Mönsterkatalog

2006-01-23

Editor: Jesper Holgersson

Bidrag av: Eva Söderström, Milena Haykowska, Jelena Zdravkovic,
Paul Johannesson, Benkt Wangler

SERVIAM-PAT-01

Draftversion 1.0

50 pages

Innehållsförteckning

1 Inledning

Denna förteckning utgör en katalog av generiska lösningar, s.k. mönster, för tjänstebaserad utveckling och tjänstebaserade arkitekturer. Mönsterkatalogen är uppdelad i två delar: scenariebeskrivningar och mönster.

Scenariebeskrivningarna fokuserar på att beskriva olika domäner som på olika sätt tillämpar SOA. Dessutom fungerar scenariebeskrivningarna som pekare till ett antal mönster som utnyttjas i respektive scenario, vilket är tänkt som exempel på hur mönsterkatalogen kan användas.

Mönstren som redovisas visar på möjliga lösningar på vanligt förekommande problem som kan uppstå när ett företag vill tillämpa SOA. Lösningarna som erbjuds är i regel på en relativt hög nivå, vilket innebär att implementationsspecifika detaljer, i de fall där detta är tillämpligt, ges som referenser till aktuell specifikation etc. Mönster i Serviam beskrivs med hjälp av en mall som innehåller följande element.

Namn och källa	Anger namn på aktuellt mönster samt varifrån mönstret är hämtat, alternativt var information hämtats som använts i utveckling av mönstret.
Även känt som	Indikerar om det finns något ytterligare namn på mönstret som används i andra sammanhang, inklusive referenser till vart detta mönster beskrivs.
Typ	Innefattar en klassificering av mönstret som indikerar vilken kategori och eventuell underkategori aktuellt mönster tillhör.
Syfte	En kort beskrivning av avsikten med aktuellt mönster
Krafter	Redogör för faktorer som påverkar problemet och även lösningen som beskrivs i aktuellt mönster. Lösningen på problemet designas ofta med dessa faktorer i minnet.
Problem	Beskriver det problem som aktuellt mönster erbjuder en lösning på.
Lösning	Besvarar det problem som beskrivs i aktuellt mönster samt även hur aktuellt mönster löser problemet. Det är inte ovanligt att lösningen kompletteras med figurer och diagram samt sätts in i ett exempel för att tydligare åskådliggöra lösningen.
Konsekvenser	Innefattar i regel möjliga bieffekter av aktuellt mönster. I första hand beskrivs möjliga negativa effekter eftersom explicita positiva effekter redan beskrivits i lösningen. Det kan även förekomma att implicita positiva effekter beskrivs.
Relaterade mönster	En hänvisning till andra mönster som har en relation till aktuellt mönster. Detta kan exempelvis vara alternativa mönster eller mönster som vanligtvis bör användas tillsammans för att komplettera varandra.

