Handelshögskolan
VID GÖTEBORGS UNIVERSITET
Institutionen för informatik
030317

Web services och dess implementeringsproblematik

Abstrakt

Web services utgörs av ett antal definierade standardteknologier som samarbetar över Internet alternativt inom en verksamhet. För att Web services skall slå igenom och tjänster skall kunna levereras finns det några problem kvar att lösa. Syftet med uppsatsen har varit att belysa hur Web services fungerar och vad begreppet innebär som ny teknik samt undersöka vilka problem som kan uppstå vid implementeringen av Web services. Vidare hade vi som syfte att undersöka eventuella förslag på lösningar till dessa problem. Arbetet har baserats på fyra valda kriterier, interoperability, integration, security och architecture. Dessa låg senare till grund för vår litteraturstudie, enkät och våra intervjuer, där det framkom att problem av olika karaktär finns, men att huvudsakliga lösningar på dessa förväntas utarbetas av mjukvaruleverantörerna genom gemensamma standarder. Då vi upplevde att företagen i vår enkät såg relativt få implementeringsproblem i dagsläget, drog vi slutsatsen att detta troligtvis beror på att tekniken inte används i så stor utsträckning och att kompletta standarder ännu inte finns.

Nyckelord: Web services, implementering, integrering, interoperability.

Författare: Charlotte Rosander, Matilda Westlindh
Handledare: Faramarz Agahi
Examensarbete II, 10 poäng

Innehållsförteckning

1 Introduktion

I detta första kapitel presenteras en problembakgrund till uppsatsen som mynnar ut i en problemformulering. Vidare kan man här läsa om uppsatsens syfte, avgränsningar och intressenter. Slutligen finns en disposition över uppsatsens fortsatta innehåll.

1.1 Problembakgrund

Web services som term lanserades av Hewlett-Packard år 2000 och har under åren som gått uppmärksammats mycket i IT-media. Tekniken som används bygger på en hel del gamla trender, det är själva begreppet som är nytt (Höij, 2002). Web services utgörs av ett antal definierade standardteknologier som samarbetar över Internet alternativt inom en verksamhet. Applikationer kan genom användandet av Web services kommunicera med varandra oberoende av plattform och programspråk. Web services grundidé är att leverera integration som en tjänst [1].

Skillnaden mellan Web services och tidigare tekniker för att integrera system så som EDI, CORBA och COM, är att Web services ska bygga på öppna standarder. Många företag använder redan idag Web services, men det är långt kvar till den breda användningen (Nordner, 2002). För att Web services skall slå igenom och tjänster skall kunna levereras finns det några problem kvar att lösa. Dessa anses vara att ena de stora programleverantörerna kring en gemensam standard, att kunna erbjuda affärsnytta samt att skapa tillräcklig säkerhet (Wallström, 2002). När det gäller utvecklingen av en gemensam standard så finns det två olika läger kring området, vilka försöker hitta lösningar. På ena sidan står IBM och Microsoft och på den andra sidan står Sun. Det är först när dessa stora program leverantörer lyckas enas som Web services kan ta verklig fart (Danielson, 2003).

Det stora genombrottet för Web service har alltså inte kommit ännu men intresset bland företag är stort och många företag funderar på att satsa på tekniken. Detta då det finns fördelar som förenkling vid integration samt att utvecklingsarbetet blir smidigare och snabbare. I början kan en Web service tillämpning ta längre tid att utveckla då koden måste vara mer flexibel för att kunna utnyttjas i flera omgivningar och på flera plattformar. Men när allt väl fungerar kan företagen få bättre fungerande tillämpningar och pengar kan sparas. Web service kan användas för att sammankoppla befintliga program med webbaserade program. Kommunikationen sker då genom att informationen bäddas in i XML (Olofsson, 2002).

Vid implementering av ny teknik uppstår ofta många problem. När det gäller Web services återstår flera av dessa problem att lösa. Vilka problem som är de största och hur de ska lösas är av intresse för dem som funderar på att implementera Web services.

1.2 Problemformulering

Utifrån företag och verksamheter som redan idag har anammat Web services har vi antagit att de har stött på problem och att de således erhållit viss kunskap som skulle behövas ta tillvara. Detta för att underlätta för andra verksamheter som är intresserade

av att implementera Web services. Med detta som bakgrund, har vi valt följande frågeställning:

1. Vilka problem kan idag uppstå vid implementeringen av Web services?
2. Vilka eventuella förslag på problemlösningar finns i dagsläget?

1.3 Syfte

Belysa hur Web services fungerar och vad begreppet innebär som ny teknik. Syftet är vidare att undersöka vilka problem som kan uppstå vid implementeringen av Web services i en verksamhet och presentera förslag på lösningar till dessa problem.

1.4 Avgränsningar

Programmeringskod för implementering av Web service kommer inte att innefattas i det här arbetet då den ofta är specifik för den enskilda implementionen. Tekniska specifikationer innefattas inte annat än ur den synvinkeln att de är av vikt för implementeringen av Web services. Vi kommer inte heller att beröra de tjänster och funktioner som erbjuds med Web services, då de fortfarande befinner sig i utvecklingsfasen. Vidare avgränsar vi oss ifrån att studera Hewlett-Packard, Bea, Versign och andra liknande företag som på något sätt är involverade i Web services utvecklingen. Detta då de inte är lika framträdande i utvecklingen som de vi valt att behandla i arbetet.

1.5 Intressenter

Systemutvecklare på IT-avdelningar som står i begrepp att implementera Web services samt de IT-avdelningar som är intresserade och följer utvecklingen för ett eventuellt framtida användande.

1.6 Disposition

Andra kapitlet innefattar metoden och där beskrivs tillvägagångssättet för utförandet av arbetet. Vidare diskuteras här valet av metoder. Slutligen finns ett avsnitt om källkritik där uppsatsens reliabilitet och validitet diskuteras. Tredje kapitlet innehåller uppsatsens teoridel vilken ligger till grund för uppsatsens fortsättning. Inledningsvis finns en bakgrund till Web services som följs av ett avsnitt med bakgrund till implementering och slutligen avslutas det med ett avsnitt om kritiska framgångsfaktorer vid implementering av Web services. Detta avsnitt är uppdelat på kriterierna; interoperability, integration, security och architecture. I fjärde kapitlet redovisas resultatet från litteraturstudien och enkäten. Vidare kan man också läsa om resultatet av intervjuerna. Samma kriterier som finns i teorin ingår också i intervjufrågorna. Femte kapitlet inleds med en diskussion baserad på våra kriterier. Detta följs av våra slutsatser och reflektioner och slutar med förslag på framtida forskning.

2 Metod

I nedanstående kapitel beskrivs hur vi gått tillväga för att skriva uppsatsen. Här beskrivs vårt val av metoder och våra tillvägagångssätt för insamling av material. Slutligen diskuteras valet av metod samt uppsatsens källkritik.

2.1 Metodval och tillvägagångssätt

Inledningsvis gjordes en litteraturstudie för att samla kunskap och få en bra grund till ämnet. Detta första angreppssätt kan betraktas explorativt då vi ville komma fram till en specifik problemformulering som undersökningen skulle kunna bygga på. Den explorativa studien är undersökande och lämpade sig bra för vår förstudie då vi använde oss av flera sätt att samla information. Detta var också ett lämpligt angreppssätt eftersom ämnet var nytt och relativt okänt för oss.

Den fortsatta undersökningen var deskriptiv då vi försökte förklara och utarbeta en noggrann beskrivning av ämnet. Den insamlade kunskapen systematiserade vi i en modellform som sedan teorin skulle baseras på. Vidare begränsade vi oss utifrån detta till att endast undersöka vissa aspekter av intresse mer grundligt.

Inriktningen på en undersökning kan vara induktiv- skapa hypoteser eller deduktiv – testning av hypoteser eller en kombination av dessa s.k. abduktion (Johansson & Lindfors, 1993). Abduktion innebär att man växlar mellan induktion där forskningen går från teori till empiri och deduktion där undersökningen går från empiri till teori. Man kan se vår inriktning som abduktiv då vi gick från teori till empiri till teori. Vi ansåg att detta angreppssätt var lämpligt då området var nytt för oss. Vidare följde vi den hermaneutiska cirkeln då vi studerade texter, tolkade innebörden och studerade texten igen för att åter igen tolka. Hermaneutiken står för tolkningslära, där forskaren är subjektiv och angriper ämnet utifrån sin egen förståelse. Forskarens kunskaper, tolkningar och intryck ses som en fördel i forskningsarbetet. Helhetssyn på problemområdet eftersträvas men bestämd utgångspunkt och slut finns inte. Detta innebär att helheten växer och utvecklas hela tiden (Patel & Davisson, 1994).

Teorin baserade vi på en modell av organisationen W3C där ett antal huvudmål specificerades. Vi omvandlade dem till lämpliga kriterier för vårt arbete. De här kriterierna återkommer sedan även i empirin.

De empiriska undersökningarna som vi gjorde innefattade intervjuer och enkäter och var en viktig del av vår undersökning. Första delen av vår empiriska undersökning var kvantitativ och inleddes med att ett antal högt standardiserade, lågt strukturerade frågor som skickades ut till 100 lämpliga företag vilka vi antog syssla med Web services. Kvantitativa metoder är strukturerade och formaliserade och här är avståndet till informationen längre och mer selektivt för att kunna göra jämförelser inom resultatet. Den kvalitativa och den kvantitativa metoden kombineras ofta (Holme & Solvang, 1997) vilket även vi har valt att göra. Den kvantitativa delen av undersökningen var tänkt att visa på problem och lösningar vid Web services implementationer. Dessutom ville vi veta hur stor kunskapen var bland företag om Web services för att lättare hitta

lämpliga personer att intervjua. Vår förhoppning var att få in många svar som skulle kunna jämföras och på så viss medföra att objektiviteten blev stor. Detta tillvägagångssätt kan tyckas enkelt men svarsfrekvensen och tiden för svarens inkommande var mycket svårbedömd.

Nästa steg i processen var en intervju av det kvalitativa slaget, vilket innebar att vi personligen intervjuade en person på ett företag. Denna intervju var till för att öka vår förståelse för implementeringsproblematiken och för att hitta problem och lösningar. Intervjun gjordes också för att tillförlitligheten förväntas vara större än vid enkätsvaren. Detta då man vid en personlig intervju har lättare för att uppfatta någon korrekt till exempel genom att ställa följdfrågor. När man använder sig av kvalitativa metoder innebär det att man inte formaliserar i någon större utsträckning. Det primära är förståelsen, generaliseringar är inte vad som eftersträvas. Det viktiga är istället att samla information för att få en djupare förståelse av det studerade problemområdet samt att beskriva dess sammanhang. En viktig aspekt är närheten till källan som informationen hämtas ifrån (Holme & Solvang, 1997). Intervjupersonen valdes utifrån de lämpligaste svaren på enkäten. Detta gjordes för att få en djupare förståelse för ämnet. Här användes standardiserade frågor för att få en hög reliabilitet. Vidare gjordes en uppföljningsintervju per telefon med en mycket teknisk kunnig person inom samma företag för att få kompletterande information. Vi hade också som målsättning att intervjua Microsoft, IBM och SUN men då de inte har gått att få tag på någon person som vill ställa upp hos dem var detta tyvärr inte möjligt.

En skillnad mellan enkäter och intervjuer är att intervjuer tar mer tid i anspråk. Detta oavsett om den sker personligen eller per telefon. En annan nackdel är den så kallade ”intervjuareffekten” som innebär att intervjuaren kan påverka personen som intervjuas, tex genom klädsel och kroppsspråk. Nackdelen med enkäter är risken för bortfall (Johansson & Lindfors, 1993). Därför valde vi att använda båda tillvägagångsätten för att minimera risken för att dessa faktorer skulle påverka empiriresultatet i allt för stor utsträckning. Dessutom för att få så mycket relevant information som möjligt.

2.1.1 Källstudier

I litteraturstudien använde vi oss av böcker, vetenskapliga artiklar, artiklar från facktidskrifter, tekniska specifikationer och webbsidor från framförallt IBM, Microsoft och Sun. Litteraturen har hittats på bibliotek eller på webbsidor genom att använda bibliotekets databaser och sökmotorer. Då ämnet är relativt nytt har litteratur i form av böcker varit få och det har ibland varit svårt att få fram relevant information fast många timmar lagts på att söka. Under uppsatsen gång har vi hela tiden stött på nya saker och varit tvungna att gå tillbaka till litteraturen för att söka nytt material. Vi studerade även olika ”newsgroups” på Internet, där kunnigt branschfolk diskuterar kring Web services. Denna information var dock av teknisk karaktär alternativt på för enkel nivå för att det skulle vara intressant för vår studie.

2.1.2 Enkät

Ett antal frågor skickades med mail ut till 100 företag. 40 av företagen var stora koncerner medan 60 stycken var datakonsulter. Vi försökte göra ett så representativt urval som möjligt. De stora företagen som valdes ut var företag som antogs ha en stor IT-avdelning, medan konsultföretagen i första hand valdes ut bland dem som på Internet beskrev sig använda Web services. I andra hand valdes konsultföretag som är relativt stora. Detta urval gjordes då vi bedömde att det skulle ge högre svarsfrekvens och sannolikheten var större för att någon skulle känna till begreppet. Vi var också ute efter att reliabiliteten, det vill säga att tillförlitligheten skulle bli hög.

Vi fick endast in 32 svar varav bara 16 var av användning för vår uppsats vilket kan anses som en relativt låg svarsfrekvens. Vi hade hoppats på en något större svarsfrekvens men lågt deltagande är ett av problemen med enkäter. När man utformar frågor till intervjuer eller enkäter är det viktigt att man tänker på hur frågorna formuleras det vill säga graden av standardisering. I en helt standardiserad enkät ställs samma frågor i samma ordning. Den här typen av enkätsvar är bra för att kunna generalisera och jämföra. En annan aspekt är hur fria frågorna är ur ett tolkningsperspektiv. Helt strukturerade enkäter leder till att personen har lite utrymme för egna tolkningar och utsvävningar. (Patel & Davisson, 1994). Frågorna till vår enkät är kraftigt standardiserade och har låg grad av strukturering. Detta för att vi ville ha kvalitativ information i form av problem och lösningar. Enkäten var utformade enligt följande för att validiteten, det vill säga relevansen för uppsatsen skulle bli hög:

Bakgrund:
1. Hur förhåller sig Ert företag till Web services?
2. Vilka är de främsta argumenten för Web services anser Ni?

