Serviamprojektet

- en översikt

2005-02-17

Stig Berild
Martin Henkel
Jesper Holgersson
Mira Kajko-Mattson
Nicklas Lundblad
Eva Söderström
Peter Söderström
Benkt Wangler
Lars Wiktorin

Innehållsförteckning

Del I

Affärsnytta, avtal och juridik

Peter Söderström
IT Plan

Eva Söderström
Högskolan i Skövde

Niklas Lundblad
Santa Anna Institute

Introduktion

"*Web Services isn't about technology; it's about successful business strategy*." (Dunn, 2003, p.17)

> "*Web Services has nothing to do with technology and everything to do with management*."
> (Murphy and Stoyanova, 2003)

Citaten ovan beskriver hur Web Services egentligen inte handlar främst om tekniken, utan minst lika mycket om lednings- och strategifrågor. Det har varit mycket uppmärksamhet och "hype" kring begreppet Web Services den senaste tiden. De flesta större organisationerna har någon form av inställning eller position rörande fenomenet (Azzara, 2002). Web Services är viktiga eftersom de representerar en grundläggande skifte i hur applikationer skapas och sprids (Wong, 2002). Trots detta är de verkliga nyttorna som kan uppnås genom Web Services fortfarande inte komna. Förväntningarna är dock stora. Mycket av det som hittills skrivits om angående Web Services handlar om tekniska frågor och standarder. Denna rapport innehåller fyra delar, där två har att göra med affärsnytta och två med juridiska och ekonomiska aspekter på Web Services. Var och en av de fyra delarna är en sammanfattning av en mer omfattande rapport. Tabell 1 visar relationerna mellan denna rapports kapitel och de rapporter som informationen hämtats ifrån.

Kapitel	Referens till rapport
2	Söderström, E. (2004), *Serviam Literature Survey, Part II, Business Value*, Report no. SERVIAM-LIT-02, August 21, 2004
3	Söderström, E. och Söderström, P. (2004), *Serviam Company Visits, Part II, Business Value*, Report no. SERVIAM-LIT-10, November, 2004
4	Lundblad, N. (2004), *Rättsliga frågor och Web Services – en probleminventering*, Report no. SERVIAM-LIT-20, April, 2004
5	Lundblad, N. (2004), *Webbtjänster och juridik – Sluttrapport rättsliga aspekter*, Report no. SERVIAM-LIT-30, November, 2004

Tabell 1: Relationer mellan kapitel och andra rapporter

Förväntningarna på Web Services är stora. Web Services kommer troligen att framför allt påverka marknader som snabbt förändras och där konkurrensen är stor. Förmågan att kommunicera och ha tillgång till information är viktig. En dynamisk omgivning tvingar organisationer att ständigt utvecklas och anpassas beroende på kundernas behov och andra aktörers krav.

Affärsnyttor

Det finns olika typer av affärsnyttor gällande Web Services. I detta kapitel summeras de nyttor och de problem som omnämnts i litteraturen.

Affärsnyttor enligt litteraturen

Det görs många löften för vad Web Services kan åstadkomma, t.ex. att Web Services kan realisera betydande kostnadsbesparingar inom sex till tolv månader samtidigt som det endast kräver små investeringar (Hagel, 2003). Sammantaget innebär Web Services att organisationer kan fokusera

på att skapa och leverera tjänster till kunder, anställda och partners mer än att oroa sig på systemkompatibilitet (Roby, 2003).

Vissa källor beskriver nyttan med Web Services genom att jämföra Web Services-teknik med traditionella lösningar. Några exempel på detta ges i tabell 1.

Traditionella lösningar	**Web Services-lösningar**
Traditionell integrering, so matt skala punkt-till-punktlösningar, kostar mycket och är komplext. Det gör den traditionella plattformen dyr (Race, 2003a; Murphy and Stoyanova, 2003; Colan, 2003)	Information i bakomliggande system blir tillgänglig och kan kopplas till högre nivåer. Detta möjliggör löst kopplade lösningar i kontrast mot t.ex. traditionell EDI (Race, 2003a; Colan, 2003)
Förr var integration en tung börda. När applikationer förändrades behövde också kopplingar ändras tillsammans med dem (Smolnicki, 2003).	Web Services för organisationer närmare realtidsdelning av data bland partners. Med Web Services förblir det standardiserade gränssnittet funktionellt även när applikationer omarbetas (Smolnicki, 2003)

Tabell 1: Traditionell vs. Web Services-lösningar

Nyttorna med Web Services kan delas in i två huvudgrupper – affärsmässiga och tekniska. Nedan presenteras respektive grupp kort i en punktlista. För mer utförlig information hänvisas till den större rapporten. De **affärsmässiga** nyttorna är:

1. *Högre intäkter*: Detta innefattar användningen av Web Services för att komma åt nya marknader, partners och kunder, samt att organisationen ska vara flexibel och på så vis snabbt kunna ställa om sig.
2. *Lägre kostnader*: Detta innefattar minskade kostnader för administration och service via integration, lägre kostnader på grund av kortare tidscykler, samt snabbare och enklare sammankopplingar med samarbetspartners.
3. *Exponerat sortiment*: Detta innefattar nya sätt att presentera information som finns i t.ex. arvsystem, där gränssnitt är anpassningsbara gentemot aktuell kund.
4. *Bättre nyttjande av intellektuella resurser*: Detta innefattar t.ex. att vissa tjänster kan outsourcas för att ge företaget möjlighet att fokusera på kärnkompetensen, samt att Web Services kan utvecklas snabbare och med färre resurser.
5. *Fusioner, uppköp och samverkan*: Detta innefattar att tillsammans med andra förbättra planering, samarbetskostnader, öka automatiserat informationsutbyte både internt och externt mellan system, för att skapa ökat värde gentemot intressenter.
6. *Minskade ledtider*: Detta innefattar t.ex. minskade lagerkostnader och ökad synlighet på grund av direkt åtkomst av aktuell information.
7. *Mer effektiva affärsprocesser*: Detta innefattar att Web Services möjliggör relativt enkel optimering av processer, t.ex. genom användningen av industristandarder.
8. *Snabbare time-to-market*: Detta innefattar att företag genom flexibiliteten som erbjuds genom Web Services snabbare kan svara på förändringar i marknader och få ut produkter snabbare.

Tekniknyttorna är:

1. *Plattformsoberoende*: Detta innefattar att tjänster kan befinna sig på och kommas åt från olika plattformar och systemtyper.

2. *Komponenttänkande*: Detta innefattar att erbjuda funktionalitet förpackad i grundläggande paket, där fler delar kan läggas till separat. Detta möjliggör dynamiskt skapande av applikationer.
3. *Återanvändning*: Detta innefattar att minska utvecklingstiden genom att återanvända befintlig kod istället för att duplicera den. Applikationer blir flexibla och kan relativt enkelt byggas om för att deras livslängd därmed ska utökas.¨
4. *Frikoppling av gränssnitt gentemot underliggande logik*: Detta innefattar att logiken kan ändras vid behov medan gränssnittet förblir intakt. Systemet påverkas inte lika mycket av förändringar. Ett företags applikationspark blir därmed mer flexibel.
5. *Flexibel användning av äldre rutiner (arv)*: Detta innefattar förlängd livslängd för gamla system, genom användning av XML-baserade Web Services.
6. *Lösa kopplingar i arkitekturen*: Detta innefattar en möjlighet till dstribuerade och löst kopplade applikationer som kan lokaliseras dynamiskt för särskilda uppgifter. Det förenklar även länkning av applikationer internt i företag.
7. *Minskade utvecklingsresurser*: Detta innefattar att Web Services ska vara enklare att skapa och kräva mindre utbildning. Nya typer av integrationsscenarier kan även stödjas enklare. Utvecklingstiden kan även den minskas, t.ex. på grund av återanvändning.

Tekniknyttorna kan i sig delas in i undergrupper. Punkterna ett till fyra handlar om faktorer som är beroende av standarder, medan fem och sex har med arkitekturer med nav att göra. Nytta nummer sju är en egenskap som syftar till en effektiv utvecklingsmiljö.

Problem och utmaningar

Web Services kommer med stora löften, men det finns några aspekter som kan orsaka problem och som därför ska ses som utmaningar och möjliga fallgropar. I detta kapitel ska vi kort ta upp fem sådana utmaningar som beskrivits i litteraturen. Utmaningarna är:

1. *Standards-relaterade utmaningar*: Standarder för Web Services är fortfarande relativt omogna och inte helt kompletta, med undantag för kärnstandarderna SOAP, WSDL och UDDI. Aspekter som återstår att lösa är säkerhet, transaktioner, processflöden, användarinteragering, etc (Smolnicki, 2003; Estrem, 2003; Dunn, 2003; Wong, 2002). De standarder som används måste även kunna hanteras internt i organisationer.
2. *Interoperabilitetsutmaningar*: Mjukvaror som utvecklas kan ibland inte hantera de möjligheter som finns i standarderna och håller sig därför endast till en begränsad del av dem. Olika mjukvaruföretag erbjuder också olika typer av Web Services-miljöer och utlovar samtidigt interoperabilitet, flyttbara applikationer, återanvändning av kod, skalbarhet, säkerhet, etc. Det återstår dock att se om de kommer kunna hålla dessa löften fullt ut.
3. *Utmaningar för nya möjligheter*: Varje organisation som ska börja med Web Services måste ta hänsyn till vilka nya initiativ de kan starta med hjälp av Web Services och hur de kan nå ut till nya marknader. Därför behövs ett visst fokus på sådana lösningar som inte varit möjliga hittills. Utmaningen ligger i att kunna ekonomiskt motivera sådana satsningar till företags-ledningen.
4. *Utmaningar för kunskaper och skickligheter*: Många av dem som arbetar med IT idag är ganska oerfarna vad gäller införande av Web Services (Smolnicki, 2003). Utan mänsklig kompetens kan inte tekniken nyttjas som önskat, vilket gör utbildning nödvändigt. Samtidigt behöver organisationer ha kontroll över vad de *inte* ska använda Web Services till. Detta är även ett långtidsperspektiv.

5. *Juridiska och finansiella utmaningar*: Några exempel på utmaningar här är att skydda kundernas integritet, intellektuellt kapital, effektiv kostnadsdelning och betalningsmodeller, ägande av Web Services, etc (Govern and Weagraff, 2003).

Råd till nybörjare

Organisationer som ska komma igång med Web Services behöver ta hänsyn och planera inför ett antal aspekter. Affärsstrategin är minst lika viktig som tekniken, vilket citaten i början av rapporten visar på. I detta avsnitt summeras tre råd till nybörjare:

1. *Analysera organisationen*: Innan en organisation börjar med Web Services måste de klargöra om och att de har en tydlig vision över vad de vill och hur de ska ta sig dit. Detta inkluderar att definiera organisationens mål (inklusive varför och till vad Web Services ska användas), att analysera det kulturella skiftet (hur det nya arbetssättet påverkar organisationen), samt att utveckla en plan för organisationen (en så kallad ”roadmap”, som innefattar att knyta resurser och aktiviteter till målen).
2. *Analysera tekniken*: Befintlig teknisk arkitektur och önskad framtida dito behöver analyseras för att kunna övergå till det nya på ett bra sätt. Detta inkluderar att analysera den nuvarande arkitekturen (inklusive vad Web Services är tänkta att lösa och stödja och vilka delar av arkitekturen som påverkas), samt att utveckla en plan för tekniken (en teknisk ”roadmap”, vilket innefattar vilka förändringar som behövs och hur dessa ska kunna åstadkommas).
3. *Utveckla exempelfall*: Syftet är att identifiera både hur och varför intern och/eller extern användning av Web Services ska ske. Detta inkluderar att utveckla exempelfall som visar vad som behövs internt i organisationen, samt exempelfall för extern användning av Web Services.

Oavsett om det handlar om extern eller intern användning av Web Services så måste en tydlig väg för hur organisationen ska utveckla en tjänstebaserad arkitektur skapas (Hagel, 2003).

Företagsbesök

Förutom den litteraturstudie som gjorts har också ett antal företagsbesök genomförts inom projektet Serviam. Detta kapitel sammanfattar de huvudsakliga kommentarer och resultat som framkommit för affärsnyttor, problem och utmaningar samt juridiska och ekonomiska aspekter. För mer omfattande beskrivning hänvisas till projektets större rapporter.

Affärsnyttor

Vid presentation av de affärsnyttor som företagen har nämnt användes samma struktur som för litteraturstudien, t.ex. för att underlätta jämförelse. Detta betyder att faktorerna som omnämnts är indelade i två huvudgrupper: affärsmässiga och teknikmässiga. De **affärsmässiga** nyttorna är:

1. *Högre intäkter*: Detta innefattar framför allt kundanpassning genom Web Services, vilket ökar värdet för både kund och företag.
2. *Lägre kostnader*: Detta innefattar främst att kostnader minskar till en följd av att administration inom organisationerna minskar när kunderna själva kan göra många enklare uppgifter direkt mot systemen.

3. *Exponerat sortiment*: Detta innefattar möjligheterna att exponera funktionalitet från arvsystem och andra befintliga system på ett nytt sätt för att skapa ökat kundvärde.
4. *Bättre nyttjande av intellektuella resurser*: Detta nämndes endast i relation till en mer effektiv utvecklingsmiljö.
5. *Fusioner, uppköp och samverkan*: Detta innefattar att Web Services möjliggör för organisationer att trigga processer från mainframes, för att på så vis kunna vara mer proaktiva mot sina kunder. Ett exempel är vid försenade leveranser.
6. *Minskade ledtider*: Detta innefattar t.ex. att uppdateringsproblem kan minskas liksom tiden det tar att uppdatera.
7. *Mer effektiva affärsprocesser*: Detta innefattar att enhetlighet kan uppnås, vilket bidrar till högre kvalitet och resultat i ett långtidsperspektiv. Även fel som begås görs på ett liknande sätt och kan därmed spåras på ett enklare sätt.
8. *Snabbare time-to-market*: Inga aspekter nämndes av företagen som kan placeras inom denna punkt.

Tekniknyttorna är:

1. *Plattformsoberoende*: Inga specifika kommentarer omnämndes kring denna punkt.
2. *Komponenttänkande*: Detta innefattar t.ex. komponent i både applikationer och utvecklingssätt.
3. *Återanvändning*: Detta innefattar att på ett nytt sätt kunna paketera information, t.ex. i form av kataloger, som sedan erbjuds till kunder. Detta ökar bland annat återanvändning av tjänster på en högre nivå i organisationer.
4. *Frikoppling av gränssnitt gentemot underliggande logik*: Detta innefattar många aspekter, t.ex. att dubbellagring av data och uppdateringsproblem kan minskas, samt att interna system kan bevaras mer eller mindre orörda genom användningen av ett Web Services-gränssnitt.
5. *Flexibel användning av äldre rutiner (arv)*: De aspekter som nämndes relaterar till det som omnämnts under återanvändning.
6. *Lösa kopplingar i arkitekturen*: Detta innefattar dels en central lösning för att nå Web Services-funktionalitet, samt att skala och skikta applikationer på ett korrekt sätt.
7. *Minskade utvecklingsresurser*: Detta innefattar främst enhetlighet i arbetssätt, och konsistenta resultat från t.ex. beräkningar. Bland annat kvaliteten ökar på så viss i ett långtidsperspektiv. Utveckling kan distribueras, samtidigt som återanvändning av både kod och tjänster möjliggörs.

Resultaten när det gäller affärsnyttor från både litteraturstudien och företagsbesöken sammanfattas i kapitel 4. Innan detta ska dock problem och utmaningar, samt de juridiska och finansiella aspekter som omnämnts i företagsbesöken presenteras i delkapitel 3.2 – 3.4.

Problem och utmaningar

Liksom de omnämnda affärsnyttorna har även de problem och utmaningar som framkom vid företagsbesöken strukturerats på ett liknande sätt som litteraturstudien. Detta innebär att fem grupperingar av problem presenteras nedan.

1. *Standards-relaterade utmaningar*: Standarder för Web Services, t.ex. säkerhet, är omogna, eller klarar inte att hantera organisationernas behov fullt ut. Dessutom kan befintliga system och protokoll hindra att nya versioner av standarder utnyttjas fullt ut.

2. *Interoperabilitetsutmaningar*: Systemändringar är ett problemområde, då det dels är svårt att ändra i gamla system på grund av den myckna affärslogik som finns i dem, och dels att det är svårt att veta hur ändringarna kommer att påverka andra system. Viss befintlig teknik klarar inte heller att hantera organisationernas krav på t.ex. schemanivåer.
3. *Utmaningar för nya möjligheter*: De flesta organisationerna trycker på att behov måste styra och inte främst tekniken. Organisationerna måste veta vilken information de ska exponera och varför, samt vilka aspekter och tjänster som är viktiga att prioritera.
4. *Utmaningar för kunskaper och skickligheter*: Det är svårt att veta hur mycket av t.ex. lösenordsutformning som kan styras gentemot kunder utan att orsaka problem i relationerna.
5. *Juridiska och finansiella utmaningar*: Det är svårt att ha en överblick över tjänsteägare och tjänstedefinitioner, liksom över Service Level Agreements (SLAs) och juridiska aspekter.

Juridiska aspekter

Organisationerna som besöktes menar att de juridiska aspekterna är viktiga, men svåra att hantera i många fall. Tre huvudsakliga aspekter lyftes fram:

1. *Kontrakt*: De är en väsentlig del i kontakten mellan företag och kund. Dessa kontrakt reglerar vilken information som kunden får fråga efter respektive få tillgång till. Behörighetsregister är vanliga att använda för att jämföra mot när förfrågningar om information inkommer.
2. *Identifiering och autentisering*: Förtroende (trust) är viktigt, men svårt. En orsak är att det är svårt att veta vad som ska identifieras när en tjänst ”anropas”, t.ex. om det ska vara en enskild eller en juridisk person. Dessa kan kräva olika lösningar. En vanlig ansats är att använda servercertifikat för identifiering, vilka jämför mot nämnda behörighetsregister.
3. *Offentlighetsrelaterad problematik*: Detta rör främst myndigheter, vilka ofta berörs av en mer omfattande lagstiftning än företag. Den aspekt som främst lyfts fram är att lagstiftningen inte har hållit samma utvecklingstakt som tekniken.

Betydelsen av att ta hänsyn till juridiska aspekter är tydlig för de deltagande organisationerna. Någon uttryckte till exempel att juridisk kunskap i framtiden kommer att vara viktig till exempel vid utveckling av nya Web Services.

Finansiella aspekter

Web Services har även finansiella implikationer för organisationer. Hittills har direkta vinster och nyttor varit svåra att mäta, men lyfts ändå fram som viktiga. De kommentarer som kommer från organisationerna i projektet har främst rört tre aspekter:

1. *Kostnader*: Web Services kostar att utveckla och underhålla, t.ex. i och med att de medför nya sätt att arbeta vilka måste föras in i organisationen, samt att uppföljningar inför underhåll tar tid. Utvecklingstiden för Web Services får inte heller vara för lång, då kostnaderna för densamma måste kunna motiveras.
2. *Vinster*: Kundanpassning är det främsta skälet som förs fram för ökade vinster. Samtidigt betonas att vinsten genom att använda Web Services är större än eventuella

risker som de måste ställas emot. Att skapa en Web Services före andra är också viktigt. Minskad administration har redan omnämnts som kostnadskapande. Tillgänglighet av en Web Services ses faktiskt som att det i sig ökar efterfrågan.

3. *Betalningsmodeller*: De modeller som nämnts är auktoriseringstjänst, engångsbetalningar samt årliga avgifter. Även betalning per enhet omnämndes som en möjlighet, men detta används inte p.g.a. svårigheter i till exempel att fastställa rättvisan i ansatsen.

Affärsnyttor och teknik för SOA

All verksamhet drivs utifrån att ett antal kriterier är uppfyllda, vi har här kallat dem ”Business drivers”. Dessa fokuserar på att ta fram den affärsnytta som eftersöks. Starkt förenklat kan man dela upp dessa kriterier i krav på

- Lönsamhet
- Konkurrenskraft/offentlig nytta

Uthållig lönsamhet i ett längre perspektiv är ett krav på all kommersiell verksamhet. Men de flesta verksamheter sträcker sig längre än så. Organisationen måste ha förutsättningar att växa och/eller förbättras. I den kommersiella verksamheten talar vi om **konkurrenskraft**, i offentlig verksamhet om **nytta**.

När det gäller tjänsteorientering har vi noterat ytterligare en bakomliggande faktor, som vi har kallat **branschpåverkan**. Det är när arbetssätt och standards utvecklas i en bransch. De behöver inte nödvändigtvis innebära lönsamhet eller nytta för en enskild aktör, men det finns ändå ett starkt branschtryck (eller via lagstiftning) som gör att den enskilda aktören måste anpassa sig. I vår modell får vi då följande business drivers.

- Lönsamhet
- Konkurrenskraft/offentlig nytta
- Branschpåverkan

Framgångfaktorer relevanta för SOA, t.ex med Web Services.

Vi har, mot bakgrunden av ett empiriskt material, studerat vilka olika framgångsfaktorer som tycks stödja dessa tre business drivers. Vi har då kommit fram till nedanstående tabell.

Perspektiv för affärsverksamheten respektive offentligt service	
Generella Business drivers	**Underliggande kritiska framgångsfaktorer**
Lönsamhet	Högre intäkter
	Lägre kostnader
Konkurrenskraft/offentlig nytta	Service
	Exponerat sortiment
	Ledtider
	Time to market
	Fusioner, uppköp och samverkan med partners
Branschpåverkan	Standardisering av affärsprocesser

Varje del i den högra kolumnen i tabellen kommer nedan att diskuteras.

Lönsamhet – intäkter

Web Services möjliggör effektivisering av IT-utveckling och underhåll, bl.a. genom de dokumenterat effektiva utvecklingsmiljöer som finns. Affärsutvecklare och kravställare blir också effektivare, speciellt om man byggt upp en bas av grundläggande tjänster som lätt kan kombineras till nya tjänster enligt ett byggklosstänkande av typ Lego. Då friställs mycket av företagets intellektuella resurser från detaljarbetet med att specificera och bygga tjänster till mer strategiska uppgifter.

Lönsamhet – kostnader

Vi kan tydligt se hur samverkan med partners effektivt sker genom tjänster. Dessa består ofta av existerande system som brutits upp i tjänster och fått generella gränssnitt. Man får en hög ambitionsnivå till ett lågt pris. Man kan vidare konstatera att utvecklingskostnaderna för t.ex. Web Services i de flesta fall rapporteras till bråkdelar av kostnaden för alternativ teknik.

Konkurrenskraft – exponerat sortiment

Kunder väntar sig i dag att ha direkt tillgång till information om leverantörernas sortiment och helts kunna lägga en order utifrån detta. SOA medger att existerande lagersystem omvandlas till tjänster som kan anropas globalt. Att snabbt kunna genomföra denna typ av förändring är en fråga om överlevnad för många företag.

Konkurrenskraft – ledtider

Att kunna leverera en vara på en bestämd, kort tid och i det utförande som kunden önskar med bindande besked vid ordertillfället är ett viktigt konkurrensmedel. Tekniskt och organisatoriskt

kräver det dock omfattande förändringar mot traditionell organisation. De flesta företag måste för att kunna priskonkurrera utnyttja partners. Det gäller att ha en realtidskoppling till dessa, som dessutom enkelt kan läggas om till andra partners eller annat sortiment. SOA är här den enda effektiva lösningen, som vi har sett inom ett flertal mycket lönsamma företag. Logiskföretagen är här föregångare med tjänster riktade till såväl mottagare som avsändare där alla parter har hög insyn i transporternas status.

Konkurrenskraft – time to market

Tiden från idé tills produkten finns tillgänglig för kunderna blir allt kortare. Inte minst inom IT och telecom där livslängden för vissa produkter bara kan vara något år. Det är då givetvis centralt att snabbt kunna ta fram alla de system som behövs. Ett SOA som bygger på ett stort antal generella grundläggande tjänster. Återanvändning är nödvändigt för att gå i land med detta liksom en effektiv och säker utvecklingsmiljö. Det gäller att kunna utnyttja äldre program samt att effektivt bygga nya tillämpningar på ett sådant sätt att tjänsterna kan användas på många plattformar.

