SERVIAM PROJEKTRAPPORT:

Förvaltning

av

Web services

Mira Kajko-Mattsson
Department of Computer and Systems Sciences
Stockholm University and Royal Institute of Technology
Forum 100
SE 164 40 KISTA
SWEDEN

2005-09-30

Dokument-id: SERV-FORV-10

Sammanfattning

Mjukvaruorganisationer världen över har nyligen börjat introducera en ny teknologi kallad Web-services. Då man inför en ny teknologi är man tvungen att granska och revidera etablerade metoder och processer för nyutveckling och förvaltning för att hantera den nya teknlogin. Annars kan man inte dra full nytta av denna.

Idag har vi tyvärr inte tillräcklig kunskap om hur man ska förvalta Web-services. I denna rapport studerar vi ett antal faktorer som måste beaktas när man går över till den nya teknologin. Först och främst tittar vi på arkitektoniska aspekter och affärsaspekter. Dessa utgör grunden för identifiering av de skillnader som råder mellan förvaltning av traditionella mjukvarusystem och förvaltning av Web-servicesystem. Skillnaderna gäller först och främst olika förvaltningsaktiviteter såsom förändringshanering, problemundersökning, verkansanalys, grundorsaksanalys, testning och releasehantering. Även aspekter rörande organisation och rollfördelning beaktas. För att kunna hantera Web services på ett effektivt sätt måste nya roller införas och organisationer måste omdesignas.

Därefter listar vi de problem som man kan möta vid förvaltning av Web-services. 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 Web-services.

Slutligen beskriver ett ramverk för hantering av Web-services och Web-servicesystem. Ramverket föreslår förändringar av förvaltningsprocesser, roller involverade i processerna och anpassningar på organisationsnivå.

Innehåll

1 Introduktion

1.1 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 Web-services. 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.

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

Web-services innebär stora potentiella fördelar för företagen såsom billigare och mindre komplex integration med arvet (legacysystem) 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 Web-services idag så kommer man att bli betydligt bättre förberedd för att möta konkurrensen imorgon. Företag som inte satsar på Web-serrvices inom den närmaste framtiden kommer förmodligen att missgynnas konkurrensmässigt.

1.2 Problem

Många Web-services 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 organisationer, förvaltningsprocesser, produkter som förvaltas, roller involverade i processerna.

Idag har vi tyvärr inte tillräcklig kunskap om hur man ska förvalta Web-servicesystem. 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.

1.3 SERVIAMS Bidrag

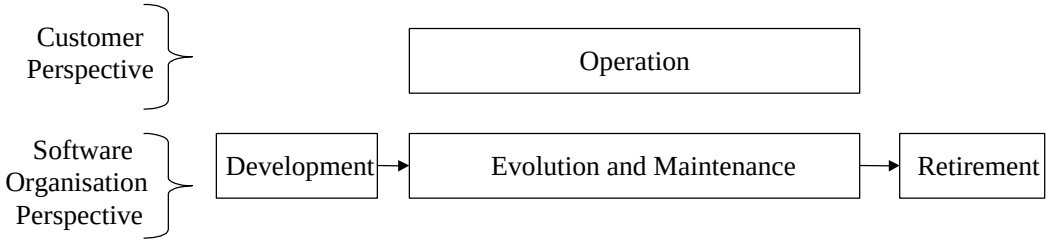
FÖRVALTNING är ett delprojekt inom SERVIAM. Dess huvuduppgift är att studera vidareutveckling och underhåll av webtjänster. Delprojektet leds av Mira Kajko-Mattsson vid Stockholms Universitet och KTH. Industripartner är SAS. Delprojektet har resulterat i ett ramverk för förvaltning av Web-services, och det är detta ramverket som presenteras i denna handbok. Handboken är indelad i tre avsnitt. Det första avsnitet introducerar området *Förvaltning.* Det andra avsnittet ger översikt av Web-servicesystem. Det tredje avsnitet beskriver Serviam-ramverket.

2 Traditionell förvaltning

I detta avsnit presenterar vi kort traditionell förvaltning. Med hänsyn till omfångsbegränsningen för denna rapport, kan vi inte beskriva hela området. Vi beskriver bara de viktigaste aspekterna inom förvaltning.

2.1 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:s definition. 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 använvändbart, avvecklar man det och ersätter det förmodligen med ett nytt och tjänligt system.



Figur 1. Huvudfaser i mjukvarulivscyckeln

2.2 Omfattning

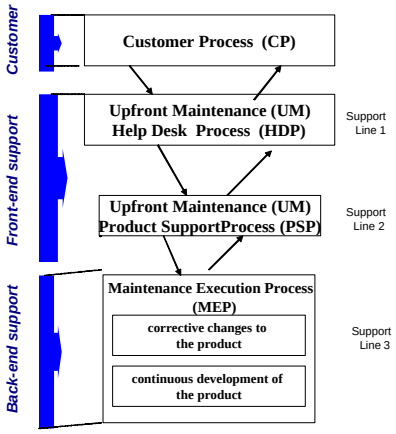
Man kan se på förvaltning ur olika perspektiv: livscykel-, process-, produkt- och organisaitonsperspektiv. Livscyckelperspektivet har redan skildrats i föregående delkapitlel och i figur 1. Ur processperspektiv omfattar förvaltning olika typer av aktiviteter. Som framgår av tabell 1 kan dessa klassificeras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

Perfektivt underhåll definieras som underhåll vilket utförs för att förbättra prestanda, underhållbarhet 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.