Implementeringsproblematiken:
3. Vad upplever Ni är det största problemet vid implementeringen av Web Services?
4. Vilka eventuella andra problem har uppstått vid implementeringen?
5. Hur har Ni löst problemen på Ert företag? (Beskriv kortfattat.)

Övrigt:
6. Övrig information som Ni bedömer vara av intresse för vår undersökning.

2.1.3 Personlig intervju

På förhand hade ett antal frågor bestämts men under intervjuns genomförande kunde vi komplettera med följdfrågor. Intentia som intervjuades valdes ut som det mest lämpliga med enkäterna som underlag. Vi utgick då ifrån vilket företag som verkade ha störst erfarenhet av Web services. Intentia är ett större konsultföretag beläget i Göteborg men med kontor över hela världen. Personen som vi intervjuade på Intentia är PR-chef och har arbetat 2,5 år på företaget. Tidigare forskade han i statistik på Göteborgs universitet. Intervjun ägde rum i mars 2003 och den varade i en och en halv timme. Denna intervju följdes sedan upp med en telefonintervju på 15 minuter med företagets systemarkitekt för att reliabiliteten skulle öka ytterligare. Han är ansvarig för hela företagets utveckling

av Web services och är den person som dragit igång utvecklingen av deras Web services verktyg. Han har arbetat på företaget och med Web services i 4 år. Frågorna som ställdes hade låg struktureringsgrad då de var utformade så personen hade stort utrymme för egna åsikter. Frågorna var utformade så att validiteten skulle bli hög för vår uppsats. Följande frågor ingick i intervjun:

Bakgrund:
1.Vilken är din befattning? Hur länge har du jobbat på företaget? Hur länge har du jobbat i branschen?
2. I hur stor utsträckning använder ni Web services ute hos kunder? Används det bara internt på företag eller externt också?

Implementeringsproblematiken:
3. Hur ser ni på implementeringen av Web services ur ett problemperspektiv när det gäller: Interoperability (samverkan), Integration, Security och Architecture
4. Hur löser ni problematiken bakom ovanstående problem?
5. Berätta om något projekt där ni använt Web services.

2.2 Diskussion av metodval

För att få en mer innehållsrik empiri kunde fler kvalitativa intervjuer gjorts. Det hade förmodligen lett till att fler implementationsproblem kunnat upptäckas. Detta är dock ett mycket tidskrävande arbete och var inte möjligt att genomföra inom vår tidsram. Vidare kunde det gjorts en uppföljning av de kvantitativa enkäterna som mailades ut till företagen, detta för att kanske få in några svar till. Vi bedömde dock att om företag inte visat intresse att svara vid första tillfället var sannolikheten inte stor att de skulle svara vid nästa förfrågan heller, då ämnet trots allt är så pass nytt. Vi valde därför att inte lägga mer tid på det. Ett annat alternativ skulle ha varit att göra en kvalitativ fallstudie ute på ett eller eventuellt flera företag. Realiserat skulle detta innebära att ett eller flera företag som använder sig av Web services skulle studeras. Då flera företag i vår enkät ansåg sig bara känna till ämnet vagt, så tror vi att det skulle bli mycket svårt att hitta företag för att genomföra en fallstudie i dagsläget. Dessutom med den knappa tid som fanns för en undersökning var det inte realiserbart.

2.3 Källkritik

Tillgången till litteratur har varit begränsad då ämnet är nytt och utbudet av böcker i Sverige är litet. Att beställa böcker från USA ansåg vi inte vara rimligt ur ett kostnadsperspektiv. De böcker som dock finns är nyutgivna vilket bör medföra att informationen är aktuell. Den form av litteratur som vi använde oss mest av var information på Internet, vilken var relativt lätt att hitta. De vetenskapliga artiklar som användes bör rimligtvis vara korrekta och vid val av andra webbsidor har vi varit restriktiva och försökt i största möjliga utsträckning välja information från företag som känns relevanta. Vi anser att vi lyckats få en bra bild av ämnet, men naturligtvis måste man alltid vara kritisk till information insamlad från Internet, eftersom man ibland kan vara osäker på källan. Det finns dock fördelar med information från Internet, då denna

information oftast uppdateras mer frekvent. Vi har i största möjliga utsträckning valt att använda information från större, väl renommerade företag och med hänsyn till detta anser vi att tillförlitligheten hos den insamlade informationen är bra och att vi därmed har en hög reliabilitet.

Vi har under undersökningens gång försökt vara kritiska till de källor som vi använt och lagt vikt på att skilja enskilda värderingar och fakta från varandra. Vidare var ämnet nytt för oss så egna erfarenheter har inte inverkat på skrivandet av uppsatsen. Därför anser vi att uppsatsen uppfyller den relevans som krävs för att validiteten ska vara god.

3 Teori

I teorin belyser vi Web services som teknik samt vilka fördelar den för med sig. Vidare presenteras en bakgrund till implementering och de kriterier som vi anser bör beaktas. De stora mjukvaruleverantörernas synsätt på Web services redogörs också kring dessa kriterier.

3.1 Web services definition, termer och organisationer

Begreppet Web services används i flera sammanhang och kan ibland vara missvisande då det inte korrekt beskriver vad det egentligen innebär. Web services betyder inte service i vanlig bemärkelse utan är en teknik som gör det möjligt att koppla ihop funktioner och servicen är de funktioner som kopplas ihop. En service är dessutom en del av en applikation som stödjer kopplingen. Servicen som erbjuds kan både användas intern och externt [2]. Servicen erbjuds över Internet och är skriven enligt en standard specifikation för att fungera tillsammans med andra komponenter i en Web services miljö. Vidare är den språkneutral och plattformsoberoende [3]. I fortsättning kommer vi att använda oss av ordet tjänst istället för service då vi anser att det överensstämmer bättre i svensk text. Nedan följer några definitioner av Web services:

"A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML based messages exchanged via internet-based protocols." [4].

"*Web services are self describing applications that can discover and engage other Web applications to complete complex tasks over the Internet*." [5].

"*Web services standards and technologies allow us to describe and deploy applications or services on a network in a consistent way so that they can be discovered and invoked in a secure and reliable manner. A Web service is an application that uses these standards and technologies.*" - Bob Sutor, chef för e-handels standarder på IBM (Sliwa, 2002).

Web service är en relativt nyuppkommen teknik som vill göra affärsutbyte genom brandväggar säkert. Företag som använder sig av Web services kan kapsla in befintliga affärsprocesser och publicera dem som en tjänst, vidare kan de söka och anlita andra tjänster och utbyta information utanför och inom verksamheten. Web services vill möjliggöra för applikation till applikation interaktion och ta bort oeffektiviteten som människan skapar [6]. Vidare underlättar Web Service för ett företag att vara dynamiskt. Dynamisk ur den aspekten att företaget kan anpassa sig efter kunders behov och lätt ändra den tjänsten de erbjuder kunden i fråga [7].

Web Services presenteras som ett gränssnitt och beskriver en samling operationer som kan nås via ett nätverk genom standardiserade meddelande baserade på XML (eXtensible Markup Language). Gränssnittet är beskrivet genom en standard XML notation, WSDL (Web Service Description Language). Den beskriver alla detaljer som

är nödvändiga att veta för att kunna kommunicera med tjänsterna. Gränssnittet döljer implementationsdetaljer för tjänsterna vilket gör att de kan användas oberoende av vilken hårdvara och mjukvaruplattform det är implementerat på och i vilket programmeringsspråk det är skrivet. Men det är också ett sätt att implementera så att funktionaliteterna definierade av gränssnittet tillhandahålls (Lublinsky, Farrell, 2002).

Web Services kan alltså utvecklas genom att använda vilket programmeringsspråk som helst och på vilken plattform som helst. Kommunikationen sker genom att alla pratar samma språk vilket är XML. Web Service använder XML för att beskriva sitt gränssnitt och för att omkoda sitt meddelande. Men enbart XML räcker inte till en fungerande kommunikation. Applikationerna behöver standard format och protokoll för att samverka med XML. Web Services är för närvarande baserat på tre teknologier; SOAP (Simple Object Access Protocol), WSDL och UDDI (Universal Description, Discovery and Integration) [7].

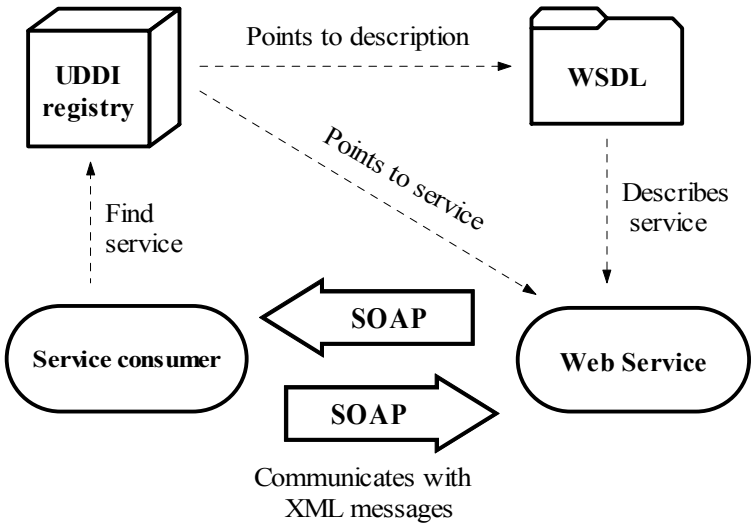

Bild 1: Relationen mellan SOAP, WSDL och UDDI. Källa: [7].

SOAP är ett XML meddelandeprotokoll som tillåter en applikation att sända ett XML meddelande till en annan applikation. Vidare har SOAP en funktion för att kunna identifiera innehållandet i ett meddelande och förklara hur det ska behandlas. Lösa kopplingar stöds också av SOAP vilket innebär att komponenter kan arbeta oberoende av varandra [7].

Automatiserad B2B interaktioner som Web services, kräver en integration av processer. Äldre tekniker som COM (Component Object Model), RMI (Remoted Method Invocation) och CORBA (Common Object Request Broker Architecture) fungerar bra inom ett lokalt nätverk men när det gäller en webbomgivning fungerar de inte, eftersom kopplingarna är kraftiga och beroendet mellan komponenterna starkt. Detta medför att de kommer i konflikt med brandväggar. SOAP:s användning av XML över enkel HTTP gör att brandväggsproblematiken undviks (Lublinsky & Farrell, 2002). SOAP är också mindre komplext än de andra teknikerna vilket gör att starttröskeln är betydligt lägre vid en implementering (Wolter, 2001). Alla objekt eller program i en verksamhet kan anslutas som en Web service genom att paketera dem inuti en SOAP implementation (Travis & Ozkan, 2002).

WSDL är ett XML vokabulär för att beskriva Web Services. Dokumentet beskriver vilka funktionaliteter Web Servicen erbjuder, hur den kommunicerar och hur det finns access till den [7]. WSDL definierar ett XML baserat protokoll för koppling till önskad Web service (Hammer, 2001). Man kan säga att en WSDL fil är ett XML dokument som beskriver ett antal SOAP meddelande och hur de meddelandena utbytes. Notationen som en WSDL fil använder för att definiera ett meddelande format är grundat på XML vilket betyder att det är både programmeringsspråksneutralt samt tillgängligt från olika plattformar (Wolter, 2001).

UDDI tillhandahåller en mekanism för att registrera och kategorisera Web Services som erbjuds och att lokalisera de Web Services som önskas. UDDI är en Web Service i sig själv och kommunicerar med hjälp av SOAP [7].

Centralt för UDDI är att det finns ett register (UDDI Business Registry) vilket är en kunskapsbas som kan utnyttjas av alla på Internet (Berlid, 2001). Registret innehåller tre typer av information; de ”gula sidorna” där verksamhetsbranschen beskrivs, de ”vita sidorna” beskriver verksamhetens namn, adress med mera och slutligen de ”gröna sidorna” som beskriver gränssnittet till tjänsten tillräckligt detaljerat så att användare kan skriva applikationer och använda Web services tjänsterna. Genom UDDI söker man denna information om tjänsterna för att kunna bygga egna applikationer med tjänsterna (Wolter, 2001).

Fördelarna med Web services
Dessa är enligt webservices.org [6]:

Mjukvara kan presenteras som en service - I motsats till mjukvarupaket kan Web services levereras och betalas för som en tjänst och tillåter access från alla plattformar. Web services tillåter inkapsling vilket gör att komponenter kan isoleras så att bara tjänsten exponeras. Detta resulterar i att kopplingen mellan komponenter är mer stabil och flexibel.

Dynamisk affärssamverkan är möjlig - Nya partnerskap kan konstrueras dynamiskt och automatiskt eftersom Web services garanterar total samverkan mellan system.

Tillgängligheten - Affärstjänster kan decentraliseras totalt och distribueras över Internet. Detta leder till att en mängd kommunikationsverktyg får tillgång till de här tjänsterna.

Effektiviteten - Verksamheter kan slipp komplexa, långsamma och dyra mjukvaruutvecklingar vid återanvändning av tjänster och kan istället fokusera på kritiska uppgifter och förbättra dem. Web services är konstruerade med hjälp av ett antal applikationer, detta medför att intern användning lätt kan bli extern användning utan att någon kod behöver ändras. Vidareutveckling vid Web services är naturligt och enkelt och eftersom Web services är deklarerat och implementerat i ett språk möjligt för människor att läsa. Detta medför också att det är mycket enklare att hitta buggar och ”fixa” dem. Det totala resultatet blir en minskning av risker och en mer effektiv utveckling.

Universellt bestämda specifikationer - Web services är baserat på universella specifikationer för utbyte av data, meddelandesändning, sökning av tjänster, gränssnitts beskrivning och affärsprocess styrning.

Integration mellan befintliga system- Större flexibilitet genom ökad integration mellan befintliga fungerande system.

Nya möjligheter på marknaden öppnas - Det kommer att bli större möjligheter för en dynamisk verksamhet och dynamisk utbyta av tjänster [6].

Tekniska utmaningar som återstår

- Säkerhetsstandarder utvecklas fortfarande. Skickandet av digitala identiteter och igenkännande är fortfarande kritiska punkter för Web services. XML-signaturer och XML-kryptering är säkerhetsinitiativ som designats för att utnyttja XMLs fördelar
- Ett XML-dokument kan krypteras i sin helhet och skickas säkert till en eller flera mottagare. Detta kan skötas genom SSL (Secure Sockets Layer) som är en standard för kryptering av webbtrafik. Men i vissa situationer är inte detta tillräckligt. Som exempelvis när olika mottagare av samma information bara har behörighet till att se en del av informationen.
- Kvaliteten på tjänsten förbättras fortfarande. Företag behöver en startpunkt till slutpunkt garanti för pålitligheten, tillgängligheten och tiden.
- Transaktionskoordinering och styrning av systemet. En komplex Web service som använder tjänster från andra verksamheter måste vara robust utifall en tjänst misslyckas eller inte producerar rätt resultat. Styrningen av systemen måste också bli lättare (Dunn, 2003).