Konkurrenskraft – fusioner och uppköp

Dagens företagsklimat förutsätter samverkan antingen genom partnerskap eller genom samgående. En av de stora problemen vid samgående är traditionellt att få IT att samverka. Övergång till SOA underlättar detta genom att de äldre rutinerna får tjänstegränssnitt och kan utnyttjas över nätet. En förutsättning för att detta skall fungera är att man har överensstämmande begrepp inom de deltagande organisationerna, vilket brukar vara svårt att uppnå.

Branschpåverkan – standardisering av affärsprocesser

Inom branschorganisationer och standardiseringsorgan har man insett att en förutsättning för rationell produktion i de flesta branscher är att man kan utbyta information på ett så enkelt sätt som möjligt. Detta började med standardisering av fakturor och liknande dokument och resulterade i bl.a. Edifact. Centralt är att företag som kommunicerar har en gemensam begreppsapparat åtminstone när det gäller de viktigaste begreppen. Inom många branscher har standardiseringsarbetet nått långt både globalt och lokalt. Ett globalt arbete är ebXML som skall skapa en utgångspunkt för E-handel. Lokalt finns många branschvisa initiativ, som den samverkan mellan livförsäkringsbolag och försäkringsmäklare som finns i Sverige. Ett företag kan knappast stå utanför sådana standardiseringar om man vill finnas kvar i branschen.

Hur passar SOA för olika typer av organisationer

I sin bok "The Discipline of Market Leaders" av Michael Treacy and Fred Wiersema identifierar författarna tre affärsstrategier som skapar mycket framgångsrika företag. Företagen är antingen:

- **Kundfokuserade**: Dessa företag känner sina kunder väl och uppfyller de flesta krav. Exempel är IBM och Avis.
- **Produktfokuserade**: Nyskapande företag som ligger i utvecklingsfronten. Exempel är Ericsson och BMW.
- **Produktionsfokuserade**: Dessa företag styr sin produktionsapparat effektivt och skapar låga kostnader. Exempel är McDonald's och IKEA.

Låt oss se hur dessa strategier passar in i vår modell. De röda fälten symboliserar var respektive framgångsfaktor är särskilt aktuell för respektive företagstyp.

Perspektiv för affärsverksamhetens respektive offentligt service		**Fokus på**		
Generella Business drivers	**Underliggande kritiska framgångsfaktorer**	**Kund**	**Produkt**	**Produktion**
Lönsamhet	Högre intäkter			
	Lägre kostnader			
Konkurrens-kraft/offentlig nytta	Service			
	Exponerat sortiment			
	Ledtider			
	Time to market			
	Fusioner, uppköp och samverkan med partners			
Bransch-påverkan	Standardisering av affärsprocesser			

Som framgår av modellen har den kundfokuserade organisationen mycket att vinna på tjänste-orientering.

Vilka är de grundläggande stegen till tjänsteorientering?

Vi har kunnat konstatera att tjänsteorientering har ett antal grundläggande steg. De är:

I. Tillhandhålla tjänster över huvud taget (t.ex. Skatteverket som går över från deklarationsblanketter till en deklarationstjänst).
II. Exponera information (t.ex. att lägg ut information om sina produkter)
III. Utbyta information (t.ex. med partners eller kunder)
IV. Automation (dvs. att låta datoriserade tjänster ersätta handläggare)

I vår modell ser vi nu att tjänsteorientering främst gynnar konkurrenskraft/offentlig nytta.

Perspektiv för affärsverksamhetens respektive offentligt service		**Fokus på**			**Grundläggande steg till tjänsteorientering**	
Gene rella Business drivers	**Underliggande kritiska framgångs-faktorer**	**Kund**	**Produkt**	**Produktion**		
Lönsamhet	Högre intäkter					
	Lägre kostnader					
Konkurrens-kraft/offentlig nytta	Service				**I** tillhandahålla	**II** Exponera information
	Exponerat sortiment					
	Ledtider				**IV** Automat-isering	
	Time to market					
	Fusioner, uppköp och samverkan med partners				**III** Utbyta information	
Bransch-påverkan	Standardisering av affärsprocesser					

Affärsstrategier som stöder de kritiska framgångsfaktorerna,

I detta avsnitt relateras ett antal affärsstrategier som stöder de förut beskrivna kritiska framgångs-faktorerna.

Lönsamhet

Bättre nyttjande av intellektuella resurser för högre intäkter: Strategiskt riktiga, marknadsanpassade affärsprocesser ger förstås **högre intäkter**. Det intellektuella kapitalet är en trång resurs i de flesta organisationer. Affärsutvecklare och ansvariga på olika nivåer är ofta upptagna med att förvalta de befintliga affärsstrategierna. Studier av företagsledningar, speciellt lågpresterande sådana, visar att man ägnar all tid att administrera befintliga strategier. Man lämnar därmed fältet fritt för konkurrenter som är uppdaterade på nya trender. Ett välkänt exempel är Facits mekaniska räknare. En effektiv och flexibel utvecklingsorganisation- och teknik avlastar affärsutvecklarna

och ansvariga det löpande och möjliggör för dem att ägna tid och organisationens strategiska frågor. Detta har beskrivits i boken Out of the box av John Hagel III.

Integration med partners ger lägre kostnad: Konkurrenstrycket och krav på höga servicenivåer medför ett ökat kostnadstryck, men det är inte alltid möjligt att öka kostnaderna. En vanlig utväg i dag är att samverka med partners. Inom industrin är detta vanligt – man utnyttjar en krets av underleverantörer. Samverkan med partners är nu även vanlig inom tjänstesektorn. Integration kräver för det mesta samverkande system. Det är viktigt att detta inte tar för mycket tid och resurser och att kopplingen kan förändras med affärsläget. SOA som teknik uppfyller dessa krav.

Perspektiv för affärsverksamhetens respektive offentligt service		
Generella Business drivers	**Underliggande kritiska framgångsfaktorer**	**Exempel på affärsstrategier som stöttar dessa kritiska framgångsfaktorer och skapar flexibilitet**
Lönsamhet	Högre intäkter	Bättre utnyttjande av intellektuella resurser
	Lägre kostnader	Integration med partners
Konkurrens-kraft/offentlig nytta	Service	Självbetjäning genom tjänster
	Exponerat sortiment	
	Ledtider	globala tjänster för lager och logistik
	Time to market	Snabbare utveckling av affärsprocesser/strategier
	Fusioner, uppköp och samverkan med partners	Integration av affärsprocesser
Bransch-påverkan	Standardisering av affärsprocesser	Utnyttja branschstandards för bibehållen (eller ökad) konkurrenskraft

Konkurrenskraft

Självbetjäning genom tjänster exponerar sortimentet och ger bättre service: Den som byggt upp ett antal ändamålsenliga tjänster för att kommunicera med kunderna kan kombinera dessa till nya tjänster. Det blir då möjligt att enkelt skapa skräddarsydda tjänstepaket för särskilda behov eller kunder. En vanlig strategi, som visat sig framgångsrik, är att exponera sitt sortiment direkt ut mot kunden via tjänster och därmed få en radikalt förhöjd kundnytta. Kunden kan t.ex. se om en viss artikel finns i lager, direkt lägga en order, osv.

Globala tjänster för lager och logistik ger kortare ledtider: Inom tillverkande industri och inom logistikföretag ser man att tjänsteorienterade arkitekturer är mycket framgångsrika. När hela kedjan är tjänstebaserad kan man optimera logistikprocessen och kunden kan anpassa sig till exakta tidpunkter för leverans.

Snabbare utveckling av affärsprocesser/strategier ger snabbare time-to-market: SOA ger större intäkter genom ett bättre utnyttjande av det intellektuella kapitalet har vi hävdat. Detta sker bl.a. genom att SOA snabbar upp utvecklingsprocessen. Man kan kombinera olika tjänster och skapa nya.

Samverkan genom integration av affärsprocesser: Vi har ovan hävdat att integration med partners ger lägre kostnad. Integration möjliggör framför allt bättre möjlighet till kundanpassning genom att ett kundärende kan genomföras i ett moment. Kvalitén höjd genom att samma uppgifter inte registreras på flera ställen.

Branschpåverkan

Branschvis standardiserade affärsprocesser: En förutsättning för att kunna konkurrera är ofta att organisationen anpassar sig till branschstandards. Det kan gälla leveranser inom bilindustrin, kundportaler inom försäkring, betaltjänster, etc.

SOA - samband med teknik och arkitektur

Alla business drivers, framgångsfaktorer och affärsstrategier som nämns i tabellen drar nytta av att SOA, t.ex. implementerat med Web Services, byggs enligt den systemfilosofi som återges i rutan nedan. Det är fråga om utnyttjande av **standards (A)** och utnyttjande av **en arkitektur med nav (B)**[1]. Dessa faktorer tillsammans med en **effektiv utvecklingsmiljö** gör att mindre resurser går åt till systemutveckling och underhåll, samt att det går snabbare.

A. Faktorer som bl.a. är beroende av **standards**
    1. Åtkomst till tjänster oberoende av vilken plattform de implementerats på
    2. Byggklosstänkande medger hög anpassningsbarhet i tjänsteutbudet
    3. Återanvändning underlättas
    4. Gränssnittet är oberoende av underliggande logik, vilket bl.a. möjliggör att tjänsterna får ett oberoende av de ofta komplexa grundsystemen

B. Faktorer som bl.a. är beroende av **arkitektur med nav**
    1. Generell och enkel åtkomst till äldre rutiner
    2. Åtkomst till tjänster möjlig utan oönskad insyn i logik, data, etc.

C. Egenskaper som förutom faktorerna A och B är beroende av **effektiv utvecklingsmiljö**
    1. Mindre resurser krävs för utveckling av tjänster
    2. Mindre resurser krävs för underhåll av tjänster
    3. Snabbare utveckling av tjänster

[1] ”Branschpåverkan” är speciellt, där gäller bara punkterna A1 t.o.m. A3

Perspektiv för affärsverksamhetens respektive offentligt service			**Egenskaper hos Web Services i en tjänstebaserad arkitektur**	
Generella Business drivers	**Underliggande kritiska framgångsfaktorer**	**Exempel på affärs-strategier som stöttar dessa kritiska framgångsfaktorer och skapar flexibilitet**	**Standards och nav** (Faktorerna A och B)	**Effektivitet** (egenskaperna C)
Lönsamhet	Högre intäkter	Bättre nyttjande av intellektuella resurser	**Alla**	**Alla**
	Lägre kostnader	Integration med partners		
Konkurrens-kraft/offentlig nytta	Service	Självbetjäning genom tjänster		
	Exponerat sortiment			
	Ledtider	globala tjänster för lager och logistik		
	Time to market	Snabbare utveckling av affärsprocesser/strategier		**C3**. (Snabbare utveckling av tjänster)
	Fusioner, uppköp och samverkan med partners	Integration av affärsprocesser		**C2**. (Mindre resurser krävs för underhåll)
Bransch-påverkan	Standardisering av affärsprocesser	Utnyttja branschstandards för bibehållen (eller ökad) konkurrenskraft	Endast **A1 – A3**	**C1**. (Mindre resurser krävs för utveckling)

Bara för integration med partners: kostnader för standardprodukter. Vi har studerat dessa egenskaper inom företagen inom Serviam och inom ett antal andra företag och man kan konstatera att överensstämmelsen är stor mellan företagen och med modellen.

Teknik för SOA

Här skall kort kommenteras hur tekniken påverkar i fallet Web Services.

A1 Åtkomst till tjänster oberoende av vilken plattform de implementerats på

Detta är en av de grundläggande egenskaperna hos Web Services.

A2 Byggklosstänkande medger hög anpassningsbarhet i tjänsteutbudet

En stor fördel med tjänster är att de kan kombineras till nya tjänster. Det enklaste sättet är när en tjänst skapar sitt eget fönster i dialogen, då är det bara att kombinera. Mer intressant är när olika tjänster kan anropa varandra och/eller utnyttjas i ett tillämpningsunikt skal. Man får ett oberoende av var tjänsten befinner sig.

A3 Återanvändning underlättas

Återanvändning kräver ett långsiktigt tänkande. Det gäller arkitektur, regler, organisation, kvalitetssäkring och informationsspridning. En förutsättning är förstås att det finns generella gränssnitt. Ett sådant är SOAP vilket underlättar återanvändning av Web Services.

A4 Gränssnittet är oberoende av underliggande logik

Så att tjänsterna får ett oberoende av de ofta komplexa grundsystemen. Inom objektorientering är det meddelanden som utväxlas via metoderna. Det inre av ett objekt, såsom data och logik döljs. Tjänster via Web Services har motsvarande egenskap. När tjänster utvecklas som en del av befintliga system med traditionell teknik blir tjänsten beroende av systemet och måste förvaltas samtidigt. När man använder SOA och löst kopplade gränssnitt begränsas beroendet till de delar som direkt exponeras i tjänsten.

B1 Generell och enkel åtkomst till äldre rutiner

Integration har alltid komplicerats av mängden äldre program. Att ersätta de gamla programmen med nya är nästan aldrig ekonomiskt möjligt. Många företag har istället provat att göra programmen till tjänster via Web Services gränssnitt. Detta går även att tillämpa på system som körs i batch. Det finns två huvudprinciper. Antingen använder man de gamla terminalgränssnitten och konverterar vart och ett till en tjänst (vilket har vissa problem). Ett bättre (och mer kostnadskrävande) sätt är att skriva om användargränssnittet som en tjänst. Allra enklast är det om man redan har en treskiktsarkitektur där användargränssnittet finns som en separat del.

B2 Åtkomst till tjänster möjlig utan oönskad insyn i logik

Integration har tidigare inneburit att direktkontakt måste ske mellan dem som ansvarar för ingående program, som till stor del måste öppnas för parterna. Gränssnittet som SOAP möjliggör åtkomst och samverkan endast på de punkter som önskas.

C1 Mindre resurser krävs för utveckling av tjänster.

Många företag har gått över helt och hållet till att utveckla nya tillämpningar i Web Services, då man anser att det långsiktigt är rationellt och för att det rent programmeringsmässigt är effektivt. Utfallet är beroende av vilken teknisk miljö som används för att framställa och underhålla tjänsterna. Det har dock visat sig att även de som använt ett minimum av programstöd, utvecklat tjänster relativt effektivt.

C2 Mindre resurser krävs för underhåll av tjänster.

SOA innebär en modularisering av systemen. Den stora vinsten är att förändringar många gånger kan isoleras till en viss modul utan att påverka omgivningen. Andra gången kan förändringar ske genom att kombinera moduler på ett nytt sätt. Vidare kan kopplingen till äldre rutiner ”legacy” göras rationellare.

C3 Snabbare utveckling av tjänster.

Det är väl dokumenterat att flera av de tekniska miljöer som används för att bygga Web Services är effektivare än äldre tekniker, bl.a. för att man strävat medvetet efter enkelhet. Detta gör att även de som endast använder de grundläggande byggklossarna SOAP och XML snabbt når resultat.

Standardprodukter som ERP-system, CRM-system och liknande bygger i dag allt mer på Web Services gränssnitt. En kund kan enklare integrera med dessa och systemen kan köpas mer modulärt till en lägre kostnad. Framförallt sjunker anspassningskostnaden

Referenser

Azzara, C. (2002), *Web Services: The Next Frontier: A Primary Research Opportunity Study*, Topical report: Enterprise Findings, Hurwitz Group, Inc.

Colan, M. (2003), "The business value of Web Services: improving IT stability, agility and flexibility – Achieving ROI", *WebSphere Developer's Journal*, January 2003, Sys-Con Publications Inc. and Gale Group

Dunn, B. (2003), A Manager's Guide to Web Services, *EAIJournal*, January 2003, pp.14-17

Estrem, W. (2003), "An evaluation framework for deploying Web Services n the next generation manufacturing enterprise", *Robotics and Computer Integrated Manufacturing*, 19 (2003), pp.509-519

Govern, M. and Weagraff, S. (2003), *Web Services: Provider Threat or Opportunity?*, Convergys Corporation, September 2003

Hagel, J. (2003), The CEO and IT Relationship: Harnessing the Value of Web Services Technology, *Internetworld*, April 1, 2003, available at: http://www.internetworld.com/

Murphy, T. and Stoyanova, D. (2003), *Optimize the Business Value of Web Services*, Computer Associates International Inc., CA World, July 13-17 2003, Las Vegas, Nevada, USA.

Race, S. (2003), "What is the Value of Web Services?", *EACommunity.com*, as is: 2003-12-10, available at: http://www.eacommunity.com/articles/openarticle.asp?ID=1820

Roby, T. (2003), *Web Services: Universal integration powers seamless, long sought after business servives*, Accenture, April 2003, available at: http://www2.cio.com/consultant/report1641.html

Smolnicki, J. (2003), *How XML and Web Services will change your business*, PriceWaterhouseCoopers, as is: 2004-01-08, available at: http://www.pwcglobal.com/Extweb/ncinthenews.nsf/docid/2C9CB295270D752DCA256DD5006D122D

Wong, S. (2002), Success With Web Services, *EAIJournal*, February 2002, pp.27-29.

Rättsliga frågor och Web Services

Ny teknik utvecklas sällan av jurister. Rättsliga problem upptäcks ofta när de redan har inträffat. Juridiken har också historiskt haft problem med att diskutera olika tekniska lösningar i detalj. Istället har det vuxit fram en ibland överdriven tro på det teknikneutrala lagstiftningsperspektivet. Huvudtanken är att det inte ska vara nödvändigt att kunna teknik för att stifta lagar och att lagarna inte ska behöva förändras när tekniken gör det. Detta är en berömvärd ambition, men orsakar även en del problem. Ny teknik kräver kanske inte nya lagar, men definitivt nya tekniska tolkningar. Lagar har också effekt på tekniknivå. Detta kapitel presenterar kort en problemkatalog för Web Services ur ett rättsligt perspektiv. Först kommer några ord om vad Web Services är ur ett rättsligt perspektiv, innan problemkatalogen beskrivs.

Web Services ur ett rättsligt perspektiv

Ur ett rättsligt perspektiv är Web Services en dubbel företeelse. Det är å ena sidan ingenting nytt eftersom tjänster inte är främmande för juridiken i form av t.ex. olika typer av avtal. Å andra sidan är det fortfarande obruten mark för rättslig forskning, eftersom Web Services-arkitekturen egentligen inte hittills har tolkats i rättsliga termer. Dessutom kan Web Services förutom att vara tjänster även ses som produkter. Några oklarheter rör om Web Services kan patenteras, om de skyddas av samma upphovsrätt som datorprogram, om de omfattas av samma skydd som databaser, och så vidare. Problematiken kring komponentbaserad utveckling aktualiseras också, t.ex. i frågor som äganderätt och ansvarsfördelning. I befintliga definitioner av Web Services nämns inte heller juridiska personer, vilka ändå är väsentliga ur ett rättsligt perspektiv. De juridiska problemen är många, men med rätt typ av rättsligt förberedande kan riskerna minimeras.

Problemkatalog

I detta avsnitt kommer åtta problemtyper att kort presenteras, där var och en kompletteras med en viktig fråga att fortsätta reda ut. Avsnittet avslutas med ett stycke med möjliga lösningar. Problemen är:

1. *Ansvarsfrågor*: Web Services beskrivs ofta som ett trepartsförhållande mellan köpare, säljare och mäklare. I varje sådant förhållande uppkommer frågor kring vem som ansvarar för vad, när och under vilka förhållanden. Det är därför viktigt att reda ut alla ansvarsförhållanden i en Web Services-baserad arkitektur innan den införs och används.

2. *Personlig integritet*: I informationssamhället blir personlig integritet allt viktigare, t.ex. på grund av ett ökande antal identitetsstölder och bedrägerier. Lagar inom området ställer krav på säker systemdesign. Frågan är om en ny och relativt oprövad arkitektur kan uppfylla dessa krav. Det är därför viktigt att de olika frågor om personlig integritet som uppkommer behandlas seriöst, diskuteras och prövas mot de lagar som finns på området innan en Web Services-arkitektur införs.

3. *Immaterialrätt*: Några viktiga frågor i detta hänseende är: hur Web Services kan skyddas (Patent? Upphovsrätt?) och vad som händer om en Web Services hanterar skyddat material på olika sätt (Vem ansvarar?). Licensproblem har redan börjat uppkomma. Problemen kan uppkomma på olika nivåer, t.ex. arkitekturen i sig, den faktiska Web

Services och innehållet/transaktionsdatan. Det är därför viktigt att utreda vilka olika rättsliga skyddsformer Web Services omfattas av innan arkitekturen i allmänhet införs, men också innan vissa Web Services börjar användas och innan visst innehåll tillhandahålles.

4. *Avtalsrätt*: Avtalsrätten erbjuder en viktig möjlighet för alla systemdesigners att reformera och bygga in olika rättsliga lösningar i sina system. Dessa möjligheter behöver tas tillvara. Det är därför viktigt att utreda vilka olika avtal som uppkommer vid användningen av en Web Services och hur dessa avtal sluts samt med vilket innehåll innan Web Services tas i bruk.

5. *Offentligrätt*: När Web Services-användning sker med staten som avsändare eller användare så uppkommer utöver redan omnämnda aspekter problem som har att göra med en offentlig aktörs rättigheter och skyldigheter. Det rör t.ex. frågor som offentlighetsprincipen, filtrering, samt förvaring och lagring hos annan part. Det är därför viktigt att förstå hur Web Services-arkitekturer påverkas av olika typer av offentligrättsliga frågor.

6. *Offentlig upphandling*: En specifik fråga som blir aktuell är om det går att bygga system och kräva att alla som vill delta i offentlig upphandling elektroniskt använder t.ex. Web Services-arkitekturen. Systemen som ska kunna användas måste även kunna hantera anbuden på ett korrekt sätt. Det är därför viktigt att förstå och klarlägga vilka krav som lagar om offentlig upphandling ställer på användningen av ny teknik.

7. *Lagen om elektronisk handel och informationssamhällets tjänster*: Eftersom Web Services kommer att inordnas i existerande mönster är det värdefullt att veta hur erbjudandet av en Web Services kommer att behandlas med tanke på existerande rätt. Det är därför viktigt och nyttigt att analysera Web Services utifrån lagen om elektronisk handel och begreppet informationssamhällets tjänster, för att förstå vilka informationskrav som kan komma att ställas på Web Services-arkitekturen. Detsamma kan gälla de fall där det kan handla om sådana avtal som omfattas av lagen om distansavtal.

8. *Elektroniska signaturer och Web Services*: Säkerheten kommer i många fall att vara beroende av elektroniska signaturer. Tekniklösningarna måste då ta hänsyn till befintlig lagstiftning. Dagens säkerhetsramverk tar ofta inte alls upp juridiska aspekter. Det är därför viktigt att klarlägga hur lagen om elektroniska signaturer påverkar utformningen av Web Services-arkitekturer som utformas för säker kommunikation med certifikat och elektroniska signaturer.

Många frågor har hittills väckts i texten. Dessa visar på områden där friktion kan uppstå mellan tekniska utvecklare och rätten. I varje särskilt fall bör de områden som nämnts analyseras med kvalificerad juridisk hjälp. Några allmänna åtgärder som kan vidtas är:

- *Checklistor*: Dessa kan utformas för diskussion av rättsliga problem och för att utveckla medvetenheten om dem.

- *Sedvana och handelsbruk*: Genom att ta fram branschgemensamma avtalsvillkor eller uttalanden kan en begynnande sedvana etableras på detta område. Mycket av den rättsliga osäkerheten kan då hanteras med relativt enkla medel. Framför allt gäller detta områden där lagen är öppen för tolkning.

- *Standardavtal*: Det borde vara möjligt att ta fram en mal för ett standardavtal som rör nyttjande, erbjudandet och integrationen av olika Web Services, kanske till och med i form av en Web Services. Kopplingar kan också göras mellan avtal och tjänstekomponenter som föreslagits på området, vilket skulle tydliggöra vilka avtal komponenter lyder under.

Juridiska aspekter kring fyra exempelfall

Som ett led i projektet Serviam inbjöds företagen till ett seminarium för att diskutera juridiska aspekter kring fyra exempelfall av Web Services (Web Services). På seminariet deltog både personer med teknisk kompetens och personer med juridisk kompetens. I varsitt avsnitt kommer en sammanfattning av diskussionen kring respektive fall att presenteras, innan en kort avsnitt med tentativa slutsatser och konstruktiva förslag avrundar kapitlet.

Fall 1

Fallbeskrivning: Banken SEB har värdefull information av olika slag. Ett annat företag kontaktar SEB och frågar om de kan få tillgång till informationen. SEB kan göra så via en Web Services, men vad gäller avtalsmässigt?