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.

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



Figur 2. Supportnivåer

Adamptivt underhåll utförs i syfte att anpassa system till den förändrade miljön. Exempel på den förändrade miljön kan vara operativsystem, mjukvaru- och hårdvaruplattformar, och klimat. För att systemet ska kunna fungera anpassar man den till den förändrade miljön utan att ändra på dess grundläggande funktionalitet.

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.

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.

3 Förvaltning av Web-Services

I det här avsnittet ges en kort introduktion till Web-services. Avsnitt 3.1 ger en översikt av Web-servicesystem. För att kunna föreslå ett ramverk för förvaltning av Web-services har vi identifierat skillnader mellan traditionella och Web-servicesystem. Dessa beskrivs i avsnitt 3.2. Slutligen i avsnitt 3.3, har vi listat problem som man kan stöta på när man förvaltar Web-services.

3.1 Överblick av Web-servicesystem

Web-services 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 Web-services som kommunicerar med varandra på det här sättet bildar ett Web-servicesystem i en tjänsteorienterad arkitektur. Web-services 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.

Web-services 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.

Deras 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. Web-services gynnar återanvändning av affärsfunktionalitet genom återanvändning av redan utvecklade och implementerade tjänster och system. Återanvändning av komponenter accelererar utveckling och vidareutveckling 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 Web-services 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. Web-services 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.

3.2 Faktorer unika för förvaltning av webtjänster

Skillnaderna mellan förvaltning av Web-servicesystem 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 Web-services 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 både kända och okända 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

Web-services har introducerat en ny affärmodell för använding, 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.

Web-services gynnar affärssamarbete. Bred och nära extern integrering med kunder, försäljare, partners och andra parter har blivit en verklighet. Med hjälp av Web-services 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/använding 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 sannolikhet för sidoeffekter. Brist på denna förståelse är en av grundorsakerna till defekter som påverkar produkternas korrekthet.

Idag möter systemförvaltarna många svårigheter 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 dem. 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 Web-services kommer att kräva ännu mer resurser. Det beror på att Web-services 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 Web-services. Web-servicearkitekturer 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 Web-services har detta drivits till sin spets. Vi har många fler komponenter, fler programmeringsspråk, fler plattformar, och annat.

Organisatoriskt perspektiv

Web-services 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 Web-servicesystem 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 Web-servicesystem, normalt mer internt fokuserade. De använder sig inte av så många externa komponenter i sina arkitekturer som Web-serviceapplikationer. Således är deras förvaltning mer intern och kan till största delen utföras av en enda organisation. Med hjälp av Web-services 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 Web-servicesystem 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 Web-services till stora delar ligger utom räckhåll 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 det faktume 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 Web-services. En annan skillnad är den stora heterogenitet av Web-serviceimplementationer (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 över 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 Web-servicesystem 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 uttrycks 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 Web-service sammanhang både använder vi själva och tillhandahåller tjänster till andra parter. En sådan situation är inte ny, men den kan ledas till sin spets med Web-servicesystem. 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 Web-servicearkitekturer 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 över 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. Web-servicearkitekturer har introducerat en flerkundsmodell för användning och delning av komponenter. Web-services kan användas av

många kunder samtidigt oavsett om de finns inom en och samma organisation eller ej. Dessutom, vilket nämnts tidigare, är Web-services 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 Web-services påverkar releasehanteringsfasen. För att kunna möta kundens behov resulterar många förändringar i ett stort antal versioner av specifika tjänster. Det orsakas av faktummet 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

Web-service applikationer 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 testning av tradtionella system. Dessutom har Web-services 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 plattformar), (2) tillgång på mjukvarukoden och (3) utvecklingsbaret.

3.3 Problem vid förvaltning av webtjänster

De store problem som man möter vid förvaltning av Web-services 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 Web-servicesapplikationer. 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 Web-services är emellertid avsevärt större.

Därför kräver några av dem ett omtag för att kunna hanteras på ett effektivt sätt. Beträffande de nya problemen så är dessa relaterade till Web-servicekedjor, 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 (implementeringsdetaljer 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.