Microsoft och IBM med samarbetspartners samarbetar i dagsläget för att få fram specifikationer för en pålitlig, definierad, lättskapad och kopplad affärsprocess för en Web services miljö (Dunn, 2003).

Utvecklingsorganisationer för Web services standarder
W3C (The World Wide Web Consortium) är ett forum för information, handel samt kommunikation och består av cirka 450 medlemsorganisationer från hela världen. W3C utvecklar samverkande teknologier så som specifikationer, riktlinjer, mjukvara samt verktyg och står i spetsen för utvecklingen av webben. Inom W3C finns det en grupp (Web Services Architecture Working Group) som arbetar med att definiera en arkitektur för Web service, och erbjuder en plattform för diskussion och för planering och skapande av ett antal tekniker för design av Web services (Haas, 2003).

WS-I (Web Services Interoperability Organization) är en öppen industriorganisation upprättad för att främja Web services samverkan över plattformar, applikationer och programmeringsspråk (Ferris, 2002). Organisationen består av medlemmar från olika mjukvaruleverantörer, affärsföretags kunder och många andra som är intresserade av Web services [8]. Detta för att bäst kunna svara för kundernas behov som att erbjuda guideledning och stötta resurser för utvecklande av samverkande Web services med mera (Ferris, 2002).

OASIS (Organization for the Advancement of Structured Information Standards) är ett ideellt, globalt konsortium som styr utveckling och antagande av e-handels standarder. OASIS producerar världsvida standarder för säkerhet, Web services, XML bestämmelser, affärstransaktioner, elektronisk publicering, ämnesmappar och samverkan inom och mellan marknadsplatser [9]. En ny kommité håller på att formas av OASIS medlemmar för att fastställa Web services handhavande [10].

3.2 Web services som ny teknik
När det kommer en ny lovande teknik så blir den genast lovordad som en ny ”silver bullet”. ”Silver bullet” är en någonting som är tänkt att lösa alla problem och vara den stora revolutionerande lösningen (DeMarco & Lister, 1990). När XML kom för att par år sedan blev det genast förklarat som den perfekta lösningen på alla problem, vilket dock inte blev fallet. Idag så är det Web services som är den nya ”hypen” som ska lösa alla integrationsproblem och detta på rekordtid samt med befintlig personal (MacVittie, 2002).

Web services har fått mycket uppmärksamhet och har alltså klassats som nästa ”silver bullet”, redo att lösa alla problem (Vinoski, 2001). Det är dock inte någon helt ny teknik utan bygger på en hel del gamla trender (Höij, 2002). Web services är en utveckling av tre stora områden inom teknik och affärsverksamhet, och baseras på en sammanstrålning av dessa (Vinoski, 2001). Områdena beskrivs nedan:

- EDI, vilken har varit ryggraden i B2B integrationer i mer en ett årtionde. EDI har aldrig haft någon stor efterföljare. Huvudsakligen beroende på sin höga implementeringskostnad. Trots de höga kostnaderna fungerar EDI bra, vilket förklarar teknikens överlevnad.

- Traditionell mellanvara vilket används normalt för A2A (applikation to applikation) integrering. System som tillhör den hör kategorin är till exempel CORBA och COM. Den här typen av mellanvara har använts i stor utsträckning och med stor framgång det senaste årtiondet som ett ”integrationsklister” för stort antal heterogena distribuerade system
- The world Wide Web har som vi alla känner till haft en explosiv utveckling under det senaste årtiondet. Detta tackvare dess otroliga kapacitet för presentering och ändlös variation av information (Vinoski, 2001).

Det är för tidigt att säga om Web services kommer att vara det ultimata tekniken för en sammanstrålning. Detta då liknande har sagts om exempelvis CORBA-baserade applikationer (Vinoski, 2001).

Fastän Web services är baserat på öppna standarder finns det stora problem. För det första garanterar inte öppna standarder att implementeringen mellan samarbetspartners fungerar, utan kompabilitetsproblem kan uppstå. Web services måste även anammas av samtliga inblandade företag som vill använda det innan det verkligen blir ett fungerande B2B alternativ. (Hill, 2002)

Innan man börjar investera i Web services bör man undersöka mognaden hos den underliggande tekniken. Tidiga signaler från mjukvaruleverantörer visar på interoperability problem som också går att finna hos CORBA. Vidare har företag som Microsoft och IBM först nu börjat med att utveckla säkerhetsrekommendationer för Web services. Om man i dagsläget vill använda Web services bör man vara säker på att det passar in i den nuvarande arkitekturen för att underlätta vid integration och interoperability. Det bästa är börja använda det inom egna Intranätet. När man gör interna integreringar behöver man inte oroa sig för säkerhet i samma utsträckning och kan i lugn och ro lära dig tekniken och göra misstagen på hemmaplan. (MacVittie, 2002)

3.3 Implementering av Web services

Kärnan i Web Services är implementeringen av själva tjänsterna. Enligt Lublinsky och Farell (2002) så är en typisk Web services implementation baserad på en affärsapplikation eller ett antal integrerade applikationer (Se bild nedan). Bra infrastruktur som säkerhet, service, transaktionsstöd, leveransgarantier o.s.v. måste stödja implementering av Web services. Överst finns sedan gränssnittstjänster som ger access till Web services tjänsterna. En robust infrastruktur behövs för att Web services ska kunna implementeras bra. Specifik infrastruktur som krävs för Web services ska kunna implementeras bra är: Säkerhet, transaktionsstöd, pålitlighet (reliability), styrbarhet (managebility) (Lublinsky & Farrell, 2002).

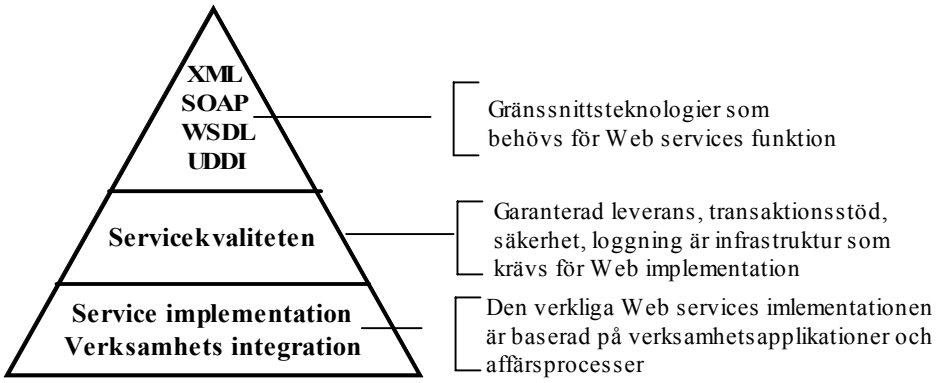

Bild 2: En typisk Web services implementation. (Lublinsky & Farrell, 2002)

Många olika arkitekturalternativ kan användas för att implementera en Web service. Det är viktigt att för varje verksamhet hitta den mest kostnadseffektiva arkitekturen för just deras del. Detta kan göras genom kapacitetsplanering för att styra kvaliteten på tjänsterna från webben och bör följas av ett antal steg. (Se nedan.). Dessa ger en mall för att planera kapaciteten för Web services och förstå behovet. Det krävs att stegen i planeringen följs systematiskt med start i företagets processer för att bestämma vilka tjänster och vilka applikationer som är centrala för målet (Almeida, 2002). Enligt Almeida (2002) bör planering ske i följande steg:

- *Förståelse av omgivningen* – Vilket innebär att få förståelse för hårdvaru- och mjukvaruresurser, nätverkskoppling och nätverksprotokoll som finns i miljön. Även toppar i användning, styrningsstruktur och krav på tjänster ska identifieras. Detta ska leda fram till en systematisk beskrivning av webbomgivningen, dess komponenter och tjänster.

- *Bedömning av arbetsbördan* - innebär att beskriva systemets totala arbetsbörda för huvudkomponenterna. I en Web services miljö interagerar användarna med en sida genom en mängd relaterade förfrågningar som kallas sessioner. Sessioner är en sekvens av förfrågningar för att exekvera e-handelsfunktioner gjorda av enskilda personer vid ett enskilt besök på en webbsida. Arbetsbördan för varje funktion karakteriseras.

- *Hitta parametrar för en modell över arbetsbördan* - Införskaffa värden för parametrarna till arbetsbördan. Här ingår även att vid utförandet av en Web service mäta antal besök per dag samt vilka de vanligaste och mest frekventerade tjänsterna är. Detta steg är viktigt för att garantera kvaliteten på tjänsten och förutse eventuella problem.

- *Prognostisera arbetsbördans utveckling* - Försöka förutse arbetsbördan i framtiden för en Web service. En bra prognos är inte bara ett mängd siffror utan ett antal scenarion och förutseenden. Tiden är en viktig aspekt att ta hänsyn till.

- *Utveckla en modell över prestanda* – Kvantitativa tekniker och analytiska modeller används för att utveckla modellen över prestanda. Modellen används för att förutse prestandan när någon del av arbetsbördan eller arkitekturen ändras.

- *Validering av prestandamodellen* - Efter att modellen har konstruerats bör den valideras. Detta innebär att prestandan i modellen ska jämföras med verkliga mätningar i systemet. Modellen anses som valid om mätningarna stämmer överens någorlunda med varandra och om de inte matchar varandra måste modellen kalibreras. Detta är normalt en iterativ process.

- *Förutse service prestanda*- Prognoser är viktiga vid kapacitetsplaneringen eftersom man måste kunna förutse hur en Web service kommer att reagera vid olika kundbeteenden och när nya affärsmodeller utvecklas. Prestandamodeller är tänkta att representera beteende i ett verkligt system.

- *Analysera framtida scenarion* - Ett antal möjliga arkitekturer analyseras för att hitta den mest kostnadseffektiva. Framtidsscenariona ska ta hänsyn till den förväntade arbetsbördan, kostnaden för webbsidan och kvaliteten på informationen till kunden. Vidare ska det här steget visa för ledningen vilka steg som bör tas för att garantera att en Web service kommer att uppfylla affärskraven för framtiden (Almeida, 2002).

3.4 Implementeringens kritiska framgångsfaktorer

W3C har fastställt sju stycken huvudmål som stödjer verksamheter och deras behov, vid användande av Web services. Genom att använda ”top-down” metoden, där man bryter ner dessa mål, får man fram de kritiska framgångsfaktorerna [11].

Nedan följer de huvudmål som W3C har ställt upp för Web services:

- *Interoperability* - möjliggöra utvecklingen av samverkande Web services över olika miljöer.
- *Reliability* - göra Web services pålitliga och stabila.
- *Integration* - Web services måste vara förenliga med dagens och framtidens utveckling av webben.
- *Security* - Web services måste erbjuda en säker miljö för on-line processer.
- *Scalability and Extensibility* - Web services måste möjliggöra för implementationer som är möjliga att utföra och som kan ändras efter behov.
- *Team Goals* – Ska säkra så att arkitekturen kommer att motsvara behoven från användarna.
- *Management and Provisioning* - Måste erbjuda en lätthanterlig, förklarlig miljö för Web services hantering [11].

Ovanstående huvudmål ligger till grund för de kriterier som vi valt för den här undersökningen. Vi har dock valt att anpassa dem, så att de bättre passar ihop med vårt syfte att upptäcka de problem som finns vid implementeringen av Web services. Genom

att bryta ner våra kriterier är målet att hitta de kritiska framgångsfaktorerna det vill säga problemen vid implementationen. Vi har valt att använda oss av följande kriterier:

- Interoperability
- Integration
- Security
- Architecture.

Anledningen till att vi valt interoperability som ett av våra kriterier är att det kan ses som ett problem vid implementeringsprocessen vilket är det område vi har valt att belysa. Reliability valde vi att inte använda som en egen kriterie då vi anser att den är en del av den relativt breda kriterien security i implementeringsprocessen. Integration valdes däremot som en kriterie då det är en viktig och stor del av implementeringen. Detta då Web services syfte är att möjliggöra integrering mellan applikationer. Scalability och Extensible som ingår i W3C:s huvudmål som femte punkt valde vi bort då den har som syfte att möjliggöra grunden till implementeringen. Detta då vi enbart är intresserade av processen vid implementeringen ligger detta utanför vårt intresseområde. Team goals gjordes om till architecture för att tydliggöra kriteriens verkliga innebörd och för att lättare kunna knyta problemen till rubriken. Det sista huvudmålet, management and provisioning, valde vi slutligen bort då det innefattar tillståndet efter implementeringen.

Då de stora aktörerna när det gäller Web services är Microsoft och IBM på ena sidan och Sun på andra sidan anser vi att de är intressanta att ställa mot varandra och mot ovanstående kriterier. Detta för att påvisa deras eventuellt olika synsätt på implementeringsproblematiken.

3.4.1 Interoperability

En Web service med olika applikationer inom en heterogen plattform som ska vara lyckad förutsätter att tjänsten kommunicerar korrekt och effektivt. Men att erhålla ”seamless” interoperability mellan deltagande entiteter i en Web service kan vara svårt trots fördelarna med standarder så finns det problem kvar att tackla. Problemen härstammar ofta från skillnader i implementationen för Web service protokollen mellan olika kommersiella och öppna källor (Siddharta & Sengupta, 2002).

Interoperability mellan applikationer förespråkas som en viktig del vid Web services. I teorin ska Web services tillåta vilken klient som helst att anropa en service över HTTP via XML dokument. Detta sätt skiljer sig från andra distribuerande tekniker som till exempel COM, som kräver en homogen omgivning för att fungera. Vissa distribuerade miljöer som exempelvis CORBA ger möjlighet för interoperability mellan olika operativsystem, programspråk och implementationer, men interoperability över webben kan inte realiseras med dem (Siddharta & Sengupta, 2002).
Det finns många standarder som kan användas. Detta kan dock kan vara ett problem då det finns för många standarder och att de är så flexibla. Två parter kan tolka dem på så olika sätt, att de till och med blir helt inkompatibla. Detta problem försöker man åtgärda

genom en organisation som inriktar sig mot interoperability inom Web services. Denna organisation, WS-I försöker skapa profiler för interoperabilty. Deras första profil är att få alla som använder standarderna att använda dem på samma sätt. Genom att få de fyra standarderna (SOAP, WSDL, UDDI och XML) för Web services att arbeta tillsammans kan man få en bra grund för interoperability mellan försäljare/utvecklare av applikationer (Travis & Ozkan, 2002).

