respondenterna men karaktären av dessa frågor är öppna frågor där respondenten har möjlighet att utveckla sina svar. Anledningen att detta valdes var att avsikten med intervjun var att ta reda på vad respondenterna själva tyckte om Web service teknologin. Om jag hade valt den kvantitativa metoden skulle respondenterna inte fått utrymme till egna tankar och deras svar hade begränsats till vissa svarsalternativ.

En annan anledning till att jag valde den kvalitativa metoden var att mitt mål med uppsatsen var att studera helheten för att kunna skaffa mig en djupare kunskap inom ett specifikt område, i detta fall Web service teknologin. Dessutom ansåg jag att denna metod var lämpligare

---

[56] Idar Holme, Bernt Krohn Solvang, *Forskningsmetodik: Om kvalitativa och kvantitativa metoder* (Lund: Studentlitteratur, 1997), 11-13.

[57] Ibid., 13 –14.

[58] Ibid., 92-94.

[59] Ibid., 79-83.

eftersom den begränsade tidsresursen och intresset för deltagandet i en intervju bidrog till att jag endast kunde samla ihop ett litet antal respondenter och utifrån deras svar ändå få en helhet.

## 3.2 Deduktiv och induktiv

De mest omtalade angreppssätt är; den *deduktiva* och den *induktiva* metoden, eller bevisandets respektive upptäcktens väg.

Det mest använda sättet att utveckla teorier är det som kallas den hypotetisk-deduktiva teoribildningen. Det innebär att ur ett antal påståenden härleds nya hypoteser. Dessa hypoteser kan därefter prövas med empiriska undersökningar, som till exempel intervjuer. Genom dessa empiriska undersökningar kan sedan tilliten till teorin stärkas eller försvagas.

I det induktiva angreppssättet är inte teoriutveckling utgångspunkten för forskningen utan är en process som förutsätter den och utvecklas parallellt med att empirisk information systematiskt insamlas. De empiriska resultaten utgör grunden för en teoretisk uppfattning som är nära kopplad med den företeelse som studeras. Detta är utgångspunkten för att utveckla en teori som till sin karaktär är generell.[60]

Jag har valt att utgå ifrån den deduktiva ansatsen genom att behandla den teori som finns inom området. Denna teori kommer att prövas empiriskt genom intervjuer. Anledningen till att valet föll på denna istället för den induktiva ansatsen var att jag valde att först skaffa mig kunskap om Web service teknologin för att sedan kunna dra en slutsats. Den induktiva ansatsen gör tvärtom, vilket innebär att det inte finns några förkunskaper om ämnet innan man går in i den. Därför anser jag att en deduktiv ansats passade bättre i mitt arbete. När jag hade skaffat mig tillräcklig kunskap inom ämnet formulerades intervjufrågor som motsvarade uppsatsens syfte. Detta ledde till att det var lättare att analysera och sammanställa svaren från respondenter som i sin tur kunde relateras till teorin.

## 3.3 Datainsamling

I detta arbete använde jag mig av både *primärdata* och *sekundärdata*. Primärdata innebär att undersökaren samlar in data just för den specifika undersökningen. Dessa insamlingsmetoder kan vara intervjuer, enkäter eller observationer.[61] Jag valde att som primärdata använda mig av intervjuer med personer som utvecklar Web services.

Sekundärdata är information som redan insamlats av andra personer och finns som producerade dokument som bearbetats. Denna information är producerat för andra ändamål. Denna information kan vara tryckta böcker, tidskrifter, tidningar med mera.[62]

### 3.3.1 Insamling av sekundärdata

Eftersom jag saknade kunskap om vad Web service teknologin var och hur den fungerade började jag med att göra en omfattande litteratursökning och studie av sekundärdata. Jag ville först få en klar bild över teknologin innan jag gick vidare. Eftersom jag inte lyckades hitta några böcker om Web services på någon av biblioteken vände jag mig till Internet. När jag sökte efter böcker på Internet om Web services hittade jag en hel del som handlade om hur man skapar en Web service, andra handlade om Web services som var skapade i ett specifikt programmeringsspråk med mera. Dock var det svårt att hitta böcker som bara beskrev vad

---

[60] Ibid., 51.

[61] Bengt-Arne Bengtsson & Hans Bengtsson. *Zigma; Foskningsboken; Om konsten att arbeta på ett undersökande och kunskapande sätt.* (Almqvist & Wiksell Förlag AB: Uppsala, 1995), 39.

[62] Ibid.

Web service teknologin var. Jag fortsatte min sökning på Internet efter artiklar om Web service teknologin, där jag hittade en hel del. Jag använde mig främst av sökmotorn Google vid sökningen av artiklar på Internet, men också av Alta Vista och Yahoo. Som nästa steg började jag undersöka vad utvecklare och företag tyckte om Web service teknologi.

### 3.3.2 Bearbetning av sekundärdata

Jag inhämtade kunskap om ämnet genom de böckerna som jag hittade på Adlibris bokhandeln på Internet samt den information som jag hade skrivit ut från Internet. Två böcker hittades som passade för ämnet och böcker som inte passade för ämnet valdes bort eftersom deras innehåll inte gav svar på uppsatsens frågeställningar. Av allt material som jag hittade på Internet gjordes sedan ett urval för att använda lämpliga artiklar för att basera uppsatsen på.. Ur de valde böckerna valdes passande avsnitt och artiklarna från Internet sorterades. Jag läste även alla tre standard specifikationerna för att få en mer teknisk förståelse för teknologin. För att få åsikter om Web service teknologin från utvecklare och företag gjordes först en insamling av tidningsartiklar som behandlade ämnet, där jag använde mig av en av Sveriges ledande IT-tidningar, Computer Sweden som är välkänd och teknisk. Jag använde mig också av passande artiklar som hittades på Internet som tog upp detta ämne.

När jag kände att jag hade tillräcklig kunskap om Web service teknologin växte uppsatsens frågeställning fram och blev klarare. Uppsatsens teoridel är resultatet av den sammanställda informationen från det material som jag har använt mig av.

### 3.3.3 Kritik av sekundärdata

Informationen om Web service teknologin har varit begränsad. Detta eftersom det mesta av informationen som hittades handlade om hur man skapar en Web service, där det redan förväntades att man visste vad teknologin var och hur den fungerar. Detta kan innebära att det finns en risk att jag har gått miste om relevant kunskap för uppsatsen. Vidare är teknologin ganska ny och det finns inga fasta standarder, vilket gör att informationen varierat från en källa till en annan.

Ett problem har varit att avgöra om informationen som hittats på Internet är trovärdig och riktig. För att kunna minska detta problem valde jag att endast hålla mig till information som var skriven av kända organisationer som W3C som är den standardorganisation som ligger bakom utvecklingen av Web service teknologin. När jag hittade information på mindre trovärdiga hemsidor undersökte jag deras referenser för att hitta mer trovärdigare hemsidor. Jag har även använt mig av information från de företag som ligger bakom Web service teknologin för att få mer trovärdig information, dessa företag är Microsoft, IBM och Sun. Jag använde mig också av publicerade artiklar där utgivaren har ansetts vara trovärdig i form av en tidningsartikel eller white pappers. Det är dock alltid ett problem med sekundärdata när det gäller trovärdigheten eftersom det är någon annan som samlat in informationen och tolkningen som gjorts kanske inte är helt korrekt. De företag och organisationer som innehåller information om Web service teknologin är partiska och tenderar därför att bara se fördelarna med teknologin. För att försöka balansera detta har jag valt många olika informationskällor både böcker, tidningar och Internet artiklar, detta för att undersöka att informationen överensstämmer.

### 3.3.4 Val för insamling av primärdata

En intervju kan genomföras på två olika sätt, den kan vara standardiserad och strukturerad eller helt ostrukturerad. Vid helt standardiserade intervjuer ställer man exakt samma frågor till alla personer som intervjuas. Man kan också begränsa utrymmet för de intervjuade till ett antal fasta svarsalternativ. Dessa alternativ kan vara ja, nej eller vet ej svar. Motsatsen till detta är ostrukturerade frågor där frågorna anpassas till varje person och svarsutrymmet är helt fritt.[63]

För att få en bättre inblick i hur Web service teknologin uppfattas och upplevs av utvecklare valde jag att göra en intervju till ett antal personer som jobbar på olika företag. Genom att välja ut olika företag kunde jag försäkra mig om att svaren var helt oberoende av varandra och erbjöd därför stor bredd. Med hjälp av intervjuerna ville jag få fram hur Web services teknologin lyckas leva upp till sina förväntningar och vilka tankar utvecklarna själva hade om denna teknologi. Eftersom dessa personer jobbar på olika företag i Stockholm kunde jag inte genomföra en personlig intervju, ansikte-mot-ansikte, utan jag använde mig istället av e-post intervjuer. Anledningen till att valet föll på denna alternativ framför en telefonintervju var att personerna på dessa företag är väldigt upptagna och det passade de bättre att få tillgång till frågorna via e-post.

Vid utformandet av frågeformuläret valde jag en blandning mellan den strukturerade och den ostrukturerade metoden. Frågeformuläret finns tillgänglig för läsaren i Bilaga 2 - Intervjuformulär. Den standardiserade metoden användes på så sätt att samma frågor ställdes till alla personerna som deltog i intervjun. Jag begränsade även antalet frågor till sexton eftersom jag tyckte att det var viktigt att veta vad respondenterna hade för åsikter om just dessa frågor. En annan anledning var av praktiska skäl eftersom det är lättare att få personer att vilja delta i en intervju om det inte är alltför många frågor som måste besvaras.

Jag använde mig dock av den ostrukturerade metoden också genom att respondenterna kunde svara på frågorna precis hur dem ville. De hade inga ramar för hur deras svar skulle se ut, vilket ledde till att svaren varierade från respondent till respondent. En nackdel med detta var förstås att det inte gick att ge en generaliserad resultat, vilket hade varit möjligt om man endast valt den standardiserade metoden. Frågorna som ställdes har sitt ursprung i litteraturstudien och var avsedda att besvara frågeställningen. Frågorna diskuterades med handledaren och några vänner för att vara säker på att de var fria ifrån missförstånd, välformulerade och meningsfulla.

### 3.3.5 Urval av intervjupersoner

Vid urvalet av de personer som skulle intervjuas avgränsade jag antalet personer, vilket skulle kunna kallas för en representativ urval av populationen, där populationen står för företag som utvecklar Web services. Vid början av arbetet samlade jag ihop ett antal tidningsartiklar som handlade om Web services, artiklarna nämnde ett antal företag som utvecklar Web service lösningar. Jag började med att göra en lista med dessa och letade sedan upp dem på Internet, för att få en inblick i deras kunskap om Web services samt kontaktinformation. Jag gick igenom företagen och valde ut tolv stycken som jag tyckte var passande för arbetet.

Nästa stag var att ta kontakt med dem. Jag började med att ringa till några av dem, men eftersom jag inte lyckades komma fram, valde jag att skicka ut en e-post istället till alla där jag frågade efter personer med kunskap inom detta område. När de flesta svaren kommit tillbaka valde jag att ringa resterande företag som inte svarat för att skynda på processen. De personer

[63] Ibid., 53.

som rekommenderades hade olika befattningar, dessa var konsulter, systemutvecklare, teknisk ansvariga eller systemarkitekter.

Jag fortsatte med att ringa till några av de kontaktpersoner på de olika företagen men det visade sig vara svårt att få tag på dessa personer. Den enda personen jag lyckades få tag på gick med på en intervju under förutsättningen att han fick tillgång till frågorna via e-post. Detta bidrog till att jag skickade ut en e-post med en förfrågan till alla de resterande personer istället för att ringa dem för att vara säker på att få tag på dem. Det visade sig alla var mycket upptagna, men att de gärna ställde upp på intervjun om de fick tillgång till frågorna via e-post och om de fick några dagar på sig att besvara dem. En ny e-post skickades till alla respondenterna med en bifogad dokument som innehöll frågorna och information om hur de skulle kunna kontakta mig vid frågor. Även ett slutdatum för inlämning av svaren angavs vilket var två veckor.

### 3.3.6 Kritik av primärdata

Vid en sammanställning av en intervju-undersökning måste man vara medveten om att det inte finns några exakta och rätta resultat. Det finns ett antal felkällor som kan uppkomma. Nedan tas ett antal sådana som påträffats under arbetets gång upp.

#### 3.3.6.1 Metodval

Jag valde att genomföra intervjun som en e-post intervju. En anledning till detta val var fördelen med att ett företags geografiska placering inte var någon begränsning, viket bidrog till att jag kunde använda mig av respondenter i Stockholm. Detta var också orsaken till att det hade varit svårare att genomföra en ansikte-mot-ansikte intervju. En annan fördel har varit att metoden har skapat en bra möjlighet för granskning av svaren där respondenterna har kunnat kontaktas igen vid oklarheter. En annan anledning till detta val var att respondenten hade mycket att göra och genom att skicka frågorna till dem kunde de få lugn och ro att svara på frågorna. På detta sätt hade respondenterna möjlighet att själva välja när de ville svara på frågorna. Detta blev dock också en nackdel eftersom respondenterna inte kunde ställa direkta frågor vid oklarheter och några respondenter glömde bort frågorna, vilket ledde till att de måste påminnas några gånger och slutade i några fall som bortfall. Bortfallet av svar var högre än väntat vilket kan bero på metodvalet. Detta har påverkat resultatet som kunde ha sett annorlunda ut om bortfallet inte varit så stor.

#### 3.3.6.2 Bortfall av intervjusvaren

Efter att intervjun skickades ut till företagen sattes en tid på två veckor innan en första påminnelse skickades ut. Av de tolv tillfrågade företagen blev det endast fem företag som verkligen svarade. Av dessa fem företag var det endast två som behövde en påminnelse om intervjun. Två av de andra företagen hörde av sig ganska direkt efter att de fått frågorna och berättade att de inte hade tillräcklig kunskap om ämnet, därför kunde de inte besvara frågorna. Resterande fem företag fick tre påminnelser om intervjun med en tidspann på en vecka mellan varje innan de hörde av sig. Tre av dessa hörde aldrig av sig alls trots mina försökt att nå dem både via telefon och e-post. Ett av dessa företag hörde av sig via telefon vid tre olika tillfällen och lovade att skicka svaret på intervjun men gjorde det aldrig. Ett annat företag hörde av sig och berättade att de inte hade tid att svara på frågorna helt enkelt.

#### 3.3.6.3 Mätfel

När ett resultat ska tas fram efter en undersökning är det viktigt att ta reda på om det finns några fel i den information som insamlats. Genom att ständigt pröva och noggrant bearbeta sitt material kan man uppnå en tillfredsställande grad av trovärdighet vid sitt resultat. Det finns två

begrepp som man kan använda sig av för att beskriva värdet av en utredning; *Reabilitet* och *Validitet*. [64]

Reabilitet uppnås genom att mätningarna genomförs på ett korrekt sätt samt att bearbetningen av informationen görs noggrant. Hög reabilitet uppnås genom att olika och oberoende mätningar av ett och samma undersökning visar samma eller liknande resultat.[65]

För att uppnå en hög reabilitet i detta arbete genomfördes ett antal intervjuer med några personer som jobbade på olika företag som var insatta i ämnet. Ett problem som upptäcktes var dock att alla företag inte hade den nödvändiga kunskapen för att kunna svara på frågorna vilket ledde till att de inte ville vara med i intervjun. Bland de fem företagen som svarat på frågorna finns det ett som inte svarar på alla frågorna eftersom respondenten inte vet svaret på frågan. Något som också upptäcktes var att svaren skiljde sig en hel del mellan respondenterna. Detta kan bero på att frågorna uppfattats på olika sätt av varje respondent. Metoddelen i denna uppsats har också beskrivit och motiverat tillvägagångssättet för denna uppsats för att vidare öka reliabiliteten i uppsatsen. Det har även använts fotnoter genomgående i uppsatsen vilket gör att läsaren själv kan söka upp den underliggande källan för att skapa sig en egen uppfattning om källans trovärdighet.