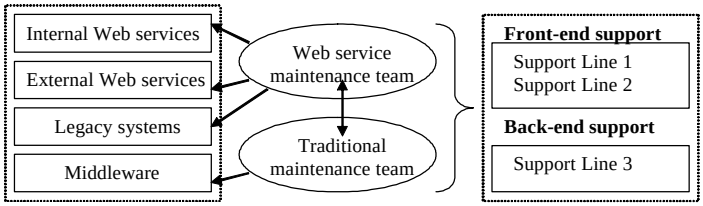
- **Begränsad insyn i förvaltningsprocessen**: Förvaltning ligger utanför en viss organisations kontroll. Alla förvaltningsaktiviteter (tex problemundersökning, förändringshantering, sidoeffektanalys, testning och liknande) inom processen är starkt beroende av samarbete med andra externa förvaltningsorganisationer,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

4 Ramverk för förvaltning av Web-Services

De egenskaper som är unika för förvaltning av Web-services utgör grunden för skapandet av SERVIAM-ramverket. Ramverket består av ett antal förslag till förändringar rörande mjukvaruorganisationer, roller och processer. Dessa presenteras i delkapitel 4.1 – 4.3.

4.1 Organisationella Förändringar

Vi lämnar två förslag till att hantera Web-services: (1) skapande av helt nya organisationer, så kallade Servicecentra och (2) förändringar av existerande organisationer. Vi tar itu med den höga distributionen av Web-servicesystem genom att föreslå primära Web-serviceentreprenörer (primary contractors).



Figur 3. Mjukvaruorganisationens struktur

ServiceCentra

Web-services kommer att leda till, och har redan lett till, skapandet av helt nya organisationer, så kallade ServiceCentra. Dessa organisationer fokuserar huvudsakligen på att skapa och förvalta Web-servicesystem. De undersöker innehållet i Web-servicekataloger och använder sig av dess innehåll för att skapa nya Web-services som de sedan gör publika på nätet.

För traditionella mjukvaruorganisationer kan förvaltning av Web-services vara mycket kostsamt. De måste genomföra nödvändiga omorganiseringar, hålla sig med en stab av högt utbildade ingenjörer och köpa nödvändig mjukvara och utrustning. Ett sätt att minska på kostnaderna är att outsourca förvaltning av Web-services till ServiceCentra. Genom att vara högt specialiserade mot att förvalta Web-servicearkitekturer, kan ServiceCentra leverera högkvalitativa förvaltningstjänster till dessa organisationer.

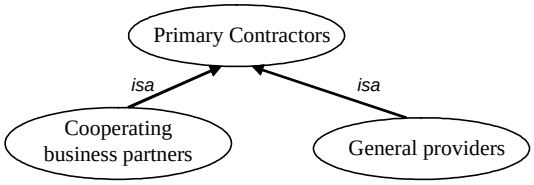
Omorganiseringar inom en Mjukvaruorganisation

För att effektivt kunna förvalta Web-services inom en organisation, föreslår vi inrättandet av ett specialdedikerat Web-serviceteam. Som framgår av figur 3, ansvarar detta team för förvaltning av både interna och externa Web-services. Vårt förslag täcker supportnivåerna 2-3.

Primära Entreprenörer (Primary Contractors)

Begreppet ”service supply chains” innebär att Web-servicesystem är distribuerade över många olika parter. Exempelvis kan externa högnivå-services bestå av många subservices, vilka kan komma från olika organisationer. För att effektivt kunna hantera denna höga grad av distribution, borde mjukvaruorganisationerna fokusera på hantering av sina interna Web-services och outsourca externa högnivå-services till primära entreprenörer (primary contractors).

Det är viktigt att man begränsar samarbetet till de primära entreprenörerna. Det skulle bli för dyrt att hantera alla entreprenörer och subentreprenörer. Det är inte effektivt och ofta inte möjligt, att hantera ett för stort antal entreprenörer.



Figur 4. Klassificering av entreprenörer

Delegering av ansvar (outsourcing) till primära entreprenörer bidrar till en mindre komplex hantering av Web-services, och därmed till förbättring av förvaltningsprocesserna. Det beror på att organisationerna bara samarbetar med ett begränsat antal entreprenörer och inte med en myriad av parter. Därför är det mer eller mindre obligatorisk att begränsa antalet entreprenörer.

Som framgår av figur 4, skiljer vi på två kategorier av primära entreprenörer. Huvudskillnaden ligger i tillit och typ av samarbete. Den första kategorin omfattar samarbetande affärspartners. Denna kategori härstammar från business-to-businessdomänen (B2B), och är därför starkt fokuserad på affärsintegration och nära samarbete mellan entreprenörer och organisationer.

Den andra kategorin omfattar generella leverantörer (providers) som erbjuder Web-services till många olika, ofta för dem okända, parter. Denna klass av entreprenörer härstammar från Web-service businessmodellen som presenterades i delkapitlet 3.2. När det gäller de generella leverantörerna, begränsas huvudsakligen affärssamarbete tlll ”sälja-köpa kontrakt”. Sålunda finns det ingen tät affärsintegration eller ömsesidigt samarbete som i det första fallet.