IBM

IBM rekommenderar att man som utvecklare av Web services använder WSDL för att få en bra interoperability mellan komponenterna. Det möjliggör för att på ett enkelt sätt kunna skicka vidare förfrågningar om tjänster från det interna nätverket till ett externt nätverk och få tillbaka ett svar och istället kunna fokusera på XML gränssnittet (Albornoz, 2002).
Vidare är det lämpligt att använda tekniker som ökar kompabiliteten av WSDL filerna med de flesta Web services gränssnitt. En av dessa tekniker är att använda SOAP kopplat med XML. Användningen av SOAP föreslås enbart för att det är den vanligaste och mest stödda och för att det finns någon anledning att inte använda SOAP i de flesta fall. Angående transportprotokoll vill man inte föreslå något specifikt eftersom det valet är av stor betydelse för användbarheten hos den specifika Web servicen. Fel transportprotokoll kan orsaka allvarliga problem när det gäller möjligheten att utöka, implementationen, säkerheten och pålitligheten. Men det är bra att var medveten om att HTTP är det mest använda och om det är ett vettigt alternativ så ska man försöka använd det (Albornoz, 2002).

Microsoft

En av utmaningarna med Web services interoperability är att specifikationen för första generationens Web services; XML, SOAP, WSDL och UDDI är ett antal standarder som gör det svårt att koordinera olika specifikationer. Användningen av samma specifikation behöver inte alltid betyda att en Web services samverkar med en annan Web services skapad på en annan plattform eller i ett annat programmeringsspråk. Efterhand som antalet specifikationer ökar så ökar också problemet. För att interoperability verkligen ska fungera är det viktigt att industriledande företag definierar implementation och testning av Web services. För att assistera utvecklare att skapa hög interoperability så erbjuder WS-I interoperabilitystandarder – profiler som är specifikationer med lösningar på hur de ska fungera ihop. T.ex. en ”basicprofile” kan innehålla SOAP, UDDI, XML och WSDL och en beskrivning på hur de ska interagera. Mer avancerade profiler finns för exempelvis sändning av meddelande och säkerhet [12].

Microsoft och IBM har lanserat nya specifikationer som kommer att förbättra interoperability inom områden som är kritiska och centrala för Web services, så som säkerheten, tillförlitligt skickande av meddelande och skickande av bifogningar. För att stödja dessa nya standarder har Microsoft lanserat WSE 1.0 (Web Services Enhancements) (Hall Gailey, 2003). WSE erbjuder avancerad Web services funktionaliteter för Microsofts utvecklingsmiljö Visual Studio .NET. Applikationer som

finns färdigutvecklade inom företaget kan utvecklas snabbt till att stödja säkerhetskrav såsom digitala signaturer och kryptering och möjliggöra för att bifoga meddelanden som inte är skrivna i XML. WSE erbjuder en viktig säkerhets-, utvecklingsbar- och utförande miljö [13].

Sun

De två vanligaste utvecklingsplattformarna för Web services är Microsoft .NET och J2EE. De stöder kommunikation sinsemellan på så sätt att en applikation inte kräver modifiering för att en Java klient som använder ett standard Java gränssnitt för XML skall kunna använda en .NET Web service. Fastän att de här plattformarna implementerar vanliga industri standarder som XML så löser inte standarderna interoperabilityproblemen på alla områden. Ett interoperabilityproblem som finns är när en Javaapplikation vill anropa en .NET tjänst asynkront det vill säga att klienten anropar tjänsten men kan inte vänta på svar på grund av tidsbrist utan vill fortsätta med sina processer och när väl svaret kommer sammanfattas processen som behövs för att sätta igång förfrågan om tjänsten. I korthet behöver den applikationen en mellanagent som kan agera som en bro mellan de två plattformarna. En lösning på detta är att låta en Javaproxy klass (som lyssnar efter tjänsten internt och sänder en förfrågan till ett externt nätverk.) agera som den här bron mellan Javaklienten och .NET servicen. Detta görs genom att en WSDL kompilator laddas ner som genererar en Javaproxy klass baserad på WSDL specifikationer för Web services och som stödjer asynkrona kommunikations metoder för att anropa .NET Web services. Vidare implementeras en svarsfunktion som stödjer asynkron kommunikation (Verma, 2003).

3.4.2 Integration

Integration är kombineringen av mjukvaru- och hårdvarukomponenter för att skapa ett totalt fungerande system. Integration kan göras med direkt kontakt av delade objekt eller komponenter, eller så kan det göras med manuell mänsklig medling mellan systemen. Integration kan också uppnås genom att sända meddelanden från ett system till ett annat. Målet med Web service integrationen är att använda plattformsoberoende öppna systemarkitekturer för att integrera systemen oavsett deras ålder, programmeringsspråk, objekt typ eller operativsystem (Travis & Ozkan, 2002).

Problem kan uppstå vid integrering av applikationer eller vid utvidgning av handels partners. Integration innebär att en applikation eller en handels partner skickar data från en till en annan och använder den utan mänskligt inblandning. Att integrera applikationer har varit ett problem ända från det att organisationer började använda två datorer. Traditionellt krävs det för att få olika system att interagerar med varandra en mellanvara som kopplar samman och kan översätta mellan systemen (Travis & Ozkan, 2002).

Innan man kan tänka på att implementera Web services, både inom och utom företaget, måste man få ordning på sina interna system. Det finns många saker att tänka på när man gör detta; Var finns datan, hur fungerar arbetsflödet normalt, vilka är dina partners

och man måste också veta vilken typ av lösningar som är möjliga för implementationen (Travis & Ozkan, 2002).

Den senaste tiden har behovet av att kunna integrera system ökat kraftigt. Web services handlar just om att använda Internetbaserade standarder för utbyte av information mellan inkompatibla applikationer. Många tror att Web services kommer att bli den dominerande infrastrukturen för att systeminteraktion (Shivram, 2002).

Web services gör det möjligt att ha en miljö där applikationer är löst kopplade till varandra och interaktionen mellan dessa sker mer eller mindre automatiskt. Användning av Web services kan vara ett bra sätt att integrera företagsinterna system och det kan vara bra att lära känna tekniken på detta sätt, innan man börjar integrera med externa partners. Ur integrationsperspektiv är det Web services förmåga att kunna skapa lösa kopplingar mellan applikationer det viktigaste. Här bör det finnas stora pengar att tjäna med tanke på de ständiga förändringar som sker i system i verksamheter idag. När man beräknar kostnader för projekt idag räknas ofta bara implementationskostnaden och underhållskostnaderna! Kostnadsaspekterna för integration är också viktigt, speciellt när det sker mot externa partner [14].

Web services implementering är i många avseende lik andra mjukvaror för service men ett karaktärsdrag för Web Services är att anrop tar mycket kraft och sker vanligen över publika nätverk. Överbelastning av nätverk är ett problem vid Web services men nätverk ska inte undvikas istället ska nätverksservicens gränssnitt designas smart. Målet är att öka förfrågnings kapaciteten. Web service måste skicka mer information i en förfrågan. Detta målet kan nås genom att definiera gränssnittet för en speciell affärsservice istället för en låg nivå för mjukvara (Lublinsky & Farrell, 2002).

För att lyckas med en Web services integrering bör man enligt Andrianopoulos (2002) tänka på att:

1. *Automatisera interna system*. Det första och viktigaste steget är att få interna system att fungerar så mycket som möjligt utan mänskligt ingripande. En Web service tillåter att två datorer att kommunicera med varandra mellan olika plattformar. Om där är människor inblandade i processen så är tiden det tar till att få information mellan dem mycket längre. Arbetsflödet kan bli automatiserat genom att man har som mål att människoberoendet blir mindre.

2. *Fastställa integreringspunkter.* När väl interna system är automatiserade är det tid att fastställa vilka punkter som ska vara accessbara vid integration. Det kan vara så enkelt som att en metod kallar på ett objekt som tillhandahåller användbar data eller det kan vara ett ställe som anropar ett komplex arbetsflöde som styr en hel rad av processer. De här integrationspunkterna kan användas av interna system såväl som externa när objekt delas.
3. *Visa integrationspunkterna för Web services.* När integrationspunkterna har fastställts kan de visas som Web services genom att använda maskinförståelig presentation av data. Web services teknologi standardiserar applikationerna,

> bygger infrastrukturen för elektronisk integration och handel. På en grund nivå visas Web services i presentationslagret precis som HTML sidor. I en mer realistisk omgivning är emellertid block skapade som kan som kan användas för att skapa ett mer sofistikerat ramverk som erbjuder säkerhet, pålitlighet, attachment, arbetsflöde och andra vanliga funktioner. De här blocken är vart och ett definierat i en standard och kan implementeras separat eller tillsammans som integrationskomponenter (Andrianopoulos, 2002).

När Web services standarderna så småningom mognar och mjukförsäljare börjar stödja dem fullt ut kommer de att ha en stor inverkan på verksamheters infrastruktur. Eftersom Web services standardiserar kopplingen mellan system medför det att samverkan kommer att öka och integrationskostnader kommer att minska. Dessutom kommer beroendet av mjukvaruförsäljare att minska och det bästa av varje tjänst kommer att vara tillgängligt. Men än så länge fungerar inte standarderna fullt ut. Det hindrar dock inte att företag redan nu börjar tänka enligt Web services strategi när det gäller arkitekturen för att stå beredda i framtiden (Andrianopoulos, 2002).

Dagens standard lämnar en del problem obesvarade när det gäller behovet vid verksamhetsintegrationen. En del av dessa problem är:

- Dataöverföring vid samverkan mellan system med komplexa ”back-end” system.
- Garanterad meddelandetjänst.
- Affärsprocesstyrningen. (Business Process Management, BPM).
- Handelspartner och protokoll styrning.
- Transaktions integritet.
- Säkerhet.

(Andrianopoulos, 2002).

Web services kommer att öka effektiviteten genom att onödiga kopplingar försvinner, men det är inte den största utmaningen. Att skapa och styra flexibla integrationer mellan system som levererar värde till företag är komplicerat och det kräver hög kapacitet på applikationsintegration robustheten. Dessutom måste förvaltning och underhåll av protokollen mellan handelspartners och affärsprocesser styras vilket måste sköts från en säker, utökbar, pålitlig och distribuerad plattform (Andrianopoulos, 2002).

Integrering i verksamheter blir mer och mer komplext och speciellt i globala företag och kommer så att förbli tills det finns en lösning som förenklar för verksamheter vid integrering. Lösningen måste fungera i realtid, internt och externt så att modifieringar kan göras när affärsförhållanden och teknologier förändras (Hammer, 2001).

IBM

WSDL beskriver Web services gränssnittet och Web services ägarna vill implementera sina gränssnitt med hjälp av SOAP. På grund av detta kommer WSDL tjänster existera som SOAP tjänster. När Web services användaren har WSDL filen vet denna användare gränssnittets detaljer. Användaren använder nu SOAP för att kommunicera med Web servicen. Man kan tänkta på Web servicen som ett objekt som man kan exponera genom

WSDL gränssnittet och som har access över Internet med hjälp av SOAP men som inte finns möjlighet att ändras i. SOAP är den enklaste mekanismen i dagsläget för att nå integration och samverkan mellan företag (Siddiqui, 2002).

Idag lägger utvecklare mycket tid och resurser på applikationsintegration. I de flesta fall, uppnår företag integration genom att bygga mellanvaruinfrastukturer tillsammans med många anslutningsdon som tillåter olika back-endapplikationer att ansluta till vanliga protokoll av något slag och utbyta data med varandra (Rudrof & Tost, 2002).
Web services tekniken har nyligen tillkallat sig mycket uppmärksamhet inom integrations området genom att definiera vanliga vägar för applikationer att interagera med varandra över olika programmeringsspråk och operativsystem. Detta har blivit möjligt eftersom Web services använder XML som grund för sitt dataformat (Rudrof & Tost 2002).

Microsoft
Genom att använda Web services kan utvecklare lösa integreringsproblematiken lättare och leverera flexibla lösningar som fungerar med förändringarna i en dynamisk industri. .NET erbjuder en samlad lösning för att bygga och integrera ett flexibelt affärssystem. I plattformen ingår tekniker för till exempel att kunna skicka meddelande, samarbete och styra databaser [15]. Se vidare om Microsoft nedan i Microsoft .Net mot J2EE Web services.

Sun
Microsoft .Net mot J2EE Web services
Sun respektive Microsoft står för varsitt sätt att integrera Web services. Det finns för närvarande 4 stycken utmaningar med Web services. Nedan beskrivs vad som anses vara de stora skillnaderna enligt Hanson (2002):

1. *Service Description* Web service måste beskrivas i någon form av struktur för att kunna profilera sig. WSDL gör detta genom att definiera XML som en beskrivning av meddelandens början och slut. J2EE möjliggör för utbyte av Web services information genom att använda WSDL för att precisera formatet för varje dokument. Tredje part som önskar delta kan slå upp information om företaget i ett register. .NET stödjer också WSDL och använder det för att dokumentet ska beskriva sig själv. XML används för att unikt identifiera slutpunkten. Vidare har .NET en klientkomponenet som tillåter en applikation använda en Web service operation beskriven av ett WSDL dokumentet.

2. *Service Implementation* Implementering innebär ofta att strukturera data och operationer i ett XML dokument som kompilerar med SOAP. När en Web service komponenet väl är implementerad skickar en klient ett meddelande till komponenten i form av ett XML dokument och komponenten skickar tillbaka ett XMLdokument som svar. J2EE Javaklasser och applikationer kan skickas med genom att använda Javas API för XML baserad RPC (JAX-RPC) och utvidgas

som Web services. JAX-RPC använder XML för att kalla RPC, (Remote Procedure Calls) och en API för att packa parametrar och returnera värden och en för att ta emot. .NET plattformen erbjuder ett flertal programmeringsspråk samt ett Microsoft verktyg för SOAP som innehåller komponenter som kan konstruera, överföra, läsa och processa SOAP meddelanden.

.

3. *Service Publishing, Discovery and Binding* När väl en Web service blivit implementerad måste den publiceras någonstans sa att intresserade kan hitta den. Information om hur en klient kopplar och interagerar med en Web service måste också anges. Kopplingen och interaktionen kallas ”binding information”. Vanligtvis används register för att publicera, upptäcka och binda Web services. De innehåller datans struktur som beskriver Web services. Ett register kan antingen innehas av en privat organisation eller en neutral tredje part. IBM och Microsoft har nyligen släppt WSIL (Web Services Inspection Language) specifikation för att tillåta applikationer att söka efter Web services på Web servrar. WSIL är lovat att komplimentera UDDI genom att göra det enklare att upptäcka möjliga tjänster på Webbsidor som inte är listade i UDDI. J2EE å andra sidan anger platsen för dess Java API för XML registreringar (JAXR) i ett enda syfte nämligen att kunna fungera med multipla register. JAXR tillåter sina klienter att få access till Web services erbjuden av en Web service implementator som visar tjänster byggda på en implemenation av en JAXR specifikation.