En första anmärkning är att eftersom det är en bank inblandad så finns en del speciella problem på grund av särreglering. Däremot liknar problematiken även hur en bank överhuvudtaget får behandla information. Följande punkter belyser viktiga synpunkter gällande fall 1:

1. Informationens *föremål* är oerhört viktigt, t.ex. om föremålet är en kund.

2. Processen måste *modelleras i detalj* för att få en helhetsbild av problemet och därefter kunna belysa lagligheten i varje enskilt steg.

3. Utlämnande av information måste ske mellan *två identifierade parter* och inte anonymt, vilket gör t.ex. elektronisk identifiering centralt.

4. *Ansvaret* för den lämnade informationen måste utredas grundligt. Här skiljer sig Web Services från vanliga webbsidor, då Web Services kan upplevas som mer seriös och pålitlig.

5. *Upphovsrättsliga aspekter* på den lämnade informationen måste diskuteras, t.ex. gällande immaterialrätt.

Seminariet drog vissa slutsatser som är viktiga att tänka på för dem som arbetar med webb-tjänster. Följande generella slutsatser kan dras om utformningen av Web Services:

- Web Services ska inte utnyttjas i avtalslöst tillstånd.

- Web Services bör byggas så att de endast kan utnyttjas av klart identifierade parter.

- Omfattande modellering krävs av varje tjänst för att kunna fastställa rättsliga aspekter att beakta.

Slutsatserna är intressanta ur ett affärsnytteperspektiv av flera skäl. För det första innebär Web Services en höggradig automatisering och därmed anonymisering av tjänsten. Detta skapar identifieringsproblem. Web Services innebär också en internationalisering, där autonoma agenter kan söka upp dem. Med ett interoperabelt och globalt användbart informationsformat ökar också ansvaret kring tjänsten, och mer eftertanke behövs angående vad tjänsten får användas till. Slutligen, även om en tjänst erbjuds gratis betyder inte det att den som erbjuder den är fritagen från ansvar.

Fall 2

Fallbeskrivning: Skatteverket hanterar årligen miljontals deklarationer. Nu vill det göra inlämningar av deklarationer on-line, kanske i flera steg. Vilka legala krav ställs på en sådan Web Services?

Denna verksamhet finns idag. Frågorna som aktualiseras är delvis av en annan natur än i fall 1. Skatteverket är en myndighet och lyder därför under en omfattande förvaltningsrättslig regelsamling. Följande punkter belyser viktiga synpunkter gällande fall 2:

1. *Identifieringsarkitekturer* är mycket viktiga även ur ett rent medborgerligt rättssäkerhetsperspektiv.

2. Arbetet med deklarationer beror också av det betydande arbetet med *formkrav*, och lagändringar är att vänta.

3. Frågor om *när ett ärende är inkommet* och vilka *eventuella misstag* som begåtts måste behandlas på rätt sätt.

Det handlar här om att vänta på en anpassning av lagstiftningens formkrav samt en utbyggnad av identifieringstjänsterna på ett rimligt sätt. Gällande Web Services är de särskilda rättsliga frågorna färre än i fall 1. Dock kan graden av automatisering diskuteras, då det finns särskilda förbud mot vissa former av automatiska förvaltningsbeslut, t.ex. i dataskyddsdirektivet.

Fall 3

Fallbeskrivning: Volvo ska skapa en Web Services ut mot kunder som vill beställa reservdelar till t.ex. lastbilar. Volvo ska sedan via kundens Web Services gå in och titta i dennes lager för att se vad de har hemma. Sedan ska Volvo kontakta leverantörerna (kanske via Web Services) som i sin tur ska skicka delen till kunden. Ett problem är hur avtalen ska se ut i allt detta med tre parter inblandade. Om kunden avbeställer men varan redan gått iväg från leverantören? Om någon av Web Services går ner? Alla kopplingar bakåt till systemen inom Volvo?

Detta exempel är mycket komplext och hanterar ett område som bekymrar många företag idag – integrationsfrågor. Det finns EDIFACT-system som kan hantera detta, men Web Services skulle kunna minska kostnaderna. Följande punkter belyser viktiga synpunkter gällande fall 3:

1. Huvudfrågan är *allokering av ansvar* mellan parterna, och att avtal finns är därför viktigt.

2. *Befogenhetsproblematik* är också relevant, t.ex. att kunna undersöka vem som agerar i systemet.

3. *Jurisdiktionsfrågor* aktualiseras i fallet, då Volvo är ett multinationellt företag.

Den viktigaste insikten fallet väckte var att graden av anonymisering i integrationsfallen inte är sådan att automatisering leder till problem. Fallet innebär nästan inga särskilda problem gällande Web Services. Däremot erbjuder Web Services särskilds problem när de används i system som öppnar upp för interaktion mellan många parter tillfälligtvis, vilket sällan sker i fall som detta.

Fall 4

Fallbeskrivning: SEB-Skatteverket-privatperson via program som skrivs av programvaruföretag. En leverantör av program för ekonomisk planering eller deklaration körs av privatperson. Programmet använder Web Services på SEB för att nå uppgifter om kundens värdepappersinnehav, kontoställningar,etc . En annan Web Services går mot skatteverket för att hämta uppgifter om skattesatser, kvarskatt, skattekonto, etc. Vad gäller vid fel, upphovsrätt, programändring, etc.?

Problemet påminner lite om diskussionen kring det amerikanska programmet Quicken. Problemen är många:

1. Vilka skyldigheter en myndighet har att vara kompatibel med en viss teknisk lösning och att lämna ut information som används i olika Web Services i ett format som kan utnyttjas kommersiellt.

2. Offentlighetsprincipen utesluter inte försäljning av offentlig information. Detta har ännu ej reglerats, även om tankar funnits på det.

Formatfrågor här intressanta och relevanta på grund av t.ex. automatiseringsmöjligheter för Web Services. De har också stora kostnadskonsekvenser, vilket särskilt blir en akut fråga gällande Web Services.

Tentativa slutsatser, öppna frågor och konstruktiva förslag

Fallen och analysen därav visar att det inte alltid går att visa på klara och tydliga problem direkt relaterade till Web Services, utan mer till Internet-användning. Tre tendenser kan dock påpekas där Web Services förstärker och har rättsliga effekter: Automatisering; Anonymisering; samt Förpackning och formatering. Slutsatsen är att rättsliga frågor är mycket viktiga att tränga in i.

På seminariet väcktes frågan om förebyggande aktiviteter för hantering av rättsliga problem. Två förslag framlades:

1. Utveckla standardavtal av olika slag.

2. Utveckla olika enkla modeller för avtalsslut vid användning av Web Services.

Tvärvetenskapliga arbeten liknande det som företagits i Serviam bör dessutom fortsätta att bedrivas i olika fora.

Del II

Arkitektur och processer

Martin Henkel
Stockholms Universitet/KTH

Lars Wiktorin
IT Plan

Intresseområden

Delprojektet arkitektur har valt att studera tre områden närmare. Dessa är

- Typarkitekturer
- Processhantering
- Arkitekturmönster

Typarkitekturer.

Avsikten med detta intresseområde är att via praktikfall identifiera vanliga arkitekturlösningar och analysera deras för- och nackdelar. Detta ger en inblick i hur man i några olika situationer bör förhålla sig till användning av Web Services. Praktikfallen har valts bland de representanter från näringsliv och förvaltning som medverkar i Serviam.

Processhantering

Arbetet inom detta område är belyser ett nytt område för web service arkitektur – användningen av processer för att koordinera exekveringen av tjänster. Eftersom området är relativt nytt så har fokus varit på att beskriva användningsområdet för exekverbara processpråk (t ex Business Process Execution Language for Web Services, BPEL4WS), samt att beskriva de begrepp som används för att konstruera exekverbara processer. Som en del i arbetet har bland annat notationen Business Process Modeling Notation, BPMN används för att beskriva processer. Resultatet av arbetet presenteras i form av kriterier för när ett processbeskrivningsspråk bör/kan användas samt definitioner och beskrivningar av de begrepp som behövs när exekverbara processer konstrueras. Arbetet med detta intresseområde fortsätter under projektets andra år.

Mönster

Detta intresseområde inom delprojektet arkitektur kartlägger och dokumenterar existerande ”best practices” i form av arkitekturmönster. Till skillnad från en typararkitektur beskriver ett mönster inte en komplett arkitektur, utan snarare en mindre del av en arkitektur. Resultatet av kartläggningen är dels en indelning i sju olika kategorier av ”best practices” inom Web Service arkitektur, dels en katalog med dokumenterade mönster. Katalogen är tillgänglig från projektets hemsida.

Arkitektur och Web Services

Web Services är en teknik för att publicera, finna och anropa tjänster i ett nätverk av interagerande datornoder. Det är en tillämpning av ett mer generellt sätt att se på strukturen

hos datasystem – tjänsteorienterad arkitektur (service oriented architecture, SOA). Här ger vi en kort beskrivning av dessa begrepp som en inledning till detta avsnitt.

Vad är arkitektur?

Med arkitektur avser vi enkelt uttryckt strukturen hos ett (data)system. Det innebär att vi beskriver systemets uppbyggnad i form av komponenter samt hur dessa samverkar med varandra. Till detta fogar vi även de principer som styr den aktuella strukturering en.

För att till fullo förstå ett systems arkitektur fordras att den beskrivs ur flera synvinklar. Detta kan ske med olika modeller, t ex komponenternas inbördes förhållande (en statisk modell) eller deras samverkan (dynamisk modell, flöden).

Komponenter och tjänster

En komponentmodell fokuserar på ett systems uppdelning i funktioner och hur dessa fördelats mellan ett antal samverkande enheter. Detta är tanken med objektorientering och dess vidareutveckling till komponentbaserad teknik.

Ett exempel på en komponentorienterad modell är att betrakta ett orderhanteringssystem som uppbyggt av komponenterna Produkt, Order, Kund. Dessa inkapslar var för sig sina egna data med tillhörande funktioner på i stort sett samma sätt som för objektorientering. Komponenter är dock mer omfattande än objekten inom objektorientering.

Ett annat sätt att se på samma system är att följa ett flöde genom systemet. De olika komponenterna ger var för sig sitt bidrag till flödet genom att de utför någon delfunktion. Man kan då se flödet som en kombination av tjänster som levereras av de medverkande komponenterna.

Att se på komponenter som leverantörer av tjänster flyttar fokus från komponentens funktionsinnehåll till vad den kan leverera till omgivningen i form av tjänster. Detta kan tyckas som en subtil skillnad, men flyttar intresset på ett avgörande sätt från komponentens innehåll till dess gränssnitt mot omgivningen. Det är detta synsätt som är grunden för en tjänsteorienterad arkitektur.

Tjänster och arkitektur

I en arkitektur baserad på tjänster flyttas intresset från hur komponenterna (eller snarare tjänsteleverantörerna) är implementerade till hur man kan nå tjänsterna. Kommunikation och samverkan blir nu den tekniska utmaningen.

Det är i detta perspektiv man får se det stora intresset för Web Services. Det är nämligen en teknisk lösning på detta problem. Web Services erbjuder möjligheter att söka, identifiera och anropa tjänster i en heterogen teknisk miljö. I princip blir det nu möjligt att kombinera tjänster till mer komplexa flöden, som med relativt ringa ansträngning kan konfigureras efter behov.

Denna teknik är dock ännu relativt ny och oprövad. Det finns flera betydande tekniska svårigheter som ännu inte är helt lösta.

Typarkitekturer

Redovisningen av detta intresseområde inleds med en sammanfattning av några praktikfall. Sedan följer en analys av dessa där vi pekar på några utmärkande och även gemensamma drag. Därefter följer en redovisning av några framgångsfaktorer som projektdeltagarna framhållit. Avslutningsvis kommer en sammanfattning med ett antal rekommendationer för dem som står i begrepp att prova Web Services i sin egen verksamhet.

Praktikfallen

Genom besök hos de medverkande företagen har projektdeltagarna fått tillfälle att studera och diskutera hur respektive företag närmat sig tekniken med Web Services. Här beskriver vi kort några av dessa praktikfall. Följande avsnitt identifierar några gemensamma drag och redovisar ett antal slutsatser och rekommendationer.

Praktikfallen är:

- SEB Trygg Liv
- SAS
- AMF Pension
- Sandvik
- Volvo IT
- Skatteverket

Varje praktikfall finns redovisat i en separat rapport som finns tillgänglig på Serviams webbsida (se referenslistan). Därför beskrivs de här endast i kort sammandrag. Till detta kommer en sammanfattning av en enkät till projektdeltagarna om deras nuvarande och kommande användning av Web Services.

SEB Trygg Liv

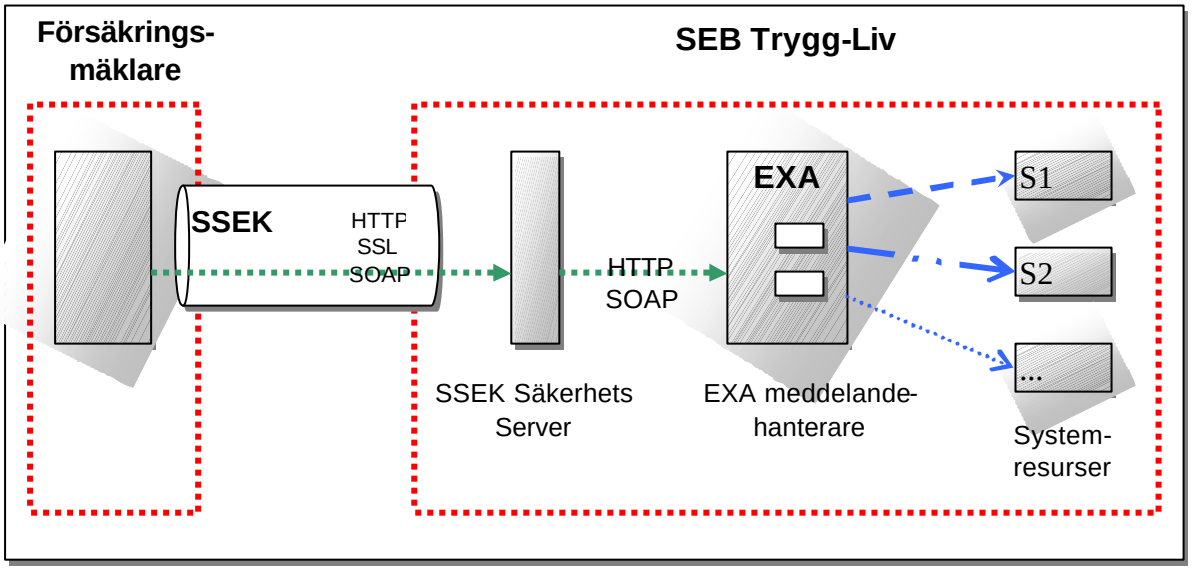
SEB har sedan länge arbetat med en komponentbaserad arkitektur för sina IT-system. Den består bl a av tre fundamentala skikt nämligen kanaler, affärslogik och resurser. Detta är en vidareutveckling av den konventionella uppdelningen av program i presentation, logik och data.

Ett område som SEB valt för att tillämpa Web Services är integration med externa samarbetsparter. I detta fall rör det sig om att ge externa försäkringsmäklare tillgång till ett internt system. Detta har man hanterat genom att skapa ett meddelandehanteringssystem, EXA, för säkert meddelandeutbyte mellan mäklarna och SEB. EXA skickar meddelanden vidare till interna system.

En viktig del i detta system är ett protokoll, SSEK, för säker kommunikation via Web Services. Protokollet använder XML och SOAP över SSL. Se mer om SSEK på www.ssek.org.

Uppgiften för meddelandehanteraren är att identifiera och förmedla in- och utgående trafik. Ett problem är att ett inkommande meddelanden kan resultera i att flera meddelanden behöver sändas till interna system.

Det är de interna systemen som levererar de önskade tjänsterna. Hela överbyggnaden med meddelandehanterare och Web Services är ett medel för att göra dessa tjänster tillgängliga för externa användare via en gemensam ingång och på ett säkert, se figur nedan.



Användning av SSEK och meddelandehanterar hos SEB

SAS

Inom SAS var man tidigt ute och experimenterade med Web Services. Redan 2001 hade SAS en av de första Web Services i världen som använde sig av UDDI. För närvarande använder man Web Services dels internt inom flera områden, bl a incheckning och marknadsföring, och dels externt för en bokningstjänst. Denna senare tjänst riktar sig till befintliga kunder med speciella avtal, t ex resebyråer.

Ur ett arkitekturperspektiv är SAS intressant därför att man har definierat ett gränssnitt, SAPI (Self booking API), som består av en uppsättning publika tjänster som förmedlar åtkomst till redan befintliga tjänster i de interna systemen. Vid utformningen av den externa tjänsten har man byggt vidare på de interna tjänster som ingår i incheckningssystemet. Dessa är i sin tur påbyggnader med hjälp av Web Services på det befintliga bokningssystemet, som körs i en Unisysmiljö. Finessen är att man inte ändrat i det gamla systemet utan enbart litat till befintliga gränssnitt.

Med hjälp av dessa gränssnitt har SAS med små resurser byggt skräddarsydd bokningstjänster åt olika kunder.

Inom SAS IT planerar man för en vidareutveckling av den interna systemarkitekturen mot en tjänsteorienterad uppbyggnad. I förlängningen kan det medföra att man kommer att införa katalogtjänster typ UDDI för att hålla reda på tjänster. För att underlätta arbetet med att identifiera nya tjänster behöver man ta fram en gemensam informationsmodell.

AMF Pension

De senaste åren har AMF Pension introducerat nya produkter och ökat sin samverkan med externa parter. Detta tillsammans med förändrade lagar och förordningar samt krav på rationalisering har medfört att trycket ökat på IT verksamheten. En önskad förändring är bättre systemsamverkan. Detta vill man åstadkomma bl a med hjälp av Web Services.

AMF planerar en successiv övergång till en integrationsarkitektur baserad på ett gemensamt integrationsnav (hub and spoke). Vägen dit går via flera intermediära lösningar.

För samverkan med försäkringsmäklare använder AMF Web Services för att tillhandahålla transaktionsorienterade grundtjänster för att skapa och underhålla försäkringar. Dessa tjänster kommunicerar via SOAP och använder meddelanden som ansluter till en XML standard för försäkringstransaktioner, MIS Life. Detta är den externa delen av systemet som kopplas till det interna integrationsnavet. Där hanteras inkommande meddelande och överförs till anrop till interna system. Detta sker via meddelandehantering men använder inte Web Services.

AMF avser att bygga ut integrationsnavet med fler funktioner i form av kopplingar till fler externa kanaler och interna system. Dessutom kommer man att sammanföra mer komplicerade pensionsberäkningar till en gemensam beräkningsplattform, som kommer att kunna nås från externa användare t ex pensionsportalen.

Sandvik

Sandvik är det företag bland praktikfallen som kommit längst i sin användning av Web Services. Bland de faktorer som bidragit till detta kan nämnas:

- En global organisation med krav på systemsamverkan
- Krav på standardiserad kommunikation med dotterbolag och underleverantörer

Sandvik har valt ett integrationsnav som kärnan i sin arkitektur. Navet är baserat på Micrsosoft Biztalk och föds med meddelanden från en meddelandeserver (IBM MQ) som bildar en fasad mot externa användare. För samverkan mellan fasaden och navet används SOAP.

Navet kommunicerar med ett antal befintliga system via olika protokoll för vilka man skrivit egna anpassningsprogram. I navet ingår funktioner för att filtrera, transformera, fördela och förmedla meddelanden.

Sandvik använder Web Service tekniken i flera system. Denna spridda användning har lett till att tre olika angreppssätt på tjänstedesign har använts, varje angreppssätt är anpassat för en viss situation.

- En stor del av tjänsterna är *meddelandebaserade*. Man har definierat ett 70-tal meddelandetyper. Definitionerna lagras i en gemensam katalog. I de slutliga meddelandena ingår förutom själva data även beskrivningarna av dessa och den operation som skall genomföras på dem. De meddelandebaserade tjänsterna används för att integrera mot underleverantörernas system, via det integrationsnav som beskrevs tidigare.

- Det procedurorienterade användningssättet innebär att man automatgenererar Web Services från klassdefinitioner i t ex C#. Procedurerna grupperas efter sin koppling till verksamhetsnära begrepp (jfr idén med verksamhetsobjekt). Denna metod att designa tjänstegränssnitt används för interna system.

- Den begränsade procedurnära användningen innebär att man valt två funktioner, hämta och ändra, och anropar dessa med ett meddelande som parameter. Innehållet i meddelandet styr till vilken resurs operationen kanaliseras. Denna design av tjänstegränssnittet gör att ändringar i gränssnittet endast medför omdefinition av funktionernas parametrar, vilket underlättar ändringar. Denna design används i ett internt system.

Volvo IT

Inom Volvo IT används Web Services i flera sammanhang. Enligt enkäten finns mellan 10-50 tjänster i drift. Man är dock fortfarande inne i en utforskande och experimenterande fas. Avsikten är att komma fram till en rekommendation för användningen av Web Services som gäller för hela IT-verksamheten. Detta ser man som en del av ett större arbete kring tjänsteorienterad arkitektur.

De exempel som beskrivs är filöverföring och åtkomst via PC av program vid felsökning av bilar. För att minska den overhead som XML innebär har man valt att använda ett tillägg till SOAP, DIME, som innebär att data kan läggas utanför XML.

Exemplet med filöverföring är ovanligt eftersom det visar hur Web Services kan användas för långa tillståndsbevarande transaktioner med stora datavolymer. Detta är det enda exempel på sådan användning som vi funnit i praktikfallen.

Skatteverket

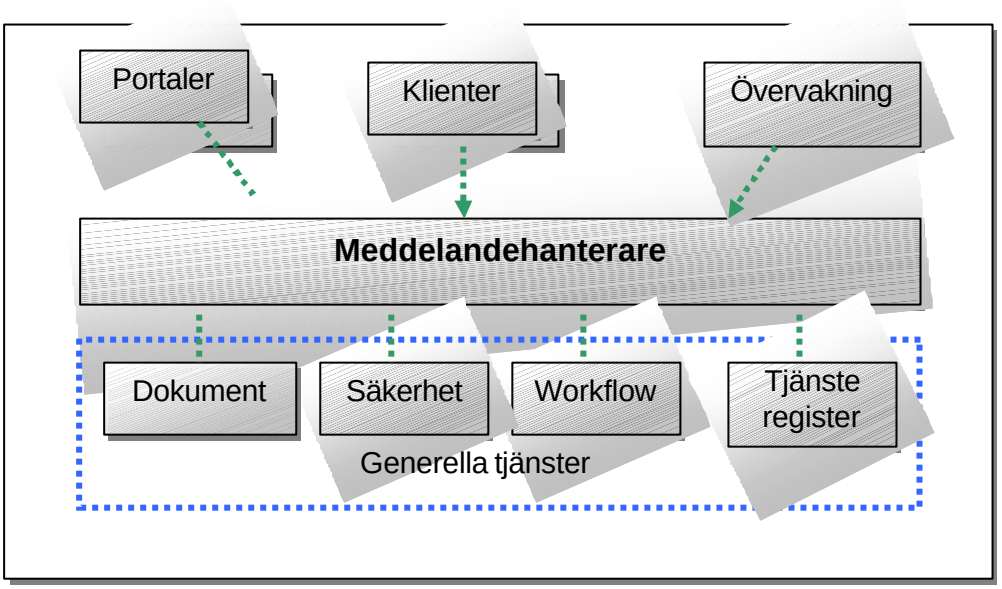
Skatteverket har idag ett fåtal Web Services i drift men ser det som ett framtida stort område. Anledningen till att man intresserat sig för Web Services är dels behov av systemintegration, dels behov av en mer flexibel arkitektur och dels krav på processorientering.

Behovet av systemintegration beror på att externa intressenter önskar nära realtidsåtkomst till systemen, bl a för informationsinhämtning och för att synkronisera datainnehåll. Detta är en effekt av ökad myndighetssamverkan.

Kraven på flexibel arkitektur beror på myndighetens utökade ansvar och på förändringar i dess organisation. De problem man brottas med skiljer sig inte från dem hos många andra företag och organisationer. Systemen är integrerade med parvisa kopplingar. Det finns flera

kopior av centrala register vilket ger problem med att sprida uppdateringar. De befintliga systemens arkitektur har ingen tydlig separation mellan verksamhetslogik, data och presentation.