4.2 Rollförändringar

I Web-servicesammanhang skiljer vi på tre grupper av förvaltningsroller:

- **Rollgrupp 1:** Roller ansvariga för att skapa, vidareutveckla och underhålla de Web-services som utgör gränsnitt mot organisationens system, till exempel mot *legacysystem*. Dessa roller måste ha djupa kunskaper både i de traditionella systemen och Web-serviceteknologin.
- **Rollgrupp 2:** Roller ansvariga för att skapa, vidareutveckla och underhålla Web-services och affärsprocesser. Dessa roller måste ha djupa kunskaper i organisationens affärsprocesser och Web-serviceteknologin.
- **Rollgrupp 3:** Roller ansvariga för att supporta Web-servicesystem på front-end supportnivåerna, först och främst på Support Line 2-nivån. Dessa roller behöver inte ha djupa tekniska kunskaper. Däremot måste de ha kännedom om affärsprocesser och deras underliggande struktur, så att de kan supporta sina kunder i deras dagliga verksamhet.

I många fall finns det inte någon skarp skiljelinje mellan de två första grupperna. Deras fastställade utnämning (designation) kan skilja sig mellan organisationer. Vårt förslag till enskilda roller är baserat på den nya ansatsen för att förstå Web-servicesystem.

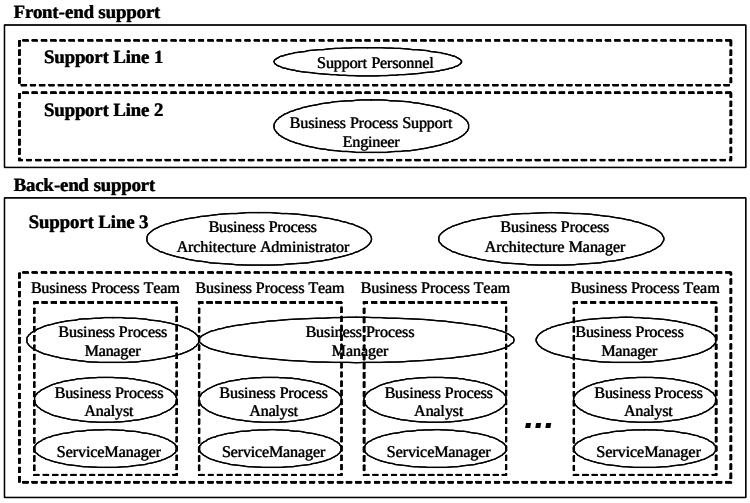
Förståelsen av Web-servicesystem skiljer sig markant åt från förståelsen av traditionella system. På grund av den dolda arkitekturen och implementationen bakom ”service supply chains”, måste förvaltarna byta fokus från att förstå implementationen till att förstå målet med enskilda Web-services (via WSDL kontrakt) deras roll och samarbete (?) med andra Web-services inom en sammanslagen affärsprocess.

Den nya ansatsen att förstå mjukvara härstammar från SOA-konceptet, där man fokuserar på affärsprocesser, aggregering av Web-services och affärsprocesser i punkt-till-punkt (end-to-end) affärsprocessflöden. En affärsprocess utför en särskild affärsarbetsuppgift. Tillsammans med andra affärsprocesser bildar de högnivåaffärsprocesser. Några av deras beståndsdelar, sådana som

Web-services eller affärsprocesser på lägre nivå, kan vara beståndsdelar i flera andra affärsprocesser.

Ur arkitektoniskt perspektiv, kan komplexiteten av affärsprocesser variera stort från små affärsprocesser implementerade med bara några få Web-services till extremt komplexa processer bestående av ett stort antal Web-services, affärsprocesser och även hela applikationer.

Affärsprocesser och processmodellering är nyckelelement i systemdesign. Därför utgör de en grund för definintion av SERVIAM-roller. Som framgår av nedre delen av figur 5, föreslår vi roller för rollgrupperna 1 och 2. Dessa är *Business Process Architecture Administrator, Business Process Architecture Manager, Business Process Manager, Business Process Analysts,* och *Service Manager*. Dessa roller bildar en *Business Process Maintenance Team* som i de flesta organisationer är verksam på Supportnivå 3.



Figur 5. Serviam roller

Vi får dock inte glömma att man förvaltar system på alla tre supportnivåer, dvs 1-3. Av denna anledning, borde vi identifiera roller för den återsående rollgruppen (rollgrupp 3) på supportnivåerna 1-2. På den första supportnivån, Help Desk, besitter supportpersonalen ganska ytliga och generella kunskaper om de supportade mjukvarusystemen. Deras roll begränsas huvudsakligen till att bekräfta att det inrapporterade problemet är ett problem och eskalera det vidare till supportnivå 2. Av denna orsak föreslår vi inte några specifika roller för supportnivå 1. Här supportar samma personal alla typer av system, både traditionella sådana och Web-service system.