4. *Service Inovocation and Execution* SOAP är ett enkelt XML baserat lättviktsprotokoll som definierar ramverket för meddelande vid utbyte av strukturerad data och information via webben. J2EE använder Javas API för XML baserad RPC för att skicka SOAP metoder och få svar. När väl den här typen av RPC har definierat en tjänst och implementerat den är tjänsten tillgänglig på en server. I .NET kan intresserade få access till en tjänst genom att implementera en web services ”lyssnare” För att kunna implementera den måste systemet förstå SOAP meddelanden, generera SOAP svar, erbjuda ett WSDL kontrakt för tjänsten och vidarbefodra tjänsten genom UDDI. Microsofts verktyg för SOAP erbjuder en klient komponent som tillåter användande av tjänster beskrivna i WSDL dokument (Hanson, 2002).

3.4.3 Security

För att denna nya teknik som Web services står för, skall kunna bli verklighet, så är hanteringen av säkerheten ett avgörande moment (Balabine & Koschel, 2002). Det finns idag säkerhetslösningar för Web Service men de flesta av dem är inte nya utan äldre tekniker som har använts för att säkra e-handelssidor. Exempel på sådan teknik är HTTP, SSL (Secure Sockets Layer) och brandväggar (Andress, 2002).

Microsoft och IBM:s WS-Security innehåller ett antal standarder men varje standard löser ett specifikt säkerhetsproblem vilket kan leda till att de kan överlappa varandra och således orsaka samarbetssvårighet sinsemellan. Det som istället skulle behövas är en helt komplett säkerhetsmodell för Web services (Lublinsky & Farrell, 2002).

I dagsläget är standardteknologier för säkerhet hos Web services, så som industristandarder och stöd för användandet av digitala signaturer, fortfarande på väg att definieras och utarbetas (Samtani, 2002).

Web services skickas vanligtvis över Internet. Affärs transaktioner som är kritiska för en verksamhets affärer innehåller ofta känslig information och bör därför vara säkrat på något sätt, så att ingen obehörig kommer åt den. Säkerhet omfattar många olika saker, men de viktigaste delarna i Web services säkerhet är enligt Samtani (2002):

Authentication försäkrar att varje entitet som är inblandad vid användandet av Web services är de som de utger sig för att vara. Authentication innebär att acceptera data från en entitet och validera dem mot en auktoritet. **Authorization** försäkrar sig om att den som vill använda en tjänst har access till den och är auktoriserad för att utföra operationen. **Data protection** försäkrar sig om att Web services förfrågan och svaret inte har manipulerats under vägen. Det kräver både dataintegritet och hemlighållande. Dock garanterar inte data protection sändarens identitet. **Nonrepudiation** garanterar att den som skickar meddelandet är den samma som skapat meddelandet (Samtani, 2002).

Eftersom externa affärstjänster används över Internet är säkerhet viktigt vid Web Services implementering. Det finns i dagsläget ett antal specifikationer för Web Services som skall kunna lösa säkerhetsbiten exempelvis XML kryptering och XML digitala signaturer (Lublinsky & Farrell, 2002).

IBM

Omgivningen måste beaktas vid design av Web services. Krav på säkerheten för tjänstepubliceringen skiljer beroende på omgivningen. Intranäts tjänster kräver inte så stor säkerhet medan B2B transaktioner kräver stor säkerhet. Ett sätt att lösa detta är att i en riskanalys ta hänsyn till dessa aspekter och erbjuda olika typ av säkerhet för olika tjänster (Gottschalk, 2000).

Traditionell förståelse för säkerhet måste ändras vid Web services arkitekturer för att stödja sökning och exekvering av tjänster i webbomgivning. Sättet att lösa säkerheten i en Web service arkitektur är att förstå och dokumentera befintliga problem och föreslå motåtgärder. IBM har identifierat följande problemområden som måste få en motåtgärd:

- Säkerheten på informationen vid exekvering delas mellan distribution, förfrågan och den som önskar tjänsten.
- Säkerheten på nätverket som tjänsten utnyttjar.
- Säkerheten på programmeringsmodellen vid designen.

Kontrollen av access tillhör egentligen applikationsutvecklingsmiljön men blir en del av Web services arkitekturen vid en B2B miljö då den måste finnas i applikationen. IBM rekommenderar att den som tillhandhåller en tjänst kontrollerar accessen från den som vill använda servicen innan den används. Det finns ett antal accesskontrollmodeller för att designa distribueringen av tjänster:

- Ingen kontroll alls- vilket innebära allmän access. Problemet med det är man inte vet vem som använder tjänsterna, oauktoriserad modifiering av datan kan ske, information kan avslöjas och tillgång till tjänsten nekas.
- Identifiering vid access till en tjänst som identifierar både den som tillhandahåller servicen och den som använder den. Detta gör att man kan välja vem som får access till tjänsten. Problemet med den här typen är att falsk identitet, oauktoriserad modifiering av data, avslöjande av information och att tillgång till tjänsten kan nekas. Den här typen distribution skulle anropas över HTTPS (Hyper Text Transfer Protocol Secure) protokoll och använda SSL för att försäkra sig om rätt identitet.
- Tredje alternativet är en fullt auktoriserad distribuering vilket innebär att accessinformation skulle lagras vid varje användning av data. Det skulle etablera ägarskap och göra så att bara auktoriserade kan modifiera datan och att bara en del av information kan ses. Hot mot denna typ skulle vara det samma som för typen innan och samma typ av protokoll samt SSL skulle användas. Samt system för att tillåta access från andra verksamheter (Gottschalk, 2000).

Microsoft

Microsofts säkerhetsspecifikation är tänkt att erbjuda de strategiska mål och hörnstenar för en Web services säkerhetsmodell WS-Security, som beskriver hur man ska koppla signaturer och krypteringsfunktioner till SOAP meddelanden. Den är uppbyggd av följande delar:

- WS- Federation som beskriver hur man ska sköta och kunna lita på relationer i en heterogen omgivning och stödjer support för identiteter som kopplas samman.
- WS-Policy beskriver kapaciteten och begränsningarna på säkerhetspolicyn för slutpunkterna.
- WS-Trust beskriver ett ramverk för modeller att kunna lita på och som möjliggör för Web services att arbeta säkert.
- WS-Privacy beskriver en modell för hur Web services och den som frågar efter tjänsterna ska kunna vara privat.
- WS-Secure Conversation beskriver hur man ska styra och autentisera meddelande utbytet mellan de inblandade delarna.
- WS-Authorization beskriver hur man ska sköta auktoriseringen av datan och auktoriserings policys [16].

Sun

I dagsläget har SUN inga säkerhets standarder implementerade i sitt Java Web services utvecklings paket. Implementering av en enkel säkerhetslösning håller på att utföras, vilken är baserad på W3C standarder och kommer att finnas tillgänglig någon gång under 2003. Nu kan man använda HTTPS/SSL för att göra överföringen säker med Web services meddelanden. Det fungerar ganska så bra när det bara är ett steg i processen och man inte måste bekymra sig om sofistikerade mellanhänder, men det finns begränsningar som att ingenting skyddas utanför överföringskanalen. Det finns också en möjlighet att göra en egen lösning med XML signaturer och XML kryptering (Maler, Jindal & Pelegri-Llopart, 2003).

Sun förväntar sig att kommande WS-I profiler ska innehålla säkerhet och standarder utvecklade av andra kommer att stödjas av SUN produkter. Det verkliga säkerhetsproblemet är komplexitet och det är lämpligt att först se till att enkla funktioner är säkra. De traditionella problemen finns också i en Web services situation och stor samverkan mellan komponenter ökar den. I dagsläget räcker HTTPS som säkerhet men snart kommer större säkerhet att behövas (Maler, Jindal & Pelegri-Llopart, 2003).

3.4.4 Architecture

När två externa applikationer integreras så skall arkitekturerna vara sådana att klienten är omedveten om implementationsdetaljerna för servern. Den enda kunskapen som nödvändig är informationen som behövs för att ta kontakt med serverarna och vetskapen om vad som förväntas i retur vilket kallas löst kopplad arkitektur. Web services arkitektur fokuserar på integrationen inuti och utanför verksamheter och målet är att integrera applikationer, program och objekt så att de definiera en tjänst för en verksamhet. Web services erbjuder tekniker för integration och kommunikation med program bland olika tjänster definierade på insidan och utsidan av en verksamhets brandvägg. Om arkitekturen för distribuerade program kommer att vara byggda på Internet, så kommer inte integrationsarkitekturen som finns, vara tillräcklig. De stängda och patentskyddade standarderna så som COM/DCOM och CORBA kan inte bli utvecklade på webben eftersom de kräver ett system där program tillstånd kan bli spårade. Ett sådant ”statefull” system är motsatsen till Internet, vilken erbjuder en ”stateless” arkitektur (Travis & Ozkan, 2002).

Web services uppkom genom utvecklingen av komponentbaserade arkitekturer och distribuerande system. För att i dagens läge kunna utnyttja Web services krävs att utvecklaren skriver egen unik kod för gränssnitt, integration, säkerhet och automatiserade affärsprocesser. Det som krävs för framtiden är en gemensam komponentarkitektur för XML vilken skulle infatta alla applikationselement så som presentation, affärslogik, data, personifiering och säkerhet (Wong, 2002).

Det är tre stora skillnader mellan Web services och traditionella applikationers sätt att kommunicera är att Web services kommunicerar med hjälp av SOAP meddelande istället för Multipurpose Internet Mail Extensions (MIME). SOAP använder XML som en väg att skicka data från en process till en annan. Vidare är inte transportprotokollet specifikt för Web services. Det går alltså att använda andra transport protokoll än HTTP, så som SMTP, raw TCP eller något annat protokoll som passar. Slutligen är Web services till skillnad från traditionella webbapplikationer självbeskrivande. De tillhandahåller metadata som beskriver meddelandena som de producerar och konsumerar (Ewald, 2002).
Web services kommer att förändra webben från en samling av information till ett distribuerat mönster. För att kunna använda dess fulla potential måste en bra beskrivning för Web services utvecklas. Detta syfte har WSMF som ger en konceptuell modell för att utveckla och beskriva Web services. Filosofin är att det ska finnas maximal koppling mellan olika komponenter och ändringsbar medlingsservice så att alla kan prata med alla när som helst. WSMF är tänkt att skapa full flexibilitet och utbyggbarhet inom e-handel

baserad på Web services. Detta mål kan nås genom en arkitektur som är baserad på två principer som komplementerar varandra nämligen: Stark koppling mellan de olika komponenterna som tillhör e-handelsapplikationen och stor möjlighet att ändra mellan vilka man pratar med så att alla ska kunna prata med alla när som helst på ett sätt som har möjlighet att växa (Fensel & Bussler, 2002).

En korrekt företagsarkitektur krävs för att stödja Web Services. Arkitekturen ska skapa gränssnitt till befintliga eller nya företagsapplikationer som knyter ihop tjänsterna. En sådan typ av arkitektur är viktig för att separera affärsprocesser från underliggande applikationer och istället centrera dem i processen för att öka flexibiliteten och möjligheten för snabb och enkel byte till existerande processer och skapandet av nya processer. Detta här baseras på att existerande applikationer utnyttjas som services. Det är också viktigt för att skapa högsta möjliga koppling mellan applikationerna, där applikationerna eller egentligen tjänsterna aldrig pratar med varandra och övergripande exekvering sköts av processorn, separat från tjänsterna. Detta ger starkt utökbara system vilket gör det enklare för installation av nya applikationer. Vidare är det viktigt att separera Web service gränssnittet från den verkliga implementationen genom att introducera affärsprocesser till iscensätta exekveringar, vilket gör det enklare att uppgradera applikationer/tjänster samt ersätta befintliga applikationer med nya som stödjer samma gränssnitt. När ett gränssnittsbyte är nödvändigt är det bara definitionen av processen som påverkas (Lublinsky & Farrell, 2002).

Att skapa en Web service startar vid processen att skapa ett presentationslager för en grupp program eller objekt som definierar en tjänst som en organisation redan har och är villig att erbjuda sina partners. Målet är att möjliggöra för kommunikation på applikationsnivån så att en affärsprocess kan nå data utanför sin processdefinition som i en partners applikation eller i en intern applikation. Så länge viktig data i organisationen kan flyttas mellan applikationer eller mellan processer är presentationslagret enkelt att bygga. Då är det infrastrukturen som är viktig (Travis & Ozkan, 2002).

De senaste integrationslösningarna och systemarkitekturerna fokuserar bara på integrationsproblem inuti verksamheten. De är byggda i distribuerade system som litar på varandra det vill säga klienten och servern, den som frågar efter tjänsten och den som tillhandahåller tjänsten känner till varandra. De vet alltså varandras objektnamn, de vet protokollen som de svarar till och språket som de pratar. I Web services miljön så kan många av dessa saker inte tas för givna (Travis & Ozkan, 2002).

IBM

Web services arkitekturen innefattar tre roller; den som erbjuder tjänster, den som efterfrågar tjänster och den som publicerar tjänster. Tre basfunktioner kan användas i arkitekturen, dessa är operationerna publicering, hitta och koppling (vilka motsvarar UDDI, WDSL och SOAP). En nätverkskomponent kan inneha vilken som av dessa roller. Vid implementeringen av en Web services arkitektur ska det finnas säkerhet för varje del och garanti på kvalitet för tjänstemodulernas krav för styrning och interaktion. Vid Web services gör den dynamiska kopplingen att implementationen blir plattforms-

och programmeringsspråksneutralt. Applikationer kommer att baseras på en sammansättning av uppsökta och ordnade tjänster vid exekvering, s.k. ”Just-in-time” integration av tjänster. Tjänsteintegration kommer att bli nästa generations e-handel (Gottschalk, 2000).

Alla komponenter i ett system är tjänster och vid en Web services innebär applikationsdesign att beskriva funktionerna och dess samverkan och vid körning är applikationsexekvering en sorts samordning. Samordningen av Web services arkitekturen kan skötas av en infrastrukturs mekanism som inte är någon funktion och som måste göras opererbar innan en tjänst kan anropas. Till exempel en speciell kommunikations mekanism som HTTPS. Denna typ av komponenter som ofta i sig är implementerade som tjänst måsta vara på plats innan en tjänst kan anropas. En tjänst kan stödja flera möjliga implementeringar för alla möjliga omgivningar som kan väljas av den som frågar efter tjänsten. På detta sätt kan samarbetet bli så säkert, pålitligt och riktigt som de samverkande önskar (Gottschalk, 2000).