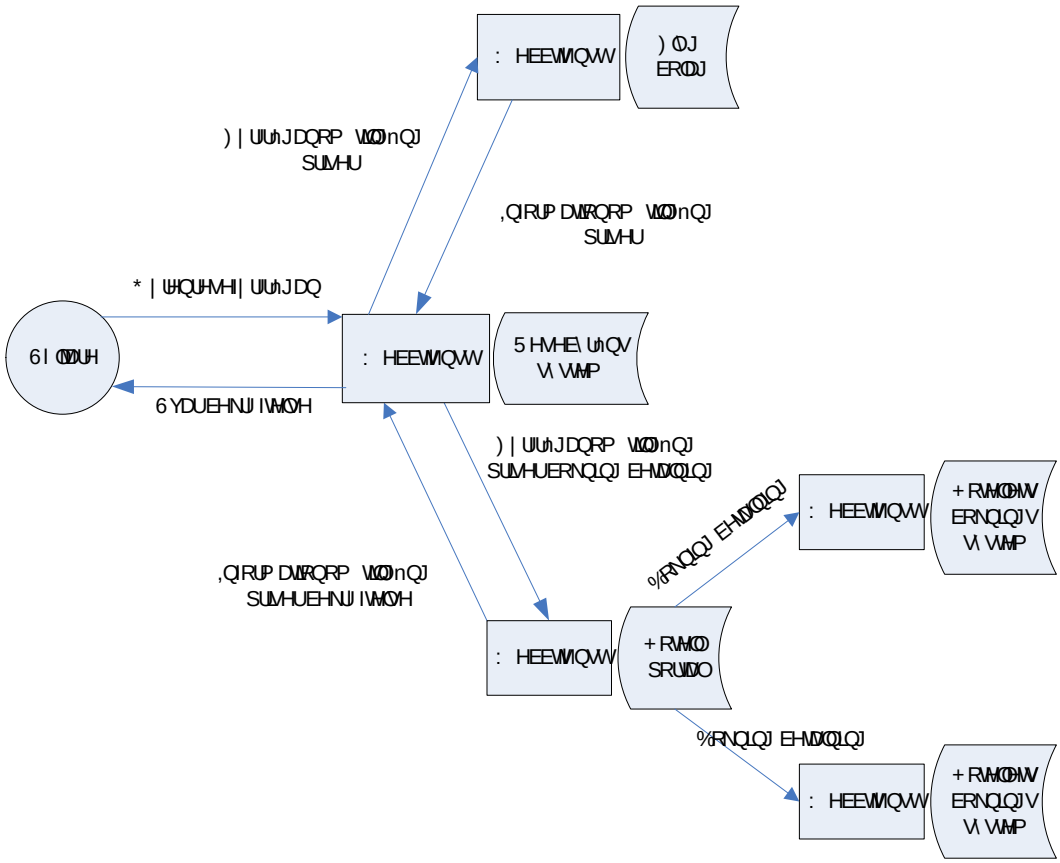
2 Verksamhetsscenarier

Mönsterkatalogen består av många viktiga och användbara mönster för hur man löser olika problem som kan bli aktuella vid utformningen av en tjänsteorienterad arkitektur. Deras generella karaktär kan dock lägga hinder i vägen för en förståelse för hur de kan tillämpas i praktiken. Därför inleds katalogen med ett antal konkreta beskrivningar med syfte att redogöra hur en tjänsteorienterad arkitektur med Web Services kan användas i olika verksamheter. Efter varje beskrivning pekas de aktuella mönstren ut.

2.1 Resebyrå – Sök, boka och betala via Web Services

På en resebyrå går verksamheten främst ut på att söka information om hotell, flyg, resepaket, hyrbilar, evenemang och mycket annat enligt kundernas önskemål. Informationen ska sedan jämföras efter kundens behov med avseende på pris, datum, destination, resplan osv. Det är en uppsjö av information som ska samlas in, jämföras, sållas och presenteras. Dessutom ska det bokas, bekräftas och betalas. En tjänsteorienterad arkitektur för denna verksamhet är en otrolig resursbesparare.

En typisk kundkontakt kan te sig på följande sätt: en kund ringer upp en resebyrå och vill veta vilka flyg som finns till Barcelona ett visst datum, vilka hotell som erbjuds inom den lägre priskategorin samt vad det kostar att hyra bil från och till flygplatsen. Försäljaren knappar in datum och destination och låter systemet söka igenom alla flygbolag som resebyrån har avtal med (eventuellt via en branschorganisation). Sökningen sker via en Web Service som ger åtkomst till flygbolagens interna databaser. Säljaren kommer inom knappt en minut få den aktuella informationen om flyg samt priser som presenteras för kunden. På samma sätt kan personalen söka efter lediga rum på hotell och biluthyrningspriser. Hotellen söks via en hotellportal som i sin tur hämtar information direkt från hotellens bokningssystem. Om kunden hittar något av intresse görs det en bokning via aktuell Web Service hos det flygbolaget och hotellet som passade bäst. Bokningssystemen uppdateras omedelbart och nästa gång det görs en sökning kommer det finnas en flygstol och ett hotellrum mindre att välja på.



Figur 1 - Bokningstjänster

Juridiska aspekter

Resebyråexemplet visar på flera intressanta juridiska aspekter. Vi kan till exempel tala om elektronisk handel och informationssamhällets tjänster. I figur 1 ser vi hur en Web Service hanterar betalning gentemot ett hotells bokningssystem. Ett problem i detta sammanhang är att det inte är helt självklart hur erbjudandet av en Web Service omfattas av lagen om elektronisk handels definitioner. Tolkningarna av lagen är även olika, vilken kan orsaka problem om oenighet uppstår. Resebyrån och hotellet/-en bör därför gemensamt analysera lagen för att bättre kunna fastställa hur samarbetet ska utformas.

En annan aspekt handlar om personlig integritet. När en säljare bokar en resa åt en kund så överförs information om kunden mellan resebyrån och dess parter. Detta ställer krav på att överföringarna är säkra så att inte den personliga informationen missbrukas och den personliga integriteten hotas. Relevanta lagar behöver studeras innan en arkitektur liknande den i vårt exempel skapas, så att problem undviks och kunderna kan känna sig säkra.