När det gäller den andra supportnivån besitter dess personal mycket djupare och bredare system- och affärskunskaper. Deras uppgift är att bekräfta inrapporterade problem i Web-servicesystem, och eskalera dem till back-end supportnvån (se figur 2). Därför dedikerar vi en roll på supportnivå 2 för denna uppgift. Vi kallar den *Business Process Support Engineer.*

Vårt förslag till roller bör uppfattas som en betecknande av ansvar som är nödvändig för att hantera Web-services. Våra roller behöver inte jämställas med individuella personer. I vissa förvaltningsorganisationer, speciellt de små, skulle alla dessa roller kunna utföras av en och samma individ. I stora organisationer, å andra sidan, skulle en roll kunna utföras av flera individer. Nedan beskriver vi *Process Maintenance Team* och dess medlemsroller.

Business Process Maintenance Teams

För att kunna maximera produktivitet och samstämmighet med SOA-konceptet rekommenderar vi att team ska anvara för förvaltning av affärsprocesser. Vi kallar dem *Business Process Maintenance*

Team. Som framgår av figur 5, ansvarar ett team för en eller flera affärsprocesser, och följdaktigen för alla Web-services som tillhör dessa processer och deras inbördes (?) beroenden. Antalet tilldelade processer till ett team beror på processernas storlek, komplexitet, och deras beroenden. Denna tilldelning är relativ och kan variera mellan olika företag, projekt, arkitektoniska lösningar och annat.

Skapande av affärsprocessteam underlättar förvaltning av Web-services. Web-services som tillhör samma affärsprocess kan samtidigt utvecklas, testas och integreras. Därför kan mängden av koordinering och planering mellan flera affärsprocessteams minskas avsevärt. Det flesta koordinerings- och planeringsuppgifter kan hanteras inom ett team.

Fortfarande kommer det dock att krävas någon form av koordinering och planering över flera team. Återanvändningsmöjligheter av Web-services inom fler än en affärsprocess gör att teamen borde spåra och hantera beroenden till andra affärsprocesser. Detta är viktigt eftersom förändringar inom en affärsprocess kan påverka andra teams processer. Om en viss komponent används i flera processer, föreslår vi att den förvaltas av det team som hanterar (eller kommer att hantera) det största antalet förändringar i den komponenten.

Service Manager

Sevice Manager ansvarar för en viss uppsättning av Web-services som tillhör en viss affärsprocess. Hennes expertis ligger på implementationsinvå Web-för interna Web-services och på gränsnittsnivå för externa Web--ervices. Hennes huvudansvar är att göra alla typer av förändringar i Web-serviceenheter. Dessa förändringar begärs av *Business Process Analyst* eller *Business Process Manager*.

Business Process Analyst

Business Process Analyst ansvarar för de affärsprocesser som tilldelas hennes team. Denna roll samarbetar tätt med *Business Process Managern* och *Service Managern*. Hon ansvarar för att (1) förse alla roller inom teamet med de nödvändiga kunskaperna om affärsprocessarkitekturen, (2) undersöka inrapporterade problem, tilldela problemrapporter till lämplig *Service Manager*, analysera föreslagna förändringar och kontrollera att de har åtgärdats på ett korrekt sätt, (3) delta i design av problemlösningar, (4) bistå *Business Process Managern* med att ta viktiga beslut och (5) hålla reda på alla beroenden till andra teams affärsprocesser.

Business Process Manager

Business Process Managern ansvarar för en eller flera affärsprocessteam. Hon ansvarar för leverans och kvalitetssäkring av hennes teams affärsprocesser. Hon bistår *Business Process Architecture Managern* och andra roller i att designa den globala affärsprocessen. Hon tar många viktiga beslut angående förändringar i affärsprocessarkitekturen och dess beståndsdelar.

Business Process Architecture Manager

Business Process Architecture Managern besitter hög expertis på den globala affärsprocessarkitekturen, dess beståndsdelar och alla dess beroenden. Hon stöds av *Business Process Architecture Administratorn* som kontinuerligt förser henne med information om status på den globala affärsprocessen.

Business Process Architecture Managern ansvarar för hantering av den globala affärsprocessen, till exempel, (1) för att designa dess arkitektur tillsammans med andra roller, sådana som *Business Process Architecture Administratorn* och *Business Process Managers*, och (2) för att besluta om större arkitektoniska förändringar i den globala affärsprocessen.

Business Process Architecture Administrator

Business Process Architecture Administratorn utför olika typer av administrativa uppgifter. I traditionell systemförvaltning kan den rollen automatiseras. På grund av den höga återanvändnings- och förändringsgraden av Web-services tror vi att den rollen är vital åtminstone under införandefasen av Web-serviceteknologin. Hennes uppgift är att hålla och sprida akutell statusinformation om den globala affärsprocessen och dess beståndsdelar.

Business Process Architecture Administratorn assisterar alla roller i många olika och varierande uppgifter. För vissa problemrapporter utgör hon den första kontaktpunkten för *Business Process Support Personal* på supportnivå 2. Hon tar emot de problemrapporter från supportnivå 2 som är svåra att eskalera till ett särskilt team, analyserar dem, och tilldelar dem till lämpliga affärsprocessteam. Tillsammans med *Business Process Managers*, bistår hon *Business Process Architecture Managern* med att ta viktiga beslut genom att leverera statussinformation om affärsprocesser. Besluten gäller den globala affärsprocessen och förändringar i den. Slutligen administrerar och leder denna roll alla CCB möten under vilka viktiga beslut om förändringar tas.