I arbetet med att minska dessa problem ser man en gemensam meddelandehanterare som en viktig komponent. Man räknar med att kunna realisera detta med den teknik som redan finns på plats inom myndigheten. Tekniken för meddelandeutväxling blir synkron. Systemsamverkan i arbetsflöden koordineras via en särskild ”workflow” tjänst. För att denna lösning skall få full effekt fordras att man tar fram anpassningar till befintliga system. Nya tillämpningar behöver utvecklas som kombinerar de tjänster som blir tillgängliga via meddelandehanteraren.



Planerad tjänstearkitekur, med central meddelandehanterare

För att få bättre grepp om vad denna nya arkitektur innebär har man provat den i några situationer.

Ett exempel är ”Navet”, som besvarar frågor om personuppgifter från externa myndigheter. Åtkomst kräver att frågaren är auktoriserad. Detta har man löst med en web service med åtkomst via SSL. Både sändare och mottagare använder certifikat.

Ytterligare ett exempel är att förenkla bolagsregisterering genom samverkan mellan bolagsverket och skatteverket. Vid registreringen kopplar man till bägge myndigheternas bakgrundssystem. Även här används certifikat för att hantera säkerheten.

Vid utveckling av tjänsterna utgår man från modeller i UML. Från dessa genererar man via XML ett flertal kodskelett. Avsikten är att eliminera manuellt arbete med den omfattande kod som krävs för att administrera Web Services och låta utvecklarna koncentrera sig på verksamhetslogiken.

Enkät kring användning av Web Services nu och i framtiden

Sex av de medverkande företagen har svarat på en enkät kring hur de ser på användningen av Web Services nu och i framtiden. Enkäten täcker följande områden: policy, intern användning, extern användning, systemintegration, verksamhetsinnehåll och teknikval. Här redovisar vi svaren i ett sammandrag.

Policy

En majoritet (4 st) har formulerat planer för introduktion av Web Services men endast två har riktlinjer utveckling av Web Services.

Fyra av företagen använder sig av avtal för Web Services mot externa parter.

Endast två använder mönster och då på nivåer från verksamhet ned till design.

Intern användning

Fyra företag har Web Services i drift. Tre av dessa kör mellan 10-50 tjänster. Användningsområdet är kommunikations mellan system och rör både interaktiva system och bakgrundssystem.

Bland fördelarna nämner man verktygsstödet, som underlättar integrationsarbetet. Svårigheter som nämns är oprövad teknik, kompetensbrist, ägarskap av tjänsterna och säkerhet

Extern användning

För extern användning är situationen något sämre. Tre använder Web Services för detta ändamål varav en har mellan 10-50 tjänster i drift och övriga två har färre än 10.

Ingen av de svarande använder Web Services för processer eller arbetsflöden.

Endast en använder dem i interaktiva externa tillämpningar.

Här nämns säkerhet som ett problemområde.

Systemintegration

Under rubriken systemintegration svarar en majoritet att de använder punkt till punkt kopplingar mellan systemen. Endast en svarande har inga sådana kopplingar.

Samtliga använder någon kommersiell integrationsprodukt och fyra har egna ramverk.

De integrationsstrategier som dominerar är filöverföring, RPC-anrop och meddelande-samverkan. Endast tre använder delade databaser och då i ringa omfattning.

Fyra svarar att de använder Web Services som integrationshjälpmedel, en arbetar med att införa det och ytterligare en ser det som en framtida möjlighet.

Verksamhetsinnehåll

Under denna rubrik finns frågor om storleken (omfattning, kornstorlek) av en web service och vilken typ av samverkansform som används. Svaren visar att Web Services främst används på funktionsnivå, mera sällan på uppgiftsnivå och nästan aldrig på aktivitetsnivå. Med funktion avses enkla operationer av typen ”ändra kundadress”. En uppgift är mer omfattande, t ex ”skapa ny försäkring”, som kan omfatta flera funktioner. En aktivitet är ytterligare mer komplex och kan byggas upp av flera uppgifter.

När det gäller samverkansform är det fråga-svar och meddelandeöverföring som dominerar med ungefär 50/50 fördelning. Endast två svarar att de använder en gemensam datamodell och en ser detta som en framtida uppgift.

Tre svarar att de använder sig av gemensamma mallar för meddelanden och att dessa baseras på XML. Övriga ser detta som en uppgift för framtiden.

Teknikval

Teknikfrågorna omfattar bl a val av protokoll och teknisk plattform. Fyra av sex använder sig av en tjänsteorienterad arkitektur och ytterligare en är på väg att införa detta. Detta svar förbryllar en smula mot bakgrund av vad som redovisats i praktikfallen. Sanningen är troligen att man endast delvis har en sådan arkitektur på plats.

SOAP använder alla utom en, som dock är på väg att införa det.

Fem använder WSDL men ingen anser sig ha bruk av UDDI. När det gäller administration av tjänster så använder sig två av egenutvecklade lösningar.

Samtliga använder något ramverk for sin utveckling av Web Services. Alla använder .NET och alla utom en J2EE. Intresset för högrenivåspråk som BPEL är minimalt. Dock inför Sandvik BPEL inom kort.

Tjänsternas gränssnitt innebär antingen att man erbjuder en funktion per anrop/meddelande eller att flera tjänster sammanförts till ett och samma funktionsanrop. Det förra används av fem av de svarande medan endast tre använder den senare principen.

Två svarar att de använder transaktioner och då endast korta sådana. Detta innebär i praktiken att man gjort bakgrundssystemens transaktionsgränssnitt tillgängligt som Web Services.

Felhantering och loggning är funktioner som endast två svarar att de etablerat regler för.

Funktionsfördelning sker genom användning av flerskiktade arkitekturer.

Processhantering

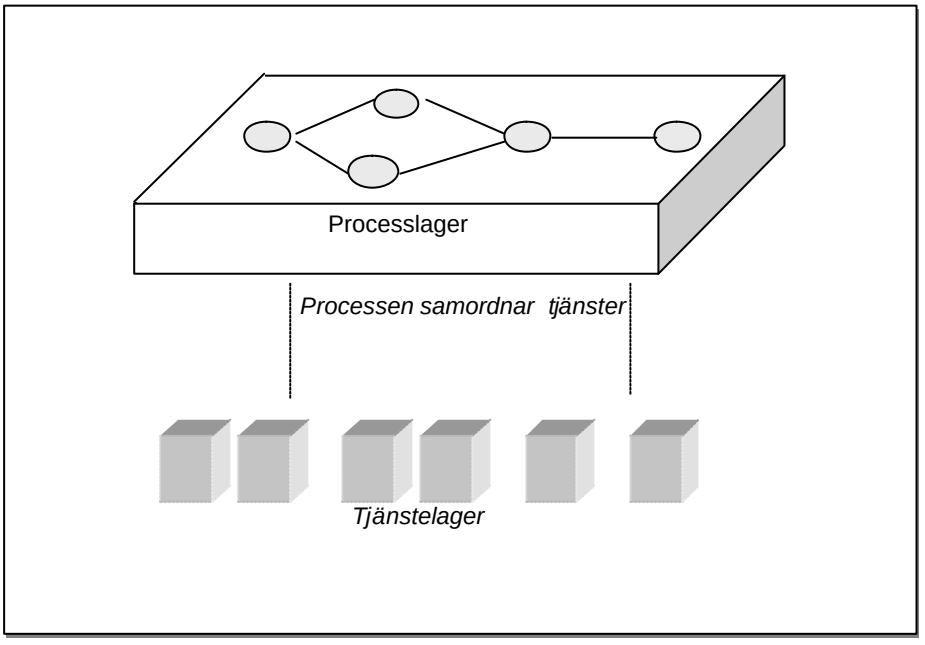
Web-service teknikerna SOAP och WSDL löser en del av integrationsproblematiken när två system ska kommunicera. När tjänster börjar användas i en större skala behövs dock tekniker för att koordinera tjänster. Ett exempel på koordinering är att kombinera ett flertal tjänster till en ny tjänst. Ett vanligt exempel är att en resebyrå kan skapa en ny tjänst (”boka resa”) genom

at kombinera tjänster från hotellkedjor och flygbolag. Att på detta sätt kombinera tjänster från flera organisationer kräver att komplexa sekvenser av meddelanden behöver utväxlas mellan tjänsterna. Detta utbyte av meddelanden behöver implementeras samt övervakas. Utan ett strukturerat angreppssätt finns risken att ett okontrollerat ”spindelnät” av tjänster skapas.

Processmodeller har länge använts som ett sätt att analysera en verksamhets aktiviteter och informationsflöden. Värt att notera är att processer beskriver en verksamhets dynamik, det vill säga aktiviteter, informationsutbyte mellan aktiviteter och aktiviteternas inbördes ordning. Det är just dynamiken som är av intresse när tjänster ska koordineras. Därmed lämpar sig processer utmärkt som en grund för att beskriva koordinering av tjänster.

I en beskrivning av ett tjänstegränssnitt (till exempel utfört med WSDL) beskrivs endast den statiska aspekten av en tjänst i form av meddelanden och operationer. Processbeskrivnings-språk kompletterar denna bild med att lägga till beskrivningar av informationsutbyte mellan tjänster. Processbeskrivningsspråket Business Process Execution Language for Web services (BPEL4WS) är ett exempel på ett språk som kan användas för att beskriva samverkan mellan tjänster.

Tjänster utgör byggstenar som kan kombineras ihop med hjälp av processer. Vanligtvis brukar därför processer finnas som ett eget skikt i en arkitektur, se figur nedan:



Koordinerande processlager

Tjänster och processer

Genom att kombinera processbeskrivningar med Web services blir det möjligt att beskriva hur en uppsättning tjänster ska interagera. Processbeskrivningen i sig kan även utgöra en tjänst, och i sin tur kombineras ihop med andra tjänster i nya processer.

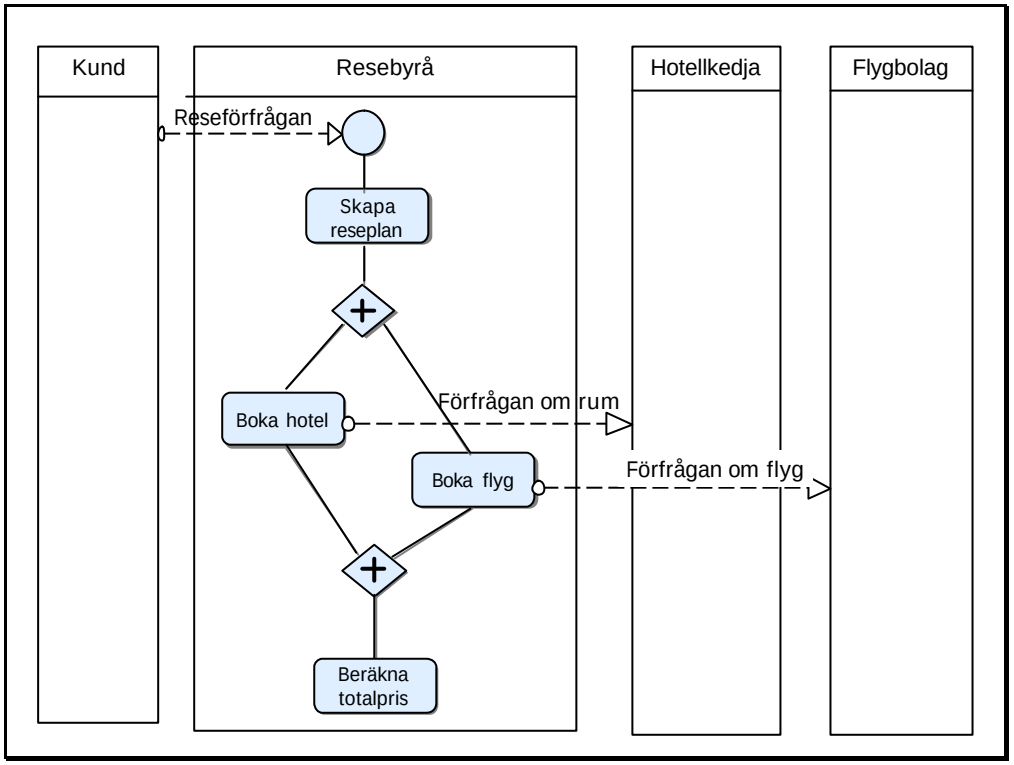
En process kan följaktligen relateras till en tjänst på tre olika sätt:

a) Aktiviteter i en process kan anropa tjänster (via operationer i en tjänsts gränssnitt).

b) En process kan implementera en tjänst, det vill säga en process kan beskriva hur en tjänst ser ut ”innanför” gränssnittet.

c) Tjänster kan starta aktiviteter i en process genom att anropa processen

De tre “sambanden” mellan tjänster kan beskrivas i en processmodell genom att ange tjänster, samt meddelandeutbyten mellan tjänster. Bilden nedan visar ett exempel på en mycket enkel processmodell där ”Kund”, ”Resebyrå”, ”Hotellkedja” och ”Flygbolag” utgör tjänster. Aktiviteterna i resebyråns process är beskrivna, samt meddelandeutbyte med de övriga tjänsterna (streckade pilar i bilden).



En enkel processmodell med tjänster och meddelandeutbyte

Processen ovan illustrerar de tre sätt på vilka tjänster och processer kan relateras:

a) Resebyråns aktiviteter anropar Hotellkedjans och flygbolagets tjänster.

b) Resebyråns process tillhandahålls till kunden i form av en tjänst (tex via Internet)

c) Tjänsten ”Kund” startar resebyråns process genom att anropa resebyråns tjänst.

Relationen mellan en tjänst och en process är därmed tämligen enkel, ur ett tekniskt perspektiv så är det samma relation som mellan en implementation av en metod i en klass (process) och gränssnittet på en klass (tjänst).

Processer sett ur ett verksamhetsperspektiv kan dock rubba denna enkla beskrivning.

Olika perspektiv på processdesign

Som beskrivits så fokuserar processbeskrivningar på de dynamiska delarna av ett system, det är därför naturligt att använda processbeskrivningar som grund för koordination av tjänster. Den mer tekniska relationen mellan processer och tjänster har beskrivits ovan. Användningen av processer för att samordna tjänster kan dock ge fördelar både från ett verksamhetsperspektiv och från ett rent tekniskt perspektiv:

- Ur *verksamhetsperspektivet* så designas processer utifrån aktörer, aktiviteter och information från verksamheten. Detta ger verksamheten möjlighet att få direkt stöd för sin verksamhetsprocess i ett IT-stöd baserat på processteknik. Processer designade ur detta perspektiv kan kallas för verksamhetsprocesser.
- Ur det *tekniska perspektivet* så kan processtekniken användas för att samordna existerande system och tjänster. Processtekniken underlättar t ex parallell exekvering, samt transaktionshantering som spänner över flera system.

Oavsett vilket perspektiv som används måste en process konstrueras metodiskt utifrån ett flertal aspekter. Som grund för design av processer så kan följande fem aspekter användas:

- *Funktions*-perspektivet beskriver de aktiviteter som finns i processen.
- *Beteende*-perspektivet beskriver de kontrollflöden som kopplar ihop processens aktiviteter. Vid design av processbeteendet så ordnas aktiviteterna i en bestämd sekvens, det är även möjligt att konstruera processens beteende så att aktiviteter utförs parallellt.
- *Informations*-perspektivet innehåller de informationsstrukturer som används som in- och utdata till processens aktiviteter.
- Den *organisatoriska* vyn definierar vilka aktörer/tjänster som ansvarar för exekveringen av aktiviteterna.
- *Transaktions*-vyn identifierar hur fel ska hanteras i processen. Detta görs gem att gruppera aktiviteter i transaktioner.

I nedanstående tabell återfinns några vägledande frågor som kan användas vid design av de fem processaspekterna:

Design aspekt	Frågeställningar	Resultat av design
Funktionell	- Hur bryts processen funktionalitet ner till aktiviteter? - Hur mycket funktionalitet ska finnas i en aktivitet? (val av granularitet)	▪ Aktiviteter
Beteende	- Kan en del aktiviteter exekveras parallellt? - Måste en del aktiviteter utföras i en bestämd ordning? - Ska en del aktiviteter endast utföras givet ett visst villkor?	▪ Beroenden mellan aktiviteter. ▪ Kontrollflöde, parallellism
Information	- Vilket meddelande innehåll ska processen producera, vila meddelanden ska kunna tas emot? - Behöver processen intern information?	▪ Struktur på meddelanden ▪ Struktur på process intern information
Organisatorisk	- Vem är ansvarig för exekvering av processen? - Vilka parter/tjänster kommunicerar processen med?	▪ Parter/tjänster.
Transaktion	- Vilka aktiviteter måste utföras som en enda transaktion?	▪ Transaktionsgränser.

När processbeskrivningsspråk används för att beskriva koordination av tjänster utgör aktiviteterna tjänster, informationen är vanligtvis XML strukturer, och aktörerna är system.

Vid design av exekverbara processer behöver alla fem processaspekterna sammanfogas mellan verksamhetsperspektivet och det tekniska perspektivet. Det vill säga funktioner, beteende, information, roller och transaktioner i verksamheten behöver ha stöd i den tekniska utformning av processen.

När gör exekverbara processer nytta?

I praktikfallen används i flera fall centrala meddelandehanterare, som styr flödet av web service anrop till rätt mottagande system. Ju fler tjänster ett företag har, desto mer intressant kan de vara att samordna och kombinera dessa tjänster för att stödja verksamheten. För framtida samordning av tjänster spelar användningen av exekverbara processer en avgörande roll. I dagsläget finns det ett flertal speciella språk som är avsedda för implementation av processer. Ett exempel är BPEL4WS. Dessa språk har ett flertal fördelar gentemot de programmeringsspråk som normalt används för att konstruera Web Services:

- Processbeskrivningsspråk innehåller speciella konstruktioner som gör det enkelt att hantera parallell exekvering och synkronisering.
- Processbeskrivningsspråk med tillhörande exekveringsmiljöer (t ex Microsoft BizTalk) gör det enkelt att hantera processinstanser. Det är t ex enkelt att implementera ett orderflöde och därefter behandla varje inkommande order som en separat processinstans. Detta gör det t ex möjligt att på ett enkelt sätt införa uppföljning på processinstansnivå.

- Användningen av ett standardiserat processhanteringsspråk gör det möjligt att använda produkter för att hantera t ex transaktioner och lastbalansering Användningen.

Ovanstående enkla lista av funktioner pekar också ut processpråkens potentiella användningsområde. Processbeskrivningsspråk är därmed tillämpliga i de fall där parallellism och synkronisering, uppföljning på processnivå, samt standardiserad hantering av transaktioner och lastbalansering utgör en större del av arkitekturproblemen.

Arkitekturmönster

Flera av praktikfallen innehåller exempel på dellösningar som baseras på väl etablerad och fungerande praxis inom IT området. Sådana fungerande lösningar på ofta återkommande problem kallas för mönster (patterns).

Flera olika kategorier av mönster kan särskiljas:

- Administrativa mönster: Dessa mönster beskriver lämpliga sätta att utveckla och förvalta Web Services. Administrativa mönster beskriver alltså inte tekniken, utan snarare vilka organisationsstrukturer som är bäst lämpade för att utveckla och driftsätta Web Services.

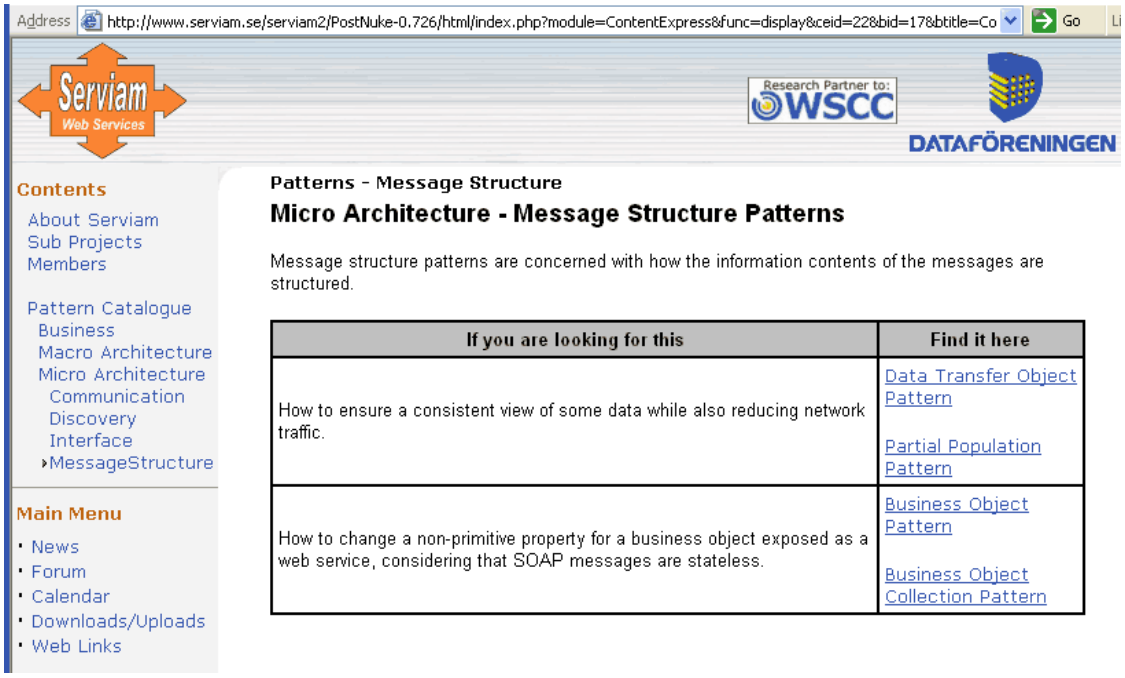
- Macro-arkitekturmönster: Beskriver översiktlig, storskalig utformning av web service arkitekturer. Dessa mönster dokumenterar arkitekturer på en hög nivå, t ex i form av indelning i lager och skikt.

- Micro-arkitekturmönster (även kallade designmönster): Dessa mönster beskriver specifika, mindre delar i en fullständig arkitektur. Ett exempel är olika mönster för att initiera en uppkoppling mot en tjänst (s k *Factory* mönster)

Ett återkommande macro-arkitekturmönster i praktikfallen är användningen av en meddelandehanterare. En grundfunktion i denna är att förmedla meddelanden. Ett mönster för detta är *mäklaren* (broker pattern). En mäklare skapar kontakten mellan sändare och mottagare. Ett ytterligare mönster, som används i en meddelandehanterare är *publicera och prenumerera* (publish and subscribe). Här registrerar sig prenumeranter som intressenter av meddelanden och /eller händelser av en viss typ.

Många av praktikfallen innebär att man erbjuder tjänster från befintliga system. Dessa tjänster implementeras ofta med hjälp av micro-arkitekturmönstret *fasad*, dvs som en isolerande och transformerande påbyggnad på det befintliga systemet.

En samling macro och micro arkitekturella mönster finns i projektet mönsterkatalog på www.serviam.se. För att ytterligare strukturera katalogen med mönster så har microarkitektur mönstren delats in i fyra underkategorier; Communication, Discovery, Interface och MessageStructure (se bild nedan).



Exempel ur Serviams mönsterkatalog

Analys

Som framgått av de korta beskrivningarna av praktikfallen är användningen av Web Services för närvarande relativt blygsam och i flera fall endast på försöksstadiet. Trenden är dock att man aktivt planerar för en ökad användning och ser stora fördelar med tekniken. Användningen är både för extern och intern samverkan.

Motiv för val av Web Services

Det finns flera faktorer som motiverat företagen att börja använda Web Services. Ur ett IT-perspektiv handlar det bl a om följande sex önskemål:

- Rationalisering och flexibilitet.
- Minska överlapp mellan system
- Öka graden av återanvändning
- Integration mellan befintliga system
- Göra funktioner hos befintliga system tillgängliga både internt och externt

- Övergång till en tjänsteorienterad arkitektur

Flera av dessa önskemål kan härledas ur verksamhetens krav på IT-stödet. Exempel på detta är:

- Mer omfattande verksamhetsförändringar anstränger IT-systemen (systemsamverkan, integration, flexibilitet)
- Stöd för genomförande och styrning av affärsprocesser (tjänsteorienterad systemuppbyggnad)
- Krav på smidigare samverkan med externa parter (gemensamma gränssnitt, tjänsteorientering)

Det finns givetvis andra sätt att hantera detta men Web Services ger så många träffar på önskelistan att det är naturligt att prova hur väl det svarar upp mot de påstådda fördelarna.