En Web services arkitektur har fördelar som att underlättar för samverkan genom att kraven för samlad förståelse som WSDL och XML finns. Genom att begränsa för bara det absolut nödvändigaste för samverkan kan Web services bli plattforms- och programspråksoberoende. En annan fördel är att den möjliggör för just-in-time integration. Vidare kan komplexiteten minskas genom att inkapsling och samverkan mellan befintliga applikationer möjliggörs. (Gottschalk, 2000).

Microsoft

Microsofts arkitektur för Web services, GXA (The global XML Web services Architecture) är en öppen arkitektur för applikationer över Internet. GXA tillhandahåller ett antal principer och riktlinjer för att främja protokollen och filformatena för dagens Web services, till mer komplexa och sofistikerade uppgifter, och är därmed ett ramverk för framtiden av Web services. GXA är baserad på fyra design teser [17]:

Modular

GXA använder utökningen av SOAP specificationen för att leverera ett antal komponerbara moduler som kan vara kombinerade så som behövs för att leverera ”end to end” möjligheter/kapabiliteter. När nya möjligheter/kapabiliteter krävs, kan nya modul-element skapas [18].

General purpose

GXA är designade för ett brett register av Web services scenarion, från B2B och EAI lösningar till ”peer to peer” applikationer och B2C tjänster [18].

Federated

GXA är helt och hållet distribuerad och designad för att stödja Web services som korsar organisationer och förtroendegränser och som inte kräver centraliserade servrar eller administrativa funktioner [18].

Standards based
Som med föregående Web services specifikationerna, kommer GXA protokollen att ge efter för lämpliga standard delar och Microsoft kommer att arbeta med intressanta parter för att färdigställa sina standardiseringar [18].

Sun
SUN Open Net Environment (ONE) arkitektur påskyndar utvecklingen och är en guide vid utvecklingen av Web services genom att den minskar komplexiteten. Detta då den erbjuder en fördesignad, förutvecklad, förtestad och dokumenterad Web serviceslösning. Arkitekturen inkluderar också en server vilket gör det möjligt att integrera nästa generations Web services enklare med befintliga applikationer och data. Arkitekturen är baserad på standarder och möjliggör för inflytande på befintliga system, plattformar och sänker kostnaden för det totala ägandet [19].

Tjänster på begäran är viktigt funktion för företag. Suns arkitektur erbjuder en integrerad lösning för utveckling och erbjudande av direkttjänster snabbt och ”seamless”. De eliminerar gissningar och riskerna för kunderna genom att erbjuda en omfattande rekommenderad lösning för utveckling på SUN ONE plattformen. SUN arbetar sida vid sida med kunderna för att testa lösningar och försäkra sig om att de klarar kraven för säker, snabb och pålitlig Web services utan att kostnaden drabbar kunden [19].

SUN vill visa för utvecklare att Web services och J2EE är en plattform tillsammans och att de passar bra ihop. Arkitekturens principer går ut på att återanvända mjukvarudesign och kod, separera stabil från instabil kod och objekt återuppbyggnad (Byous, 2002).Web services erbjuder en löst kopplad kommunikation mellan affärsapplikationer vilket är en viktig komponent på Suns plattform J2EE. En typisk Web service blandar olika interaktioner och processmodeller med synkron och asynkron kommunikation [20].

Inkommande klientförfrågningar i form av SOAP meddelande samlas till metodanrop på klassen som implementerar Web services gränssnittet. Tjänstens slutpunkt ska tillhandahålla säkerhetsvalidering och överföringsparametrar för de här förfrågningarna innan förfrågningen delegeras till Web servicens affärslogik. Parametrarna skickas antingen som javaobjekt, XML dokument eller ibland som SOAP dokument delar. Parametrar med javaobjekt är enkelt att hantera men vid XML dokument bör man tänka på: Tjänstens slutpunkt bör validera det inkommande XML dokumentet mot dess schema. Slutpunkten bör också överföra dokumentet till ett intern fungerande schema. Slutligen bör slutpunkten bryta upp dokumentet och samla det i domänobjekt [20].

4 Resultatredovisning

I resultatredovisningen redovisas litteraturstudien i sammanfattad form, vidare presenteras här enkätsvaren som inkommit. Därefter kommer intervjun som gjorts på Intentia och kompletteringsintervjun som gjordes per telefon också den med Intentia. De innehåller frågor baserade på kriterierna interoperabilty, integration, security och architecture.

4.1 Litteraturstudie

Nedan följer en presentation av litteraturstudien i sammanfattad form kring våra kriterier det vill säga problemen och lösningar som kan uppstå vid implementering av Web services. Detta redovisas utifrån IBM, Microsoft och Suns uppfattningar vilka framkommit av litteraturstudien.

Interoperability

IBM:s rekommendation är att man använder WSDL för att tjänster enkelt ska kunna skicka vidare förfrågningar om tjänster via såväl interna som externa nätverk. SOAP bör användas eftersom det är det som har mest stöd däremot bör man välja transportprotokoll med hänsyn till interoperability med den specifika tjänsten.

Microsoft anser att bara för att man använder sig av standarder behöver inte interoperability fungera med Web services. Efterhand som antalet standarder ökar kommer problematiken att öka. Företag som leder utvecklingen bör vara måna om att definiera implementationen av Web services så som WS-I gör med sina profiler. IBM och Microsoft arbetar tillsammans med nya specifikationer för interoperability så att kritiska områden som säkerhet ska fungera.

Sun anser att ett interoperability problem är när tjänster anropas asynkront för då behövs en applikation mellan de olika plattformarna. Detta kan lösas igenom att en Javaproxy klass fungerar som bro mellan.

Integration

IBM menar att SOAP är det enklaste sättet för att få integreringen mellan företag att fungera i dagsläget. Men det som verkligen gör integreringen smidig är att XML används som grund för dataformatet.

Microsofts .NET erbjuder en samlad lösning för att kunna integrera ett flexibelt affärssystem. Medan Sun å andra sidan erbjuder sin variant.

Security

IBM menar att man måste ta hänsyn till vilken typ av omgivning som finns då man bestämmer säkerheten. Intranät kräver till exempel inte fullt så hög säkerhet. Det bästa sättet att lösa säkerhetsproblematiken är att förstå problemen och dokumentera dessa

samt föreslå lämpliga motåtgärder. Följande problemområden definieras av IBM: säkerheten på informationen vid exekvering, säkerheten på nätverket och säkerheten på designen av programmeringen. Vidare rekommenderar IBM att den som tillhandahåller en tjänst kontrollerar accessen från den som vill använda tjänsten.

Microsofts har utvecklat en säkerhetsspecifikation som är tänkt att fungera som hörnstenar i en Web services säkerhets modell. Här ingår funktioner som beskriver hur man ska kunna lita på relationer, begränsningar i säkerheten, ramverk för säkerheten, privathetspolicys, autentisering av meddelanden och auktoriserings policys.

Sun har i dagsläget ingen säkerhets standard i sitt utvecklingspaket. Dock är utvecklingen igång. De rekommenderar att man använder HTTPS/SSL idag eller en egen lösning med XML signaturer och XML kryptering. Sun förväntar sig att kommande WS-I profiler ska innehålla säkerhetsstandarder. Vidare anser de att det verkliga säkerhetsproblemet är komplexiteten så först bör man säkra enkla funktioner. De bedömer att HTTPS är tillräckligt idag men snart behövs bättre säkerhet.

Architecture
Microsoft menar att deras arkitektur för Web services GXA främjar protokollen och filformaten för dagens Web services och blir därmed ett ramverk för framtida komplexa och sofistikerade uppgifter.

Sun ONE arkitekturen är en guide som vid utvecklingen av Web services minskar komplexiteten som kan vara ett problem. Guiden erbjuder en fördesignad, förtestad och dokumenterad Web services lösning.

4.2 Enkät

Hur förhåller sig Ert företag till Web services?
Majoriteten av de företag som svarade, använder inte Web services alls i sina verksamheter. Det var dock många som svarade att de är positivt inställda och att de följer utvecklingen med stort intresse. Några räknar också att med att börja använda sig utav Web services inom en snar framtid.

Ibland de företag som var mest positiva nämndes *”Vi följer det med stort intresse. Vi tror mycket på detta, och ser fram emot vårt första riktiga projekt. ”* samt *”Vi betraktar det som en potentiellt mycket viktig teknologi för oss själva, våra partners och kunder. Därför gör vi research och följer noggrant utvecklingen av Web services. ”*

Minoriteten av de som svarade var ”negativt” alternativt ”delvis negativt” inställda till Web services. De svarade bl.a. att ”*Samtidigt så kan detta vara en fluga. Det är rätt många nya tekniker som kommit under åren som floppat rejält*.” och ” *Tyvärr så är det svårt för företagen (och ofta med Microsoft i spetsen) att enas om en gemensam standard. Så vi ser nog inte så ljust på detta med Web Services*.” Det var bara ett företag som svarade att de inte visste vad det var.

Vilka är de främsta argumenten för Web services anser Ni?
Svaren på denna fråga varierade från företag till företag. Några var dock eniga om att det var interoperability samt oberoendet av plattform och programmeringsspråk som var det främsta argumentet för Web services. Några andra tyckte att det var det breda stödet från mjukvaruleverantörerna. Möjligheten att löst integrera sina interna system med andras externa system sågs också som ett viktigt argument. Vidare var det ett argument för att standarderna som ingår är enkla och väl spridda i branschen och att tjänsterna är tillgängliga över Internet. Att kunden blir mindre beroende av sin IT-leverantör såg också som ett argument bland de tillfrågade. Vidare ansågs att tjänster kan delas och utnyttjas från andra företag också som en fördel. Då detta bidrar till att man inte alltid internt måste utveckla alla tjänster som behövs. Detta i sin tur leder till att utvecklingskostnader kan minska. Några synpunkter tagna från enkäterna:

”Web Services förenklar integration av olika applikationer på olika plattformar, skrivna i olika språk, utifrån en överenskommen standard”

”Alla applikationer kan via XML-baserad kommunikation nyttja Web Services för att komma åt tjänster hos företag på Internet.”

”Det faktum att samtliga stora mjukvarubolag ställt sig bakom Web Services innebär att standarden får en trovärdighet… ”

Vad upplever Ni är det största problemet vid implementeringen av Web Services?
Även svaren på denna fråga varierade från företag till företag. Då var inte så många svarade på denna fråga, så att generalisera är inte möjligt. Bland de inkomna svaren nämndes problem som att i framtiden kan inkompabilitetsproblemen bli större i takt med att Web services standarderna växer och blir allt fler och komplicerade. Även semantikfrågorna kan bli ett problem i framtiden. Då företag kan välja och förmodligen kommer att definiera dessa begrepp annorlunda kan det uppstå problem. Även hur de olika stegen i en flerstegsdialog definieras och får betydelse kan påverka och orsaka problem i framtiden. Då standarder i dagsläget inte är helt färdiga vet man inte hur de kommer att utvecklas i framtiden och vid införande av nya standarder och ny tekniker tar det alltid tid för utvecklare att lära sig hur de fungerar. Säkerheten nämns endast av två företag som det största problemet när det gäller Web services implementationer. Integration mellan olika komplexa system upplevs av ett företag som det allvarligaste problemet vid implementationer. Vidare anser ett annat företag att plattformsberoende delar orsakar problem. Några andra röster om det som upplevs som det största problemet är:

”Hålla reda på om den som tillhandahåller en viss Web service tjänst har ändrat sitt sätt att bearbeta datan… så att du från ett tillfälle till ett annat får ett annat resultat, än vad du förväntar dig. ”

”Standardiseringsläget oklart, vilket är allvarligt med tanke på att interoperabilitet förutsätter någon form av standards”

Vilka eventuella andra problem har uppstått vid implementeringen?
Andra problem som man på företagen upplevde i samband med implementeringen av Web services var att det var problem att skapa en överenskommelse mellan parterna som håller juridiskt, organisatoriskt, ekonomisk och tekniskt mellan de som ska kommunicera med hjälp av Web services. Detta kan ta mer tid än man ibland tror. Utbildningsresurser upplevs också som ett problem samt att verktygen är omogna vilket ger långa projekttider och ibland onödiga problem. I vissa lösningar kan dessutom komplexitetsgraden bli hög och detta kan leda till problem.

Hur har Ni löst problemen på Ert företag? (Beskriv kortfattat.)
Ett företag skriver att problemen som uppkommer kräver egen kod och anpassning, för att man skall kunna gå runt dem. Andra företag menar att problemen skall lösas innan de uppstår, genom att innan projektstart utvärdera vilka typer av tillämpningar som Web services kan stödja samt genom *”att provköra tekniken tidigt” och ”att ha realistiska förväntningar”*.

Ett par företag gav rådet att använda ett visst utvecklingsverktyg eller koncept, så som Computer Associates utvecklingsverktyg ”*Advantage Plex*” samt Suns koncept ”*Sun ONE*”.

Övrig information som Ni bedömer vara av intresse för oss.
Ett företag sammanfattade med att skriva*:*
”Mycket viktigt att branschstandarder kommer fram och blir accepterade för att det skall bli verklig fart på denna användning. Dessutom måste säkerhetsfrågor lösas på ett effektivt sätt.”

4.3 Intervju
Nedan följer resultatet av intervjun med Intentias PR-chef. Han har jobbat på Intentia i ca 2,5 år och tidigare forskade han inom statistik. Intentia är ett konsultföretag som grundades för 20 år sedan i Sverige. Det har expanderat till Europa, Nord Amerika och Asien och har i dagsläget drygt 3300 anställda i världen varav ca 1100 i Norden.

I hur stor utsträckning använder ni Web services ute hos kunder? Används det bara internt på företag eller externt också?
Intentia har använt sig av Web services de senaste ett och ett halvt åren. De var bland de första att utveckla Web services när de började 1999. Då hette det dock inte Web services eftersom det begreppet myntades först 2000. I dagsläget finns det en kund som

använder Intentias verktyg MWSF (Movex Web Service Framework) i ett pilotprojekt. Det lanseras officiellt först i april 2003.

Hur ser ni på implementeringen av Web services ur ett problemperspektiv när det gäller nedanstående kriterier och hur löser ni problematiken bakom problemen?