4.3 Processförändringar

Införande av Web-servicesystem kommer att påverka de flesta förvaltningsprocesser. Förvaltningsprocesserna kommer att omspänna flera organisationer. Därför måsta man granska alla dess beståsprocesser för att kunna anpassa dem för att hantera den höga distributionsgraden. Denna typ av arbete är resurskrävande och det kan ta tid innan man kan komma med lämpliga förslag till processförändringar. Inom projektet hinner vi bara komma med två generella förslag. Dessa är (1) processer för hantering av *Service Level Agreements* (SLA) och (2) konfigurationshantering. Dessa två processer utör en grund för förändringar inom alla resterande förvaltningsprocesser.

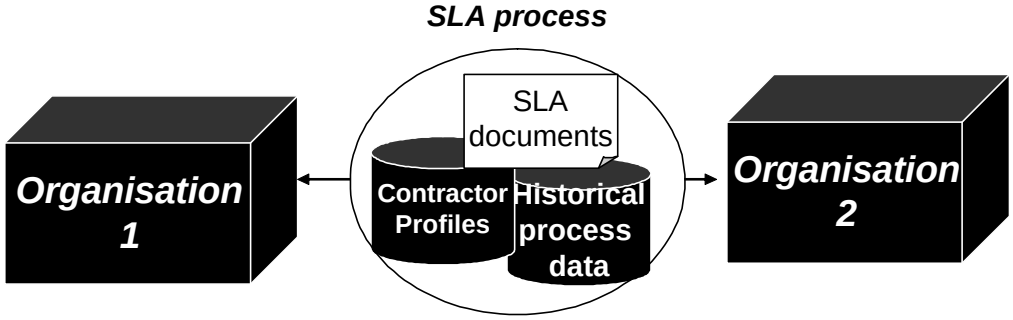


Figure 6. Hantering Web-service systems

4.3.1 Service Level Agreements

En Service Level Agreement (SLA) är ett skrivet kontrakt mellan samarbetande parter. Det anger vad en tjänstekonsument har rätt att förvänta sig av en tjänsteleverantör. Det är ett dokument som identifierar huvudmål och ansvar för involverade parter. Dess innehåll granskas kontinuerligt av en SLA-process, en process som utförs gemensamt av de samarbetande parterna (se figur 6). Under denna process, registrerar respektive partner informantion om sin egen och sina partners effektivitet, utreder om tjänsterna levereras enligt gällande SLA-avtal, justerar och förhandlar om ny ansvarsfördelning. Detta gör man för att möta förändrade behov.

I Web-service sammanhang skiljer vi på två typer av SLA: elektroniska och statiska. De elektroniska SLAs används för automatisering av förhandlingar mellan parter som genomförs av

mjukvara vid runtime, till exempel när man upptäcker och köper Web-services. I vårt ramverk beaktar vi inte denna typ av SLA. Vi är mer intresserade av statiska SLAn, de som används mellan tätt samarbetande organisationer.

SLA-domänen är ett starkt försummat område inom programvaruteknik. För närvarande finns det inga generella SLA-processmodeller. Införandet av Web-services har emellertid illustrerat vikten av frågan. Behovet av SLA processer har skapats av det faktum att Web-Services ligger utanför gränsen för en organisation. Organisationer är starkt beroende av ett tätt och omfattande samarbete med externa parter. Den redan nämnda service supply chain och businessmodellen resulterar i en mängd frågor som borde täckas explicit i SLA-dokumentet. Kvaliten på förvaltningsprocessesn är starkt beroende av SLA-dokumentet och SLA-processen.

SLA-Document

Samarbetande parter måste först och främst enas om det specifika innehålet och de initiala mål som ska införlivas i ett SLA-dokument. Innehållet varierar med typen av SLA. Emellertid finns det element som är gemensamma för de flesta dokument. De är ganska många och av utrymmesskäl kan vi inte specificera dem alla i den här handboken. De som är intresserade av att studera dem är välkomna att läsa [1]. Nedan föreslår vi bara de element som vi tror är specifika för Web-service system. Dessa är följande: .

- *Gemensamt definierad och exekverad förvaltningsprocess:* De förvaltningsprocesser som påverkas av Web-serviceteknologin borde definieras i termer av roller, aktiviteter, tidsramar, synkronisering, informations- och datautbyte, gemensamma verktyg, och liknande. Det är viktigt att man kommer överens om hur man ska utföra gemensamma förvaltningsprocesser.
- *Förvaltningsansvar:* Explicit angivelse av vem och under vilka omständigheter äger och ansvarar för vilka förvaltningsaktiviteter och deras omfång inom den gemensamma processen.
- *Leverabler:* Specifikation av all dokumentation som ska levereras med Web-services, till exempel design, kod, dokumentation och gränsnitt.
- *Kvalitetsfaktorer:* Specifikation av de kvalitetsattribut som ska implementeras under utveckling och förvaltning av Web-services, och kriteria för att mäta dem. Här ingår även identifiering av tekniska kvalitetsfaktorer såsom nätverksprestanda, uptime availability, säkerhet, packetförluster under en given tidsperiod och liknande.
- *Web-servicekritikalitet:* Specifikation av all funktionalitet som är kritisk och som behöver extra uppmärksamhet och hantering.