Validitet är beroende av vad som mäts, för att nå en hög validitet är det viktigt att hålla sig till ämnet och inte mäta oviktiga saker runtomkring.[66] Validitet försökte jag uppnå i detta arbete genom att utforma relevanta intervjufrågor som höll sig till ämnet. För att vara någorlunda säker på att bra frågor ställdes utgick jag ifrån frågor som tagits upp i artiklar som inhämtats från Internet och tidningsartiklar från Computer Sweden. Jag lät även handledaren samt några vänner att granska frågorna i frågeformuläret innan det skickades till respondenterna för att avgöra om frågorna var tillräckligt tydliga i sin utformning. För att ytterligare öka undersökningens validitet fick respondenterna ta del av frågorna i två veckor innan de behövde skicka sina svar. Frågorna besvarades av kunniga personer inom ämnet och deras svar har varit innehållsrika, däremot är resultatet som undersökningen genererar inte generellt. Detta eftersom endast ett fåtal respondenter deltog i undersökningen.

#### 3.3.6.4 Bearbetningsfel

Efter att intervjuerna samlats in genomfördes en analys och bearbetning av dem. Här upptäcktes att respondenterna svarat olika på svaren. En av respondenterna hade väldigt kortfattade svar, vilket gjorde det svårt att utläsa ett tydligt resultat. En annan av respondenterna svarade inte på alls på vissa frågor eftersom kunskapen om just den frågan saknades. Eftersom bearbetningen av svaren alltid kräver en tolkning från den frågande parten, så är jag medveten om att de tolkningarna som jag har gjort utifrån svaren påverkat slutresultatet. Dock har jag försökt att så objektivt som möjligt göra en redogörelse för svaren och slutresultatet.

---

[64] Holme & Solvang, 163.

[65] Ibid.

[66] Ibid.

# 4 Resultat

Syftet med detta arbete är att fastställa huruvida Web services är ett teknologi för framtiden. För att kunna svara på denna fråga tyckte jag att det var nödvändigt att undersöka vad utvecklare av Web services tycker om denna nya teknologi. Nedan följer en kort presentation av varje företag som deltog i intervjun.

- **Marotz AB**

  Webbadress: http://www.marotz.se/sv/index.htm

  Marotz är ett konsultföretag med fokus på kundanpassade systemlösningar för ett stort antal affärsområden. Marotz erbjuder konsulttjänster inom systemutveckling. Marotz AB startade sitt kontor i Kista i september 1998 och i september 2001 blev Marotz uppköpta av Hansen Technologies. Representanten för företaget i intervjun är Henry Aspenryd, System Arkitekt.

- **Borland Nordic**

  Webbadress: http://www.borland.se

  Borland är ledande i sin roll att tillhandahålla teknologi som kan hjälpa till att utveckla, sprida och integrera mjukvara hos företag. Företaget bildades 1983 i Kalifornien, USA och har idag mer än 1,1 miljoner anställda världen över och företag i över 20 länder. Huvudkontoret i Sverige ligger i Solna, Stockholm. Representant för företaget i intervjun är Robert Lecklin, Teknisk ansvarig.

- **Toolkit Software**

  Webbadress: http://www.toolkitsoftware.se/phpweb/index.php

  Toolkit Software tillhandahåller expertkonsulter inom systemutveckling. De säger sig vara experter på att utveckla affärskritiska, transaktionsintensiva och tekniskt avancerade system som ställer höga krav på tillgänglighet och säkerhet. Bland dessa kan nämnas börssystem, e-postsystem, elektroniska arkiv samt bank och finanssystem. Företaget grundades 1998 med målet att vara ett personalägt konsultbolag med expertkunskap om marknaden, dess produkter och teknologier. Representant för företaget i intervjun är Jonas Lindskog, Systemutvecklare.

- **Sun Microsystems AB**

  Webbadress: http://www.sun.se

  Sun Microsystems är en ledande leverantör av kraftfull maskinvara, programvara och tjänster som gör att nätverket fungerar. Sun är skaparen av programmeringsspråket Java och är en av de drivande företagen bakom Web service teknologin. Sun finns i mer än 170 länder och på webben. I Sverige startade Sun Microsystems AB 1988 och har idag ca 450 anställda. Huvudkontoret ligger i Kista i norra Stockholm. De största kunderna finns inom telekommunikation, bank och finans, myndigheter, tillverkande industri, media samt universitet och högskolor.

  Sun Microsystems AB är som Microsoft AB också en av de som står bakom utvecklingen av Web service teknologin och är därför inte bara utvecklare utan också skapare till teknologin Representanten för företaget i intervjun är Per-Olof Litby, ansvarig för affärsutveckling inom Web services.

- **XML Sweden**

  Webbadress: http://www.xml.se

  XML Sweden är ett konsultföretag som har specialiserat sig på dataformatet XML med tillhörande standarder. Eftersom Web services teknologin använder sig av XML är det av intresse för företaget att innefatta den kompetens avseende de standarderna kring Web services för att kunna ge kvalificerade råd och utbildning för företag. Bland företagets kunder finns Stockholms stad, Bankgirot, Telia Mobile och AstraZeneca. Representanten för företaget i intervjun är Gustaf Liljegren, Konsult.

## *4.1 Intervjuresultat*

En sammanställd resultat över svaren på intervjuerna redovisas nedan. De är grupperade för att skapa en överblick.

### 4.1.1 Utvecklarna och Web services

Frågorna under denna rubrik är tänkta att fungera som en introduktion till företaget och Web services.

- **Varför valde Ert företag att börja utveckla Web Services applikationer?**

  Enligt **Marotz** var anledningen till att de började utveckla Web service applikationer att de ansåg att de vill lära sig mer om den nya och intressanta teknologin som uppkommit.
  För **Sun** var det en självklarhet att börja utveckla sådana applikationer eftersom de är ett av de drivande företagen bakom Web services teknologin. **XML Sweden** började utveckla applikationer av intresse för Web service teknologin. De kände att de ville kunna svara på kraven från sina kunder om en eventuell förfrågan skulle uppkomma. I **Toolkit Softwares** fall var det just kundernas efterfrågan som bidrog till att de började utveckla Web service applikationer. Merparten av deras kunder vill ha just Web service lösningar.
  Eftersom **Borland** är ett mjukvaruföretag som levererar produkter till sina kunder var det naturligt för dem att även kunna erbjuda sina kunder Web service lösningar. Kunderna kände behovet av att kunna utbyta information mellan sina system inom företaget, vilket bidrog till att Borland började utveckla Web service applikationer.

- **Vilka uppgifter sköter Web Services applikationerna som Ert företag skapar?**

  Företaget **XML Sweden** hade inget svar på denna fråga. **Marotz** däremot berättade att deras applikationer används främst för läsning och hämtning av data från större databaser. Eftersom **Sun** är ett globalt företag så arbetar de med globala lösningar och de anser att det är viktigt att deras applikationer kan göra våra informationstillgångar tillgängliga på ett enkelt sätt i globala nät. **Toolkit Software** utvecklar applikationer som är väldigt olika, det är allt ifrån börssystem till administrativa system för olika statliga förvaltningar.
  **Borlands** representant har främst utvecklat Web service lösningar för ett företag som tillverkar stål. Detta har medfört att applikationerna som utvecklats är relaterade till de uppgifter som ett sådant företag kan tänkas behöva få hjälp med. Ett exempel är automatisk sökning och hämtning av ståldata från företaget över Internet med information om vikt, dimensioner och potentiella fel av stålen. Ett annat exempel är att möjliggöra att en rörtillverkare kopplas samma till stålföretaget för gemensamma leveranser till kunder som finns inom samma område

- **På vilket sätt har Web Services teknologin underlättat för Er att lösa problem som annars varit svårare att lösa?**

  **XML Sweden** hade inget svar på frågan. Något som är gemensamt med svaret på denna fråga är att de andra företagen är väldigt positiva till Web services. Både **Marotz** och **Sun** anser att Web services kan i stort sett användas till att bygga i princip alla tjänster.
  **Sun** anser vidare att Web services bidragit till stora kostnadsbesparingar, vilket är positivt för underlättandet av lösningar. **Marotz** säger att de har kunnat utveckla lösningar utan att behöva ta hänsyn till vilka utvecklingsverktyg som använts tidigare. De tror vidare att tjänster som levererar data kommer att vara dem som kommer att utvecklas mest av.
  **Toolkit Software** kunde underlätta sitt arbete tack vare att Web service erbjuder en enkel exponering av tjänster, vilket bidrar till att det går snabbare att integrera olika system för att på så sätt kunna skapa nya applikationer mycket snabbare. I **Borlands** fall har den största vinsten varit att det har gått fort att utveckla lösningarna, eftersom Web service teknologin är robust och enkel, vilket bidrar till att det är enkelt att plugga in nya tjänster och system.

- **Vilka nackdelar eller hinder ser ni med Web Services?**

  **XML Sweden** anser att svarstiderna på Internet kräver en annan typ av programmering både för enskilda datorer och datorer på lokala nät. De menar att Web service skulle behöva ett asynkront protokoll för att få möjlighet till svarstiden på flera dagar om det skulle behövas. **Sun** ser emellertid inte några speciella hinder eller nackdelar med Web service förutom att det är en teknologi under utveckling. **Marotz** skulle vilja se förändringar vad gäller säkerhet, sessionshantering, transaktionshantering och händelsehantering. De är dock medvetna om att Web services är en teknologi under utveckling som ännu inte är helt färdigutvecklad.
  **Toolkit Software** ser standardiseringen av teknologin och Suns ovilja att delta i arbetet som ett hinder. De tycker vidare att standardiseringsprocessen går för långsamt framåt och de tycker att det är en nackdel att tre olika standardiseringsorganisationer är inblandade.
  **Borland** ser inga direkta hinder, men påpekar att Web service teknologin är ganska ny vilket gör att man endast ser toppen på isberget. Infrastrukturen för kommunikation finns idag, men det är viktigt att lyckas hålla ihop dessa transaktioner mellan olika system genom att få enhetliga och fungerande specifikationer för informationsutbytet mellan organisationer.

### 4.1.2 Typ av applikationer

Frågorna under denna rubrik syftar till att få en överblick över vilka slags applikationer som Web service används mest till för tillfället och anledningen till det.

- **Vilken typ av applikation utvecklar Ni flest av, för användning internt (program till program kommunikation) inom företag eller för extern användning (tjänster) över Internet?**

  **XML Sweden** hade inget svar på denna fråga. Annars skilde sig svaren lite mellan de andra företagen. De applikationer som **Marotz** utvecklar används mest för internt bruk. Detsamma gäller för **Toolkit Software**. **Borland** utvecklar interna applikationer som används mellan olika affärsenheter. **Sun** däremot utvecklar applikationer som fungerar som portaler.

- **Om det är flest för internt bruk (program till program kommunikation), varför är det så?**

  Eftersom det endast var Marotz, Toolkit Software och Borland som utvecklade applikationer för internt bruk var det också endast de som svarade på denna fråga.
  Marotz menar att eftersom Web services inte alltid har funnits, har det inte heller alltid funnits något stort utbud för att hämta information från någon extern källa. Detta bidrar till att företagen än så länge behöver applikationer som hjälper dem att ta hand om sina egna privata nätverk. Dock ökar mängden Web services hela tiden och detta kommer att förändras i framtiden. Enligt Toolkit Software är anledningen att kunderna saknar kunskapen om vilket mervärde de kan få av att exponera sina system till andra aktörer. Detta bidrar till att externa projekt inte är så vanliga. Borland menar istället att behovet av interna applikationer är störst, eftersom kunder som är intresserade av externa applikationer har redan idag integrerat sina system med *Electronic Data Interchange* (EDI).

- **Är det svårare att utveckla applikationer för extern användning och i så fall varför?**

  **XML Sweden** hade inget svar på denna fråga. Enligt **Marotz** är det betydligt svårare att utveckla en applikation för extern bruk än för intern bruk. De menar att applikationer som levererar data externt har högre krav på säkerhet. De måste också vara helt oberoende av utvecklingsspråk och operativsystem. **Toolkit Software** hade samma åsikt i frågan.
  Även **Sun** höll med, de ansåg också att bristen på helt färdigutvecklade standarder försvårar arbetet med externa applikationer. De anser vidare att det måste finnas klara affärsrelationer för att kunna underlätta arbetet med externa applikationer. **Borland** tyckte också att det är svårare att utveckla externa applikationer på grund av säkerheten. Tillgängligheten blir också viktigare och bidrar till svårigheten.

### 4.1.3 Andra teknologier

Frågorna under denna rubrik ska leda till att få en inblick i fördelar respektive nackdelar mellan Web services och några liknande teknologier.

- **Hade Ni några andra teknologier som alternativ innan Ni bestämde Er för Web Services och i så fall vilka?**

  Svaret på denna fråga var enligt **Marotz** beroende på vilken typ av system som skall byggas. De menar att det alltid finns alternativa lösningar. **XML Sweden** höll med dem i detta svar genom att de väljer att anpassa sig beroende på vad kunderna efterfrågar. I **Suns** fall menade dem att de inte hade haft något annat alternativ, för dem var Web services det enda tänkbara. **Toolkit Software** hade *Java Enterprise Edition* (J2EE) med RMI för distribuerade system och IIOP som transportprotokoll som alternativ till Web services. I **Borlands** fall har deras kunder använt sig av deras utvecklingsplattform *Borland Delphi* vilket erbjuder stöd för Web service teknologin, därför är valet också naturligt för dem.

- **Hur ser Ni att Web Services skiljer sig från tidigare teknologier som tex. CORBA?**

  **XML Sweden** menar att CORBA, liksom DCOM och RMI har problemet att vara synkrona protokoll. Detta gör att CORBA inte är något bra alternativ för kommunikation över Internet. Internet använder sig av asynkron överföring och att programmera felhantering för alla olika former av avbrutna anslutningar skulle ta för långt tid och stå i vägen för den egentliga uppgiften, därför menar XML Sweden att SOAP är ett bättre alternativ trots bristerna kring säkerheten. Något som också är en nackdel med CORBA är att den är definierad som ett API

vilket gör att det finns regler som styr överföringen. Detta slipper SOAP eftersom den använder sig av XML. **Sun** tycker att styrkan i Web service gentemot CORBA ligger i att Web service är enklare att använda och är baserat på öppna standarder. **Marotz** anser att CORBA är svårare att implementera, den har dock större funktionalitet.
En nackdel med DCOM är att den är beroende av ett operativsystem, vilket Web services inte är. Enligt **Toolkit Software** är fördelen för Web services att den är datacentrerad till skillnad från CORBA som är metodcentrerad. Detta tillsammans med att Web services använder sig av XML-protokollet leder till färre interoperabilitetsproblem. En annan fördel är att Web services är mer accepterad på alla plattformar och mycket lättare att förstå och implementera. Några nackdelar är dock att Web services är långsammare och inte lika moget som CORBA. Enligt **Borland** är fördelen med Web service teknologin att den bygger på en enkel web teknologi med textmeddelanden som skickas över Internet. CORBA är dock mer mogen och en mycket mer kraftfull teknologi i jämförelse med Web services. CORBA är överlägsen när det gäller distribuerade tillämpningar där prestanda, skalbarhet och feltolerans är nödvändiga. CORBA kan hantera säkerhet och transaktioner på ett bra sätt idag i en heterogen system miljö.