Systemintegration

Ett användningsområde för Web Services är integration mellan befintliga system. Problemet med parallella men skilda system är väl känt. Det fordras att systemen kan samverka för att de skall kunna stödja och styra företagets affärsprocesser. Integration kan ske genom datautbyte eller genom funktionssamverkan. Datautbyte kan finnas på flera nivåer, från datautbyte genom filer via databasdelning till meddelandeutbyte. Samverkan genom funktionssamverkan kan innebära att system kopplas mer eller mindre tätt tillsammans. Den traditionella kopplingen via proceduranrop ger en tät koppling som inte alltid är önskvärd. Ett komplement är samverkan via händelser. En affärshändelse genererar då en signal som vidarebefordras till de system som abonnerat på denna händelse.

En stor del av den interna systemintegrationen innebär datautbyte och då i många fall genom filöverföring. Man ser dock en trend mot integration via meddelanden. Detta drivs av kravet på kortare eftersläpning mellan systemen, dvs ökade krav på nära realtidsuppdatering. Här har Web Services inneburit en stor hjälp. Web Services via SOAP underlättar sammankopplingen av systemen och XML underlättar identifikation och transformation av innehållet i meddelandena. Notera att SOAP använder sig av RPC vilket medför att denna användning av Web Services i botten handlar om proceduranrop, dvs synkron koppling.

Systemintegration är ett stort och växande område. Där har användningen av Web Services visat sig ge betydande vinster genom att det går snabbare att ta fram fungerande lösningar. En nackdel kan vara sämre prestanda än specialkodade lösningar. För stora datavolymer passar inte Web Services. Där kommer framgent filöverföring att vara den mest praktiska lösningen. Notera dock det undantag som exemplet från Volvo IT beskriver.

Extern samverkan

Flertalet av de tjänster som i praktikfallen förmedlas till externa nyttjare är paketeringar av funktioner i befintliga system. Detta är ett sätt att öka användbarheten och livslängden hos dessa system samtidigt som man undviker att göra ingrepp i dem. Det är en tydlig strävan att använda befintliga gränssnitt i dessa system och lägga nödvändiga anpassningar som ett lager utanför dem. Detta är i praktiken en användning av mönstret fasad.

Ett stort problem vid samverkan med externa parter är säkerheten. Det behandlas därför som ett särskilt delprojekt inom Serviam. På grund av att säkerhet inom Web Services ännu inte är fullständigt specificerad har man i praktikfallen tvingats till egna lösningar. SSEK som används av SEB och AMF är ett intressant exempel på detta.

Ett problem vid arbetat med att definiera SSEK var att få flera intressenter att enas om en gemensam lösning. Beroende på vilken lösning som väljs kan intressenterna tvingas till stora ingrepp i de egna systemen. Det kan därför vara frestande att framhäva egna lösningar som ger minimal egen systempåverkan men som kanske inte är optimala i ett större perspektiv. Detta är givetvis inte ett problem som rör enbart säkerhetsområdet. Det förekommer i alla sammanhang där man skall enas om samverkan över organisatoriska gränser. Det lyckade resultatet med SSEK visar att man inom begränsade områden med stor intressegemenskap snabbt kan nå resultat. Jämför detta med de problem och fördröjningar som arbetet med standardiseringen inom Web Services ger otaliga exempel på.

En framkomlig väg för att hitta samverkansgränssnitt är att identifiera det som är gemensamt och generellt och skapa transformationer från det speciella till det generella och omvänt. Den breda acceptansen av XML har underlättat ett sådant arbete i samband med datautbyte.

Gemensamma informationsmodeller

I flera av praktikfallen har man observerat behovet och nyttan av en gemensam informationsmodell. Detta underlättar definition av meddelandeinnehåll och transformationer från/till det gemensamma formatet och de medverkande systemen. Man ska dock inte underskatta svårigheterna med att ta fram och förvalta en sådan modell. Spåren från många havererade projekt inom detta område förskräcker.

Arkitekturförändringar sker långsamt

En övergång till en tjänsteorienterad arkitektur är ett långsiktigt mål. Som praktikfallen visar är det vanligt att detta sker genom stegvisa transformationer.

I ett första skede experimenterar man med Web Services inom ett begränsad område. Detta innebär också att man inför någon form av meddelandehantering eller mer avancerad integrationsmotor.

Nästa steg kan innebära att man bygger på med fler tjänster och permanentar några tillämpningar. Samtidigt brukar behovet av mer genomarbetade meddelandedefinitioner bli uppenbart. Detta leder till önskemål om gemensamma definitioner och begreppsmodeller.

När man övertygat sig om att Web Services och även tjänster i en mer generell betydelse är användbart som grund för en IT-arkitektur kommer kraven på att exponera fler system som tjänster. Detta leder till att man behöver se över ansvarsförhållande för förvaltning och ägande. Det kan alltså ske en påverkan på sättet att organisera IT-verksamheten.

Ytterligare ett steg är när man inser behovet av en gemensam resurs för att beskriva och administrera alla tjänster och meddelanden. Det är detta som ingår i UDDI och WSDL men som även kan lösas på andra sätt.

Teknikval

Den teknik som används för att realisera Web Services varierar bland praktikfallen. Några gemensam drag finns dock.

Som transportprotokoll väljer samtliga åtminstone http, ibland kompletterat med andra protokoll som smtp eller https, det senare i samband med krav på säker kommunikation.

För meddelandeförmedling väljer man SOAP och för att beskriva innehållet används XML..

För andra funktioner varierar teknikvalet. I de flesta fall handlar det om att ta tillvara de funktioner som erbjuds av den produkt man valt för meddelandehantering eller integration. Där kan t ex finnas stöd för säker meddelandeöverföring och transaktionshantering.

Slutsatser och rekommendationer

Baserat på praktikfallen kan vi konstatera att Web Services grundläggande teknik fungerar väl. De initiala problemen med användning av de grundläggande teknikerna WSDL och SOAP är lösta. XML har etablerat sig som en grund för att beskriva meddelanden. När allt fler organisationer vill använda tekniken för att integrera system över organisationsgränser så uppkommer dock andra problem. Framförallt behöver arkitektur och utveckling samt driftsättning och övervakning av Web Services anpassas för att hantera en större mängd tjänster. Dagens hantering räcker endast till för ett fåtal tjänster. Vi vill särskilt peka på följande behov:

- Utveckla gemensamma informationsstrukturer (t ex XML schema), samt skapa rutiner för att hålla dessa uppdaterade.
- Använd framförallt ett meddelandebaserat angreppssätt när tjänstegränssnitt konstrueras. Detta sätter den gemensamma informationsstrukturen i centrum.
- Även om användning av processbeskrivningsspråk idag är liten, så är det lämpligt för större organisationer att redan nu utvärdera tekniken.

Standardiseringen inom området pågår och är långt ifrån avslutad. Några viktiga områden som ännu inte är klara är säkerhet och transaktionshantering. I de exempel på användning av Web Services där säkerhet är viktig har man valt att ta fram egna lösningar.

Framgångsfaktorer

Här följer några exempel på framgångsfaktorer saxat ur den mer fylliga redovisningen av praktikfallen.

- Välj kunniga utvecklare för att kunna komplettera med egna lösningar där valda produkter inte räcker till. Arbeta i små grupper.
- Börja med begränsade funktioner som inte är verksamhetskritiska
- Ta fram en vision av hur den framtida arkitekturen bör se ut
- Skapa en gemensam ingång mot bakomliggande system
- Låt inte generella funktioner som säkerhet och loggning hamna i tjänsterna utan placera dem som en del av infrastrukturen
- Meddelandedefinitioner kräver medverkan av verksamhetskunnig personal. Var nöjd med tillräckligt bra meddelanden. Sök inte den perfekta lösningen.

Referenser

Följande dokument är framtagna av projektets arkitekturgrupp:

Dokumentnummer	Titel
SERVIAM-LIT-03	Henkel, M., “Serviam Literature Survey Part III – Web Service Design”
SERVIAM-LIT-04	Zdravkovic, J., “Serviam Literature Survey Part IV –Service-based Processes”
SERVIAM-ARC-01	Wiktorin, L., “Ett exempel på dokumentation av en typarkitektur”
SERVIAM-ARC-02	Wiktorin, L., “Questions on Architecture”
SERVIAM-ARC-03	Johansson, T., Henkel, M., "Pattern Template"
SERVIAM-ARC-04	Henkel, M., "Architectural Case: SEB"
SERVIAM-ARC-05	Henkel, M., "Architectural Case: Sandvik"
SERVIAM-ARC-06	Wiktorin, L., “Architectural Case: AMF Pension"
SERVIAM-ARC-07	Henkel, M., “Architectural Case: The Swedish Tax Agency”
SERVIAM-ARC-08	Wiktorin, L., “Architectural Case: SAS"
SERVIAM-ARC-09	Wiktorin, L., “Architectural Case: Volvo"
SERVIAM-ARC-10	Wiktorin, L., “Web service usage survey”
SERVIAM-ARC-11	Henkel, M., Zdravkovic, J., “Service Orchestration- Applying Executable Processes”. Första versionen, etapp 1.
SERVIAM-ARC-R1	Henkel, M., Zdravkovic, J., Johannesson, P., "Service-based Processes - Design for Business and Technology", the International Conference on Service Oriented Computing, New York, USA, November 15-18, 2004.
SERVIAM-ARC-R2	Henkel, M., Zdravkovic, J., "Architectures for Service-oriented Processes", the Nordic Conference on Web Services, Växsjö, Sweden, November 22-23, 2004.
SERVIAM-ARC-R3	Henkel, M., Zdravkovic, J., “Approaches to Service Interface Design”, To appear in: The First International Conference on Interoperability of Enterprise Software and Applications; Workshop on Web Services: Business, Financial and Legal Aspects, Geneva, Switzerland, February 23-25, 2005.

Del III

Stödfunktioner- en översikt

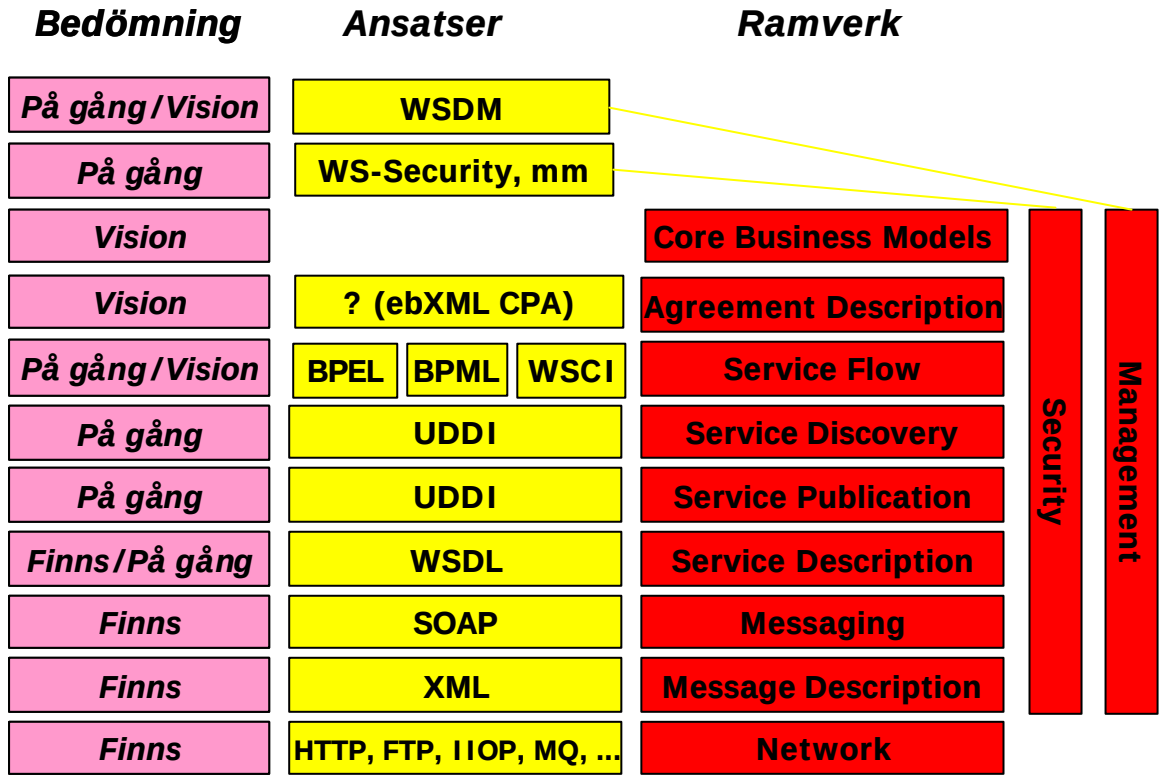
Jesper Holgersson
Högskolan i Skövde

Stig Berild
Santa Anna Institute

Benkt Wangler
Högskolan i Skövde

Inledning

Web Services är ett koncept som syftar till att göra det möjligt att använda Internet på ett mer dynamiskt sätt [6]. Web Services pekar ut en uppsättning standarder som avses möjliggöra utbyte av information samt tjänstebaserad programvaruutveckling och är som sådan del av en mer generell strävan mot informationsutbyte och programvaruutveckling baserat på välavgränsade tjänster som görs tillgängliga, mer eller mindre universellt, via katalogtjänster och väldefinierade gränssnitt. I fortsättningen kommer vi att använda benämningen webbaserade tjänster när vi avser den mer allmänna trenden och Web Services när vi avser den specifika tillämpning av webbaserade tjänster som Web Services är. Web service-standarderna är stadda i fortvarig stark utveckling. Det är endast för de understa nivåerna i den s.k Web service-stacken av standards (se Figur 1), som det föreligger fixa och reellt användbara standards. Figur 1 indikerar hur långt utvecklingen i skrivande stund hunnit. Exakt var utvecklingen landar och vilka standards som verkligen kommer till användning är f.n. omöjligt att säga.



Figur 1. Web-service-stacken [13]

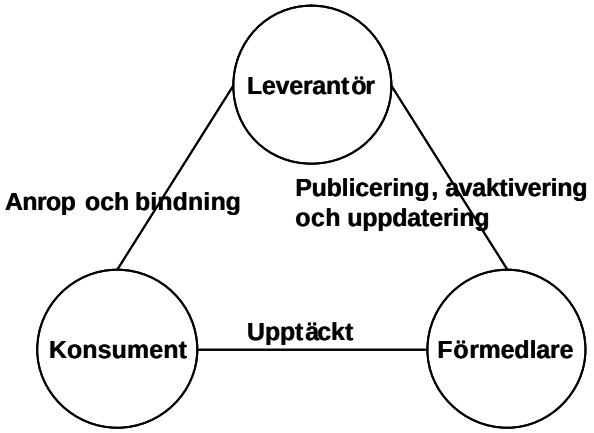
Webbaserad tjänsteutveckling tillåter publicering av affärsfunktioner på Internet och möjliggör genom detta potentiellt universell tillgång till dessa affärsfunktioner [6], [17]. Genom att tillämpa metodiken för webbaserade tjänster får företag också nya möjligheter att utbyta data inom och utom företaget på ett enkelt och billigt sätt. Webbaserade tjänster

erbjuder också en möjlighet att på ett enhetligt sätt definiera gränssnitt till nya och gamla tillämpningar (legacy systems) och kan på detta vis underlätta integration mellan funktioner. Att använda Web Services för att integrera nya och gamla applikationssystem innanför brandväggar eller i punkt-till-punkt-lösningar med erkända partners är troligen det som för närvarande tilldrar sig det största intresset.

För att en webbaserad tjänst skall kunna fungera enligt visionen, alltså ett globalt tjänsteutbyte över Internet, krävs ett antal grundläggande aktiviteter[1]:

- Beskrivning
- Publicering, uppdatering och avaktivering
- Upptäckt
- Anrop och bindning

För att utföra dessa aktiviteter behövs, enligt [1] och [17], ett antal aktörer eller roller. Det behövs en aktör som erbjuder en tjänst (leverantör), en aktör som utnyttjar en tjänst (konsument) samt en aktör som förmedlar tjänsterna (förmedlare). Figur 2 illustrerar hur dessa aktörer samverkar. För en tydlig förklaring av rollers och aktiviteters funktioner hänvisas till [17].



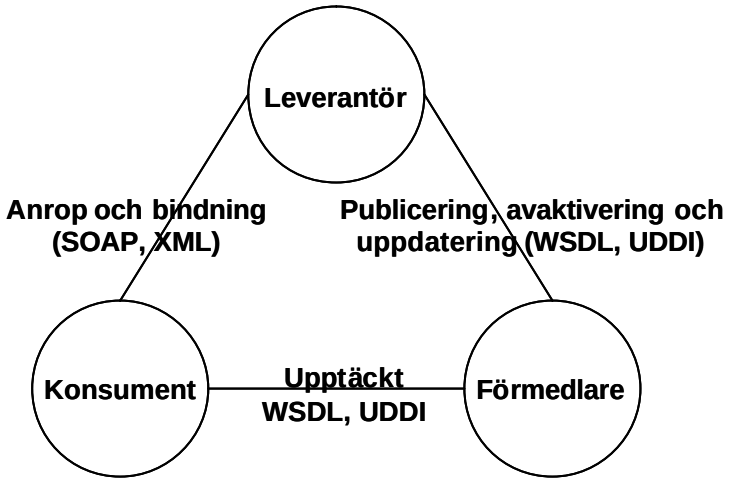
Figur 2. Rollfördelning och aktiviteter för webbaserade tjänster. (Efter [17])

I Figur 3 har vi placerat in de standarder ur Web service-stacken som gör det möjligt att hantera de olika aktiviteterna.

SOAP är ett XML-baserat protokoll som hanterar utbytet av anrop och svar mellan två parter. SOAP är, liksom XML, plattformsoberoende eftersom det enda SOAP egentligen gör är att definiera paketeringsmodell och kodningsmekanismer. SOAP erbjuder således möjligheter att skicka XML-dokument mellan olika system och därmed erbjuder det all den information som är nödvändig för att systemen skall förstå hur XML-dokumenten skall tolkas. För en djupare inblick i SOAP rekommenderas [17, 18, 19].

WSDL och UDDI kommer att beskrivas mer ingående i senare delar av denna rapport men för att ge en kort inblick i deras funktion och karaktärsdrag beskrivs de här kortfattat.

Avsikten med WSDL är att beskriva en viss tjänsts egenskaper och hur denna tjänst är uppbyggd. WSDL är enkelt uttryckt ett XML-schema som formellt definierar det ramverk som beskriver en tjänsts gränssnitt, det vill säga ett standardiserat sätt att representera datatyper i meddelanden, de operationer som är nödvändiga att utföra på meddelandet samt hur meddelandet skall mappas till nätverket för vidare transport. För en djupare förklaring hänvisas till [17, 18, 19].



Figur 3. Roller, aktiviteter och semantik för Web Services. (Efter [17]).

UDDI kan liknas vid en stor databas där tjänster skall publiceras för att senare kunna återfinnas av andra aktörer. I figurerna 2 och 3 representerar förmedlaren denna distribuerade databas. Syftet med förmedlaren är att beskriva alla nödvändiga attribut för en tjänst som exempelvis vilken kategori av tjänst det handlar om, vem som erbjuder tjänsten samt beskrivningar och förklaringar för en viss tjänst. Den information som erhålls via förmedlaren ska vara tillräckligt detaljerad för att en intresserad konsument skall kunna fatta ett beslut om att utnyttja tjänsten i fråga. UDDI kan, som redan påtalats, betraktas som en helt ordinär databas som innehåller information om de tjänster som finns lagrade i databasen. Dessutom måste UDDI kunna erbjuda information till en global marknad, innefattande alla upptänkliga branscher och där inblandade parter måste kunna beskriva sina tjänster på ett gemensamt sätt som de finner lämpligt. Allt detta gör att UDDI måste vara betydligt mer flexibelt än en ordinär situationsanpassad databas [8, 9, 17, 18, 19]. En något utförligare diskussion på en högre nivå om UDDI:s roll kommer i senare delar av rapporten. Se också [9, 10].

Det är i dagsläget knappast aktuellt med en publik Web Service-arkitektur som det tidigare var tänkt. Det finns ett par huvudsakliga skäl till detta. Ett skäl är att det finns en del frågor som återstår att lösa på säkerhetsområdet vilket resulterar i att företag inte vill öppna sina kritiska källsystem mot Internet. Ett annat tungt vägande skäl är att, som framgår av Figur 1, semantiken för webbaserade tjänster inte är färdigstandardiserad på alla plan ännu [9, 10].

Web service-stacken innehåller som synes en mängd delar och många av dessa är inte färdigutvecklade ännu. I denna rapport kommer fokus därför att ligga dels på de lägre nivåerna i stacken, alltså de lager som redan diskuterats i rapporten, och dels på WS-security.

Generellt är det inte helt lätt att uttala sig om hur och vad som används av Web Services idag. De flesta företag är i ett initialt skede i sin satsning mot webbaserad tjänsteutveckling. Det finns därför inte så mycket information att utgå ifrån. Dock har de företag vi varit i kontakt med ganska mycket idéer och visioner då det gäller att skapa tyngre webbaserade tjänster baserade på en Web service-arkitektur. Många av dessa ansatser verkar både lovande och intressanta. Innan dessa tillämpningar har sett dagens ljus och kan studeras mer ingående är det svårt att göra en reell bedömning av dessa. Det är emellertid uppenbart att företag som deltar i Serviam-projektet planerar att använda WSDL och kanske också UDDI i framtida Web service applikationer, det senare dock i första hand för internt bruk. Redan i dagsläget finns mindre applikationer i drift som utnyttjar WSDL.

Semantik

XML är, som redan diskuterats i denna rapport, en grundläggande byggsten för att tillämpa Web Services. XML används inom Web Services som ett standardiserat format för att skapa och utbyta dokument mellan olika applikationer. Något som ännu inte diskuterats i denna rapport är hur XML används för att uttrycka hur ett visst dokument skall specificeras. I detta sammanhang bör även WSDL nämnas då även detta till viss del är relaterat till specificering av dokument. Dock är WSDL inblandat på en högre nivå, beskrivande en tjänst, jämfört med XML:s dokumentspecifikationer som det kan finnas flera av i varje tjänst.

Det finns två olika möjligheter att specificera XML-dokument: XML schema samt XML 1.0. Det är viktigt att poängtera att dokument som uttrycks i XML schema och XML 1.0 ser exakt likadana ut. Det som skiljer XML 1.0 och XML schema är alltså inte hur dokumenten ser ut. Istället är det hur ett dokument specificeras, alltså hur villkor och krav på struktur och innehåll formuleras, som skiljer XML 1.0 och XML schema åt. XML 1.0 utnyttjar DTD:er (Document Type Definiton) för att specificera dokument. En DTD är en mall som beskriver vilken struktur, vilka villkor och vilka datatyper som kan användas.

Till skillnad från XML 1.0, erbjuder XML schema större möjligheter att uttrycka villkor, datatyper och krav på strukturer. Som sagts innan är XML-koden för själva dokumentet densamma och schemat har samma syfte som DTD har, alltså att reglera villkor för dokuments innehåll. Skillnaden ligger i att XML schema ser på dokument ur en annan synvinkel, vilket i sin tur ger en annan uppsättning av modellbegrepp. Dessutom finns det mer uttrycksfulla språkkonstruktioner för att uttrycka villkor och restriktioner för dokument än vad som är fallet med användning av DTD. Därtill är ett schema direkt uttryckt i XML, till skillnad från DTD:er som utrycks i ett separat syntax. För en utförligare beskrivning av XML schema hänvisas till [7].

XML 1.0 och XML-schema används i dagsläget i en mängd olika applikationer, främst i syfte att underlätta eller framtidssäkra integrering både inom och mellan företag. Sålunda finns inte speciellt mycket att säga om XML och XML-schema, det fungerar och används vilket borgar för ytterligare vidareutveckling. Ett litet varningens finger angående XML 1.0 och XML-schema kan dock höjas. [7] belyser detta potentiella problem som innebär att det finns risk att utvecklingen kommer att gå två vägar beroende på vilken vilja företag har att förnya sig från XML 1.0 till XML-schema, alltså från XML med användning av DTD:er till användning av XML i kombination med XML-schema. Många företag har i dagsläget fullt upp med att anpassa sina system för XML 1.0 och det finns en risk att somliga företag inte har vilja, tid och ekonomi att fortsätta med ytterligare anpassningar mot XML-schema. Om två ”skolor”

kring XML uppstår finns det vissa risker att utbyte av Web Services kan påverkas av vilken typ av XML som används, eftersom interagerande tjänster måste kunna tolka varandras dokument. Detta scenario är inte speciellt lockande.