Interoperability
På Intentia har man idag inte något interoperabilty problem vid implementering av Web services. Detta på grund av att de redan 1999 började utveckla verktyg för att integrera med hjälp av Web services. På Intentia skapade man verktyget MWSF, vilket är ett verktyg som är mer anpassat till Web service för affärssystem. Exempelvis anser de att Microsofts verktyg inte är anpassade för affärssystem med 20 000 funktioner/tjänster då varje liten tjänst tar 2-3 dagar att implementera. Att använda MWSF vid integrering tar enbart 15-20 minuter och inlärningen av hur det fungerar är väldigt snabb, då det normalt tar cirka 1,5 timmar. Vidare behövs det inte någon kunskap om bakomliggande standarder som SOAP, WSDL osv. Interoperability går därför väldigt snabbt, smidigt och problemfritt. Verktyget bygger på metadata vilket gör att när det måste göras förändringar för till exempel XML uppgraderingar görs detta i verktyget och användaren av verktyget påverkas inte.

När företag ser att integrationskostnaderna verkligen minskar kommer Web services verkligen att bli något företag vill satsa på.

Integration
Intentia kan inte se några problem vid integrering om man använder deras verktyg. I detta verktyg bestämmer man vilka funktioner som ska göras tillgängliga för Web services. Detta verktyg skiljer sig enligt Intentia från IBM/Microsoft genom att integrationen är mycket mer stabil. Vidare tycker man att tjänster går att koppla till Web services mycket snabbt, på ett par minuter, i motsats till Microsoft och IBM som tar dagar att för att integrera en tjänst/funktion. Kostnaden med detta verktyg blir betydligt lägre. Just nu används det bara för interna tjänster men man har provat att koppla det till externa applikationer vilket fungerade bra.

Security
Applikationer med speciella säkerhetskrav fungerar inte ur säkerhetsperspektiv utan här måste man använda den teknik som används idag det vill säga EDI. Bandbredden kan eventuellt bli ett problem här också. Annars är säkerheten inget problem.

Architecture
Intentia jobbar med en Java-plattform. Förändringar i till exempel standarder eller plattformar påverkar inte användningen av MWSF för konsulterna, detta då alla förändringar integreras direkt i verktyget av systemarkitekter på Intentia. Ändringarna sker alltså i metadatan, vilket leder till att det ändras i hela integreringsprocessen. Detta medför att konsulterna inte behöver veta om något förändras. Arkitekturen motsvarar

inte kraven för mer komplexa kopplingar. Överföringsproblem kan uppstå beroende på bandbredd.

Intensiva transporter fungerar inte utan här måste man använda den teknik som annars används idag det vill säga EDI. Annars är bandbredden det enda problem som eventuellt skulle finnas. Intentia tror inte att Web services kommer att slå ut EDI vid tyngre systemintegrationer ens i framtiden.

Berätta om något projekt där ni använt Web services.
Intentia har inte själva internt använt sig utav Web services och MWSF, detta då de inte har haft resurser för interna projekt, men det är på gång!

Företaget i pilotprojektet som nämndes innan är Flextronic, Telias servicebolag, vilka sköter servicen vid eventuella problem på telenätet. Deras system måste integreras på grund av att Flextronic måste åka ut och undersöka vad det är för fel och vad som behövs för att åtgärda det. Sedan måste de åka till Telias lager för att undersöka om komponenterna de behöver finns där. Detta kan skötas mycket smidigare med hjälp av Web services då Flextronic direkt via mobil/dator, kan se vad som finns på lager. Intentia har också märkt av ett stort intresse för Web services från sina övriga kunder. Det företag som idag använder verktyget kopplar själv ihop de tjänster som önskas med hjälp av verktyget. Med hjälp av Web services kan man sy ihop det bästa från olika databaser. Det fungerar exempelvis inte med Microsofts verktyg som inte är inte anpassat för affärssystem enligt Intentia. Verktyg anpassade för affärssystem saknas i dagsläget på marknaden, här är Intentia unika. I framtiden kommer integrationskostnaderna att falla med hjälp av Web services. Intentias mål är inte att tjäna pengar på att konsulterna är ute så länge som möjligt hos kunderna och integrerar, utan på att deras system fungerar och att service ges på systemet efter implementation.

Ett generellt problem för Intentia är att få folk att förstå att verktyget finns och vad det kan göra. Intentia menar att det finns alltid en skepsis mot sådant som är nytt.

Idag använder man Web services till att skapa nya affärsmodeller inom företaget. Där man tar det bästa ur olika system och sätter ihop en ny applikation. Kopplingen mellan dessa funktioner sköts med Web services.

4.3.1 Kompletteringsintervju

Nedan följer en intervju med Intentias systemsarkitekt. Han har jobbat på Intentia i 4 år och är den som drog igång utvecklingen av Intentias Web services verktyg. Han är idag ansvarig för vidareutvecklingen av detta.

Hur ser ni på implementeringen av Web services ur ett problemperspektiv när det gäller nedanstående kriterier och hur löser ni problematiken bakom problemen?

Interoperability
Själva konceptet Web services är tänkt för att underlätta vid just interoperability och här kan Intentia inte se några problem. Vid införandet av Web services så måste en kodbas till för att koppling mellan affärssystem ska fungera. Detta lager av kod, eller själva ramverket, för funktionaliteten som man kan välja att bygga på exempelvis IBM's plattform eller Microsoft .NET är ett 20-30 tal rader kod. Det som kan ses som ett problem när det gäller den här koden är att den måste underhållas, dokumenteras och testas. Det är vad som kostar både pengar och tid.

Integration
Om man vill utöka sin verksamhet med Web services kan detta medföra att stabiliteten inte är så bra. Detta är största risken med dagens verktyg samt att kostnaderna kan skena iväg. "*Specifikationerna inga problem om man gör rätt, vilket inte Microsoft och IBM gör!*"

Security
Säkerheten är inget problem enligt Intentia. Specifikationerna är färdiga, men det är som med alla andra system att man får ingenting gratis utan man måste själv se till att den säkerheten man vill ha uppfylls. Den säkerheten som man sedan väljer att ha specificerar man i sitt gränssnitt och den som använder tjänsten ser då vilken typ av säkerhet just den tjänsten har. Säkerhet ingår inte som någon gratis funktion i Web services utan man måste själv lägga till sina funktioner. I princip kan man få sin säkerhet precis så hög som man önskar, det är inte Web services som koncept som sätter begränsningen. "*De som säger att säkerhet är ett problem vet inte vad de talar om. De vill ha allt serverat på ett fat vilket man inte får med något system. Säkerheten måste var och en ta ansvar för själv. Säger någon att det är säkert skall man vara skeptisk. Inga program eller dylikt kan erbjuda tillräcklig säkerhet när man köper det*."

Architecture
Arkitektur finns idag i form av färdiga plattformar som IBM:s och Microsofts men man kan också välja att bygga sin egen för att få de funktioner man önskar. I dagsläget finns väldigt många standarder och fler lär komma, men det behöver inte vara ett problem. Exempelvis utvecklade Microsoft en standard för routing för ca 2,5 år sedan och sedan utvecklade SUN med OASIS som styrorgan en standard för samma sak vilken släpptes för bara ett par månader sedan! Konkurrensen är dock bra. Den standard som finns idag för WSDL är en sammanslagning av det bästa från flera olika utvecklare från konkurrerande företag.

Standarder för versionshantering saknas i dagsläget det är någonting som skulle behövas.

5 Diskussion

I detta kapitel förs en diskussion kring resultatet av undersökningarna. Här redovisas också våra slutsatser och egna reflektioner. Slutligen presenteras här idéer till framtida forskning inom området.

5.1 Diskussion kring implementeringsproblematiken

Web services är inte den första tekniken för att integrera system och distribuera data. Däremot är Web services unikt när det gäller integration över Internet med oberoende av plattform och programmeringsspråk. Web services har ett flertal fördelar som att en applikation kan presenteras som en tjänst och att effektiviteten hos verksamheter kan öka då inte allting måste utvecklas från början utan kan läggas till som en tjänst. Men ingen ny teknik är felfri och kärnan med Web services är själva implementeringen av tjänsterna och där kan problem uppstå. Nedan diskuteras och besvaras uppsatsens frågeställning:

1. Vilka problem kan idag uppstå vid implementeringen av Web services?
2. Vilka eventuella förslag på problemlösningar finns i dagsläget?

Frågorna är komplexa och många olika problem och lösningar upplevs bland utvecklarna av Web services. Dock går det att urskilja vissa konkreta problem vid implementering. Dessa går att hänföra till våra valda kriterier; interoperability, integration, med security samt med architecture.

Interoperability-problemen upplevs vara skillnader i implementationen av protokollen. Tolkningar kan medföra att de i värsta fall blir inkompatibla. Standarderna blir allt fler och interoperability mer komplex vilket också är ett problem. Anrop av tjänster asynkront är ännu ett problem då det behövs någon form av applikation mellan de olika plattformarna.

Interoperability problem som berör tolkning kan lösas genom att WS-I skapar profiler för interoperability standarder och om dessa används så kan tolkningsproblematiken förenklas. Vidare bör WSDL användas för att förfrågningar om tjänster enkelt ska kunna läsas och skickas vidare såväl internt som externt. Anledningen till att välja SOAP är att det protokollet är vanligast och används mest, därmed skapar det också minst problem. När det gäller standarder och komplexitet bör också organisationer som WS-I definiera implementationer. Applikationen mellan plattformarna vid asynkront anrop kan lösas med en Javaproxy klass som fungerar som bro mellan menar Sun. Egen kod för anpassning av olika standarder är en annan lösning som framkommit.

I takt med att allt fler företag börjar använda Web services så kommer förmodligen problemen med interoperability att öka till att börja med. Vi tror dock att detta löser sig då fler standarder utvecklas vilket leder till många av de här problemen kommer att minska.

Integrations-problemen kan uppstå när antalet handelspartners utökas eller fler tjänster integreras. Inkompabilitet och överenskommelser mellan partners vid extern integration blir då ett problem vilket kan bli ännu större i framtiden då antalet integreringar ökar. Andra problem som kan uppstå är överbelastning av nätverk samt att kapaciteten för förfrågningar efter tjänsten kan bli för låg. Dagens standarder har också en del brister exempel på sådana brister är; dataöverföring vid komplexa system, garanterad meddelande tjänst, affärsprocesstyrning, styrning av handelspartners och protokoll, transaktionsintegritet och säkerhet.

Integrationsproblematiken kan lösas igenom att man innan man börjar integrera tänker på det egna systemets funktioner med hänsyn till var datan finns, arbetsflödet och vilka samarbetspartners som finns. Processerna bör i största möjliga mån automatiseras och accesskriterier bör bestämmas. Överbelastning av nätverk kan lösas igenom att tjänsternas gränssnitt designas efter detta och förfrågningskapaciteten kan lösas genom att mer information skickas i varje förfrågan. Dataöverföring vid komplexa system är inte lämpligt för Web services. Här är det bättre att använda äldre tekniker så som EDI. IBM rekommenderar SOAP för att få integreringen mellan företag att fungera. Framtida problem som inkompabilitet och överenskommelser mellan partners ser man i dagsläget ingen lösning på. Vidare måste robustheten hos applikationsintegreringen vara hög för att integreringen ska fungera bra. Dessutom måste underhåll och förvaltning av protokoll mellan handelspartners skötas säkert, pålitligt och med möjlighet till utökning.

Web services används inte idag i komplexa system och vi tror inte att det är något som kommer att ske i framtiden heller, utan vi tror att man kommer att använda äldre och mer beprövade tekniker även då. Vi tror dock att integreringen av Web services kommer att underlättas, när mjukvaruleverantörsparterna enats om vilka standarder som ska gälla för Web services.

Security-problemen är viktiga då mycket information transporteras över Internet, och detta är ännu viktigare vid känslig information. Verifiering, auktorisering, dataskydd och icke reproducering är viktiga aspekter. Ett av de verkliga säkerhetsproblemen är komplexiteten. Applikationer med speciella säkerhetskrav är ett annat problem med Web services.

Säkerhetsproblemen kan idag lösas med hjälp av äldre tekniker som HTTPS och SSL. Det är viktigt att förstå säkerhetsproblematiken för att hitta dess lösning. Det är av vikt att den som tillhandahåller en tjänst kontrollerar accessen från den som önskar använda tjänsten. Microsoft har utvecklat en säkerhetsspecifikation tänkt för att fungera som grund i en säkerhetsmodell. Sun håller på med utveckling av en enkel säkerhetslösning, vilken är baserad på W3C standarder och kommer att finnas tillgänglig någon gång under 2003 men förväntar sig också att kommande profiler från WS-I ska innehålla säkerhet. Komplexiteten när det gäller säkerhet bör lösas genom att man börjar med de enkla problemen för att gå vidare till lite mer komplicerade när de väl fungerar. Applikationer som kräver speciell säkerhet bör inte använda sig av Web services utan förlita sig på äldre tekniker som EDI. Standarder för säkerhet håller alltså fortfarande på

att definieras i dagsläget. De specifikationer som används idag på marknaden är till exempel XML kryptering och XML digitala signaturer.
Vi upplever säkerheten som den del som har längst kvar till ett färdigställande. Detta då säkerhet över Internet kan vara relativt komplext. Då Web services används över Internet är det extra viktigt att säkerheten är tillräcklig för känslig information. Vad som behövs är en komplett säkerhetsmodell för Web services.

Ett problem med **architecture** är att det krävs en gemensam komponentarkitektur för XML som innefattar applikationselement som presentation, data, säkerhet och affärslogik. Andra problem är att komplexiteten i arkitekturen kan uppfattas som en svårighet, problem kan också uppstå vid intensiva transporter samt vid användning av plattformsberoende delar.

Arkitekturproblemen kan lösas genom Microsofts arkitektur för Web services GXA. Denna främjar filformat och protokoll och kan fungera som ett ramverk för framtida uppgifter som är mer komplexa. Sun ONE arkitekturen kan användas som en guide vid utvecklingen av Web services för att minska komplexiteten. Guiden ger en Web services lösning som är fördesignad, förtestad och dokumenterad. Vid intensiva transporter bör äldre teknik som EDI användas.

Arkitekturproblem när det gäller Web services tror vi inte hör till de mest komplicerade problemen när det gäller implementering. Mjukvaruleverantörerna utvecklar idag sina egna arkitekturer för Web services. Vi tror dock inte att de kommer att enas om en gemensam arkitektur, men det är i säg inget problem då Web service är plattforms- och språkoberoende. Det kan trots allt vara en fördel att de på olika håll utvecklar och provar nya standarder och tekniker som leder utvecklingen framåt mot gemensamma standarder.