### 4.1.4 Standard

Eftersom Web services lider av standardproblem var tanken med dessa frågor att ta reda på hur mycket detta problem påverkar utvecklarna egentligen.

- **Det finns ett problem när det gäller standard för Web Services, hur angriper Ni problemet?**

  **XML Sweden** anser att de första generationsstandarderna, alltså SOAP, WSDL och UDDI, nästan är färdigutvecklade och därför finns det inte mycket mer att göra i dem. Därför är deras viktigaste koncentration att förvalta dessa standarder och göra justeringar där det behövs. **Sun** löser problemet genom att endast använda sig av öppna standarder och sedan implementera dessa när det behövs. **Marotz** håller sig endast till SOAP, detta eftersom Sun och Microsoft inte lyckats nå någon kompromiss och de vill inte använda någon av deras verktyg tills en standard har godkänts. **Toolkit Software** följer de rekommendationer som Web Service organisationen ger så lång som det är möjligt. **Borland** avvaktar och använder tekniken för de ändamål som tekniken erbjuder idag.

- **Hur tror Du att detta problem kommer att lösas i framtiden?**

  **XML Sweden** menar att det är helt avgörande för Web services att det finns standarder. De är också viktigt att dessa är enkla att lära sig och implementera. De tror att Microsoft .Net kommer att vinna kampen och på så sätt kommer Web services att enas runt en enda standard. Enligt **Sun** kommer de öppna standarderna att dominera, standarder där ingen är huvudansvarig för dem. **Marotz** tror i sin tur att kampen kommer att avgöras genom att en av alla standarderna kommer att få övertaget helt enkelt, vilken nämns inte, men fram tills detta inträffar får man vänta sig att det kommer att finnas en hel del diskussioner.
  **Toolkit Software** tror inte alls att problemet kommer att lösas i framtiden. De menar att det alltid kommer att finnas tillfällen där man måste utveckla applikationer som inte bygger på någon standard. **Borland** tror att det kommer att hända mycket med teknologin framöver eftersom den är ganska ny. De tror att teknologin kommer att vara tillgängligt i alla kommersiella produkter, som mobiltelefoner och system. Web services kommer att bli de facto standard i för kommunikation mellan system enligt Borland.

### 4.1.5 Säkerhet

Säkerheten är ett hett ämne just nu när det gäller Web services och då måste man få veta vad utvecklarna tycker om det.

- **Säkerheten kring Web Service teknologin sägs inte vara tillräcklig, hur ser Ni att den kan förbättras?**

  **XML Sweden** tycker att det är ett krav att SOAP standarden använder sig av ett annat transportprotokoll än HTTP, detta eftersom det använder sig av port 80 som är den port där all webbtrafik går igenom. Visserligen för användningen att denna protokoll att SOAP-meddelanden inte får några problem med brandväggar, men brandväggarna måste anpassas och XML Sweden menar att detta endast är en kortsiktig lösning. **Sun** tycker att standarderna som är under utvecklig måste färdigställas för att säkerheten ska öka. **Marotz** tycker att säkerheten inte är något stort problem eftersom dess hantering kommer att byggas och bli tillräcklig. De tycker att redan idag erbjuds tillräckliga tillämpningar som dock inte är gratis. **Toolkit Software** tror att säkerheten kan förbättras genom att man använder sig av de specifikationer som *Microsoft*, *IBM* och *VeriSign* utarbetat. **Borland** tycker att transaktionssäkerhet, tillgänglighet, kryptering och komprimering på protokoll nivå borde förbättras.

### 4.1.6 Tillförlitlighet

- **Experter säger att tillförlitligheten hos Web Services måste bli bättre för att den ska bli mer accepterad för extern användning, håller Du med och hur kan den förbättras?**

  Både **XML Sweden, Sun** och **Toolkit Software** håller med experterna om att tillförlitligheten hos Web services måste förbättras. **XML Sweden** anser att den kan förbättras genom att byta transportprotokollet som används idag, HTTP. **Sun** anser istället att det är standarderna som kan förbättra tillförlitligheten, eftersom de har förmågan att bli föremål för enighet. Därför är det viktigt att de standarderna blir färdiga.
  **Toolkit Software** anser att det behövs Quality of Service och bättre hantering av leveransen av ett meddelanden. **Marotz** däremot tycker att det idag redan finns teknologi för att erbjuda en tjänst som tillgodoser tillförlitligheten. **Borland** tycker att det inte finns tillräcklig stöd för feltolerans i SOAP- protokollet idag. De menar att det skulle vara bra om en Web service skickade ett svar på om en förfrågan kommit fram eller inte, detta är speciellt viktigt när det gäller monetära transaktioner. Det är också viktigt att vara säker på att det är rätt system som skickar förfrågningar, så ingen känslig information lämnas ut till någon otillbörlig part som lyckats hacka sig in i systemet.

### 4.1.7 Framtiden för Web service

Det var intressant att ta reda på om utvecklarna tror på Web services i framtiden, vilket ledde till dessa frågor.

- **Vad tror Du att Web Services behöver förbättra eller lägga till för att kunna slå igenom ännu mer och bli en framtidsteknologi?**

  **XML Sweden** anser att säkerheten runt Web service måste förbättras för att fler ska våga använda sig av teknologin. Det behövs även vissa förändringar i dagens förslag till standarder till exempel att transportprotokoller HTTP bytes ut mot HTTP/S för att lättare kunna angripa alla typer av säkerhetsaspekter. **Sun** tycker att fokusering måste ändras för att Web services ska kunna slå igenom bättre. Nu är det visioner som styr, men de ser hellre till att affärsnyttan stod

mer i fokus. **Marotz** tycker att det behövs förbättringar inom säkerhet, sessionshantering, transaktionshantering och händelsehantering för att teknologin ska kunna få framgång och bli det enda alternativet. Förutom att standarderna måste bli klara tycker **Toolkit Software** också att det behövs Quality of Service samt en bättre hantering av leveransen av ett meddelanden. Detta för att vara säker på att ett meddelande verkligen kommer fram. Enligt **Borland** måste standarderna mogna och få en enhetlighet som gör att alla förstår dem på samma sätt.

- **Vilka fördelar tror du kommer att göra att Web Services till en vinnare?**

  **XML Sweden** tror att det är transportprotokollet HTTP tillsammans med XML formatet som står för den största fördelen. Användningen av XML bidrar till en sann språk och plattformsoberoende. SOAP bidrar också till en enkel metod för överföring av data över Internet. En annan fördel är att Web services står under standardorganisationen W3C.
  **Sun** tror att enkelheten och möjligheten till nya affärssamarbeten kommer att bidra till att Web services blir en vinnare. **Marotz** tror att plattforms- och operativsystem oberoendet tillsammans med enkelheten av dess användning kommer att göra teknologin till en vinnare.
  **Toolkit Software** tror som Sun att det är enkelheten som kommer att vara den avgörande faktorn till Web service framgången. Plattformsoberoende, språkneutralt och leverantörsoberoende är dem egenskaperna som kommer att göra Web service teknologin till en vinnare enligt **Borland**. Detta tillsammans med att företag som IBM, Sun och Microsoft är eniga och stöder utvecklingen gör teknologin mer framgångsbenägen.

- **Är Web Services en teknologi att räkna med i framtiden? Förklara Din synpunkt.**

  Teknologin där datorer som transparent initierar kontakter med andra datorer kommer absolut att växa i framtiden, namnet däremot på denna teknologi, om det är Web services eller inte är något som **XML Sweden** inte har något svar på, de är dock övertygade att teknologin kommer att finnas, även om förändringen kommer att dröja några år. **Sun** tror absolut att Web services är att räkna med. Detta på grund av att Web services erbjuder en standardiserad teknologi att bygga nya affärsrelationer på. **Marotz** tycker som Sun att Web services är något att räkna med. Detta eftersom Web services inte har någon egentlig konkurrent, trots CORBA med flera.
  **Toolkit Software** tror definitivt att Web services är en teknologi för framtiden eftersom det är så många i industrin som står bakom den.
  Detta tillsammans med att kunderna redan börjat implementera teknologin styrker hans åsikt.
  **Borland** tror absolut att Web service teknologin är att räkna med. De tror att teknologin kommer att vara den standardiserade metodiken och tekniken för att kommunicera mellan system. Den är flexibel och enkel och kommer i framtiden även att vara säkert och tillförlitlig. Tyvärr tappar man prestanda på grund av flexibiliteten i teknologin men de tror att även detta kommer att lösas i framtiden genom den tekniska utvecklingen inom kommunikationsområdet.

# 5 Diskussion

Den huvudsakliga frågeställningen inför denna uppsats var om Web services teknologin har en framtid. Jag valde att undersöka svaret på denna fråga genom att först sätta mig in i vad Web services teknologin är och hur den fungerar. Nästa steg i min undersökning var att ta reda på vad utvecklare av denna teknologi hade för åsikter. För att undersöka detta genomförde jag ett flertal intervjuer med några ledande utvecklare inom detta område.
Jag baserade min undersökning på de framgångsfaktorer som tagits fram under teorin och jag har kommit fram till att även om det är arbete kvar så kan nog dessa framgångsfaktorer uppfyllas.

## *5.1 Redovisning av framgångsfaktor*

Genom att undersöka Web service teknologin utifrån de uppsatta framgångsfaktorer har jag funnit det vara möjligt att dra en slutsats. Dessa faktorer kommer nu att diskuteras.

### 5.1.1 Standard problemet

Efter en litteraturundersökning har det visat sig att Web services är en samling befintliga standarder, som samlats ihop till en teknologi. Själva grundtanken med Web services är att bryta ner stora och oöverblickbara system till mindre delar för att kunna hantera det lättare och denna teorin har funnits i ett antal år. Däremot är själva grunden för hur dessa anrop ska utföras och vad som ska användas till överföringen av data ett relativt nytt påfund. De flesta utvecklarna anser att det krävs att Web service teknologin uppnår standard nivå på alla sina delar innan en större genombrott kan ske.

Både respondenterna och litteraturkällor anger att något som är viktig för att företagen ska vara säkra på att teknologin har en framtid, är att den blir en standard i alla avseenden. Idag råder det inte någon total enighet kring de olika standarderna. Den enda enigheten finns i SOAP, som i och för sig är grunden i Web services. Många av respondenterna och andra källor tror att den som plockar hem den största delen av marknaden kommer att vara den som också kommer i slutändan att definiera standarderna i framtiden. Detta oavsett om det är den bästa lösningen eller inte. Enhetligheten är dock ganska hög i dagsläget. Både SOAP och WSDL är brett accepterade och UDDI håller på att bli mer och mer accepterad och använd.

Litteraturstudien visar att en orsak till att Web service teknologin kommer att lyckas enas runt en standard är att alla protokoll som tillsammans bygger upp teknologin för Web services hanteras av standardiseringsorganisationen W3C. Det är en fördel om ett världstäckande och vedertaget standardiseringsorgan som W3C-organisationen beslutar när en teknik anses vara standard. Det gör det enklare för företag att enas om framtida standarder.

En annan orsak är också att det stora antal företag som hjälper till och utvecklar teknologin för Web services, där de största parterna är *Microsoft*, *SUN* och *IBM*. På grund av att företag som dessa utvecklar teknologin för Web services kommer genomslagskraften bli stor då de alla har stort inflytande över stora delar av IT-industrin. De ökar även intresset för Web services genom stora marknadsföringskampanjer. Exakt när alla leverantörerna enas kring en standard är inte säkert, men respondenterna och litteraturen tror att det kommer att ske inom en snar framtid.

### 5.1.2 Säkerheten i Web services

Enligt respondenterna och litteraturen är säkerheten i Web service teknologin en bristande del i utvecklingen. Eftersom Web services distribueras över Internet är de väldigt lättåtkomliga och känsliga för attacker. Av den anledningen är det viktigt att säkerheten blir färdigutvecklad, innan något större genombrott sker. Litteraturen visar att det går visserligen att använda sig av redan befintliga metoder, såsom SSL, men metoden är inte utvecklad för Web services, utan lämpar sig mer för att kryptera hela tunnlar med trafik. För en kedja med Web services är det mer intressant att varje nod kan kryptera delen den ansvarar för. Vanligtvis binds SOAP mot HTTP eftersom kommunikationen vanligtvis inte hindras av brandväggar. Dock är säkerheten inte tillfredställande vilket alltid är svårt för distribuerade system, detta eftersom ett meddelande kan passera flera noder. Det finns inget standardiserat sätt för SOAP att utföra krypteringar eller signeringar av meddelanden. I det avseendet går det idag inte att använda SOAP till något som innehåller känslig data, utan att införa egna krypteringsmetoder. Respondenterna ansåg att säkerheten måste förbättras, dock kände de fullt förtroende för att det nog skulle komma en lösning som skulle vara tillfredsställande. Det görs hela tiden framsteg i säkerheten genom att det utvecklas flera olika verktyg. Litteraturen angav några lösningar för detta där det kommer att erbjudas verktyg för att filtrera Web service meddelanden över brandväggar. Andra lösningar är *SOAP-DSIG* för kryptering, *SAML* för autentisering och *XKMS* för hantering av krypteringsnycklar. Respondenterna ansåg att det var är viktigt att tillmötesgå de krav som ställs på säkerheten, men de hade också tilltro till att detta också kommer att göras. Det kanske tar ett tag till innan en fungerande säkerhetsstandard blir verklighet, men ett sådant är på gång. Detta bidrar till att säkerheten kommer att bli bättre, vilket i sin tur är ett bidragande faktor till att teknologin kommer att slå igenom.

### 5.1.3 Affärsnyttan

Litteraturen visar på att när en ny teknologi dyker upp är det viktigt för den att visa på affärsnyttan för användarna. Om användaren inte känner behovet av den nya teknologin kommer den heller inte att slå igenom. Oftast är också en nyare teknologi dyrare vid starten eftersom det är så många faktorer som måste stämma. Det är då ännu viktigare att motivera användarna att våga lägga ut kostnaderna och testa. Det är viktigt att teknologin tillhandahåller affärsfördelar till rimliga priser även för småföretagare.

Enligt litteraturen erbjuder Web service teknologin idag lösning till många problem. För interna Web services erbjuds lättare hantering av befintliga system, för externa Web services erbjuds en uppsjö av olika tjänster. Efterhand som teknologin blir mognare och stabilare kommer fler tjänster att kunna erbjudas. Detta bidrar till att företag kan känna att just deras behov blir mötta.