Likaså SOAP kan i dagsläget anses som så stabilt att det används i flera olika tyngre tillämpningsmiljöer. Naturligtvis kommer nya versioner av SOAP att dyka upp men detta är ju bara en naturlig del av en standards utveckling. Det viktiga är att SOAP idag kan användas och att detta även sker. Dock är det hittills absolut vanligaste att data skickas internt i företag eller möjligtvis punkt-till-punkt med hjälp av SOAP [12].

Även WSDL tycks bli allt stabilare och används i realiteten för att uppvisa information om tjänster och även för specifikationsutbyte mellan olika aktörer. Ett exempel som avviker från detta är Amazon som har WSDL-specifikationer över produktkataloger, generell kundinformation mm. Överföring av data sker med hjälp av SOAP och XML. Även en del andra företag gör försök att öppna vissa tjänster med hjälp av WSDL. Dock är det ännu inga tjänster som exponerar data som ligger djupt skyddade i företagens nätverk. För de flesta företag gäller det i dagsläget att få igång mindre testtjänster med mer lättviktiga applikationer [12]. Det är dock uppenbart, som redan påtalats, att WSDL håller på att stabiliseras och att det sannolikt kommer att användas mer och mer och till allt tyngre tillämpningar. En begränsande faktor för hur stor användningen kan bli är naturligtvis hur säkerheten för informationen kan hanteras. En översikt av säkerhetsaspekter och säkerhetslösningar för Web Services kommer i kapitel 4. Den som vill ha ännu mer information hänvisas till [1, 2].

Katalogtjänster

Genom Internet öppnas möjligheter till global samverkan mellan företag på en mängd olika sätt. För att hantera denna globala samverkan behövs teknikoberoende standarder med tillhörande regler som erbjuder en stabil plattform att operera på. För att åstadkomma allt detta behövs beskrivningar, etableringar och avtalsförfaranden. Dessutom måste företag på något sätt få en överblick över vilka tjänster och möjligheter som erbjuds, var dessa finns, vem som tillhandahåller dem samt vad som krävs för att utnyttja dem.

Vi har tidigare i denna rapport diskuterat rollen förmedlare inom en generell tjänsteorienterad arkitektur. Förmedlaren är den aktör som ska se till att den mängd av tjänster som erbjuds finns beskrivna och exponerade med en tillräcklig detaljrikedom för att en intresserad användare i slutändan ska kunna sälja eller utnyttja (köpa) en tjänst. Det är följaktligen förmedlaren som erbjuder en katalogtjänst för alla olika tjänster.

Det finns två grundläggande delar i en katalogtjänst; publiceringen av tjänsten så att potentiella konsumenter kan hitta tjänsten samt beskrivning av tjänsten så att potentiella konsumenter kan bedöma om en tjänst är passande eller ej. Dessutom behövs naturligtvis ytterligare tilläggsinformation om bindningar, specifikationer etc. som krävs när en tjänst väl ska utnyttjas, det vill säga mer detaljerad information om hur en konsument måste anpassa sitt eget system så att ett utbyte av information kan ske mellan konsument och leverantör.

Beskrivningen av en webbaserad tjänst är naturligtvis mycket viktig för att denna skall kunna klassificeras, upptäckas och användas. Beskrivningen av tjänsten måste kunna förstås av både människor och maskiner och innehålla både funktionella krav (vad tjänsten gör) samt ickefunktionella krav (krav på säkerhet, autentisering osv.).

Beskrivning och publicering av tjänster kan vidare delas in i syntaktiska och semantiska beskrivningar. Syntaktisk information beskriver hur en tjänst skall användas och relaterar ofta till ickefunktionella krav, exempelvis säkerhet, genom att specificera vilka certifikat som krävs för att utnyttja en viss tjänst. Semantisk information innefattar information om den aktör som erbjuder en tjänst, en beskrivning av vad tjänsten gör tillsammans med hur tjänsten tillämpar olika detaljer, exempelvis säkerhet. Inom Web service-stacken hanteras syntaktisk och semantisk beskrivning av tjänster med hjälp av WSDL och UDDI. Det finns även andra sätt att hantera beskrivningar som exempelvis ebXML, ADS, SDL etc. För en djupare inblick i hur olika ansatser hanterar beskrivning av tjänster hänvisas till [17].

Förutom beskrivning är publicering av webbaserade tjänster nödvändigt då tjänsten måste göras känd och tillgänglig för den stora massan. Publicering av webbaserade tjänster måste, liksom beskrivning, vara av både syntaktisk och semantisk natur. UDDI erbjuder möjligheter för både syntaktisk och semantisk informationsbeskrivning då information om verksamhet, vilka tjänster som finns, bindningar samt information om specifikationer för tjänster går att lagra och publicera. Dock finns det även för publicering av webbaserade tjänster andra ansatser som kan hantera detta. Dessa ingår dock inte i Web service-stacken. Vi hänvisar till [17] för en mer generell bild av beskrivningsområdet.

Det aktuella läget för UDDI är inte så positivt. *”Teknikfokus i kombination med bristande respekt för och förståelse för hur producenter och konsumenter av tjänster (företrädesvis webbtjänster) upplever sina respektive roller utifrån ett handels- och affärsperspektiv har bidragit till att UDDI av många snarare upplevs som en hype än en stabil byggsten i Web Services-ramverket”* ([9], sid 3). Denna problematik har lett till att UDDI än så länge endast används internt, d.v.s. användning i slutna miljöer som företag, enskilda verksamhetsområden eller möjligtvis branscher. Inom ett avgränsat område finns det större möjligheter att komma överens om hur webbaserade tjänster skall beskrivas i en gemensam katalog. Problematiken med UDDI behandlas mer detaljerat i [8] och [9]. Se även djupare diskussioner om beskrivande data i [10] och [11].

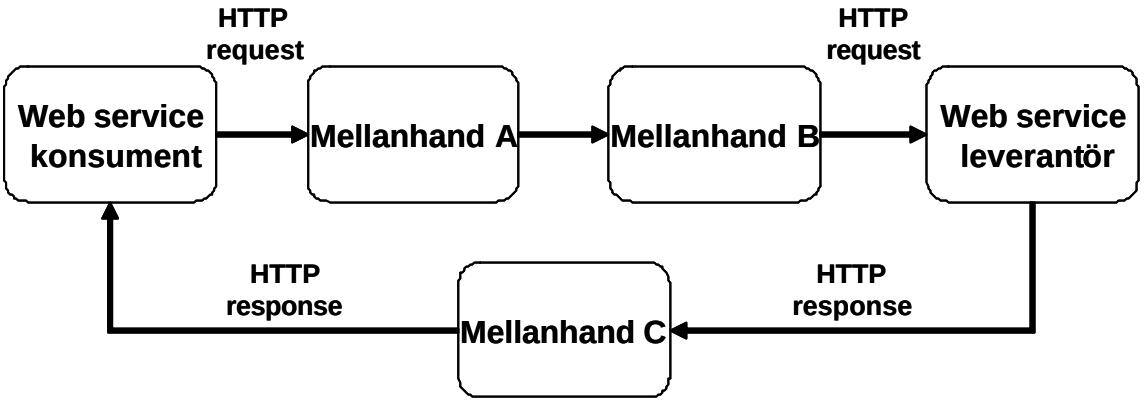
Säkerhet

Publik användning av Web Services introducerar nya vägar in till företags källsystem som, om den individ som är inne i systemet är obehörig, kan ställa till med mycket problem både genom allmänt sabotage samt åtkomst av känslig information. Inom Serviam-projektet ligger fokus i första hand på de tekniska aspekterna av säkerhet, alltså vad ett företag rent tekniskt kan göra för att skydda sina system och Web Services. Dock bör man ha i åtanke att tekniska lösningar är verkningslösa om individer på ”insidan” av ett företag har möjligheter att komma åt känsliga källsystem. Detta scenario gäller generellt och inte bara för Web Services.

En faktor som är viktig att beakta då det gäller säkerhet för Web Services är de olika roller som samverkar i en Web Service-arkitektur. Som redan nämnts finns tre grundläggande roller: sändare, mottagare samt föremedlare/register. Dessutom finns oftast en eller flera mellanhänder, som kan ses som en specialroll för Web Services.

Kommunikationen inom Web Services sker mellan noder som kan sända meddelanden, ta emot meddelanden alternativt både sända och ta emot meddelanden. I en affärstransaktion kommuniceras i allmänhet en sekvens av meddelanden mellan två eller möjligen flera noder. I en sådan sekvens är det vanligt att det finns en eller flera mellanhänder mellan start – och

slutpunkter, se figur 4 som ett exempel. En mellanhand kan ses som en komponent som skickar meddelanden vidare (alltså en form av routing) och/eller erbjuder någon form av funktionalitet relaterat till ursprungliga kommunikationen mellan två ändnoder. Ett exempel på en mellanhand som erbjuder funktionalitet skulle kunna vara en Web Service som utnyttjas av och inom den Web Service som används av ändnoderna. Det är inte heller ovanligt att en sändare utnyttjar flera olika mellanhänder eller Web Services som delar i en större Web service-arkitektur, vilket kan leda till ett komplext nätverk av Web Services och beroenden mellan dessa [1].



Figur 4. Ett exempel på utnyttjandet av mellanhänder (Efter [16])

Anledningen till att mellanhänder får så stort fokus i denna rapport är att användningen av mellanhänder är den enskilt största möjligheten att få krypterade data exponerade för obehöriga användare. Skälet till detta är att existerande och allmänt beprövade säkerhetstekniker för dataöverföringar via WWW (exempelvis VPN, SSL/TLS) inte fungerar fullt ut för Web Services som transporterar data via mellanhänder. Problemet ligger i att när ett SOAP-meddelande, krypterat och skickat över exempelvis SSL, anländer till en mellanhand, måste hela meddelandet dekrypteras (hela meddelandet måste dekrypteras eftersom SSL endast arbetar på de lägre lagren i OSI modellen) för att mellanhanden skall komma åt information om vart meddelandet skall skickas vidare etc. I det ögonblick som meddelandet är helt dekrypterat är hela interaktionen mycket sårbar för obehörig åtkomst [1, 5, 20]

Ett annat problem relaterat till existerande krypteringstekniker är att dessa i sin ursprungsform endast erbjuder krypterad överföring. När data väl är överförda, lagras de helt okrypterat, vilket naturligtvis gör det enklare att komma åt information än om data är krypterade även i källsystemet.

Nu är det inte bara transport av data via mellanhänder som gör Web Services sårbara. [1] listar ett antal problem och hot som främst gäller för publika Web Services. En kort sammanställning av dessa kommer dock att återges här. Det är viktigt att återigen poängtera att det i första hand är publika Web Services som behandlas. Så länge Web Services används internt inom ett företag eller på skyddade kanaler utan mellanhänder finns inte alls samma säkerhetsproblematik.

Autentisering

För i stort sett alla system finns det någon form av behörighetskontroll som undersöker om en viss person eller applikation har rätt att utnyttja ett givet system eller applikation. Samma resonemang gäller naturligtvis även för Web Services. Autentisering är en hörnsten för att kontrollera behörighet. En användare (fysisk eller applikation) av en Web service måste på något sätt kunna identifiera sig (autentisera sig), emot den Web service som skall utnyttjas för att fastställa vilken form av behörighet som gäller. Genom att utge sig för att vara någon annan och följaktligen få behörigheter och tillgång till data bakom ett företags brandvägg, det vill säga källsystemen, kan den obehöriga åsamka leverantören av en Web service stor skada. Obehörig åtkomst kan även nås på andra sätt, exempelvis genom avlyssning av mellanhänder som i sin tur kan resultera i falsk autentisering.

Avlyssning

Ett problem som uppstår vid användning av mellanhänder mellan sändare och mottagare är möjligheter till avlyssning. Eftersom ett meddelande måste packas upp i klartext vid en mellanhand innebär detta att det finns goda möjligheter att komma över information som sedan kan användas för att komma åt den Web service som meddelandet gäller. Äldre krypteringstekniker fungerar säkert vid punkt-till-punkt-kommunikation men kan inte hantera säkerheten vid mellanhänder.

Integritet

Ett annat säkerhetsproblem är integritet, alltså hur en sändare/mottagare kan veta att innehållet i meddelandet är äkta, alltså att inget i meddelandet har manipulerats under dess färd mellan ändnoderna. Exempelvis öppnar avlyssning för att integriteten kan brytas genom att ett meddelande i klartext kan manipuleras för att utföra oväntade operationer eller ge obehörig åtkomst till system som annars inte ska kunna nås.

Hacker-attacker

Alla ovanstående scenarier innefattar någon form av attack utförd av en individ med oärliga avsikter. Web Services inbjuder även till andra former av attacker som är av en än mer destruktiv natur, alltså utan ett direkt ekonomiskt intresse. Exempel på attacker av det mer destruktiva slaget kan exempelvis vara DoS attacker (Denial of Service) vilket innebär att en Web service bombarderas med data vilket i sin tur leder till att denna Web service inte kan hantera alla inkommande förfrågningar. Att en Web service är indisponibel kan få katastrofala följdverkningar om en användare av en Web service är beroende av den för att kunna utföra exempelvis viktiga transaktioner. Andra former av hacker-attacker kan härledas till bruten integritet för ett meddelande. Innehållet i meddelandet kan ha utökats med elak kod, exempelvis i form av virus, som kan skapa stor förödelse i ett företags källsystem.

Omogen plattform

Förutom ovanstående scenarier finns det troligtvis ytterligare ett antal svagheter som ännu inte upptäckts. Alla luckor och svagheter för Web service-arkitekturen kan inte täckas in i

förväg och användning av nya standarder och teknologier kommer, innan en viss användningsmognad uppstått, alltid att innebära vissa risker. Ett tydligt exempel på ovanstående resonemang, dock inte direkt relaterat till Web Services, är utvecklingen av service pack 2 för Windows XP som var nödvändigt att läggas till då det fanns alltför många och stora säkerhetsluckor i originalversionen.

Dessutom innebär en omogen plattform att andra oanade problem som exempelvis oönskade interaktioner mellan mjukvaror och andra oförutsedda konsekvenser kan inträffa. Web service-arkitekturens komplexitet ökar naturligtvis riskerna för oönskade biverkningar.

Hur ser säkerheten ut på ett generellt plan?

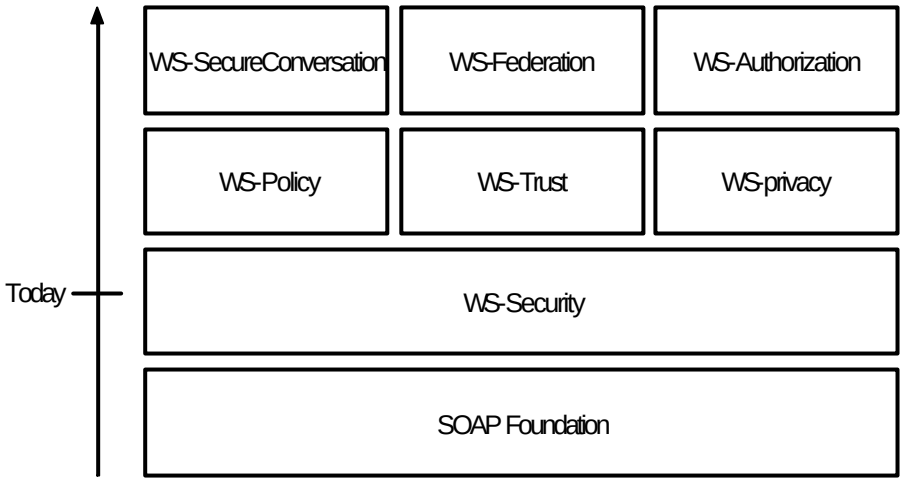
Säkerheten för webbaserade tjänster förbättras hela tiden, men fortfarande är det många tjänsteapplikationer som inte är publikt exponerade på grund av bristande säkerhet i tidigare versioner av SOAP. Exempelvis nämns i specifikationen för SOAP 1.1: *"Not stated in this document are methods for integrity and privacy protection. Such issues will be addressed more fully in a future version(s) of this document."* [14], [1]. Utvecklingen har sedan dess (SOAP 1.1 släpptes i maj 2000) gått framåt och i SOAP 1.2 finns det ett större utrymme för att lägga till funktioner relaterade till säkerhetsaspekter [14], [1].

WS Sequrity beskrivs i [5] som en hörnsten i den säkerhetsarkitektur som är under utveckling för att komma tillrätta med säkerheten för webbaserade tjänster, i första hand behörighetskontroller samt skyddad överföring av data. Som nämnts i ovanstående stycke har tidigare initiativ för att åstadkomma webbaserade tjänster lämnat litet utrymme för säkerhet då detta inte var explicit uttalat i tidigare försök till standardisering. Säkerhetsaspekter har på senare tid tagits på större allvar och det är för närvarande stor aktivitet inom olika standardiseringsorgan för att utveckla standards för säkra publika webbaserade tjänster. I juni 2002 överlämnade IBM och Microsoft ett förslag till säkerhetsansats, WS Security, till OASIS (Organization for the Advancement of Strucured Information Standards) för fortsatt standardiseringsarbete. WS Security är i dagsläget en ledande standard för att åstadkomma säker överföring av meddelanden mellan sändare och mottagare. Det går till exempel att skicka meddelanden via mellanhänder utan att innehållet i dessa någon gång kan läsas i klartext. Specifikationen för WS Security är en grundsten i den större specifikation som ska innefatta lösningar på i stort sett alla kända hotbilder mot Web Services. Dock är dessa tilläggsspecifikationer fortfarande under utveckling och det är oklart när det finns en helhetslösning för att säkert tillämpa publika Web Services. Figur 5 redovisar andra specifikationer som är tänkta att komplettera WS-Security.

Vad alla olika specifikationer innebär och hur de hänger ihop kommer inte att diskuteras vidare i denna rapport då det redan finns översiktligt beskrivet i [5].

I dagsläget finns alltså en grundläggande specifikation för att säkra vissa aspekter, främst säker (krypterad) överföring av meddelanden via mellanhänder, av publika Web Services. WS Security hanterar överföringen via mellanhänder genom att utöka SOAP-protokollet. På så sätt kan säkerhetskrav som integritet för meddelande, sekretess och autentisering för enskilda meddelanden uppnås hela vägen mellan sändare och mottagare [5].

En annan fördel med WS Security är att XML-kryptering kan användas vilket gör att olika delar av meddelandet kan krypteras med olika nycklar. Detta gör det möjligt att skilja mellan olika delar av meddelandet som är avsedda för olika mottagare [5].



Figur 5. WS Security stack enligt förslag från IBM och Microsoft (Efter [15]).

Det är dock inte bara fördelar med att anamma WS Security-paketet. Det första uppenbara problemet är att standarden är ny. En första version följs i regel av flera andra som rättar till de problem och felaktigheter som kan ha förekommit i föregående versioner. Det finns ingen anledning att tro att det skulle vara någon skillnad för WS Security. Innan standarden använts en tid under realistiska förhållanden kommer olika problem med stor sannolikhet att uppstå [5].

Ett mycket närliggande problem till detta är att mycket få företag idag använder annat än mycket enkla webbaserade tjänster utanför företagets interna nät, vilket i sin tur leder till att inte så mycket ny kunskap om användningen av WS Security kommer fram [21] [2].

Ytterligare ett problem som kan skönjas är att nästa lager i säkerhetsarkitekturen inte är färdigt ännu. Detta innebär att det i dagsläget inte finns någon standardisering för säkerhetspolicys, mekanismer för att etablera autentisering och auktorisering (WS-Trust) samt hur sekretess för meddelanden skall behandlas mellan parter (WS-Privacy). Det är dock viktigt att poängtera att WS Security kan användas utan de specifikationer som kommer att läggas till senare [22].

Som redan poängterats är WS Security en ny standard som saknar djupare utvärderingar. Alternativa äldre säkerhetsstandarder som exempelvis SSL och VPN är betydligt bättre dokumenterade. Så länge ett företag inte använder sig av publika webbaserade tjänster, det vill säga arbetar innanför brandväggen, eller har direkt kommunikation mellan sändare och mottagare (punkt till punkt) finns det egentligen ingen anledning att använda WS Security då de äldre standarderna fungerar på ett tillförlitligt sätt.

Hur ser det ut i företagen?

I dagsläget använder sig inget av de i Serviam deltagande företagen av publika webbaserade tjänster. Det finns förvisso en del projekt som har börjat utnyttja WSDL för att beskriva sina tjänster för potentiella mottagare. Dock är detta informationsbyte i dagsläget mycket begränsat [3]. Ett uttalande som stödjer denna observation kommer från IBM & Microsoft som hävdar att webbaserade tjänster utanför företagets nätverk i dagsläget är svårt att åstadkomma med säkerheten garanterad [3]. Detta uttalande är i första hand baserat på de svårigheter som upplevts i att hantera säkerhetsaspekter för publika webbaserade tjänster.

Det intryck man får från deltagande företag är att det för tillfället inte finns speciellt många färdiga tjänster, vare sig internt eller publikt. Dock finns det gott om planer och framtida projekt som i mycket större utsträckning utnyttjar någon form av webbaserade tjänster. Likafullt är de flesta av dessa ansatser ämnade för intern användning eller för interaktion med närstående partners [3], [4].

Säkerheten för det fåtal existerande applikationerna i företagen är i regel baserad på beprövad teknik som SSL/VPN/TLS kommunicerad över främst http. Det finns dock planer i vissa företag att tillämpa delar av WS Security, främst XML Signature, men efter vad som framkommit är detta inget som ännu är realiserat i praktiken [3], [4].

Några slutsatser

Grundtanken med webbaserad tjänstearkitektur, det vill säga ett världsomspännande nätverk av tjänster som kan utnyttjas med relativt enkla medel, är i dagsläget ett rörligt mål. Dock har vissa fundament för Web Service-arkitekturen såsom XML, SOAP och WSDL uppnått rimlig stabilitet. Utvecklingen av Web Service-arkitekturen är fortfarande intensiv vilket borgar för en fortsatt vidareutveckling och komplettering av dess olika delar.

Web Services vinner idag ökad terräng i företag och offentlig förvaltning. De främsta användningsområdena är för närvarande realisering av tjänster som görs tillgängliga internt i företaget eller i punkt-till-punkt-lösningar med etablerade partners. Ett annat område där viss framgång nåtts och där stora förhoppningar knyts till Web Services rör integrationslösningar som involverar både befintliga och nya system. Parallellt med att utvecklingen går vidare kommer även användningen av Web Service-arkitekturen att accelerera.

Till företag som överväger att tillägna sig detta arbetssätt och att användas sig av Web Services skulle vi vilja ge följande råd:

- Nöj er inte med XML 1.0, Använd i stället XML:s fulla potential, inklusive XML Schema.
- Tänk noga igenom vilka säkerhetskrav som krävs om en tjänst skall användas publikt samt om det kommer att finnas mellanhänder inblandade.
- Om ni känner tvivel angående säkerheten runt publik exponering av Web Services, använd punkt-till-punkt kommunikation som bevisligen är säker.
- Tänk på att tekniska lösningar är verkningslösa om individer på ”insidan” av ett företag har möjligheter att komma åt känsliga källsystem.

- Ge för tillfället upp illusionen om UDDI som en allomfattande distributör av tjänster, fokusera istället på de tekniker som har uppnått en tillräckligt hög mognadsgrad, det vill säga XML, SOAP och WSDL.

- Ha i åtanke att Web service-arkitekturen fortfarande är ung, nya brister kommer att upptäckas likaväl som nya fördelar.

Att Web Service-arkitekturen kommer att fortsätta utvecklas och anammas av allt fler företag är mycket troligt, med tanke på de fördelar som den uppvisar i form av flexibilitet, ekonomi m.m. Det finns mycket som talar för att vi är på väg in i ett nytt sätt att bygga system som kännetecknas av utveckling baserad på tjänster generellt. Man talar redan om tjänstebaserad utveckling (service-oriented computing) som ett begrepp, alldeles oavsett om det rör sig om Web Services eller elektroniska tjänster realiserade på annat sätt.