SLA-Process

Det räcker inte med att bara skapa SLA-dokument. För att förbättra det långsiktigaa samarbetet, borde organisationerna definiera en gemensam SLA-process i vilken man följer, utvärderar och förbättrar samarbetsprocessen.

En viktig förutsättning för en lyckad SLA-process är en mogen förvaltningsprocess som ger tillräckligt med insyn i samarbetsprocessen för att registrera historiska fakta om processens förlopp (se figur 6). Organisationerna borde också skapa entreprenörsprofiler i vilka man samlar all information om sina samarbetsparnters och uppfyllnaden av deras åtaganden som de definierats i SLA-dokumentet och som registrerats i historiska processposter. Allt detta borde hjälpa organisationerna att säkra att förvaltningstjänsten levereras enligt de gemensamma SLA-avtalen och ge feedback för ansvarsjusteringar och omförhandlingar.

Genom att hålla detaljerade register om förvaltningsprocesserna, deras steg och erfarenhet har organisationerna tillräckligt med underlag för att kunna utvärdera samarbetsorganisationer, deras beteende och ansvarsuppfyllande. Det borde i sin tur hjälpa dem att ställa villkor för fortsatt

samarbete eller ta beslut om man ska fortsätta att samarbeta med organisationen eller om man ska välja en annan organisation.

4.3.2 Configuration Management

Web-servicesystem kan bestå av ett stort antal samarbetande Web-services som kan komma från olika källor. Varje Web-service kan bestå av, eller ingå i, många olika komponenter. Web-servicearkitekturen kan ses som ett nätverk av komponenter med många ömsesidiga relationer, som dessutom kan förändras mycket snabbt, snabbare än traditionella systemarkitekturer. Både Web-services och deras beroenden kan ändras ofta. Vid en viss tidpunkt kan det finnas många versioner av en viss Web-service.

Alla dessa arkitektoniska faktorer tyder på att det är mycket viktigt att ha en precis och korrekt konfigurationsinformation, dvs information om Web-services, deras beroenden, versioner, varianter, och liknande. Det är också viktigt att ha en bra konfigurationshanteringsprocess. Därför föreslår vi att förvaltningsorganisationerna lägger tillräckligt med resurser på konfigurationshantering av Web-services.

Det är svårt att föreslå en lämplig granularitetsnivå för Web-servicesystem. Systemkonfigurationen borde brytas ner och identifereas unikt på den nivå som möjliggör för organisationerna att effektivt kontrollera, lagra, rapportera och versionshantera konfigurationskomponenterna. En viktig del i konfigurationshantering är att bestämma rätt konfigurationstruktur och nivå, med högnivåkomponenter nedbrutna till lägrenivåkomponenter och så vidare. Konfigurationsstrukturen borde beskriva förhållanden, versioner och position för varje konfiguratonskomponent.

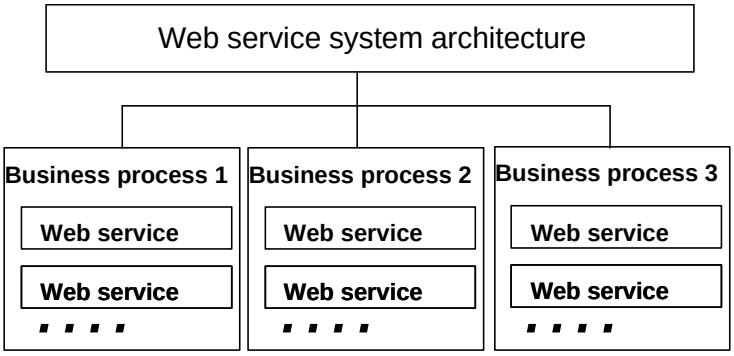


Figure 7. Nedbrytning av konfigurationsstrukturen

Som framgår av figur 7, föreslår vi att man organiserar konfigurationsstukturen kring affärsprocesser. Affärsprocesser består av en samling Web-services. Därför borde de grundläggande konfiguratinselementen (CI) vara Web-services. Konfigurationshantering borde hantera data om alla Web-services som implementerar en viss affärsprocess. Dessutom borde den identifiera all information om versioner, varianter och dess beroenden till andra Web-services och/eller affärsprocesser.

Som tidigare nämnts är det viktiga med konfigurationshantering att bestämma rätt konfigurationsgranularitet. I Web-servicesammanhang är det särskilt viktigt. Det beror på det faktum att Web-servicegranulariteten kan skilja sig markant. Vissa Web-services kan vara mycket små och bestå av bara två rader kod. Andra Web-services kan vara komplexa applikationer bestående av hundratals andra Web-services. Genom att välja rätt granularitetsnivå kan man spåra förändringar i dem och därmed bättre uppskatta de totala resurser som krävs för att förvalta Web-services..