### 5.1.4 Enkelhet

Enligt respondenterna och litteraturen är den stora fördelen med Web service teknologin gentemot andra tidigare teknologier just dess enkelhet. Bland de respondenterna fanns en gemensam tro på att enkelheten i Web services skulle vara en bidragande effekt till dess framgång. Litteraturen beskriver att enkelheten ligger i att Web services baseras på XML, vilket är själva byggstenen för hela konceptet och som redan är en rekommenderad standard av W3C. All dataöverföring samt överliggande protokoll för Web services är definierade i just denna syntax. Orsaken till valet beror på att det är ett enkelt språk, som både maskiner och människor snabbt kan tolka. En annan fördel med XML är dess språkoberoende och att den enkelt kan utnyttjas av alla plattformar. Vilket i sin tur bidrar till att hela Web service teknologin är plattforms-, programmeringsspråk- och försäljaroberoende.

Litteraturen visar att teknologin är enkel att lära sig även för människor utan någon större erfarenhet av system utveckling. En del av enkelheten ligger i att teknologin använder sig av XML, som är ett dataformat som är läsbart för människor medan andra teknologier använder binära format istället. Något som också bidrar till enkelheten är att teknologin använder sig av transportprotokollet HTTP. Detta bidrar dock både till för och nackdelar, vilka redan tagits upp under säkerhetsdiskussionen.

### 5.1.5 Tillförlitlighet

Företagen måste kunna räkna med att Web service klarar av påfrestningar och säkerheten. Enligt respondenterna är kravet för en ökad tillförlitlighet att samsas runt standarder, både när det gäller teknologistandarderna och säkerheten. När dessa uppfylls är sannolikheten stor att fler företag kommer att känna att Web service teknologin är en tillförlitlig teknologi. Innan dessa krav möts är inte tillförlitligheten tillräcklig.

Litteraturen visar att eftersom Web service teknologin är ganska ny har det inte hunnits testas färdigt. Många företag befinner sig fortfarande under en testfas, där de först koncentrerar sig på interna integrationsprojekt, eftersom interna projekt är mindre riskfyllda och enklare att testa. Men det finns även exempel på företag som har startat med externa projekt, där Web services används för att koppla ihop affärspartner. Trots detta tror litteraturen att det lär dröja något år innan de flesta företag flyttar sina Web services projekt utanför brandväggen. Många företag resonerar som så att bara för att det finns en lättanvänd teknik innebär det inte att de måste ändra sin affärsmodell över natten, vilket är rätt. Men det är ändå viktigt för dem att fortsätta att experimentera med Web services eftersom arbetet med Web services bara har börjat.

### 5.1.6 Återanvändning av gamla system

Det är viktigt för företag att kunna återanvända gamla beståndsdelar eftersom det håller kostnaderna nere. Detta har litteraturen bevisat. Det är dyrt för företag att skapa nya system och därför är det bra om Web services kan erbjuda återanvändning av gamla system. Detta tycker respondenterna är något som är bland de främsta fördelarna med Web services; den förenklar återanvändning av gamla system i nya lösningar. Det är inte längre nödvändigt att utveckla funktioner som redan finns på nytt, utan det går att utnyttja existerande system. Detta bidrar till att företag inte behöver slänga något som de redan har utan Web services ger möjligheten att knyta ihop funktioner i existerande system till nya.

## *5.2 Reflektion över framgångsfaktorerna*

I början av detta arbete sattes sex framgångsfaktorer upp för att kunna svara på uppsatsens frågeställning. Dessa faktorer togs fram med hjälp av litteraturen i form av tidningsartiklar, böcker och dokument på Internet. Anledningen till att just dessa sex faktorer valdes var att det upptäcktes att dessa faktorer togs upp separat av andra tekniker. Genom att samla ihop dem i detta arbete kunde de sedan användas för att komma fram till ett resultat. Eftersom en omfattande litteraturstudie hade gjorts innan dessa faktorer samlades ihop bidrog det till att faktorerna valdes efter om de var möjliga att besvaras. Eftersom faktorerna ligger i tiden var det inga svårigheter att få respondenterna att förstå vad som efterfrågades. Det var också lättare att hitta information i tidningar och på Internet som behandlade dessa faktorer. Om några andra faktorer hade valts istället hade dessa visat en annorlunda sida av Web service teknologin, en sida som hade gjort att resultatet hade varit helt annorlunda.

# 6 Slutsats

Sammanfattningsvis kan det sägas att teknologin för Web services ännu är ganska ny och behöver tid på sig att mogna. Det kommer att ta en tid innan teknologin färdigställs helt, och hur lång tid det kommer att ta vet man ännu inte. Vad som dock är viktigt är att de underliggande protokollen är rekommenderade standarder som redan används. Detta bidrar till att säkerhetskraven höjs och åtgärder kommer att börja dyka upp mer och mer. Det kommer att dröja ytterligare ett par år innan teknologin ges ut som en mer komplett version, där alla de tre grundprotokollen och säkerheten är standarder. Orsaken till detta är att protokoll, som exempelvis UDDI och säkerhetsprotokollen, måste synkroniseras och utvecklas ytterligare innan de kan antas som standarder.

Min övertygelse är att när standardiseringen har inträffat kommer företag att börja använda Web services i större skala, men det är klokt att redan nu börja i liten skala för att vara redo när genombrottet kommer. Web services är nämligen något som världen kommer att anpassa sig efter. En anledning till detta är på grund av den mängd stora företag som ligger bakom.

Teknologins svagheter kan överbyggas med gemensamma krafter och dess fördelar är redan idag slående. Efter min undersökning kan jag dra slutsatsen att trots befintliga brister hos Web services kommer den ändå att bli framtidens teknologi. Därför är min rekommendation att dagens företag redan idag sätter sig in i denna teknologi.

## *6.1 Fortsatt forskning*

Eftersom teknologin för Web services inte har nått en färdig status ännu, finns det fortfarande mycket som kan utforskas i området. Något som skulle kunna undersökas är implementationer och tester över prestanda för Web services. I detta ingår också undersökning på om dagens nätverk är redo att bemöta en uppsjö av dessa tjänster.

Det skulle också vara intressant med en utvärdering av de olika säkerhetsprotokoll som redan finns och som troligen kommer att bli klara eftersom tiden går. Det borde också utföras implementationer innehållande säkerhet för att undersöka hur prestanda påverkas och om det verkligen är praktiskt genomförbart.

# Efterord

Jag skulle vilja framföra min tacksamhet till några personer som gjorde denna uppsats möjlig.

Först och främst vill jag tacka min handledare Kari Wahll som varit ett stöd för mig och hjälpt mig att hålla rätt kurs i mina undersökningar. Jag körde fast några gånger men efter samtalen med Kari gav hon mig ny inspiration och skärpa så att jag kunde gå vidare.

Jag vill också rikta min tacksamhet till alla personer som deltagit i mina intervjuer, trots att de alla befann sig mitt inne i en hektisk period av sina arbeten.

Jonas Lindskog från Toolkit Software, Robert Recklin från Borland, Gustaf Liljegren från XML Sweden, Henry Aspenryd från Marotz och Per-Olof Litby från Sun Microsystems.

Tack allihopa för att ni gav er tid att svara på mina frågor.

Sist men inte minst vill jag också tacka min man som varit ett stöd och inspirationskälla för mig under hela arbetets gång.

Göteborg, december 2002

Roxanna Gustafson

# Referenser

## *Böcker*


- Backman, Jarl. *Rapporter och uppsatser*. Studentlitteratur: Lund, 1998.
- Bengtsson, Bengt-Arne & Bengtsson, Hans. *Zigma; Foskningsboken; Om konsten att arbeta på ett undersökande och kunskapande sätt*. Almqvist & Wiksell Förlag AB: Uppsala, 1995.
- Cerami, Ethan. *Web Services Essentials*. O´Reilly & Associates: Sebastopol, 2002.
- Clabby, Joe *Web Services Explained* Prentice Hall: New Jersey, 2002
- Holme, Idar & Solvang, Bernt Krohn. *Forskningsmetodik: Om kvalitativa och kvantitativa metoder*. Lund: Studentlitteratur, 1997.
- McLaughlin, Brett. *JAVA and XML*. O´Reilly: Sebastopol, 2000.


## *Tidningar*


- Höij, Magnus ”Web services - Drömmen som blev sann”, *Computer Sweden*, 2 oktober 2002.
- Olofsson, Kent ”Snabbare utveckling”, *Computer Sweden*, 16 oktober 2002
- Olofsson, Kent ”Web services vid brytpunkten”, *Computer Sweden*, 16 oktober 2002
- Olsson, Henrik ”Web services bra men inte nytt”, *Computer Sweden*, 28 oktober 2002
- Svidén, Henrik ”Myndihetssverige suddar gränserna-Web services knyter ihop alla myndigheter”, *Computer Sweden*, 14 oktober 2002
- Wallström, Martin, ”Snabb expension lockar många aktörer”, *Computer Sweden*, 9 oktober 2002
- Wallström, Martin, ”Tre tunga utmaningar”*, Computer Sweden*, 23 oktober 2002


## *Elektroniska källor*

### Artiklar


- CERN. (1997).
  *History and growth.* [online]. Tillgänglig:
  http://public.web.cern.ch/Public/ACHIEVEMENTS/WEB/history.html .
  [2002-09-16]
- Chopra, Vivek, Zoran, Zaev, Damschen, Gary, Dix, Chris et al. (2001).
  *Professional XML Web Services* [online]. Tillgänglig:
  http://www.vbip.com/books/1861005091/chapter_5091_01.asp
  [2002-10-01]
- Dumser, Johann (2001).
  *Internal or External Web Services? - Which One Will Dominate?* [online]. Tillgänglig:
  http://www.webservicesarchitect.com/content/articles/dumser03.asp
  [2002-12-01]
- MacRoibeaird, Sean (2002).
  *Universal Description, Discovery & Integration (UDDI).* [online]. Tillgänglig:
  http://wwws.sun.com/software/xml/developers/uddi
  [2002-10-18]

- Randell, Brian, Aaron Skonnard (1999).
  *A Guide to XML and Its Technologies*. [online]. Tillgänglig:
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnxml/html/xmlguide.asp
  [2002-11-05]
- Ryman, Arthur (2000).
  *Understanding Web Services*. [online]. Tillgänglig:
  http://www7.software.ibm.com/vad.nsf/Data/Document4362?OpenDocument&p=1&BCT=#&Footer=1
  [2002-12-02]
- Sterling, Bruce. (1992).
  *Short History of the Internet*. [online]. Tillgänglig:
  http://w3.aces.uiuc.edu/AIM/scale/nethistory.html.
  [2002-09-16]
- Techstreet (2002).
  *The What, Why, and How of Standards*. [online]. Tillgänglig:
  http://www.cssinfo.com/whystandards.html
  [2002-12-04]
- World Wide Web Consortium (2002).
  *Leading the Web to its Full Potential*...[online]. Tillgänglig:
  http://www.w3.org
  [2002-10-05]


## Specifikationer


- Berners-Lee Tim, R. Fielding, U.C Irvine, L. Masinter (1998).
  *Resource Identifiers (URI): Generic Syntax*. [online]. Tillgänglig:
  http://www.ietf.org/rfc/rfc2396.txt?number=2396
  [2002-10-15]
- Box, Don, Ehnebuske, David, Kakivaya, Gopal et al. (2000)
  *Simple Object Access Protocol (SOAP) 1.1, W3C Note* [online]. Tillgänglig:
  http://www.w3.org/TR/SOAP/
  [2002-09-15]
- Christensen, Erik, Curbera, Francisco, Meredith, Greg, Weerawarana, Sanjiva (2001)
  *Web Services Description Language (WSDL) 1.1 W3C note 15 march 2001*. [online]. Tillgänglig:
  http://w3c.org/TR/wsdl .
  [2002-09-29]
- Ehnebuske, Dave, McKee, Barbara, Rogers, Dan (2002).
  *UDDI Version 2.04 API Specification, UDDI Published Specification , 19 July 2002* [online].
  Tillgänglig: http://www.uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf
  [2002-10-18]
- Ehnebuske, David, Rogers, Dan, Riegen, Claus von (2002).
  *UDDI Version 2.03 Data Structure Reference, UDDI Published Specification, 19 July 2002*.
  [online]. Tillgänglig: http://www.uddi.org/pubs/DataStructure-V2.03-Published-20020719.pdf
  [2002-10-15]


## Tidningar


- Danielsson, Lars (2002). *Web services förändrar utvecklingen i grunden*.
  [online]. Tillgänglig:
  http://computersweden.idg.se/ArticlePages/200209/26/20020926155442_CS465/20020926155442_CS465.dbp.asp
  [2002-12-17]

- Eriksson, Ingemar (2002). *Web services i praktiken: Storebrand har gått från hajp till handling.* [online]. Tillgänglig: http://cio.idg.se/artikelarkiv/artikel.asp?ID=373 [2002-12-17]
- Lidström, Karin (2002). *Platt skärm för dyr.* [online]. Tillgänglig: http://computersweden.idg.se/ArticlePages/200211/28/20021128123532_CS508/20021128123532_CS508.dbp.asp [2002-12-06]
- Olofsson, Kent (2002), *5 heta tekniktrender* [online]. Tillgänglig: http://computersweden.idg.se/ArticlePages/200209/10/20020910170703_CS303/20020910170703_CS303.dbp.asp [2002-11-29]
- Ricknäs, Mikael (2001). *Genombrott för het SIP-protokoll.* [online]. Tillgänglig: http://nyheter.idg.se/display.asp?ID=011123-CS10 [2002-12-09]
- Uvell, Markus (2002). *Från affärsmän till tonårstjejer.* [online]. Tillgänglig: http://computersweden.idg.se/ArticlePages/200207/18/20020718164058_CS963/20020718164058_CS963.dbp.asp [2002-12-04]


## White Pappers


- Bellwood, Thomas A.(2001). *UDDI - A Foundation for Web Services.* [pdf]. Tillgänglig: http://www.idealliance.org/papers/xml2001/papers/pdf/03-02-03.pdf [2002-10-15]
- Deitel, Harvey M, Paul. J, Deitel. (2002). *Web Services A Technical Introduction.* [pdf]. Tillgänglig: http://www.deitel.com/newsletter/20020523/articles/WebServices_SOAP-WSDL-UDDI.pdf [2002-10-01]
- Systinet (2002). *Introduction to Web Services.* [pdf]. Tillgänglig: http://www.systinet.com/download/332fd7be0e8021ff85e15a95d73a7d96/wp_Introduction_to_Web_Services.pdf [2002-10-29]
- UDDI Organisation (2000). *UDDI Technical White Paper.*[pdf]. Tillgänglig: http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf [2002-10-15]