Referenser

Serviamdokument

[1] Toms A., ”Serviam Literature Survey, Part VII, Web Services Security”, Serviam 2004.
[2] Berild S., ”Konferensrapport från ’1st European Semantic Web Symposium”, Serviam 2004’”.
[3] Toms A., “Notes from meeting at SEB regarding Web Service and Security”, Serviam 2004.
[4] Toms A, “Delprojekt Stödfunktioner-Säkerhet” Power Point bilder, Serviam 2004.
[5] Toms A., ”WS Security-An overview”, Serviam 2004.
[6] Henkel M., “Serviam Literature Study Part I Introduction” Serviam 2004.
[7] Berild S., “XML Schema – en översikt” Serviam 2004.
[8] Berild S., “UDDI – kokar soppa på en spik?” Serviam 2002.
[9] Berild S., ”UDDI – ett par år senare” Serviam 2004.
[10] Berild S., ”Metadata – Vad, När, Hur och Varför?” Serviam 2003.
[11] Berild S., ”Mycket ’meta’ är det” Serviam 2003.
[12] Berild S., ”Web Services – håller dimman på att skingras? – några intryck från konferensen Web Services Edge 2003” Santa Anna IT Research Institute 2003.
[13] Berild S., ”Power point presentation på skatteverket 2004-10-07” 2004.

Övriga källor

[14] World Wide Web Constortium. Tillgänglig på Internet (041204): www.w3.org.
[15] IBM & Microsoft. (2002) Security in a Web Services World: A Proposed Architecture and Roadmap. Version 1.0 April 7, 2002. Tillgänglig på Internet (041217): http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp.
[16] Clark, M. & Irani, R. (2002) Web Services Intermediaries. Fletcher, P & Waterhouse, M (ed.) (2002) *Web Services Business Strategies and Architectures.* Chapter 12. UK: Expert Press Ltd. ISBN: 1-59059-179-8.
[17] Tsalgatidou, A & Pilioura, T (2002) An overview of Standards and Related Technology in Web Services. *Distributed and Parallel Databases.* Pages 135-162.
[18] Fletcher, P & Waterhouse, M. (ed.) (2002) *Web Services Business Strategies and Architectures.* Chapter 17. UK: Expert Press Ltd. ISBN: 1-59059-179-8.
[19] Newcomer, E. (2002) *Understanding Web Services: XML, WSDL, SOAP and UDDI.* UK: Addison-Wesley. ISBN: 0-201-75081-3.
[20] Boncella, R, J (2004) Web Services and Web Service Security. *Proceedings of the Americas Conference in Information Systems.* NY, August 2004. Tillgänglig på Internet (041008): http://www.washburn.edu/cas/cis/boncella/tutorialAMCIS2004.doc.
[21] Web Services Security (2003) *Computer Fraud & Security, Volume 2003, Issue 3, March 2003, Pages 15-17.*
[22] Chou, D,C & Yurov, K (2004) Security development in Web Services environment. *Science Direct*, tillgänglig på Internet (041008): http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6TYV-4D98BTW-1-7&_cdi=5628&_orig=search&_coverDate=09%2F22%2F2004&_sk=999999999&view=c&wchp=dGLbVtb-zSkWb&_acct=C000034819&_version=1&_userid=646852&md5=71b447e76b1cffc986b58a013f93afac&ie=f.pdf.

Del IV

Förvaltning av webbtjänster

Mira Kajko-Mattson
Stockholms Universitet/KTH

Introduktion

Bakgrund

Då man inför en ny teknologi påverkas även relaterade metoder och tekniker. Man kan inte dra full nytta av den nya teknologin om inte etablerade metoder och tekniker anpassas till denna. Ett exempel inom dataområdet är introduktionen av relationsdatabaser. Ett annat är införandet av objektorienterade programmeringsspråk. I båda dessa fall var man tvungna att granska och revidera processerna för nyutveckling, vidareutveckling och underhåll för att hantera den nya teknologin. Man stötte dock på en hel del problem varav många inte gick att förutsäga.

Nyligen har mjukvaruorganisationerna börjat introducera en ny teknologi kallad webbtjänster. Denna innebär att man går över från tätt integrerade och centraliserade system till system bestående av löst integrerade och distribuerade komponenter. Komponenter är dessutom implementerade i olika programmeringsspråk, de exekveras på olika platformar och de kommunicerar med varandra genom väldefinierade gränssnitt som är oberoende av platform och programmeringsspråk.

Webbtjänster utgör en viktig grund för utveckling av nya affärsmöjligheter. De kan tillgodose behoven hos flertalet kunder och allehanda affärsverksamheter. Exempel på användning av webbtjänster kan hittas i allt från B2B och B2C applikationer till e-lärande och liknande.

Webbtjänster innebär stora potentiella fördelar för företagen så som billigare och mindre komplex integration med arvet (legacy systems) och bättre intra- och inter-företagsintegration. Därför lanseras de nu som nästa generations e-businessystem. Man tror nämligen att om man utforskar och satsar på webbtjänster idag så kommer man att bli betydligt bättre förberedd för att möta framtiden. Företag som inte satsar på webbtjänster inom den närmaste framtiden kommer förmodligen att missgynnas konkurrensmässigt.

Problem

Många webbtjänster genomgår ständiga förändringar. Därför innebär de en ny utmaning inom programvaruteknik. Denna utmaning har ännu inte undersökts. Framgångarna för den nya generationens affärssystem beror inte bara på hur man implementerar den nya teknologin, utan även på hur man förvaltar dem. Ur förvaltningsperspektiv finns det många aspekter som måste undersökas. Dessa inkluderar förvaltningsprocesser, produkter som förvaltas, roller involverade i processerna och de förändringar som krävs för att anpassa organisationen för att kunna hantera webbtjänsteapplikationer.

Idag har vi tyvärr inte tillräcklig kunskap om hur man ska förvalta mjukvarusystem av webbtjänster. Denna brist kan innebära att man utför underhåll på ett ineffektivt sätt vilket kan leda till höga förvaltningskostnader och missnöjda kunder. Dessutom kan den medföra att organisationerna reagerar alltför långsamt på nya affärkrav.

SERVIAM delprojekt: Evolution and Maintenance Förvaltning

FÖRVALTNING är ett delprojekt inom SERVIAM. Dess huvuduppgift är att studera vidareutveckling och underhåll av webbtjänster. Delprojektet leds av Mira Kajko-Mattsson vid Stockholms Universitet och KTH. Dess industripartner är SAS.

Syfte och Avgränsningar

Systemförvaltningsområdet är mycket brett. Det beskrivs kort i avsnitt 2.1. Att hantera hela området inom delprojektets ramar är praktiskt omöjligt. Preliminärt har delprojektet därför avgränsats till korrigerande underhåll.

Planer för delprojektet

SERVIAM-delprojektet sträcker sig över två år. Under det första året planerades och utfördes följande huvudaktiviteter:

- Aktivitet 1: Genomförande av litteraturstudie för att ta reda på status inom forskningen
- Aktivitet 2: Studier av industriella processer och identifiering av deras status.
- Aktivitet 3: Utarbetande av förslag till projektaktiviteter för det andra projektåret.

Delprojektets resultat

Under det första projeketåret har följande producerats:

- Aktivitet 1- Literaturstudie: Aktiviteten har resulterat i rapporten [SERV-FORV-9]. Den består av två delar. Första delen beskriver statusen inom traditionell förvaltning. Den andra delen beskriver statusen inom förvaltning av webbtjänster. Rapportens innehåll sammanfattas i Kapitel 2 och 3.

- Aktivitet 2 – Status inom industrin. Aktiviteten har resulterat i en serie rapporter som beskriver statusen inom SAS [SERV-FORV-3 – SERV-FORV-8]. Som redan nämnts i Kapitel 1.4, är förvaltningsområdet mycket omfattande. Att hantera hela området inom SERVIAM-delprojektet är praktiskt omöjligt. Dock har delprojektet haft ambitionen att hantera så mycket som resurserna medgett. Delprojektet ska förtsätta med denna aktivitet under det andra året.

- Aktivitet 3 – Framtida arbete. De leverabler som producerades under Aktivitet 1 och 2 gav underlag för aktiviteter planerade till år 2.

Projektresultaten redovisas i de SERVIAM-rapporter som listats i slutet av denna rapport. Många av dessa är dock fortfarande under arbete.

Näringslivssamverkan

Delprojektet samverkar med företaget SAS, Scandinavian Airline Systems. SAS är Nordens största flygbolag och resegrupp och den fjärde största flygbolaget i Europa mätt i antal passagerare och operativ avkastning. SAS fokuserar både på affärs- och fritidsresor. Dess

huvuduppdrag är att erbjuda transporter till sina kunder. Bolaget erbjuder också andra tjänster såsom hotell.

Idag har SAS ca 3500 personer anställda. Av dessa är 150 personer involverade i utveckling och underhåll av mjukvarusystem. Det låga antalet IT-personal beror på att det mesta av IT-verksamheten är outsourcad till dels ett annat företag, CSC (Computer Sciences Corporation), dels till externa konsulter.

SAS hanterar mjukvarusystem som stödjer deras huvudverksamhet. Just nu har företaget ca 200 datasystem såsom traditionella reservationssystem, inventariesystem, system för spårning av flygaktiviteter, databaser med kundprofiler, hanteringssystem för EuroBonus, passagerarinformationssystem och webbaserade system för informations- och reservationshantering.

Rapportöversikt

Rapporten är indelad i fem avsnitt.

1. *Introduktion* diskuterar bakgrunden till förvaltningsdomänen. Det presenterar också delprojektet Förvaltning inom SERVIAM.

2. *Traditionell Förvaltning* beskriver statusen inom traditionell systemförvaltning.

3. *Förvaltning av Webbtjänster* ger en översikt av webbtjänstsystem.

4. *Forskning på SAS* beskriver de forskningsaktiviteter som delprojektet utförde på SAS och dess resultat.

5. *Förslag för År 2* listar förslag till projektaktiviteter och resultat som ska utföras/produceras under det andra projektåret.

Traditionell förvaltning

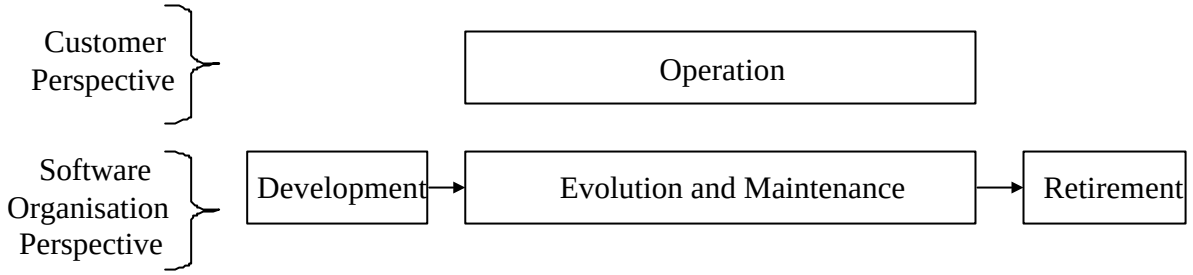
Jämfört med nyutveckling är fortfarande förvaltning ett omoget område. Trots sin ålder lider det fortfarande av många barnsjukdomar. För att hjälpa våra läsare att förstå dem presenterar vi först statusen för området mjukvaruförvaltning och pekar ut problem inom detta. Statusen presenteras i avsnitt 2.1 och problemen beskris i avsnitt 2.2. Med hänsyn till omfångsbegränsningen för denna rapport, kan vi inte beskriva hela området. För den som vill veta mer rekommenderar vi rapporten [SERV-FORV-9].

State of the Art

Definition

IEEE definierar förvaltning som “en process för modifiering av mjukvarusystem eller mjukvarukomponenter efter leverans för att rätta fel, förbättra prestanda eller andra egenskaper, eller anpassa systemet till en förändrad miljö” (IEEE Std. 610.12-1994). Figur 1 åskadliggör IEEE definitionen. Den förutsätter att förvaltning börjar först när systemet har levererats till kunden. Den förutsätter också att livcyckeln består av tre huvudfaser (1) nyutveckling, (2) drift

och förvaltning (på engelska evolution and maintenance) och (3) avveckling. I den första fasen bygger man ett helt nytt mjukvarusystem. Efter det att systemet har utvecklats färdigt och levererats till kunden pågår två parallella faser, drift och förvaltning. Medan kunder använder sig av systemet stödjer (*supportar*) mjukvaruorganisationen dem i deras dagliga verksamhet. Samtidigt fortsätter mjukvaruorganisationen att vidareutveckla och underhålla systemet. Slutligen, när systemet inte längre är användbart, avvecklar man det och ersätter det förmodligen med ett nytt och tjänligt system.



Figur 1. Huvudfaser i mjukvarulivscyckeln

Omfattning

Man kan se på förvaltning ur olika perspektiv: livscykel-, process-, produkt- och organisaitonsperspektiv. Livcyckelperspektivet har redan skildrats i föregående delkapitlel och i figur 1. Ur processperspektiv omfattar förvaltning olika typer av aktiviteter. Som framgår av tabell 1 kan dessa klassificeras som antingen korrigerande, perfektiva, adaptiva eller preventiva.

Korrigerande underhåll definieras som en aktivitet under vilken man rättar fel i hård- eller mjukvara. Det förstås som en process under vilken man löser problem som rapporterats av användarna av berörda mjukvarusystem. Ett exempel på ett mjukvaruproblem kan vara det faktum att en robotarm svänger till vänster och inte till höger i en viss situation.

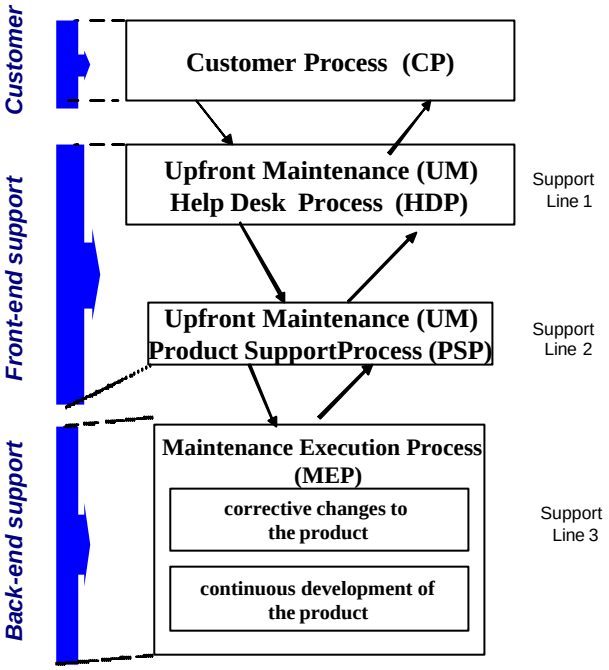
Tabell 1. IEEE:s definitioner av förvaltning (ANSI/IEEE STD-610.12, 1990)

Corrective maintenance: Maintenance performed to correct faults in hardware or software. **Adaptive maintenance**: Software maintenance performed to make a computer program usable in a changed environment. **Perfective maintenance**: Software maintenance performed to improve the performance, maintainability, or other attributes of a computer program. **Preventive maintenance**: Maintenance performed for the purpose of preventing problems before they occur.

Perfektivt underhåll definieras som underhåll vilket utförs för att förbättra prestanda, underhållbarhet (lättare att underhålla) eller andra egenskaper hos ett datorprogram. Det förstås huvudsakligen som processen för att lägga till ny funktionalitet till ett redan befintligt

mjukvarusystem. Ett exempel på ny funktionalitet kan vara en ny sorteringsfunktion i en industrirobot eller en ny algoritm som gör att roboten arbetar snabbare.

Preventivt underhåll definieras som underhåll som utförs i syfte att förebygga problem innan de inträffar. Detta innebär att förvaltningsorganisationen själv letar och hittar problem innan kunderna klagar. Ett exempel kan vara att en mjukvaruingenjör noterar en defekt i ett redan levererat mjukvarusystem. Han är säker på att denna defekt, om den exekveras, kommer att orsaka operationella problem för kunderna. Han beslutar därför att korrigera defekten och leverera den defektfria mjukvaran till kunden så snart som möjligt.



Figur 2. Karta över supportnivåer och deras kommunikationslänkar

Beträffande organisationsperpektivet ger olika team/grupper, avdelningar eller organisationer support på olika (support-) nivåer. Som framgår av figur 2 utgör supportnivå 1 den första kontaktpunkten mot kunden. Den stödjer kunden i dennes dagliga arbete. Supportnivå 2 stödjer Supportnivå 1 vid hantering av mer komplexa kundkrav. Supportnivå 3, slutligen, gör förändringar i mjukvarusystemet vilka har förmedlats via supportnivåerna 1 och 2.

När det gäller produktperspektivet inom mjukvaruunderhåll definierar systemförvaltning ett kvalitetsattribut som heter underhållbarhet. Underhållbarhet täcker olika kvalitetsaspekter som ska implementeras i systemet för att underlätta underhållsarbetet i framtiden.

Problem inom förvaltning

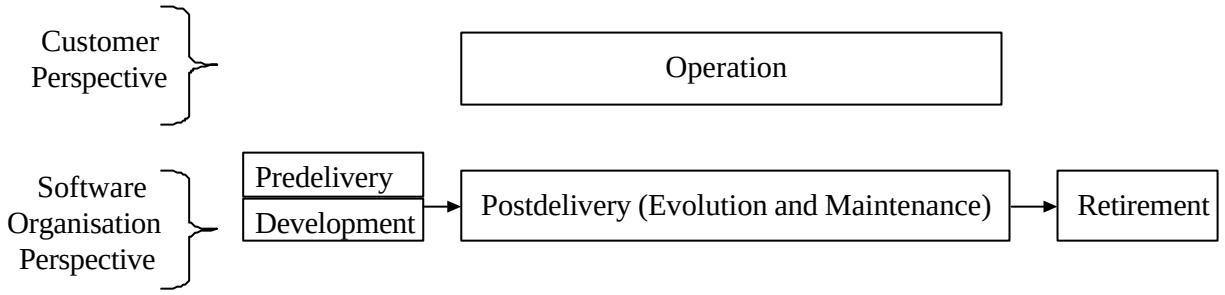
Förvaltningsområdets brist på mognad gör att detta idag lider av många osäkerheter och problem. Några av problemen är (1) val av ett rättvisande namn för området, (2) definitionen av förvaltning, (3) klassificering av förvaltningsaktiviteter, (4) oenighet i användandet av förvaltningsterminologin, (5) brist på förvaltningsprocessmodeller, (6) mätningsproblem, och (7) avsaknaden av en underhållbarhetsmodell. Dessa problem bidrar starkt till det kaos som för nuvarande råder inom förvaltningsområdet. Nedan beskriver vi dem kort.

Benämningen på området

Bland vissa råder uppfattningen att den engelska benämningen på förvaltningsområdet - "maintenance" - är felaktig. Den leder tankarna till "att fixa buggar" men den återspeglar inte de vidareutvecklingsaktiviteter under vilka ny funktionalitet läggs till, så kallade utvidgningar. Av denna anledning borde man i stället kalla detta område för "evolution och maintenance". [2]

Definition av förvaltning

Det har rått stor oenighet gällande definitionen av begreppet mjukvaruförvaltning. Fortfarande råder det oenighet om omfånget av förvaltning, dess beståndsdelar, tidsspännvid och om var man ska dra skiljelinjen mellan nyutveckling och förvaltning. Den största kritiken mot IEEE:s definitionen gällde dess påstående att förvaltning är en typisk efterleveransaktivitet (se IEEEs definition ovan). Det har föreslagits att förvaltning i stället borde påbörjas före leverans. Som visas i figur 3 borde denna aktivitet pågå parallellt med nyutveckling. Förvaltarna borde följa utvecklingen av systemet och kontinuerligt utvärdera dess underhållbarhet.



Figur 3. Förslag till nya livs cyckelfaser

Förvaltningstyper

Invändningar har rests mot IEEE:s definitioner av förvaltningstyper. Det hävdas att klassificeringen inte är uttömmande och ömsesidigt uteslutande. Deras typdefinitioner är för generella, de överlappar varandra och förstås olika av mjukvaruutvecklarna. De är inte tillräckligt förklarande och vilket kan leda till missförstånd och oklara gränsdragningar dem emellan.

Oenighet i användandet av förvaltningsterminologi

Mjukvarusamfundet använder sig inte av en gemensam begreppsapparat inom förvaltningsområdet. Terminologin som används av forskare och industri skiljer sig avsevärt. Även inom respektive grupp, t o m inom en och samma organisation, förekommer det att olika termer används för ett och samma koncept.

[2] Som kontrast till mjukvaruområdets snäva tolkning av begreppet "underhåll" kan nämnas att exempelvis flygindustrin talar om tre typer av underhåll – förebyggande underhåll, avhjälpande underhåll och modifieringar. Modifieringar omfattar allehanda förändringar inklusive tillägg av ny funktionalitet.

Brist på underhållsprocesser

Än idag saknas det tillräckligt detaljerade processmodeller för förvaltning. Enligt rapportförfatterens kännedom finns det bara tre internationellt accepterade standardmodeller – IEEE 1219, den nyligen introducerade model ISO/IEC FDIS 14764 (ISO 1999) och Service Supportstandarden ITIL (IT Infrastructure Library Service Support Model) (OCG, 2004).

De första två modellerna är emellertid mycket generella. Den tredje modellen är den mest detaljerade idag. Dock är den fortfarande alltför generell. Dessa modeller ger inte en tillräckligt djup förståelse av omfånget av varje förvaltningstyp. De ger heller inte tillräcklig insyn in i respektive förvaltningstyp. I princip täcker de alla förvaltningstyper med en och samma processmodell. Därför kan de vilseleda de organisationer som försöker tillämpa dessa standarder i sitt arbete att implementera och förbättra sina förvaltningsprocesser. Organisationer skulle idag behöva förslag till processaktiviteter och nogranna förklaringar och motivieringar för implementering av dem. Vad vi känner till finns det bara en sådan detaljerad modell. Just nu är den under utveckling. Den heter Corrective Maintenance Maturity Model (CM^3) och den täcker fn en del av förvaltningensdelområdet – korrigerande underhåll.

Mätning av underhållskostnaden

Mjukvarusamfundet är enigt om att förvaltningkostnaderna är mycket höga idag. Enigheten slutar emellertid här. När man undersöker förvaltningskostnaderna i relation till de totala utvecklingskostnaderna i vad som hittills publicerats ser man att resultaten avviker starkt ifrån varandra. Som redan nämnts visar de publicerade siffrorna att förvaltning kostar mellan 40 – 90% av de totala livcyckelskostnaderna. En förklaring till denna stora diskrepans i mätresultaten är faktumet att vi inte delar en gemensam och objektiv förståelse av förvaltningsområdet, förvaltningstyper och ingående processer. Vi har inte ens en samstämmig syn på vad förvaltning innebär, dess omfång och dess förhållande till nyutveckling.

Brist på underhållbarhetsmodeller

Underhållbarhet är en viktig produktivitets- och överlevnadsfaktor. Den innebär högre produktivitet till lägre förvaltningskostnad. Den är en förutsättning för att enkelt behålla mjukvarusystemen friska, för att förlänga deras livscyckel och för att minimera deras livscyckelkostnader. Idag är underhållbarhet definierad på en mycket abstrakt och generell nivå. På en mer detaljerad nivå uppfattas underhållbarhet olika. Det finns ingen universiellt accepterad och detaljerad underhållbarhetsmodell. Till följd av detta är det idag omöjligt att uppskatta underhållbarheten för systemen.

Förvaltning av webbtjänster

I det här avsnittet ges en kort introduktion till webbtjänster. Avsnitt 3.1 ger en kort översikt av webbtjänstbaserade system. För att kunna föreslå ett ramverk för förvaltning av webbtjänsteapplikationer har vi identifierat skillnader mellan traditionella och webbtjänsterbaserade system. Dessa beskrivs i avsnitt 3.2. Vi har också listat problem som man har stött på eller kan stöta på när man förvaltar webbtjänster. Dessa beskrivs i avsnitt 3.3.

Överblick av webbtjänstsystem

Webbtjänster är kompletta, självbeskrivande och modulära applikationer som kan publiceras och anropas över nätet. De är baserade på ett antal teknologier som tillåter att oberoende, löst kopplade enheter som är implementerade i olika programmeringsspråk och som körs på olika plattformer kan kommunicera med varandra via väldefinierade gränsnitt. En grupp av webbtjänster som kommunicerar med varandra på det här sättet bildar ett webbtjänstsystem i en tjänsteorienterad arkitektur. Webbtjänster kan användas som byggklossar i många olika applikationer och därför kan de lätt återanvändas inom och utanför en organisation. De kan också erbjudas till en rad användare från samma eller andra organisationer.