5 Epilog

Web-servicesystem adderar ytterligare complexitet till processerna för underhåll och utveckling av mjukvara. För att hantera denna komplexitet har vi föreslagit ett generellt ramverk för utveckling och underhåll av Web-servicesystem. Vårt ramverk omfattar organisations- roll- och processförändringar för att stödja Web-serviceprodukter. Förändringarna har summerats i tabell 2. Tabellen utgör ett verktyg som ska stödja organisationerna vid implementeringen dessa förändringar.

Tabell 2. SERVIAM-ramverkets verktyg

Organisationella förändringar
• Skapa ett team som ansvarar för hantering av Web-servicesystem. • Den höga distributionen av Web-services hanteras av primära Web-service entreprenerör.
Rollförändringar
• **På supportnivå 2**: Skapa en roll som ansvarar för support av Web-servicesystem. • **På supportnivå 3**: o Skapa roller som ansvarar för hantering av förändringar i Web-servicesystem. o Skapa en roll som ansvarar för den globala processarkitekturen.
Processförändringar
• Implementera en process för kontinuerlig utvärdering av samarbetet mellan samarbetande organisationer. Processen heter SLA processen. • Definiera gränsnitt mot samarbetande organisationer och definiera den gemensamma processen, roller och ägarskap.

Vårt ramverkhar presenterats för tio organisationer i Polen och Sverige. Organisationerna i Sverige var Volvo IT, Personec AB (ingår i TietoEnator), Scandinavian Airline Systems (SAS), Skandinaviska Enskilda Banken (SEB), Centrala Studiestödsnämnden (CSN), AMF Pension, Stockholms Läns Landsting and Riksskatteverket (RSV). Organisationerna i Polen var Prokom Software SA och ATENA Uslugi Informatyczne i Finansowe Sp. z o.o (ATENA).

Flertalet företag uttryckte åsikten att ramverkets förslag till förändringar var sunda, innovativa och nödvändiga för att optimera processerna för utveckling och underhåll av Web-services. Organisationerna hävdade dock att de skulle bli kostsamma att implementera.

På basis av våra intervjuades åsikter drar vi slutsatsen att ramverket är värt att pröva. När man gör så får man inte glömma att göra en kostnads-/intäktsanalys för att utvärdera dess effektivitet. Vi inbjuder härmed hjärtligt svenska mjukvaruorganisationer att experimentera med ramverket I en industriell miljö.

Referencer

Citerade referenser:

[1] Kajko-Mattsson M, Ahnlund C, Lundberg E, CM^3: Service Level Agreements, In Proceedings, IEEE International Conference on Software Maintenance, IEEE Computer Society Press: Los Alamitos, CA, 2004, pp. 432-436.

Delprojektets resultat

[2] Kajko-Mattsson M, Evolution and Maintenance of Web Service Applications, In Proceedings, IEEE International Conference on Software Maintenance, IEEE Computer Society Press: Los Alamitos, CA, 2004, pp. 492-493.

[3] Kajko-Mattsson M, Tepczynski M, Future of Evolution and Maintenance in the World of Integrated Web Service Systems, In Proceedings, the Seventh International Conference on Integrated Design and Process Technology (IDPT), Society for Design and Process Science, 2005, no page numbering, proceedings on CD.

[4] Kajko-Mattsson M, Winther P, Vang w, Petersen A, Eliciting a Model of Emergency Corrective Maintenance at SAS, in Proceedings, the International MultiConference in Computer Science & Computer Engineering (SERP 2005), 2005.

[5] Kajko-Mattsson M, Winther P, Vang w, Petersen A, An Outline of CM^3: Emergency Problem Management, in Proceedings**,** IEEE, 31st EUROMICRO CONFERENCE on Software Engineering and Advanced Applications (SEAA)**,** Software Process and Product Improvement (SPPI) track, 2005.

[6] Kajko-Mattsson M, Tepczynski M, A Framework for Evolution and Maintenance of Web Service Applications, In Proceedings, IEEE International Conference on Software Maintenance, IEEE Computer Society Press: Los Alamitos, CA, 2005.

[7] Kajko-Mattsson M, Meyer P, *Evaluating the Acceptor Side of EM^3: Release Management at SAS*, In Proceedings, IEEE 4th International Symposium on Empirical Software Engineering, IEEE Computer Society Press: Los Alamitos, CA, 2005.

[8] Kajko-Mattsson M, Tepczynski M, *SERVIAM Maintenance Framework*, In Proceedings, IEEE International The 9th World Multi-Conference on Hawaii, IEEE Computer Society Press: Los Alamitos, CA, 2006.

[9 Kajko-Mattsson M, Evaluating CM^3: Upfront Maintenance at SAS, in Proceedings, the 9th World Multiconference on Systemics, Cybernetics and Informatics (WMSCI 2005), 2005.

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

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

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

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