## Bilaga 1 - Ordlista

| | |
|---|---|
| Applikationswebb | Statusen som Internet har idag. Vilket medför möjligheten att generera HTML-dokument via affärslogik på webbserver sidan. |
| ARPANET | *Advanced Research Projects Agency Network*, grundstenen för dagens Internet. |
| ASP | *Active Server Pages*, ett serverskriptspråk som utvecklats av Microsoft. |
| CERN | *Le Conseil Européen pour la Recherche Nucléaire*, ett världsledande europeiskt laboratorium för partikelfysik som grundades 1954. Tim-Berners Lee jobbade där när han utvecklade WWW. |
| CGI | *Common Gateway Interface*, ett samlingsnamn för serverbaserade skriptspråk. |
| CSF | *Critical Success Factors*, en metod för att identifiera informationsbehovet för marknadschefer som introducerades av John Rockart. |
| CORBA | *Common Object Request Broker Architecture*, en metod för att utföra distribuerad kommunikation vilken inte använder Internet som bas. |
| DCOM | *Distributed Component Object Model*, är ett protokoll som gör det möjligt för mjukvarukomponenter att kommunicera med varandra direkt över ett nätverk. |
| Dokumentwebb | Den första statusen som Internet hade. Internet bestod då till största delen av statiska dokument. |
| DTD | *Document Type Definition*, används för validering av innehållet i ett XML dokument. |
| EIP | *Enterprise Information Portal*, en portal som kan bestå av komponenter som en Web service har behov av. |
| FTP | *File Transport Protocol*, ett protokoll för överföring av filer över Internet. |
| HTML | *Hypertext Markup Language*, det språk som används för att skapa dagens hemsidor. |
| HTTP | *Hypertext Transform Protocol*, kommunikationsprotokollet som används för att överföra HTML-dokument eller allmän webbtrafik på Internet. |
| Internet | Ett publikt nätverk som är lättillgängligt både för allmänheten och företag. |
| IP | *Internet Protocol*, ett protokoll som specificerar formatet på ett paket och deras adressystem. |
| ISO | *International Organization for Standardization*, ett internationellt standardiseringsorgan. |
| JSP | *Java Server Pages,* ett serverskriptspråk baserat på Java som utvecklats av Sun. |
| RPC | *Remote Procedure Call,* ett standardiserat sätt för ett program att kommunicera med ett annat program på en annan dator, utan att behöva känna till och förstå nätverksdetaljerna. |

| | |
|---|---|
| SAML | *Security Assertion Markup Language,* en öppen XML-standard utvecklad av OASIS för att klara av säkra elektroniska transaktioner. |
| SGML | *Standard Generalized Markup Language*, är det språk som XML baseras på och är en delmängd av. |
| SMTP | *Simple Mail Transport Protocol*, ett vanligt protokoll för att skicka e-post. |
| SOAP | *Simple Object Access Protocol*, ett XML-baserad protokoll för utbyte av data mellan system som finns distribuerade över ett nätverk. |
| SOAP-DSIG | *SOAP Security Extensions:Digital Signature*, Digitala signaturer för SOAP meddelanden. |
| SSL | *Secure Socket Layer*, en vedertagen tunnelkrypteringsmetod för Internet. |
| TCP | *Transport Control Protocol,* ett kopplingsorienterat protokoll som används för att skicka data på Internet. |
| Tjänstewebb | Den status som Internet är på väg att övergå till. Denna kommer innefattas av bland annat teknologin för webbtjänster. |
| UDDI | *Universal Discovery Description and Integration*, ett sökregister för webbtjänster. |
| URI | *Uniform Resource Identifier*, en unik sträng av tecken som identifierar en källa. Den kan vara av två typer URN eller URL. |
| URL | *Uniform Resource Locator*, identifierar en unik adress för en webbsida |
| URN | *Uniform Resource Name*, ett unikt och globalt namn på en resurs. |
| W3C | *World Wide Web Consortium*, ett standardiseringsorgan för Internet teknologier. |
| WSDL | *Web Services Description Language*, ett XML-baserat dokument som beskriver en webbtjänst. |
| WWW | *World Wide Web*, ett system av Internets servers som tar emot speciellt utformade dokument. |
| XKMS | *XML Key Management Services*, ett XML-baserat protokoll som definerar registrering och åtkomst av publika nycklar för webbtjänster. |
| XML | *eXtensible Markup Language*, en syntaxnotation för att på ett strukturerat och standardiserat sätt överföra data mellan två parter. |

## Bilaga 2 - Intervjuformulär

### Intervju om Web Services

*Var snäll och ge så uttömmande svar som möjligt.*

1. Varför valde Ert företag att börja utveckla Web Services applikationer?
2. Hade Ni några andra teknologier som alternativ innan Ni bestämde Er för Web Services och i så fall vilka?
3. Vilka uppgifter sköter Web Services applikationerna som Ert företag skapar?
4. Vilken typ av applikation utvecklar Ni flest av, för användning internt (program till program kommunikation) inom företag eller för extern användning (tjänster) över Internet?
5. Om det är flest för internt bruk (program till program kommunikation), varför är det så?
6. Är det svårare att utveckla applikationer för extern användning och i så fall varför?
7. På vilket sätt har Web Services teknologin underlättat för Er att lösa problem som annars varit svårare att lösa?
8. Vilka nackdelar eller hinder ser ni med Web Services?
9. Det finns ett problem när det gäller standard för Web Services, hur angriper Ni problemet?
10. Hur tror Du att detta problem kommer att lösas i framtiden?
11. Vad tror Du att Web Services behöver förbättra eller lägga till för att kunna slå igenom ännu mer och bli en framtidsteknologi?
12. Hur ser Ni att Web Services skiljer sig från tidigare teknologier som tex. CORBA?
13. Säkerheten kring Web Service teknologin sägs inte vara tillräcklig, hur ser Ni att den kan förbättras?
14. Experter säger att tillförlitligheten hos Web Services måste bli bättre för att den ska bli mer accepterad för extern användning, håller Du med och hur kan det förbättras?
15. Vilka fördelar tror du kommer att göra att Web Services till en vinnare?
16. Är Web Services en teknologi att räkna med i framtiden? Förklara Din synpunkt.

Tack för Din medverkan!

# Bilaga 3 - Web Service Arkitekturen

I denna bilaga finns en närmare beskrivning av Web service arkitekturen, några av delarna har nämnts tidigare men tas upp här igen för att få en sammanhängande version. Detta för att läsaren ska kunna gå till denna del utan att ha läst hela arbetet innan.

Web Service arkitekturen består som tidigare redovisats av tre standarder som tillsammans skapar en Web service. Dessa är SOAP, WSDL och UDDI. Dessa tre standarder möjliggör en kommunikation mellan en Web service och en anropande applikation på ett system, där kommunikationen är oberoende av programmeringsspråk, operativsystem och hårdvaruplattform. [67]

Standarden SOAP tillhandahåller en kommunikationsmekanism mellan en Web service och en applikation, standarden WSDL erbjuder en gemensam metod för att beskriva en Web service för andra program och standarden UDDI gör det möjligt att söka efter en specifik Web service. När dessa tre standarder slåss samman kan en utvecklare skapa en applikation som erbjuder en tjänst vilken kan publiceras på webben. [68] Grunden i Web services är *eXtensible Markup Language* (XML). XML är ett regelverk för att skapa märkspråk, ett så kallat metaspråk, och som gör det möjligt att utbyta information mellan olika applikationer oberoende av dataformat som används internt. [69] Genom att använda XML i Web services bidrar det till att Web services kan använda sig av Internet för att erbjuda och utnyttja tjänster. I kommande avsnitt återfinns en närmare beskrivning av varje del.

## *XML (eXtensible Markup Language)*

XML är en öppen och accepterad standard som rekommenderades av *World Wide Web Consortium (*W3C) 1996 för att underlätta dataöverföring. [70] W3C är en organisation som utvecklar teknologier, standarder, mjukvaror och verktyg för att utveckla webben till dess fulla potential. [71] Syftet med XML är att på ett strukturerad och standardiserad sätt överföra data mellan två parter.

När webbens popularitet exploderade under 1990-talet upptäckte man att *Hypertext Markup Language* (HTML) hade vissa begränsningar. HTML saknar förmågan att förändra eller lägga till egenskaper, den har ingen förmåga att beskriva data som den formaterar. Detta gjorde att utvecklare strävade efter ett bättre alternativ. Som ett svar till detta uppkom XML. [72]

XML är en delmängd av *Standard Generalized Markup Language* (SGML) som är en definierad *International Organization for Standardization* (ISO) standard. SGML är inget språk utan en standard för hur man skapar språk. Den beskriver den logiska strukturen i ett dokument. Den anger kapitel, rubriker, stycken och fotnoter, men inte hur dessa ska se ut. Eftersom SGML är alltför komplext har dess stora användning uteblivit. SGML är inte lämpad för att användas över webben. Liksom SGML är XML ett *meta language* men XML är enklare än SGML och anpassad för användning på webben. Till skillnad mot HTML definierar XML inte bara hur data presenteras, utan även själva innehållet. [73]

---

[67] Cerami,, 3-5.

[68] Ibid., 15-18

[69] McLaughlin, 2.

[70] Ibid.

[71] World Wide Web Consortium (2002). *Leading the Web to its Full Potential*...[online].
Tillgänglig: http://www.w3.org [2002-11-05]

[72] Deitel & Deitel, http://www.deitel.com/newsletter/20020523/articles/WebServices_SOAP-WSDL-UDDI.pdf

[73] McLaughlin, 4

## XML syntax

Syntaxen för ett XML-dokument är väldigt likt HTML. Figur 5 visar ett enkelt exempel på ett XML-dokument.

```
<?xml version = ”1.0” ?>
<order>
   <product>Table</product>
   <price>1050 Kr</price>
   <productID>154888</productID>
</order>
```

*Figur 5 Ett exempel på ett XML-dokument. Dokumentet innehåller information angående en beställning av ett bord med dess pris och produktID.*

Varje element består av en start- och en sluttagg, `<product>` och `</product>` samt ett innehåll som finns emellan dem, `Table`. Starttaggen kan även innehålla ett antal attribut, som kan se ut på följande sätt:

```
<product type = furniture>
```

Där namnet för attributet är `type`, och attributets värde läggs in efter lika med tecknet `(=)` alltså `furniture`. En sluttagg däremot börjar alltid med tecknet `</` och avslutas med `>`. Texten däremellan är den exakta kopian av startaggens namn, som sluttaggen avslutar. I detta fall, `</product>`.

XML är till skillnad från många andra språk *case-sensitive,* vilket innebär att den skiljer mellan stora och små bokstäver i sina element. `<Order>` är inte samma sak som `<order>`, dessa element skulle uppfattas som två olika element av XML.

Det är också viktigt att ett XML-dokument är välformat, vilket innebär att varje element som öppnas måste också stängas, alltså en start- och en sluttag. Den måste också innehålla en *rot-element*, i detta fall är det <order>, samt ett antal barn-element som är nästlade i varandra och avlutade på ett korrekt sätt. [74] I exemplet skulle dessa vara `<product>`, `<price>` och `<productID>`.

## Namespaces

Eftersom alla element i XML inte är fixerade tillåter det för användarna att skapa sina egna element. Detta kan leda till problem. Ett problem skulle kunna uppstå när två företag utbyter XML-dokument mellan varandra. Deras element skulle kunna vara lika men dess betydelse olika. En kollision mellan dessa skulle då vara oundviklig och det skulle bli svårt att avgöra vad varje element innebär. [75] Figurerna nedan förtydligar problemet som kan uppstå.

```
<table>
   <tr>
       <td>Apples</td>
       <td>Bananas</td>
   </tr>
</table>
```

*Figur 6 Detta XML-dokument bär information i en tabell.*

---

[74] Brian Randell and Aaron Skonnard (1999). *A Guide to XML and Its Technologies.* [online]. Tillgänglig: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnxml/html/xmlguide.asp [2002-11-05]

[75] McLaughlin, 6

```
<table>
   <name>African Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```

*Figur 7 Detta XML-dokument bär information om en tabell, en möbel.*

Om de ovanstående dokumentet skulle läggas ihop skulle det uppstå en element konflikt. Båda dokumenten anger en tabell, men deras innehåll är väldigt annorlunda.

Som lösning på problemet införde W3C den 14 januari 1999 en specifikation kallad för XML *namespaces* där detta problem tas upp. XML namespaces ska tillföra en enkel metod för att göra varje element och attribut i ett XML-dokument unikt. Detta kan åstadkommas genom att binda samman dokumentet till ett namespace som identifieras av en *Uniform Resource Identifier* (URI) referens. [76] Ett URI är en sträng kan anta två olika typer, en lokaliserare *Uniform Resource Locator* (URL) eller ett namn *Uniform Resource Name* (URN). URL identifierar en unik adress för en webbsida och URN identifierar ett globalt och unikt namn på en resurs som aldrig försvinner, även om resursen skulle upphöra att existera består dess namn. [77]

Ett namespace deklareras genom att använda det reserverad attributet xmlns. Attributets värde är en URI referens som också är namnet på det namespace. Genom att göra på detta sätt får elementen ett unikt värde.[78]

```
<?xml version = ”1.0”?>
<o:order xmlns:f="http://www.w3schools.com/furniture">
   <o:product>Table</product>
   <o:price>1050 Kr</price>
   <o:productID>154888</productID>
</o:order>
```

*Figur 8 Detta XML-dokument har fått ett unikt namespace angivits, den heter `f:` och har det påhittade värdet http://www.w3schools.com/furniture*

Istället för att behöva ange `f:` i varje element kan ett namespace tilldelas ett fördefinierad värde som gäller för hela dokumentet.[79] Figur 9 visar ett XML-dokument med ett fördefinierad namespace.

```
<?xml version = ”1.0”?>
<order xmlns="http://www.w3schools.com/furniture">
   <product>African Coffee Table</product>
   <price>1050 Kr</price>
   <productID>154888</productID>
</order>
```

*Figur 9 Hela XML-dokumentet tilldelas ett unikt namespace som gäller för hela dokumentet.*

---

[76] W3C (1999). *Namespaces in XML.* [online]. Tillgänglig: http://www.w3.org/TR/REC-xml-names [2002-11-06]

[77] Tim Berners-Lee, R. Fielding, U.C Irvine, L. Masinter (1998). *Resource Identifiers (URI): Generic Syntax.* [specifikation]. Tillgänglig: http://www.ietf.org/rfc/rfc2396.txt?number=2396 [2002-11-06]

[78] Ibid.

[79] Ibid.

## Document Type Definition (DTD)

Eftersom användare av XML kan ange egna element är det viktigt att dessa följer vissa krav för att de ska kunna översättas på rätt sätt. För att kunna validera ett XML-dokument kan man använda sig av DTD som är ett språk som sätter restriktioner på ett XML-dokument. Det är definierat som en del av XML specifikationen. En DTD kan anges innanför ett XML-dokument eller referera till en extern DTD. [80]

En DTD beskriver hur ett XML-dokument bör utformas och kan användas för att definiera validiteten i ett XML-dokument. Den kan också ange relationen mellan alla element och hur attributen relaterar till olika element. En DTD ger portabilitet till ett XML-dokument, vilket gör att den kan bearbetas och valideras av vilken maskin som helst som kan lokalisera dokumentets DTD. [81] Figur 10 visar hur en extern DTD-dokument kan se ut och Figur 11 visar hur DTD-dokumentet används.

```
<?xml version="1.0"?>
<!ELEMENT order (product, price, productID)>
<!ELEMENT product (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT productID (#PCDATA)>
```

*Figur 10 En extern DTD dokument med alla element och dess värden. Filen får namnet "order.dtd".*

```
<?xml version="1.0"?>
<!DOCTYPE order SYSTEM "order.dtd">
<order>
   <product>Table</product>
   <price>1050 KR</price>
   <productID>154888</ productID >
</order>
```

*Figur 11 DTD-dokumentet används i XML-dokumentet.*

## XML Schema

XML schema har utvecklats av W3C för att ersätta DTD. Det är enkelt uttryckt en mängd regler som beskriver ett XML-dokument. Det definierar de element som kan uppkomma i ett XML-dokument tillsammans med deras attribut. Det innehåller också information om strukturen på XML-dokumentet som vilka element som är barn, antalet barn-element och sekvensen i vilket dessa uppträder. [82] Figur 12 visar hur ett XML schema kan se ut.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:complexType>
       <xs:sequence>
           <xs:element name="product" type="xs:string"/>
           <xs:element name="price" type="xs:int"/>
           <xs:element name="productID" type="xs:int"/>
           </xs:sequence>
   </xs:complexType >
</xs:schema>
```

*Figur 12 Figuren utvecklar vidare det tidigare exemplet av en order med ett XML-schema.*

---

[80] McLaughlin, 6.