Webbtjänster har många fördelar. De leder till utökad interoperabilitet (samverkan) via en standariserad integrationsmekanism. De leder till högre automationsnivå, och därmed till en avsevärd minskning av mängden manuellt arbete. De leder till ett standardiserat sätt för att integrera alla typer av system, från gamla stordatorsystem till mobila klienter, interna tjänster och externa system som tillhandahålls av affärsparnters.

Webbtjänsternas lösa koppling leder till stor flexibilitet. Implementeringen av individuella komponenter kan bytas ut utan att orsaka någon skada på andra systemdelar så länge som gränsnittet inte ändras. Webbtjänster gynnar återanvändning av affärsfunktionalitet genom återanvändning av redan utvecklade och implementerade tjänster och system. Återanvändning av komponenter accelererar utveckling, vidareutveckling och underhåll genom att låta utvecklarna använda sig av färdiga komponenter som byggklossar. Applikationer och nya lösningar kan implementeras snabbare, billigare och med lägre risk.

Potentiellt erbjuder webbtjänster möjligheter att binda tjänster vid runtime, dvs under exekvering. Detta i sin tur leder till lägre användningskostnader. Organisationer kan betala för tjänster när de behöver och använder dem. Webbtjänster skyddar existerande mjukvaruinvesteringar genom att erbjuda enkla och billiga metoder för integration och återanvändning av gamla, ofta affärskritiska system. Implementation av tjänsteorienterade arkitekturer tillåter företagen att bygga dynamiska *e-business*-arkitekturer, som är extremt *agila* till nya affärskrav. Snabbare svar på nya, uppkommande och oundvikliga affärskrav hjälper företagen att erbjuda högkvalitativa tjänster, ger konkurrensfördelar och genererar därmed avsevärda vinster.

Faktorer unika för förvaltning av webbtjänster

Skillnaderna mellan förvaltning av webbtjänstsystem och traditionella mjukvarusystem bör beaktas från flera olika perspektiv. Dessa är arkitektur-, affärsmodell-, process-, produkt-, roll-, och organisationsperspektiv. De första två perspektiven ger emellertid grunden för identifiering av skillnader för de övriga perspektiven.

Arkitektonisk perspektiv

Med hjälp av webbtjänster kan man bygga komplexa mjukvarusystem genom att integrera återanvändbara komponenter tagna från olika källor. Några tjänster kan byggas *in-house*, medan andra kan komma från externa leverantörer eller samarbetande organisationer. Tjänster kan utvecklas i olika programmeringsspråk, de kan exekveras i heterogena miljöer och de kan distribueras inom olika organisationer och över vidsträckta geografiska avstånd. Denna ansats erbjuder möjligheter till stor samverkan mellan olika organisationer och möjligheter till extern och intern integration.

Affärssperspektiv

Webbtjänster har introducerat en ny affärmodel för användning, marknadsföring och försäljning av mjukvaruapplikationer. Först och främst har tjänstebegreepet introducerats vilket innebär att man kan använda sig av en viss tjänst men man behöver inte äga den. Denna ide är analog med dagens affärsvärld, där tjänster erbjuds och köpes när de behövs. Därför har konceptet av mjukvaruägarskap förändrats, från något som ägs till något som används.

Webbtjänster gynnar affärssamarbete. Bred och nära extern integrering med kunder, försäljare, partners och andra parter har blivit en verklighet. Med hjälp av webbtjänster har dagens affärer blivit mer globala och de kan drivas av många samverkande parter. Därjämte har möjligheterna att publicera, upptäcka och använda sig av tjänster över nätverket resulterat i en tjänstemarknad där kunder lätt kan hitta och anskaffa de behövliga tjänster som levereras av tjänsteleverantören. Tjänster kan användas av många kunder samtidigt oavsett var de befinner sig. Denna affärsmodell främjar direkt en bred mjukvaruintegration, anskaffning/användning och ombesörjande av externa och interna komponenter.

Produktperspektiv

Effektiv förvaltning kräver att man har bra förståelse av mjukvarusystem, deras funktionella krav, design, kodens interna stuktur, systemets stabilitet, modiferingsbarhet och sannolikheten för sidoeffekter. Brist på denna förståelse är en av grundorsakerna till defekter som påverkar produkternas korrekthet.

Idag möter systemförvaltarna många hinder som hindrar dem från att få en fullständig kunskap om mjukvarusystemen. Några av dessa hinder är (1) undermålig och bristfällig systemdokumentation, (2) begränsat utbud av lämpliga verktyg, (3) starkt varierande releaser, (4) kontinuerligt ökande systemstorlek och komplexitet, och (5) tidsrestriktioner. Dessa hinder resulterar i svårigheter att kontrollera negativa sidoeffekter som uppstår vid systemförändringar. Förvaltarna kan ovetande introducera nya fel varje gång de gör förändringar i mjukvarusystemen. Med tiden kan förvaltarna förlora kunskapen om mjukvarusystemen, vilket i sin tur leder till ännu större svårigheter att förvalta systemen. Oavsett om mjukvarusystemen är ordentligt dokumenterade eller inte måste förvaltarna alltjämt underhålla sina kunskaper om deras funktionalitet och struktur.

En av de största kostnaderna inom förvalning är tiden som förvaltarna tillbringar med att förstå sig på existerande mjukvarusystem. Olika undersökningar har visat att de tillbringar mellan 40-60% av den totala tiden när de arbetar med förändringar. Man tror att förståelsen av webbtjänster kommer att kräva ännu mer resurser. Det beror på att webbtjänster är distribuerade och att deras arkitektur är uppbyggd i flera lager. De består av många tjänster, som ofta kommer från olika leverantörer. Därtill kommer att denna teknologi tillåter en organisation att integreras med affärsprocesser tillhörande olika kunder, leverantörer eller samarbetspartners. Därför förlyttar sig gränsen av mjukvara mot externa parter. Denna trend leder till skapandet av begreppet ”service supply chains” eller på svenska ”tjänsteförråds/lager-kedjor” där den kontrakterade funktionaliteten, högnivåtjänster som erbjuds av huvudentreprenörer, kan bestå av mindre subtjänser, och så vidare, i rekursivt bygga tjänstekedjor. Dessutom exponerar vi våra tjänster för externa parter, t ex samarbetande organisationer. Alla dessa faktorer resulterar i en arkitektonisk modell vars systemdelar (tjänster) kan spänna över flera samverkande applikationer som tillhör olika organisationer, kunder, och som kan erbjudas av flera leverantörer.

Nästa stora skillnaden är den stora utvecklingsbarheten hos webbtjänster. Tjänstearkitekturer underlättar snabbare komponentförändringar. Den orsakas av en flerkundsmodell och en finfördelad stuktur. En flerkundsmodell innebär att en tjänst kan användas av många applikationer samtidigt. En finfördelad stuktur innebär att man kan bryta ner arkitekturen i en mängd individuella enheter som lätt kan förändras och återanvändas. Tjänster utvecklas snabbt och kontinuerligt, och varje förändring introducerar potentiellt nya problem. Alla dessa faktorer resulterar i en situation där komponenter förändras och uppgraderas mer frekvent än traditionella system.

Andra skillnader, som redan pekats ut, är distribuerad arkitektur, många samverkande komponenter, bred integration, olika programmeringsspråk, och olika driftsmiljöer. Man kan säga att ingenting är nytt jämfört med de traditionella systemen, men, i kontexten av webbtjänster har detta drivits till sin spets. Vi har många fler komponenter, fler programmeringsspråk, fler plattformar, etc.

Organisatoriskt perspektiv

Webbtjänster och deras integrationsmöjligheter både inom och utanför organisationer har markant påverkat den moderna affärsmodellen. Gränserna för affärsorganisationer och affärssystem som stödjer dem har förflytats till den yttre världen. Denna situation har introducerat en ny stor utmaning för förvaltningsorganisationerna. Hur ska man förvalta webbtjänstesystem som till stora delar ligger utanför kontroll av förvaltningsorganisationen? Systemen exekveras över många externa parter och därför är deras hantering och förvaltning distribuerad och utförd av flera samverkande förvaltningsorganisationer.

Traditionella mjukvarusystem är, i motsats till webbtjänstsystem, normalt mer internt fokuserade. De använder sig inte av så många externa komponenter i sina arkitekturer som webbtjänstapplikationer. Således är deras förvaltning mer intern och kan till största delen utföras av en enda organisation. Med hjälp av webbtjänster har affärsorganisationer blivit mer globala så även förvaltning av mjukvarusystem. Tjänstearkitekturer har avsevärt förändrat sättet att se på förvaltningsorganisationen, från en enskild organisation till en samling av distribuerade och samverkande organisationer.

Roller

I föregående avsnitt nämnde vi att webbtjänstebaserade system är distribuerade, att de utvecklas mycket snabbt och att de består av många moduler som kommer från olika håll. Vi visade också att förvaltning av webbtjänster till stora delar ligger utanför räckvidden för en enstaka förvaltningsorganisation och därför måste utföras av flera samarbetande organisationer. Denna skillnad gör att vi måste anta att det även finns skillnader i förvaltningsteamstrukturen, dvs, dess roller, ansvar och kompentens.

Den mest synliga skillnaden är faktumet att det inte finns några tydliga gränser mellan system och subsystem. Systemgränser är inte fastställda, de spänner över flera parter och de är kandidater för kontinuerlig förändring. Istället för att ha klart urskiljbara system har vi en mängd av tjänster som tillsammans tillhandahåller någon funktionalitet. Därtill kommer att samma tjänster kan bestå av och användas av många olika applikationer. De traditionella förvaltningsrollerna tilldelas huvudsakligen vissa system eller moduler med klart definierade gränser och relationer till samverkande parter. Antalet relationer till andra applikationer är avsevärt lägre jämfört med webbtjänster. En annan skillnad är den stora heterogenitet av webbtjänstimplementationer

(många olika programmeringsspråk och operationella miljöer) som måste hanteras av supportrollerna.

Problemundersökning

Huvudmålet med problemundersökning är att återskapa och diagnostisera rapporterade problem. Supportingenjörerna börjar med problemundersökning för att kunna samla in information om de rapporterade problemen. De undersökta systemen är emellertid distribuerade och spridda over många externa parter. Detta resulterar i ett tjänsteleveransnätverk (service supply chains) som tillhandahåller en avsevärd del av funktionalitet från externa källor. Huvudskillnaden jämfört med traditionell förvaltning är det faktum att arkitekturkunskapen som krävs för att undersöka problemen delvis är dold i dessa tjänstenätverk. Man har begränsad insyn i arkitekturen, tjänstestrukturen och implementeringsdetaljer. Insynen inskränker sig bara till de tjänster som finns inom organisationen. Denna situation har också stor påverkan på andra underhållsaktiviteter, såsom förändringshantering, testning, releasehantering, sidoeffekt- (verkans) och grundorsaksanalyser.

Bred inter-organisationell integrering och användandet av externa tjänster skapar en situation där problemundersökningsprocessen inte längre ligger på en enda förvaltningsorganisation. I stället måste den utföras i samarbete med en rad supportenheter som tillhör olika organisationer. En annan fundamental skillnad är att problemundersökningsaktiviteter görs i en mycket snabbt förändrad miljö, till exempel tjänster som kontinuerligt förändras. Det beror på att tjänstebaserade system betjänar många olika kunder och att alla deras krav måste mötas. Båda dessa skillnader är vanliga för alla förvaltningsaktiviteter.

Verkansanalys

Målet med verkansanalys är att bedöma effekten av de förändringar som görs i systemet. Effekten utrycks normalt som en mängd av förändringar avseende vissa systemdelar och tiden och ansatsen som krävs för att implementera dessa förändringar. De skillnader som gäller vid problemundersökning gäller även här vid verkansanalys. Det finns emellertid några nya frågeställningar som måste analyseras.

Under verkansanalys koncentrerar vi oss inte bara på våra system (som består av olika slags interna och externa tjänster) utan även på system vilka vi förser med funktionalitet, tex system som är beroende av oss. Man bör inte glömma att i webbtjänstsammanhang både använder vi själva och tillhandahåller tjänster till andra parter. En sådan situation är inte ny, men den kan ledas till sin spets med webbtjänstebaserade system. Effekten av förändringar påverkar ett stort antal kunder. Analysprocessen måste utföras över ett stort antal tjänster, system och organisationer.

Grundorsaksanalys

Under grundorsaksanalys försöker förvaltarna att identifiera grundorsakerna till mjukvaruproblem. De underligande problemen kan finnas i processer, produkter, och resurser. Därför går grundorsaksanalyser mycket mer på djupet än problemundersökningar, som huvudsakligen koncentrerar sig på att reproducera och diagnosticera de rapporterade problemen. Den söker de underliggande orsakena till problemen och initierar åtgärder för att förbättra och rätta situationen, dvs förebygga en upprepning av problemen i framtiden.

När det gäller tjänstearkitekturer med många externa tjänster tillhörande andra affärspartners, verkar traditionell grundorsaksanalys vara utom kontroll för en enda organisation. Den insyn i data som krävs för den här processen är begränsad, dvs den inskränker sig till bara en organisation. Även om vi får tillgång till information från externa organisationer och hittar orsakena, kan vi inte direkt göra rättningar. Vi behöver en helt ny ansats för att hantera grundorsaksanalys vars hantering är spridd over flera olika organisationer.

Förändringshantering

Förändringar är huvudsakligen ett resultat av bekräftade problem eller nya affärsbehov. Huvudmålet med förändringshantering är att säkerställa att de standariserade metoderna och procedurerna används för att hantera alla förändringar och för att minska effekten av förändringsrelaterade problem på mjukvarukvaliteten. Webbtjänstarkitekturer har introducerat en fler-kundsmodell för användning och delning av komponenter. Webbtjänster kan användas av många kunder samtidigt oavsett om de finns inom en och samma organisation eller ej. Dessutom, vilket nämnts tidigare, är webbtjänster under kontinuerlig och snabb utveckling jämfört med andra arkitekturer. Denna situation driver fram nya frågor rörande förändringshantering.

Ett stort antal samtidiga användare i kombination med med kontinuerligt uppkommande affärskrav kan resultera i ett avsevärt antal samtidiga förändringskrav för specifika tjänster. Det totala antalet krav förväntas vara högre än i de traditionella arkitekturerna. Dessutom kan dessa krav se olika ut och även stå i strid mot varandra. För att lägga ytterligare ved på brasan kommer förändringshantering att utföras av många underhållsorganisationer.

Releasehantering

Releasehanteringsfasen ansvarar för att leverera mjukvara och dokumentation till kunden. Den snabba utvecklingen av webbtjänster påverkar releasehanteringsfasen. För att kunna möta kundens behov resulterar många förändringar i ett stort antal versioner av specifika tjänster. Det orsakas av faktumet att våra tjänster kan användas av många kunder parallellt och att alla dessa kan ställa olika krav. För att kunna införa nya komponentversioner måste vi säkerställa kompatibilitet med alla kunder som använder sig av dem.

Från ett övergripande perspektiv består vårt system av en samling av tjänser och var och en av dessa kan ha många olika parallella versioner. Således är denna situation mycket annorlunda än för traditionella systemen.

Testning

Webbtjänstapplikationer tillåter byggande av komplexa affärssystem. För att säkerställa hög kvalitet hos dessa system måste vi implementera en effektiv testningsprocess. Testning av distribuerade system är emellertid svårare än av tradtionella system. Dessutom har webbtjänster många fler versioner av specifika moduler, vilket genererar mer testning. De skillnader som ska tas hänsyn till när man testar distribuerade system är (1) heterogenitet (systemen består av komponenter skrivna i olika programmeringsspråk och exekveras på olika platformar), (2) tillgång på mjukvarukoden och (3) utvecklingsbaret.

Problem vid förvaltning av webbtjänster

De store problem som man möter vid förvaltning av webbtjänster härstammar från det faktum att det är en ny teknologi som används. Ur förvaltningsperspektiv är denna teknologi ny för förvaltningsorganisationerna. Dessutom fattas det förvaltningsprocesser eller *best practices* som skulle kunna hjälpa dessa organisationer att hantera denna teknologi.

Vi har identifierat två grupper av problem: gamla kända problem som är gemensamma för alla teknologier och arkitekturer, och nya problem som är typiska för webbtjänstapplikationer. När det gäller den första gruppen är de bara yttringar av gamla problem i en ny miljö. Exempel på sådana är kvalitetssäkring, hantering av externa komponenter och testning av distribuerade system. Omfattningen av dessa problem i kontexten av webbtjänster är emellertid avsevärt större.

Därför kräver några av dem ett omtag för att kunna de ska kunna hanteras på ett effektivt sätt. Beträffande de nya problemen så är dessa relaterade till webbtjänstkedjor, samarbete med många externa företag, och förekomsten av tjänster som man inte själv äger. Oavsett vilken grupp av problem det gäller måste följande problem hanteras:

- **Begänsad insyn i produktstukturen**: Den arkitekurella kunskapen (implementerings-detaljer och tjänststrukturen) som krävs för förvaltningsaktiviteter är delvis inte åtkomlig för förvaltarna, dvs, den är dold i tjänsteleveransnätverk. Organisationer har ett lagstadgat intresse av att skydda implementeringsdetaljer för sina tjänster. Dessutom kan de ofta av samma skäl inte ge information om leverantörer längre ned i leveransnätverket. Därför har vi inte en fullständig kunskap om och overblick över det förvaltade systemet.

- **Begränsad insyn i förvaltningsprocessen**: Förvaltning ligger utanför en viss organisations kontroll. Alla förvaltningsaktiviteter (tex problemundersökning, förändringshantering, sidoeffektanalys, testning och liknande) inom processen är starkt beroende av samarbete med andra externa förvaltningsorganisationer, dvs organisationer som tillhör olika parter och är utanför en organisatons räckvidd. Därför är kvalteten på, och framskridandet för, förvaltningsprocessen beroende av garantier från externa partners.

- **Ökad komplexitet i förvaltningsprocesser**: Webbtjänster är höggradigt distribuerade och består av många enheter som kommer från olika ställen. Dessutom förändras komponenterna mycket snabbt. Detta kan introducera avsevärd komplexitet i förvaltningsprocessen.

- **Oklart ägarskap**: Webbtjänster gynnar skapandet av tjänsteleveransnätverk, där den slutgiltiga (kontrakterade) funktionaliteten kan bestå av många subtjänster. Dessutom har dessa tjänster inte alltid en ägare. De kan användas utan att ägas. Dessa faktorer resulterar i en situation där det är oklart vem som ansvarar för förvaltningen av dessa komponenter.

- **Ineffektiv teamstruktur**: Idag bygger de vanliga teamstrukturerna huvudsakligen på system- eller modulägarskap, dvs roller som tilldelas ansvaret för vissa system eller moduler. Dessa stukturer är inte optimala för att hantera webbtjänstebasede system. I dessa system kan vissa komponenter delas och bestå av många andra tjänser. Detta leder i sin tur till ett komplext nätverk med många beroenden. Det finns inga tydliga gränser mellan system och subsystem. Därför måste existerande roller och deras ansvarsområden förändras för att kunna hantera supportaktiviteterna effektivt.

- **Brist på nödvändig kunskap**: Webbtjänstearkitekturer kräver att förvaltarna har bredare tekniska och affärsmässiga kunskaper för att kunna förvalta systemen på ett effektivt sätt. Detta omfattar kunskaper om gällande distribuerade arkitekturer, tjänstemodellering och kommunikation, olika driftsmiljöer, olika språk samt integrationsaspekter såsom middleware. Dessutom förväntnas supportingenjörerna att ha bredare affärskunskaper (de som gäller affärsprocesser) i domänen som de arbetar i. Det är viktigt eftersom tjänsteorienterade system består av affärsprocesser.

Forskning på SAS

I det här delkapitlet beskriver vi aktiviteter som genomförts på SAS.

Projektaktiviteter

Förvaltningsdomänen är mycket stor. Därför är det omöjligt för Serviam-delprojektet att undersöka den i dess helhet. Vi har därför tillsammans med SAS kontaktpersoner kommit överens om valet av lämpliga delområden. Vi har också kommit överens om att delprojektledaren kan utöka antal delområden, om tiden och resurserna tillåter detta.

- De initialt valda delområderna var:
- Problemhaneringsprocessen
- Servicenivåavtal

Dessa delområden utökades sedan med:

- Akut (emergency) problemhantering
- Testning inom korrigerande underhåll
- Förändringshantering
- Releasehantering.

För närvarande utvärderar vi dessa proesser. Utvärderingen genomförs mot existerande processmodeller såsom CM^3 och ITIL. Vid slutet av projektets andra år ska denna utvärdering kompletteras med en utvärdering mot ramverket för förvaltning av webbtjänster vilket vi presenterar i nästa avsnitt.

Ramverk för förvaltning av webbtjänster

Under delprojektets andra år kommer förvaltningsdelprojektet att skapa ett ramverk för förvaltning av webbtjänser. I ramverket kommer att föreslås generella förändringar som ska införas i förvaltningsprocesserna och var dessa förändringar ska implementeras. Dessa förändringar kommer att gälla olika förvaltningskomponenter såsom roller, processer, och organisationer. I initiala faser av ramverkskonstruktionen identifierade vi skillnader mellan förvaltning av webbtjänstsystem och traditionella system. Vi har också redogjort för de problem som man kan möta när man förvaltar webbtjänstapplikationer.

Vi är övertygade om att identifieringen av problemen som presenterades i avsnitt 3.3 kommer att hjälpa organisationer att få en generell överblick av vad man kan förvänta sig när man går över till webbtjänster. Dessutom kommer det att hjälpa dem att börja planera för hur dessa system ska hanteras på ett produktivt och kostnadseffektivt sätt. Där utöver är identifieringen av skillnader och förändringar i förvaltningsmodellen ett nyckelelement när man designar och utvecklar processer och andra förvaltningsaspekter. Detta kommer att hjälpa företagen att peka ut de processelement som måste modifieras och anpassas.

Även om vi har tillhandahållit ett ramverk så kanske detta inte alltid är återanvändbart i sin helhet inom olika organisationer. Genom att lista de allmänna skillnaderna ger vi organisationerna möjlighet att bättre förstå vår modell. Detta kommer i sin tur att hjälpa dem att anpassa sina processer för förvaltning av webbtjänstsystem. Sålunda kommer det att hjälpa dem att uppnå sina affärsmål, t ex nöjda kunder och/eller leverantörer, kvalitetsförbättringar i erbjudna tjänster och lägre kostnader.

När vi konstruerar ramverket kommer vi att beakta följande:

- Hur bygger vi upp förvaltningsorganisationen för att försäkra oss om effektivitet och kostnadseffiktivitet i förvaltningsprocessen?
- Hur kan vi synkronisera och försäkra oss om kvalitet i förvaltningsaktiviteterna som utförs av många externa parter?
- Ska vi begränsa förvaltningssamarbetet till de primära leverantörerna, dvs leverantörer som tillhandahåller den slutgiltiga funktionaliteten?
- Hur kan vi utföra förvaltningsaktiviteterna då synlighet för både arkitekturen och affärssystemens struktur är begränsad?
- Hur bör vi strukturera förvaltningsteamen (roller, ansvar) inom förvaltningsorgaisationen för att hantera webbtjänster på ett optimalt sätt?
- Hur kan vi försäkra oss om att den information om arkitekturen som krävs för alla förvaltningsaktiviteter är aktuell för system som utvecklas snabbt och kontinuerligt?
- Vem kommer att utföra förvaltningsaktiviteter för kedjor av tjänster och då tjänsterna som används saknar ägare?

Det är inte säkert att vi kommer att kunna besvara all dessa frågor inom vårt framtida ramverk, men vi ska göra vårt bästa.

Referenser

SERV-FORV-6: Kajko-Mattsson M, Fernis U, 2004, Problem Management at EDISS Sales at SAS, SERVIAM Report, 2004.

SERV-FORV-7: Kajko-Mattsson M, Petersen A, 2004, Problem Management at HB System Support, SERVIAM Report, 2004.

SERV-FORV-8: Kajko-Mattsson M, Petersen A, Winther P, Vang Brian, Emergency Problem Management at SAS, SERVIAM Report, 2004.

SERV-FORV-9: Kajko-Mattsson M, The State of Art within Evolution and Maintenance of Web Services, SERVIAM Report, 2004.

SERV-FORV-17: Kajko-Mattsson M, Fernis U, Acceptance testing at SAS, SERVIAM Report, 2004.