5.2 Egna reflektioner

Begreppet Web services var helt nytt för oss och verkar fortfarande ligga i startgropen hos de flesta konsultföretag. Dock så kommer det förmodligen att användas mer då folk i branschen får mer kunskap samt får upp ögonen för alla fördelar med tekniken. Dessutom så kommer förhoppningsvis standarder och säkerheten att förbättras inom en snar framtid och då blir intresset större. De företag som använder Web services idag känns väldigt entusiastiska och när andra företag som inte använder tekniken idag ser fördelarna och dessutom fler tjänster finns tillgängliga så blir förmodligen fler intresserade. Eftersom IT-branschen är inne i en svacka så är företag mindre benägna att satsa på ny teknik men så fort potentialen upptäcks tror vi på en stark utveckling av Web services. Detta då teknik över webben ofta är mer lättförståelig än många andra tekniker.

Syftet med uppsatsen var att försöka undersöka de problem och lösningar som kan uppstå vid implementeringen av Web services. Efter att ha genomfört vår empiridel, reflekterade vi över de problem och lösningar som samlats in samt om företagen verkligen ville delge oss problem som de haft och sina lösningar på dessa. Detta då vi inte fick in så många svar på enkäten samt att företaget som intervjuades knappt såg några problem alls. Dock

borde problem finnas då de stora företagen Microsoft, IBM och Sun på sina hemsidor tar upp flertalet implementeringsproblem. Det verkar inte sannolikt att endast dessa företag skulle ha problem och således inte ligga på samma utvecklingsnivå av Web services som de svenska företag som vi kontaktat. Detta då Microsoft, IBM och Sun räknas som ledande i den globala utvecklingen av Web services. Slutligen kan nämnas att vi möjligen var för positivt inställda beträffande utbredningen av Web services användande hos de företag vi kontaktade.

Vidare reflekterade vi över att Sun ofta har en mer teknisk aspekt på sin information än IBM och Microsoft. Detta ledde till att deras information var svår att jämföra med IBM:s och Microsofts information.

6 Avslutning

Detta avslutningsavsnitt beskriver de slutsatserna som kan dras av undersökningen. Här lämnas även förslag till framtida forskning.

6.1 Slutsats

Vår slutsats är att företagen i Sverige upplever få implementeringsproblem i dagsläget, vilket troligtvis beror på att få har provat eller använder Web services. Vi tror också att det beror på att tekniken inte är fullt utvecklad. De stora mjukvaruleverantörerna har dock börjat arbeta med problemen inom området, vilka det i dagsläget inte finns några fullständiga lösningar på. Vid användning av Web services internt så upplevs idag inga större problem. I dagsläget används därför Web services lämpligen i sådana system i väntan på tillräckliga standarder för externa tjänster. Vidare drar vi slutsatsen att det kommer att ta några år innan Web services kommer vara en utbredd teknik för dataöverföring över Internet. Detta då standarder kring Web services måste bli färdigutvecklade samt att alla problem som uppstår vid implementeringen måste lösas, det vill säga att tekniken måste bli väl beprövad.

6.2 Framtida forskning

Under arbetets gång med skrivandet av uppsatsen har en del tankar och funderingar beträffande områden till vidare studier dykt upp. Nedan presenteras de idéer som vi anser är mest intressanta:

- Fallstudie på ett företag som implementerar Web services skulle kunna göra att fler implementationproblem upptäcks samt fördelar och nackdelar med Web services. Vad kan Web services tillföra en verksamhet?
- Kostnadsbesparingar med användandet av Web services. Vad kan en verksamhet spara på att använda färdigutvecklade tjänster istället för att utveckla själv?
- Web services standarder. Är standarderna tillräckliga? Saknas något?
- Web services ur ett säkerhetsperspektiv. Hur långt räcker dagens säkerhet? Vad fattas för att säkerhetskraven ska anses uppfyllda?
- Hur utbrett är Web services på marknaden om 3-5 år? Har det tagits emot väl och används det så utbrett som antagits?

Referenser

Böcker och artiklar

Almeida, V. (2002). *Capacity Planning for Web Service*. Lecture Notes in Computer Science 2459, pp. 142-157, Berlin Heidelberg: Springer-Verlag.

Andrianopoulos, A. (2002). The Framework Behind Web Services Integrtion. *EAI Journal june 2002*. pp 22-23.

Backman, J. (1998). *Rapporter och uppsatser*. Lund: Studentlitteratur.

DeMarco, T., Lister, T. (1990). *Software state of the art: Selected papers*, New York: Dorset House Publishing Co., Utvald artikel: Brooks, Jr., F., *No Silver Bullet: Essence and Accidents of Software Enineering*

Dunn, B. (2003). A Managers Guide to Web Services, *EAI Journal January 2003.* pp 14-17.

Fensel, D., Bussler, C. (2002). *Web Service Modeling Framework WSMF.* Science Direct. Elsevier Science B.V.

Hammer, K. (2001). Web Services and Enterprise Integration. *EAI Journal., November 2001*. pp12-15.

Hansen, M., Madnick, S., Siegel, M. (2002). *Process Aggregation using Web Services*. pp 12-27. Lecture Notes in Computer Science 2512. Berlin Heidelberg: Springer-Verlag.

Holme, I. M., Solvang, B. K. (1997). *Forskningsmetodik, Om kavalitativa och kvantitativa metoder.* Lund: Studentlitteratur.

Johansson Lindfors, M-B. (1993). *Att utveckla kunskap: om metodologiska och andra vägval vid samhällsvetenskaplig kunskapsbildning*. Lund: Studentlitteratur.

Nordner, A. (2003, januari, 17). Säkerheten är på väg. *Computer Sweden*. Stockholm:IDG.

Patel, R., Davidson, B. (1994). *Forskningsmetodikens grunder Att planera, genomföra och rapportera en undersökning*. Lund: Studentlitteratur.

Siddharta, P., Sengupta, S. (2002). *Web Services Interoperability: A Practioner´s Experience*. Lecture Notes in Computer Science 2519. pp. 587-601. Berlin Heidelberg: Springer-Verlag

Travis, B., Ozkan, M. (2002). *Web Services Implmentation Guide*, Denver: Architag Press.

Internetdokument

Albornoz, J. (2002). *Finding your way through Web service standards, Part 3: More WSDL and best practices.* [WWW document]. URL http://www-106.ibm.com/developerworks/webservices/library/ws-stand3.html?dwzone=webservices (2003-03-16)

Andress, M.(2002). *The road to secure Web Service*. [WWW dokument]. URL http://www.infoworld.com/article/02/01/10/020114tcsecure_1.html (2003-03-16)

Balabine, I. Koschel, A. (2002). *Komma igång med projekt*. [WWW dokument]. URL http://www.dfs.se/kompetens/konferenser/default.asp?d=4&id=791&subid=790 (2003-03-16)

Berlid, S. (2001). *UDDI – kokar soppa på en spik?*. [WWW dokument]. URL http://www.rockad.nu/0102/rockad_0102-2.pdf (2003-03-16)

Danielsson, L. (2003). Utan standarder faller visionen. [WWW dokument]. URL http://www.mediearkivet.se/mediearkivet/sok/skolor/skolor.html (2003-03-16)

Ewald, T. (2002). *Understanding XML Web Services: The Web Services Idea.* [WWW document]. URL http://msdn.microsoft.com/webservices/understanding/readme/default.aspx (2003-03-16)

Ferris, C. (2002). *First look at the WS-I Basic Profile 1.0.* [WWW document]. URL http://www-106.ibm.com/developerworks/webservices/library/ws-basicprof.html (2003-03-16)

Gottschalk, K. (2000). *Web Services architecture overview.* [WWW dokument]. URL http://www-106.ibm.com/developerworks/web/library/w-ovr/ (2003-03-16)

Haas, H. (2003). *Web Services Activity Statement*. [WWW document]. URL http://www.w3.org/2002/ws/Activity (2003-03-16)

Hall Gailey, J. (2003). *Using Web services Enhancements to send SOAP Messages with Attachments.* [WWW document]. URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/wsedime.asp (2003-03-16)

Hanson, J. (2002). *NET versus J2EE Web Services Acomparison of Approaches.* [WWW dokument]. URL http://www.webservicesarchitect.com/content/articles/hanson01print.asp (2003-03-16)

Hill, K. (2002). *Report: Web Services No Silver Bullet.* [WWW dokument]. URL http://www.crmdaily.com/perl/story/17589.html (2003-03-25)

Höij, M. (2002). *Web services – drömmen som blev sann*. [WWW dokument]. URL http://www.mediearkivet.se/mediearkivet/sok/skolor/skolor.html (2003-03-16)

Jacobs, I. (2003). *About the World Wide Web Consortium (W3C)*. [WWW document]. URL http://www.w3.org/Consortium/ (2003-03-16)

Lublinsky, B., Farrell, M. (2002). *Web Services The Implementation Iceberg*. [WWW dokument]. URL http://www.eaijournal.com/PDF/WebServicesLublinsky.pdf (2003-03-16)

MacVittie, D. (2002). *Do-It-All Web Services?, Reality Check, Please*. [WWW dokument]. URL http://www.networkcomputing.com/1310/1310buzz2.html (2003-03-25)

Maler, E., Jindal, A., Pelegri-Llopart, E. (2003). *Web Services Security*. [WWW dokument]. URL http://developer.java.sun.com/developer/community/chat/JavaLive/2003/jl0211.html (2003-03-16)

Olofsson, K.(2002). *Perspektiv*. [WWW document]. URL http://www.mediearkivet.se/mediearkivet/sok/skolor/skolor.html (2003-03-16)

Rudrof, D. Tost, A. (2002). *Integrate enterprise applications with Web services and J2EE*. [WWW document]. URL http://www-106.ibm.com/developerworks/webservices/library/ws-eai/ (2003-03-16)

Samtani, G. (2002). *Top 10 Web service security requirements. [WWW dokument]. URL* http://builder.com.com/article.jhtml?id=u00320020610GXS01.htm (2003-03-16)

Shivram, A. (2002). *Nyttan med web services*. [WWW dokument]. URL http://www.dfs.se/kompetens/konferenser/default.asp?d=4&id=791&subid=790 (2003-03-16)

Siddiqui, B. (2002). *Deploying Web services with WSDL: Part 2.* [WWW document]. URL http://www-106.ibm.com/developerworks/webservices/library/ws-intwsdl2/ (2003-03-16)

Sliwa, C. (2002). *Defining Web Services Is No Easy Task.* [WWW document]. URL http://www.computerworld.com/developmenttopics/development/webdev/story/0,10801,73923,00.html (2003-03-16)

Verma, M. (2003). *Platform Interoperability: Sun[tm] ONE and Microsoft .NET, Achieving Asynchronous Communications.* [WWW dokument]. URL http://sunonedev.sun.com/building/tech_articles/async_paper.html (2003-03-16)

Wallström, M. (2002). *Tre tunga utmaningar* [WWW dokument]. URL http://www.mediearkivet.se/mediearkivet/sok/skolor/skolor.html (2003-03-16)

Vinoski, S. (2001). *Web services.* [WWW dokument]. URL http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci777321,00.html (2003-03-25)

Wolter, R. (2001). *XML Web Services Basics.* [WWW document]. URL http://msdn.microsoft.com/webservices/understanding/webservicebasics/default.aspx?pull=/library/en-us/dnwebsrv/html/webservbasics.asp (2003-03-16)

Wong, G. (2002). *Nyttan med web services*. [WWW dokument]. URL http://www.dfs.se/kompetens/konferenser/default.asp?d=4&id=791&subid=790 (2003-03-16)

Ofullständiga referenser

[1]
Fördelarna med Web services. [WWW dokument]. URL http://www.attachmate.se/article/0,1012,3856_23_7926,00.html (2003-03-16)

[2]
Web Services definition. [WWW dokument]. URL http://www.service-architecture.com/web-services/articles/web_services_definition.html (2003-03-16)

[3]
Wilson, M., *Web Services - a definition.* [WWW document]. URL, http://www.w3c.rl.ac.uk/pasttalks/slidemaker/W3C_web_services/slide1-6.html (2003-03-16)

[4]
What is a Web Service?. [WWW document]. URL http://www.wslabs.com/ (2003-03-16)

[5]
Web services definition. [WWW document]. URL http://pt.sun.com/noticias/kits/images/epitch.pdf (2003-03-16)

[6]
Why Web Services?. [WWW document]. URL http://www.webservices.org/index.php/article/articlestatic/75 (2002-03-16)

[7]
Introduction to Web Service. (2002). [WWW dokument]. URL http://dev.systinet.com/library/white_papers/index (2003-03-16)

[8]
About us. [WWW document]. URL
http://www.ws-i.org/AboutUS.aspx (2003-03-16)

[9]
Organization for the Advancement of Structured Information Standards. [WWW document]. URL
http://www.oasis-open.org/who/ (2003-03-16)

[10]
OASIS Technical Committee Addresses Management of Web Services. [WWW document]. URL
http://xml.coverpages.org/ni2003-02-25-b.html (2003-03-16)

[11]
Web Services Architecture Requirements. [WWW document]. URL
http://www.w3.org/TR/wsa-reqs (2003-03-16)

[12]
Web Services – Interoperability across platforms, applications, and programming language, [WWW dokument] URL
http://www.microsoft.com/net/downloads/ws-i.doc (2003-03-16)

[13]
Web Services Enhancements for Microsoft .NET. [WWW dokument]. URL
http://msdn.microsoft.com/webservices/building/wse/default.aspx (2003-03-16)

[14]
[WWW dokument]. URL
http://www.frontec.se/Integration/news/news_integration_webservices.htm (2003-03-16)

[15]
Microsoft Drives XML Web Services Integration Through .NET Enterprise Servers. [WWW document]. URL
http://www.microsoft.com/presspass/press/2001/jun01/06-18FlessnerPR.asp (2003-03-16)

[16]
Security in a Web Services World: A Proposed Architecture and Roadmap. [WWW document]. URL
http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnwssecur/html/securitywhitepaper.asp (2003-03-16)

[17]
An Introduction to GXA: Global XML Web Services Architecture. [WWW document]. URL
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dngxa/html/gloxmlws500.asp (2003-03-16)

[18]
XML Web Services Infrastructure. [WWW document]. URL
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconwebservicesinfrastructure.asp (2003-03-16)

[19]
Sun Microsystems accelerates delivery of Web services with Sun One based application services reference architecture. [WWW document]. URL
http://www.sun.com/smi/Press/sunflash/2002-06/sunflash.20020625.1.html (2003-03-16)

[20]
Using Web services Effectively. [WWW dokument]. URL
http://java.sun.com/blueprints/webservices/using/webservbp.html (2003-03-16)