[81] Ibid.

[82] Ibid., 10.

## Parser

För att kunna läsa den data som finns i ett XML-dokument behöver alla program en XML-parser. Den tolkar rådatan som finns i dokumentet och ger den en betydelse, det innebär att den skapar en händelse eller ny datastruktur av den. [83]

En parser har också som uppgift att kontrollera att dokumentet är välstrukturerat, om en DTD eller XML schema finns med kan den även avgöra validiteten på dokumentet. Genom att bearbeta rådata som inkommer tillåter det för andra XML-verktyg att arbeta vidare på dokumentet. [84]

Att välja en parser är inte så lätt, det finns två egenskaper man måste beakta. Snabbheten av parsern är viktigt, detta eftersom dokumentens komplexitet ökar och kräver mer kraft och snabbhet av parsern. En annan viktigt egenskap är parserns överensstämmelse med XML-specifikationen. Många anser att det är viktigare att parsern är snabb och bryr sig inte lika mycket om parsern följer kraven som XML-specifikationen anger. För att välja rätt parser är det viktigt att hitta en balans mellan dessa båda egenskaperna. [85]

## *SOAP (Simple Object Access Protocol)*

SOAP är ett XML-baserat protokoll för utbyte av data mellan system som finns distribuerade över ett nätverk. Den första versionen av protokollet blev tillgängligt för allmänheten 1999 och var ett resultat av ett samarbete mellan utvecklare på *Microsoft*, *DevelopMentor* och *UserLand Software*. Deras arbete ledde till att den första versionen SOAP 1.1, som överlämnades till W3C för standardisering, släpptes den 8 maj 2000.
W3C har senare lagt till ändringar och släppt ytterligare en version, SOAP 1.2 i december 2001 som ännu inte antagits som en rekommendation. [86]

Eftersom SOAP använder sig av XML bidrar det till att protokollet är plattforms- och språkoberoende. SOAP består av tre delar: ett *envelope* som specificerar ett antal regler för inkapslad överföring mellan datorer, *encoding rules* (kodningsregler) som används för omvandling av datatyper och *RPC conventions* som är regler för meddelandesystemet. [87]

## SOAP Message

När SOAP-meddelanden utbytts finns det två parter som är inblandade; en sändare och en mottagare. En begäran (*request*) skickas från en klient och servern skickar tillbaka ett svar (*response*). När ett meddelande anländer finns det tre saker som applikationen måste genomföra. [88]

1. Identifiera alla delar på meddelandet som är ämnade för applikationen.
2. Se efter att alla de obligatoriska delarna stöds av applikationen och bearbeta dem därefter. Om det inte skulle vara så kan applikationen slänga meddelandet.
3. Om applikationen som behandlat meddelandet inte är den slutgiltiga destinationen tas alla delar som identifierats bort och skickas sedan vidare.

---

[83] McLaughlin, 23.

[84] Ibid.

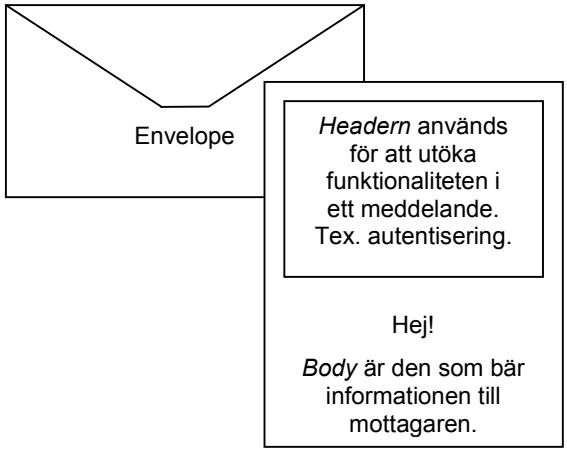
[85] Ibid.

[86] Vivek Chopra, Zaev Zoran, Gary Damschen, Chris Dix et al. (2001). *Professional XML Web Services* [smakprovskapitel -online]. Tillgänglig: http://www.vbip.com/books/1861005091/chapter_5091_01.asp [2002-10-01]

[87] Cerami, 50.

[88] Ibid., 52.

För att kunna behandla ett meddelande krävs det att processorn i SOAP applikationen förstår växelmönstret för att på så sätt veta vad den ska göra. Ett SOAP-meddelande består av tre delar; ett obligatiriskt `envelope`, en frivillig `header` och en obligatorisk `body`.[89] Figur 13 Ger en närmare förklaring över meddelandets delar.




*Figur 13 Förklaring över de olika delarna i ett meddelande*

## Envelope

Ett *envelope* packar ihop ett meddelande och beskriver hur meddelandet ska behandlas samt dess innehåll Den anger också meddelandets mottagare.

SOAP *encodingStyle attribute* är ett globalt attribut bestående av en URI, som används för att kunna definiera datatyperna som används i ett meddelande. Attributet kan användas av vilken element som helst och den kommer att appliceras till elementets innehåll och dess barn-element. När meddelandet sedan packas upp kan mottagaren följa samma regler som avsändaren. Ett SOAP-meddelande har ingen *default encoding.* [90] Figur 14 visar ett `encodingStyle` attribut.

```
<SOAP ENV:Envelope
SOAP ENV:encodingStyle="http://schemas.xmlsoap.org.soap">
```

*Figur 14 Det globala attributet anger reglerna som används vid omvandlingen av ett meddelande.*

SOAP definierar inte någon traditionell versionshanteringsmodell som kan användas mellan olika versioner. Som lösning på problemet infördes att varje meddelande måste innehålla ett namespace i sitt `envelope` för att ange version 1.1. [91] Detta namespace är ett URI och ser ut som följande:

---

[89] Don Box, David Ehnebuske, Gopal Kakivaya et al. (2000) *Simple Object Access Protocol (SOAP) 1.1, W3C Note* [online]. Tillgänglig: http://www.w3.org/TR/SOAP/ [2002-09-15]

[90] Box, Ehnebuske, Kakivaya et al, http://www.w3.org/TR/SOAP/

[91] Ibid.

```
<SOAP ENV:Envelope
xmlns:SOAP ENV = "http://schemas.xmlsoap.org/soap/envelope"
SOAP ENV:encodingStyle="http://schemas.xmlsoap.org.soap">
```

*Figur 15 Ett meddelande som innehåller en enkel versionshantering genom att lägga till ett namespace.*

Om ett meddelande som tas emot består av ett annat än den ovannämnda namespace, hanterar den mottagande applikationen meddelandet som felaktigt och skickar tillbaka ett felmeddelande av typen `VersionMismatch` kod till avsändaren och meddelandet slängs. [92]

## Header

Genom att använda en *header* kan funktionaliteten utökas i ett meddelande. En header består av sådan information som berör säkerhet och routing. Den måste inte förekomma i ett SOAP-meddelande utan är frivillig. Headern inkapslar denna utökning av utan att behöva påverka strukturen i SOAP. Det finns ett antal attribut som kan användas i samband med hanteringen av en header. Dessa är `Actor attribute` och `mustUnderstand attribute`.

### Actor attribute

Ett SOAP-meddelande förflyttar sig från en avsändare till en mottagare. Den passerar ett antal mellanhänder efter sin färd mot destinationen, dessa mellanhänder kallas för noder. En nod är en applikation som har förmågan att både ta emot och skicka vidare ett meddelande, både noderna och mottagaren identifieras via ett URL. [93]

Alla delar som ingår i ett SOAP-meddelande är inte ämnade för den slutlige mottagaren utan för noderna efter vägen. Genom att använda sig av actor attribute specificeras vilken del data i headern och bodyn som skall hanteras av noden. För att ange dessa noder används ett antal URI:er. Om denna attribut utelämnas antas det att noden som först tar emot meddelandet också är den slutliga destinationen och mottagaren för meddelandet. [94]

### MustUnderstand attribute

Attributet `mustUnderstand` är global och kan användas för att ange om mottagaren av en header måste förstå och hantera alla element i headern. Värdet för denna attribut är antingen ”0” eller ”1”. Om värdet är ”1” innebär det att mottagaren måste hantera alla element i headern, om den inte gör det måste ett felmeddelande rapporteras. [95]

---

[92] Ibid.

[93] Ibid.

[94] Ibid.

[95] Ibid.

### Body

Informationen som skickas mellan en avsändare och en mottagare finns i en body. Denna element är obligatorisk och får inte utelämnas i ett meddelande. Den återfinns innanför ett `envelope` element och om ett `header` element finns ligger body elementet direkt efter. [96]

```
<SOAP:Envelope>
   <SOAP:Header>
   </SOAP:Header>

   <SOAP:Body>
   </SOAP:Body>
</SOAP:Envelope>
```

*Figur 16 En enkel beskrivning på strukturen hos ett SOAP-meddelande*

### SOAP Fault

Ett element som klarar av att hantera fel eller status information är `fault` elementet. Den återfinns innanför ett SOAP-meddelandes body element. Genom att använda detta element kan ett fel beskrivas. Den består av fyra definierade subelement. [97]

- `Faultcode element` – består av ett värde som identifierar feltillståndet hos en applikation. Värdet är avsedd för maskiner och har inte för avsikt att visas för människor.
- `Faultstring element` – består av information som är avsedd för en användare. Den består av en sträng som kort beskriver felet för användaren och skrivs ut som en dialog.
- `Faultactor element` – anger i vilken nod felet uppstod. Värdet för en faultactor består av ett URI som identifierar noden.
- `Detail element` – avsedd för att innehålla information som är specifikt för en applikation relaterad till en body element. [98]

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <soap:Body>
       <soap:Fault>
           <faultcode>SOAP:Server.productIDDoesNotExist</faultcode>
           <faultstring>The productId 154888 does not
exist</faultstring>
           <faultactor>http://www.w3schools.com/furniture</faultactor>

           <detail>
           <w:error xmlns:w="http://www.w3school.com/">
           <desc> The productId 154888 does not exist.</desc>
           </w:error
           </detail>
       </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

*Figur 17 Visar hur ett meddelande ser ut med felhanteringselement.*

---

[96] Ibid.

[97] Chopra, Zoran et al., http://www.vbip.com/books/1861005091/chapter_5091_03.asp

## SOAP Encoding

SOAP definierar ett antal regler för att omvandla datatyper. Detta gör det möjligt för SOAP-meddelanden att ange specifika datatyper som *int, float, double* och *arrays* med mera. Dessa regler är inte obligatoriska att följas trots att W3C stödjer dess användning, utan kan ersättas med ett annat *encoding schema* om nödvändigt. Dessa datatyper kan vara av typen *scalar* eller *compound*, alltså enkla datatyper eller sammansatta datatyper. [99]

Enkla datatyper består av exakt ett värde, det kan vara ett efternamn, pris eller en produktbeskrivning. För att två applikationer ska kunna förstå datatyperna i ett meddelande är det viktigt att dessa är överens om hur datatyperna ska se ut. Genom att använda samma *encoding schema* löses detta problem.[100]

Sammansatta datatyper består av flera värden som en order eller en lista med aktier. Sammansatta datatyper är även indelade i två typer; *arrays* och *structs*. Arrays innehåller flera värden där varje värde identifieras unikt av talet på positionen, istället för att använda ett namn. Det enda som är viktigt är innehållet och dess ordning i arrayen. Structs består också av flera värden, men varje värde är istället unikt identifierad av namnet på en post. [101]

## SOAP via HTTP

SOAP är inte bundet till något tranport protokoll, SOAP kan överföras via *Simple Mail Transport Protocol* (SMTP), *File Transport Protocol* (FTP) eller något annat protokoll. Dock har SOAP specifikationen endast detaljer för hur HTTP fungerar och HTTP förblir den populäraste alternativet för transport av SOAP.[102] HTTP är webbens protokoll och eftersom det är en accepterad och standardiserad standard är det ett bra val för SOAP.

SOAP begäran (request) skickas via HTTP request och svaren via HTTP response. Fastän SOAP begäran kan skickas via HTTP GET, är det endast HTTP POST som finns detaljerad i specifikationen. Dock behövs både request och response för att sätta innehållet i SOAP till text/xml , detta är ett krav eftersom ett SOAP-meddelanden använder sig av XML. HTTP POST skickar block av data till en särskild URI. I fallet för SOAP är det SOAP-meddelandet som är datablocket. [103]

En klient måste ange ett `SOAPAction` header, som är ett URI på en specifik server som används för att ange meningen med den skickade begäran. Detta gör det möjligt att snabbt och enkelt avgöra ett SOAP-meddelandes avsikt och dess slag utan att behöva undersöka den närmare. [104]

Svar på SOAP-meddelanden som går via HTTP följer samma *status kod* som gäller för HTTP. Om meddelandet kommit fram utan problem är status koden `200 OK` annars består koden av ett kodtal samt typen av fel som inträffat, till exempel: `500 Internal Server Error`.[105]

---

[98] Ibid.

[99] Cerami, 56.

[100] Ibid., 56-58.

[101] Ibid., 56-59.

[102] Ibid., 60.

[103] Chopra, Zoran et al., http://www.vbip.com/books/1861005091/chapter_5091_05.asp

[104] Cerami, 61.

[105] Ibid., 62.

### *WSDL (Web services Description Language)*

WSDL är en specifikation som definierar hur en Web service beskrivs med hjälp av XML. Ett WSDL-dokument beskriver vilka funktioner en Web service erbjuder, hur den kommunicerar och var den kan nås. WSDL tillhandahåller en strukturerad mekanism för att beskriva funktionerna som en Web service kan genomföra, formatet på meddelanden som den kan hantera, protokollen som den stödjer och åtkomsten av en instans i en Web service. [106]

WSDL utvecklades av *IBM* och *Microsoft*. WSDL 1.1 specifikationen överlämnades till W3C i mars 2001 för standardisering. Den har dock ännu inte nått standardnivå. [107] Trots att teknologin fortfarande befinner sig under utveckling tillhandahåller nästan alla produkterna stöd för WSDL.

WSDL är oberoende av plattform och programmeringsspråk, den används främst för att beskriva tjänster. Varje Web service som publiceras på Internet medföljer ett WSDL-dokument som listar dess tjänster och instruktioner för dess användning. Ett WSDL-dokument fastställer typen på meddelanden som en Web service kan skicka och ta emot samt specificerar vilken data den kallande applikationen måste tillhandahålla för att Web servicen ska kunna genomföra sin uppgift. WSDL språket och dokumentet är endast avsett för applikationer och inte för människor, därför är det bara applikationer som förstår dess innehåll. [108]

#### Strukturen för WSDL

Ett WSDL-dokument definierar *tjänster* som en samling ändpunkter i ett nätverk, *portar*. Dessa ändpunkter och meddelanden är avskilda från den riktiga nätverksplaceringen, detta tillåter återanvändning av abstrakta definitioner: *meddelanden* är abstrakta beskrivningar av data som utväxlas och *porttyper* är en abstrakt samling av *funktioner*. Protokollet som används och specifikationen av dataformatet för en porttyp bildar en återanvändningsbar *binding*.
En *port* uppstår av en ansluten nätverksadress med en återanvändningsbar *binding* och en samling av *portar* skapar i sin tur en tjänst. [109] WSDL använder sig av ett antal element, dessa är:

`<definitions>` – namnet på WSDL-dokumentet.

`<types>` – datatyperna som används mellan klienten och servern.

`<message>` – anger meddelandets format.

`<portType>` – definierar ett set funktioner.

`<binding>` – kartlägger funktioner och meddelanden i en `portType` till ett protokoll och en specifikation för dess dataformat.

`<port>` – anger adressen för en `binding`.

`<service>` – en samling portar som bildar en tjänst.

Innan detaljer för varje element som ingår i WSDL beskrivs visas ett exempel som visar hur ett WSDL-dokument med sina element kan se ut. Exemplet som visas i Figur 18 är hämtat ur WSDL specifikationen.

---

[106] Systinet, http://www.systinet.com/download/332fd7be0e8021ff85e15a95d73a7d96/wp_Introduction_to_Web_Services.pdf

[107] Ibid.

[108] Deitel & Deitel, http://www.deitel.com/newsletter/20020523/articles/WebServices_SOAP-WSDL-UDDI.pdf

[109] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana (2001) *Web Services Description Language (WSDL) 1.1 W3C note 15 march 2001*. [specifikation]. Tillgänglig: http://w3c.org/TR/wsdl . [2002-09-29]

```
<?xml version="1.0"?>

   <definitions name="StockQuote"

   targetNamespace="http://example.com/stockquote.wsdl"
   xmlns:tns="http://example.com/stockquote.wsdl"
   xmlns:xsd1="http://example.com/stockquote.xsd"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns=" http://schemas.xmlsoap.org/wsdl/>

   <types>
       <schema targetNamespace="http://example.com/stockquote.xsd"
           xmlns="http://www.w3.org/2000/10/XMLSchema">
           <element name="TradePriceRequest">
               <complexType>
                   <all>
                           <element name="tickerSymbol"
type="string"/>
                   </all>
               </complexType>
           </element>
           <element name="TradePriceRequest">
               <complexType>
                   <all>
                           <element name="price" type="float"/>
                   </all>
               </complexType>
           </element
       </schema>
   </types>

   <message name="GetLastTradePriceInput">
       <part name="body" element="xsd1:TradePriceRequest"/>
   </message>

   <message name="GetLastTradePriceOutput">
       <part name="body" element="xsd1:TradePrice"/>
   </message>

   <portType name="StockQuotePortType">
       <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
       </operation>
   </portType>

   <binding name="StockQuoteSoapBinding"
type="tns:StockQuotePortType">
       <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
       <operation name="GetLastTradePrice">
       <soap:operation soapAction="http://example.com/GetLastTradePrice">
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
       </operation>
   </binding>
```

```
<service name="StockQuoteService">
   <port name="StockQuotePort" binding="tns:StockQuoteBinding">
       <soap:address location="http://example.com/stockquote"/>
   </port>
   </service>

</definitions>
```

*Figur 18 En WSDL-dokument som beskriver en enkel Aktietjänst. Tjänster stödjer en enda funktion kallad* `GetLastTradePrice`*, som tar emot en sträng och lämnar ifrån sig ett*

WSDL-dokumentet i Figur 18 beskriver den information som behövs för att kunna sätta upp en kommunikation mot den Web service som dokumentet beskriver. Det första som anges i dokumentet är de typer som används till parametrar och returvärde för dokumentet. I `types` elementet går det att se två olika definitioner, nämligen `TradePriceRequest` och `TradePrice` . Där definieras `TradePriceRequest` som en sträng som innehåller värdet på en aktie, genom att använda elementet `tickerSymbol` är det möjligt att omvandla strängen till ett riktigt tal. `TradePrice` är ett decimaltal som representerar en priset på aktien.[111]

Efter typdefinitionerna kommer två stycken `message` element. Första elementet beskriver parametrarna för anropet och det andra returvärdet från anropet. p`ortType` elementet och `message` elementet är nära kopplade till varandra. Det går inte att se vad `message` elementen innebär innan `portType` elementet fyllts in. När `portType` elementet studerats blir resultatet att det första `message` elementet (`TradePriceRequest`) representerar en inparameter till anropet och det andra `message` elementet (`TradePrice`) motsvarar returvärdet.[112]

När anropet är definierat anges det också hur meddelandet ska skickas, det innebär vilket kommunikationsprotokoll som skall användas. Detta anges i `binding` elementet och i detta exempel är det meningen att SOAP ska användas. Det som inte har nämnts är dock var Web servicen är belägen. I elementet `service` kan det utläsas att Web servicen finns på adressen http://example.com/stockquote .[113]

### Definitions

WSDL består av ett rotelement som heter `definitions`. Den definierar namnet på Web servicen och deklarerar ett antal namespaces (`targetNamespace`) som används på hela dokumentet. Den innehåller alla beskrivna element som Web servicen består av. [114] Figur 19 visar hur `definitions` elementet ser ut i dokumentet.

```
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
   xmlns:tns="http://example.com/stockquote.wsdl"
   xmlns:xsd1="http://example.com/stockquote.xsd"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns=" http://schemas.xmlsoap.org/wsdl/ >
```

*Figur 19 Syntaxen för* `definitions` *elementet.*

---

[110] Christensen et al., http://www.w3.org/TR/wsdl

[111] Ibid.

[112] Ibid.

[113] Ibid.

[114] Cerami, 120.

### Types

Detta element används för att kunna definiera egna datatyper, sådana som är relevanta för meddelandet. WSDL använder sig av det rekommenderade schema som finns definierad i *W3C XML schema part 2* som regel för att definiera dessa datatyper, schemat stöder både enkla och sammansatta datatyper. Det finns inga krav på att detta schema används utan en egen datatypschema kan användas istället, detta eftersom det är omöjligt för ett enda schema att innehålla alla varianter av typer som finns. Därför är det viktigt att ange vilket schema som används. SOAP verktyg använder informationen i `types` elementet för att koda och avkoda data i SOAP-meddelanden. [115] Figur 20 visar syntaxen för `types` elementet på ett förenklat sätt.

```
<types>
   <xsd:schema. . . . />
</types>
```

*Figur 20 Syntax för att definiera `types` element.*

### Message

Detta element definierar formatet på meddelandet. Meddelanden används som input och output för funktioner. Ett meddelande kan bestå av en eller flera logiska delar, kallad parts. Var och en av dessa är kopplad till en typ i typschemat. Dessa parts beskriver endast hur meddelandet är uppbyggt, men inte dess innehåll. [116] Namnet på ett `message` element måste ha ett unikt namn, detsamma gäller för varje `part` element. [117] Figur 21 visar syntaxen för `message` elementet.

```
<message name="nmToken">
   <part name="nmToken" element="qname"? type = "qname"?/>
</message>
```

*Figur 21 Syntaxen för att definiera `message` element.*

### PortType

Detta element definierar funktioner som använder sig av `message` elementen. Namnet för `portType` består av ett unikt namn bland alla `portType` element som kan förekomma. Ett `portType` element kan stödja fyra olika primitiva funktioner.[118]

- *One-way* – Tjänsten tar endast emot ett meddelande från en klient. Därför har `operation` elementet ett enda `input` element.[119] Figur 22 visar ett exempel.

```
<portType...>
   <operation name="nmToken">
       <input name="nmToken"? message="qname"/>
   </operation>
```

*Figur 22 Syntax för ett `portType` element med One-way funktionen.*

---

[115] Christensen et al., http://www.w3.org/TR/wsdl

[116] Cerami, 124.

[117] Christensen et al., http://www.w3.org/TR/wsdl

[118] Ibid.

[119] Ibid.

- *Request-response* – Tjänsten tar emot ett meddelande och skickar tillbaka ett svar. Därför består `operation` elementet av både en `input` element och en `output` element. För att kapsla in fel skulle ett `fault` element kunna läggas till.[120] Figur 23 visar ett exempel.

```
<portType>
   <operation name=nmToken>
       <input name="nmToken"? message="qname"/>
       <output name="nmToken"? message="qname"/>
       <fault name="nmToken"? message="qname"/>
   </operation>
</portType>
```

*Figur 23 Syntax för ett `portType` element med Request-response funktionen.*

- *Solicit-responce* – Tjänsten skickar ett meddelande och tar emot ett svar. Detta gör att operation elementet har ett output element följt av ett input element. Även här kan ett fault element delges.[121] Figur 24 visar ett exempel.

```
<portType>
   <operation name=nmToken>
       <output name="nmToken"? message="qname"/>
       <input name="nmToken"? message="qname"/>
       <fault name="nmToken"? message="qname"/>
   </operation>
</portType>
```

*Figur 24 Syntax för ett `portType` element med Solicit-response funktionen.*

- *Notification* – Tjänsten skickar ett meddelande. Därför består `operation` elementet av endast ett `output` element.[122] Figur 25 visar ett exempel.

```
<portType>
   <operation name=nmToken>
       <output name="nmToken"? message="qname"/>
   </operation>
</portType>
```

*Figur 25 Syntax för ett `portType` element med Notification funktionen.*

[120] Ibid.

[121] Ibid.

[122] Ibid.

### Binding

Detta element tillhandahåller specifika detaljer på hur ett `portType` element verkligen kommer att överföras. Det anger vilket kommunikationsprotokoll som kommer att användas för anropet. Själva `binding` elementet anger ett namn som är unikt och ett `type` attribut, där type attributet refererar till `portType` elementet. För att konkret kunna ange ett meddelandes input, output och fel kan `binding` elementet utökas genom att använda ett `binding extensibility` element. Ett `binding` element får endast specificera ett protokoll och den behöver inte specificera någon adress information. [123] Figur 12 visar hur ett `binding` element ser ut.

```
<binding name="nmToken" type="qname">
   < -- extensibility element (1)-- >

   <operation name="nmToken">
       < -- extensibility element (2)-- >
       <input>
           < -- extensibility element (3)-- >
       </input>
       <output>
           < -- extensibility element (4)-- >
       </output>
       <fault>
           < -- extensibility element (5)-- >
       </fault>

   </operation>
</binding>
```

*Figur 26 Syntaxen för ett `binding` element.*

### Port

Ett `port` element definierar en adress för ett `binding` element som anger via vilket protokoll meddelandet ska skickas. Elementet består av två attribut; *namn* som är unikt för dokumentet och *binding* som refererar till `binding` elementet som `port` elementet är kopplat till. Elementet `binding extensibility` kan användas för att specificera en adress för `port` elementet. [124] Figur 27 visar `port` elementet på ett förenklat sätt.

```
<service. . .>
   <port name="nmToken" binding="qname">
       < -- extensibility element (5)-- >
   </port>
</service>
```

*Figur 27 Syntaxen för `port` elementet.*

[123] Ibid.

[124] Christensen et al., http://www.w3.org/TR/wsdl

**Service**

Detta element är det som specificerar tjänstens fysiska position. Den består av ett eller flera `port` element. Namnet för `service` elementet är unikt för hela WSDL-dokumentet. De portar som anges kommunicerar inte med varandra. En tjänst kan ha flera `port` element som är kopplade till samma `portType` element skiljer sig via `binding` elementet eller adressen. [125] Figur 28 visar ett `service` element.

```
<service name="mnToken">
   <port. . . />
</service>
```

*Figur 28 Syntaxen för `service` elementet.*

## *UDDI (Universal Discovery Description and Integration)*

UDDI 1.0 uppkom i september 2000 som ett resultat av samarbetet mellan *Microsoft*, *IBM* och *Ariba*. Sedan den offentliggjordes har den växt till att idag omfatta mer än 280 företag. I juni 2001 publicerades UDDI 2.0 av samma företag och den 19 juli 2002 publicerades UDDI 3.0. [126]

UDDI är tänkt att vara en allmän katalog- och integrationstjänst som lagrar och ger information om tillgängliga Web services. Det tillåter användare att publicera och söka efter Web services på Internet. När en förfrågan sänds från en applikation svarar UDDI-registret med att ge information som omfattar upptäckta tjänster, deras status och tillgängligheten av andra tjänster som stämmer överens med förfrågan. UDDI ger därmed företag en världsomspännande webbaserad distribuerad katalog som de kan publicera sina tjänster i. [127]

Företag kan lagra sin information antingen i en privat UDDI-katalog som endast är tillgänglig för godkända företag eller i allmänna UDDI-kataloger där alla får tillgång till informationen. Strukturen för UDDI kan liknas vid en telefonbok. Data som samlas inuti UDDI är indelad i tre kategorier; *Vita sidor* som innehåller allmän information om ett företag, exempelvis dess namn, adress, telefon och annat som kan vara av intresse. *Gula sidor* som innehåller information om verksamheten där data klassificeras för varje företag eller tjänst. *Gröna sidor* Innehåller tekniskt information om en tjänst. Denna information kan vara programmeringsspråket plattform med mera. Den har också en adress för att kunna anropa en tjänst. [128] Den tekniska arkitekturen för UDDI består av tre delar; *UDDI data model* (datastrukturen) som anger hur företag och tjänster beskrivs, *UDDI API* som beskriver hur data publiceras och söks samt *UDDI cloud services* som tillhandahåller information om hur en nod i UDDI registret skall hanteras. För att kunna kommunicera med UDDI använder man SOAP-meddelanden. [129].

### UDDI Data Model

All information som finns i UDDI är skriven i XML. Anledningen till att man valt XML är densamma som för de andra standarderna, XML är plattformsoberoende. XML gör det också möjligt att använda sig av en hierarki av element och det är lätt att validera och lägga till en

---

[125] Ibid.

[126] Ibid.

[127] Cerami, 157-159.

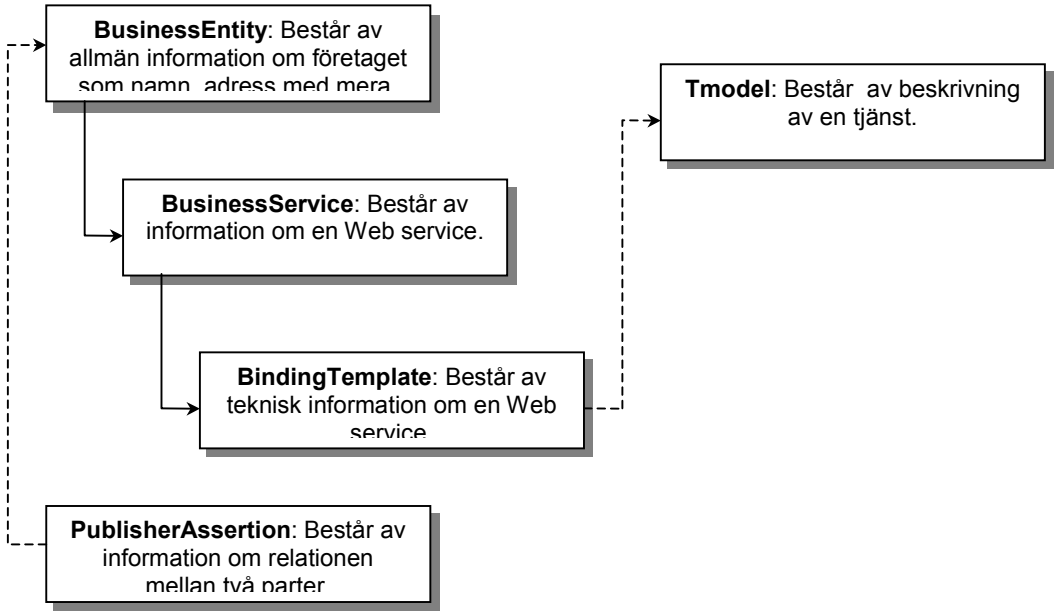
[128] UDDI Organisation (2000). *UDDI Technical White Paper.*[pdf]. Tillgänglig: http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf [2002-10-15]

[129] Cerami, 161.

mängd olika datatyper. UDDI XML schema anger fem olika datatyper Uppdelningen har gjorts för att lättare kunna lokalisera och förstå informationen som används. Dessa är:

- `<businnesEntity>` innehåller företagsinformation.
- `<businessService>` innehåller information om tjänsten.
- `<bindingTemplate>` innehåller information om binding.
- `<tmodel>` innehåller information om specifikationer för en tjänst.
- `<publisherAssertion>` innehåller information om relationerna mellan två parter, försäkrad av någon av dem.

En närmare beskrivning av dessa element kommer men först visa Figur **29** hur dessa element fungerar tillsammans.




*Figur 29 Visar strukturen för UDDI-registret. Heldragna pilar syftar till instanser och den streckade pilen till en referens.*

### businessEntity

Detta element består av den kända informationen som ett företag lämnar ut som en beskrivning av sig själv och tjänsterna som den erbjuder. Denna information finns på de gula sidorna av ett UDDI-register. [130] Figur 30 visar strukturen på ett `businnessEntity` element.

```
<element name="businessEntity" type="uddi:businessEntity"/>

<complexType name="businessEntity">

   <sequence>
       <element ref="uddi:discoveryURLs" minOccurs="0"/>
       <element ref="uddi:name" maxOccurs="unbounded"/>
```

[130] David Ehnebuske, Dan Rogers, Claus von Riegen (2002*). UDDI Version 2.03 Data Structure Reference, UDDI Published Specification, 19 July 2002.* [online]. Tillgänglig: http://www.uddi.org/pubs/DataStructure-V2.03-Published-20020719.pdf [2002-10-15]

```
      <element ref="uddi:description" minOccurs="0"
maxOccurs="unbounded"/>
      <element ref="uddi:contacts" minOccurs="0"/>
      <element ref="uddi:businessService" minOccurs="0"/>
      <element ref="uddi:identifierBag" minOccurs="0"/>
      <element ref="uddi:categoryBag" minOccurs="0"/>
   </sequence>

   <attribute name="businessKey" type="uddi:businessKey"
use="required"/>
   <attribute name="operator" type="string" use="optional"/>
   <attribute name="authorizedName" type="string" use="optional"/>

</complexType>
```

*Figur 30 Exempel på strukturen för* `businessEntity`.

`discoveryUrls` är ett valfritt element som består av en lista med URL:er. Den tilldelas alltid en URL som returnerar `businessEntity` elementet. `name` elementet är obligatorisk och består av namn på `businessEntity` som kan läsas av människor. Den innehåller namnet på personen som registrerat tjänsten i UDDI-registret. Det är tillåtet med flera `name` element. `operator` elementet innehåller namnet på operatören och `authorizedName` elementet innehåller namnet på personen som publicerade den data som återfinns i ett `businessEntity` element.

Elementet `description` är ett valfritt element som består av en enda eller flera korta beskrivningar av ett företag. Den kan innehålla en beskrivning för varje språk. Genom att använda sig av `contacs` elementet kan en lista med information om alla kontakter göras, men elementet är valfritt att användas. Det går även att göra en lista med en beskrivning av tjänsterna genom att använda `businessServices` elementet som också är valfritt att använda. `identifierBag` elementet består av en lista med identifierare för företaget för att lättare kunna söka i registret, även denna element är valfritt att användas. För att kunna lista företaget till en kategori används elementet `categoryBag`. Den listar ett företags produkter, industrityp på företaget och dess geografiska position. `businessKey` elementet är ett unikt identifierare för den specifika `businessEntity` strukturen. [131]

**businessService**

Detta element beskriver en tjänst. Varje businessService är ett barn till businessEntity. Detta fastställs genom att undersöka värdet i businessKey som kopplar ihop dessa två. När företag vill återanvända eller dela tjänster kan den använda sig av businessService elementet genom att använda en projektion av den redan publicerade businessService. Vid en uppdatering av en businessService som genomförs av businessEntity kommer automatiskt att gälla även för den projicerade elementet. Det elementet som projicerar och den som blir projicerad ska innehålla samma businessService nycklar[132]. Figur 31 visar strukturen för `businessService` elementet

```
<element name="businessService" type="uddi: businessService"/>
<complexType name="businessService">

   <sequence>
      <element ref="uddi:name" maxOccurs="unbounded"/>
      <element ref="uddi:description" minOccurs="0"
maxOccurs="unbounded"/>
      <element ref="uddi:bindingTemplates" minOccurs="0"/>
      <element ref="uddi:categoryBag" minOccurs="0"/>
   </sequence>
```

[131] Ibid.

[132] Ibid.

```
   <attribute name="serviceKey" type="uddi:serviceKey"
use="required"/>
   <attribute name="businessKey" type="uddi:businessKey"
use="optional"/>

</complexType>
```

*Figur 31 Exempel på strukturen för* `businessService`.

Elementet för `businessService` består av två annorlunda element; `bindingTemplates` och `serviceKey` elementerna. Övriga element är samma som återfinns på `businessEntity` elementet. Elementet `bindingTemplates` innehåller teknisk information om en tjänst som är kopplad till ett företag. För att kunna komma åt ett `businessService` tilldelas den ett unikt värde i `serviceKey` elementet. [133]

**bindingTemplate**

Detta element består av information om åtkomsten av en tjänst genom att beskriva var och hur tjänsten kan nås. Varje `bindingTemplate` struktur har en `businessService` som förälder som i sin tur har en `businessEntity` som sin förälder. [134] Figur 32 visar strukturen för `bindingTemplate`.

```
<element name="bindingTemplate" type="uddi: bindingTemplate"/>
<complexType name="bindingTemplate">

   <sequence>
       <element ref="uddi:description" minOccurs="0"
maxOccurs="unbounded"/>
       <choice>
           <element ref="uddi:accessPoint"/>
           <element ref="uddi:hostingRedirector"/>
       </choice>

       <element ref="uddi:tModelIntanceDetails"/>

   </sequence>

   <attribute name="serviceKey" type="uddi:serviceKey" use="optional
"/>
   <attribute name="bindingKey " type="uddi: bindingKey "
use="required"/>

</complexType>
```

*Figur 32 Exempel på strukturen för* `bindingTemplate`.

Detta element består av tre speciella element; `accessPoint` elementet är en textfält som används för att peka på ingångsporten i form av ett URL-typ för att kunna komma åt en särskild Web service. Dessa typer kan vara en URL, en *e-postadress*, en *ftp-adress* eller till och med ett *telefonnummer*. Om det inte finns någon `accessPoint` element kan `hostingRedirectory` elementet användas istället. Denna element har en `bindingKey` attribut som anger omdirigeringsreferens till ett annat `bindingTemplate`. Själva beskrivningen av hur Web servicen ska användas finns i `tModelInstanceDetails` elementet. [135]

---

[133] Ibid.

[134] Cerami, 166.

[135] Ehnebuske et al., http://www.uddi.org/pubs/DataStructure-V2.03-Published-20020719.pdf

**tModel**

Ett viktigt mål är att kunna beskriva en Web service och sedan göra denna beskrivning meningsfull för att kunna använda den vid sökning av Web servicen. Ett annat mål är att göra dessa beskrivningar användningsbara nog för att lära ut om interaktionen med en tjänst som en användaren inte vet något om. Denna struktur består av data om data. Informationen som bildar ett tModel struktur är en nyckel, ett namn, en valfri beskrivning och ett URL som pekar på en plats där mer utförlig information om en Web service finns tillgänglig. Elementet beskriver hur en tjänst uppträder, tjänstens specifikation och standarder. Ett viktigt element i denna struktur är `overviewDoc` som kopplar ihop WSDL med UDDI. [136]

```
<element name="tModel" type="uddi:tModel"/>
<complexType name="tModel">

   <sequence>
       <element ref="uddi:name/>
       <element ref="uddi:description" minOccurs="0"
maxOccurs="unbounded"/>
       <element ref="uddi:overviewDoc" minOccurs="0"/>
       <element ref="uddi:identifierBag" minOccurs="0"/>
       <element ref="uddi:categoryBag" minOccurs="0"/>
   </sequence>

   <attribute name="tModelKey" type="uddi: tModelKey "
use="required"/>
   <attribute name="operator" type="string" use="optional"/>
   <attribute name="authorizedName" type="string" use="optional"/>

</complexType>
```

*Figur 33 Exempel på strukturen för* `tModel`.

**publisherAssertion**

Detta element används för att relatera två företag med varandra. För att detta ska fungera måste båda företagen relatera till varandra. Det räcker inte med att bara ett av företagen anser sig vara relaterat till den andra utan båda företagen måste publicera exakt samma information. När detta genomförs blir informationen synlig.[137]

```
<element name="publisherAssertion" type="uddi:publisherAssertion"/>
<complexType name="publisherAssertion">

   <sequence>
       <element ref="uddi:fromKey/>
       <element ref="uddi:toKey"/>
       <element ref="uddi:keyedReference"/>
   </sequence>

</complexType>
```

*Figur 34 Exempel på strukturen för* `publisherAssertion`.

Detta element består av tre barnelement. FromKey, toKey och keyedReference. Dessa element är obligatoriska. Elementet `fromKey` är en unik referens till den första `businessEntity`, `toKey` är en

[136] Ibid.

[137] Ibid.

unik referens till den andra `businessEntity` och `keyedReference` anger typen av relation mellan företagen.[138]

## UDDI API

För att kunna hantera informationen i UDDI-registret finns det ett UDDI API. Det är baserat på SOAP och är indelat i två delar. Det finns *Inquiry API* och *Publisher API*. Vidare är Inquiry API indelat i två delar, en av delarna används för att skapa program som tillåter att söka och bläddra mellan informationen som finns i registret. Den andra delen tar hand om felhanteringen i Web services. Publisher API används för att skapa gränssnitt mot UDDI registret och gör det möjligt för en person att hantera informationen som publiceras. [139]

### Inquiry API

Denna del består av ett antal anrop för att kunna få tillgång till information som återfinns i UDDI-registret. Den gör det möjligt att lokalisera företag och Web services för att sedan gå djupare in i informationen som finns. Funktionerna visas som SOAP meddelanden över HTTP och det behövs inte någon säkerhetsfunktion för att använda funktionerna i Inquiry API. [140] Det finns tre olika steg för att hantera informationen. *Browse pattern, drill-down pattern* och *invocation pattern*. Dessa ska beskrivas närmare nu.

Första steget, *browse pattern*, ger en möjlighet till att upptäcka och undersöka data genom att söka i registret. Denna steg karakteriseras av att informationen är bredd, sedan genomförs en sökning och ett resultat hittas som bidrar till en avgränsning där informationen blir mer specifik. Sökningen genomförs med ett `find_xx` anrop, där `xx` kan bytas ut mot ett ord, exempelvis `find_business`. Denna returnerar alla alternativ som matchar sökningen. Sedan kan ett av alternativen väljas för att gå djupare ner i sökningen och undersöka om företaget erbjuder den tjänsten som söks. Denna sökning genomförs med hjälp av en `find_service` anrop. När den sökta tjänsten hittas kommer också en unik nyckel att hittas tillsammans med tjänsten. Denna nyckel är viktig för vidare steg. [141]

Det andra steget, *drill-down pattern*, använder sig av den unika nyckeln för att få full tillgång till all information om den strukturen som nyckeln är kopplad till. För att göra detta används ett `get_xx` anrop tillsammans med den unika nyckeln. Exempelvis `get_business` tillsammans med det unika nyckeln för `businessEntity`. Genom att den unika nyckeln finns tillgänglig kan användaren få tillgång till all information som finns i någon av strukturerna *businessEntity*, *businessService*, *bindingTemplate* eller *tModel.* [142]

I tredje steget, *invocation pattern*, hämtas informationen från `bindingTemplate` strukturen i dokumentet för att använda tjänsten som erbjuds. Ett `bindingTemplate` erbjuder fakta om gränssnittet och tjänstens URL. Denna information borde lagras i applikationen och användas för att koppla sig till tjänsten varje gång applikationen behöver kommunicera med en instans av tjänsten. När en uppkoppling med den lagrade informationen misslyckas finns möjligheten att koppla sig direkt mot UDDI-registret för att fråga efter ny information av `bindingTemplate strukturen`. Den nya informationen kommer att ersätta den gamla för senare användning. [143]

---

[138] Ehnebuske et al., http://www.uddi.org/pubs/DataStructure-V2.03-Published-20020719.pdf

[139] Cerami, 162-195.

[140] Sean MacRoibeaird (2002). *Universal Description, Discovery & Integration (UDDI).* [online]. Tillgänglig: http://wwws.sun.com/software/xml/developers/uddi [2002-10-18]

[141] Dave Ehnebuske, Barbara McKee, Dan Rogers (2002). *UDDI Version 2.04 API Specification, UDDI Published Specification , 19 July 2002* [online]. Tillgänglig: http://www.uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf [2002-10-18]

[142] Ibid.

[143] Ibid.

**Publishing API**

Detta API erbjuder möjligheter att spara och ta bort någon sparad information. För att få åtkomst till detta API krävs användaren autentiseras och att *Secure Socket Layer* (SSL) används som transport för meddelandena som en säkerhetsåtgärd, detta för att otillåtna användare inte ska kunna ändra den lagrade informationen[144]. Denna autentisering genomförs genom att användaren väljer en UDDI-operatör (webbsida) där de kan publicera och uppdatera sin information. Varje operatör sätter upp sina egna regler för autentisering och det rekommenderas att varje användare endast har en operatör för att inte skapa redundans. All data från denna operatör kommer alltid att sprida sig till andra noder, men uppdatering får endast ske hos operatören.[145]

Det är operatören som godkänner och tillåter publicering av informationen. När information ska sparas eller tas bort krävs det att en *token* används i anropen. Den verifierar en användare och utfärdas av en operatör. För att erhålla ett token görs ett `get_authToken` anrop som kräver ett användarnamn och ett lösenord. När användaren sedan vill ändra eller spara ny information krävs detta token.[146]

För att sedan lagra information behövs en datastruktur och den första som skapas är en `busineesEntity`. För att skapa den används ett `save_business` anrop. Denna anrop kräver den redan genererade token för att godkännas. Samma process gäller för de resterande strukturerna `busineesService`, `bindingTemplate` och `tModel` med ett liknande `save_xx` anrop. För att ta bort elementen utförs en delete_xx anrop istället , där xx står för de olika delarna i datastrukturen. [147]

---

[144] MacRoibeaird, http://wwws.sun.com/software/xml/developers/uddi

[145] Cerami, 182-183.

[146] Ehnebuske et al., http://www.uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf

[147] Ibid.