Ett tredje och sista exempel handlar om avtal som sluts mellan de parter som samverkar. I vårt exempel i figur 1 ser vi att det handlar om resebyrån, flygbolag och hotell. Ofta saknas juridiskt bindande avtal, och om de finns är de ofta bristfälliga. Ett exempel är att giltighetsperiod saknas. Resebyrån skulle i detta fall kunna agera genom att i sin branschorganisation verka för att ta fram standardavtal för samarbete via Web Services.

Relaterade mönster

Viktiga frågor som bör diskuteras och tas hänsyn till vid utvecklingen av de Web Services som beskrevs ovan är sökbarheten, koordineringen och samordningen.

Observera att mönsterhänvisningarna inte är uttömmande utan pekar ut endast några typiska mönster för scenariot.

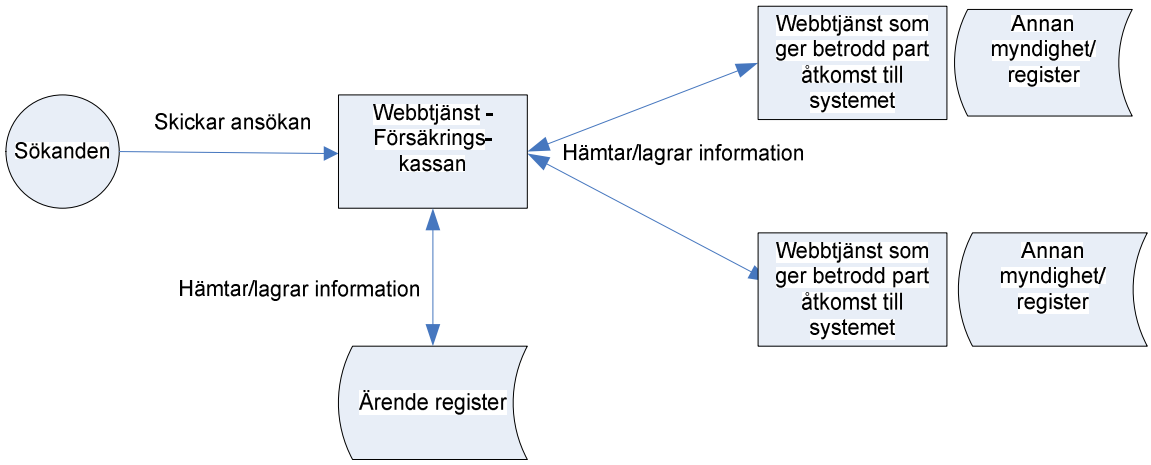
- *Att göra en Web Service sökbar (se avsnitt 5.2)*
  Med tanke på det stora antalet Web Services som måste anropas i sökningen efter flygstolar, priser och hotell vore det önskvärt att den egna applikationen automatiskt kunde identifiera nya Web Services som finns tillgängliga för att göra sökningen ännu mer breddad och effektiv.
- *Att utnyttja koreografi (se avsnitt 5.3)*
  Kommunikationen mellan olika Web Services måste i vissa fall ske i en viss ordning när bokning, alternativt betalning, skall göras hos olika aktörer. Då krävs det att processerna föredefinierats hos berörda parter så att en aktivitet inte utförs förrän den föregående är slutförd. Exempelvis ska den aktivitet som hämtar prisuppgifter inte aktiveras innan en tidigare aktivitet har undersökt att det finns lediga flygstolar.
- *Att utnyttja message broker (se avsnitt 5.5)*
  Ett hotell kan erbjuda flera olika Web Services att hämta information, boka samt att betala. Finns det en mängd Web Services kan det vara lämpligt att ge den anropande applikationen hjälp med att identifiera vilken av alla tillgängliga Web Services som ska användas. På så sätt kan en klientapplikation om med begränsad kunskap om hotellets alla Web Services ändå nå fram med sin förfrågan till den Web Service som är bäst lämpad i sammanhanget.
- *Att uppnå meddelandesäkerhet – med mellanhänder (se avsnitt 5.4)*
  Då det finns en hotellportal via vilken sökningen sker innebär det att när betalningen ska göras går meddelandet inte direkt till hotellets system utan först tar portalen emot anropet för att sedan förmedla anropet vidare. Förbindelsen är alltså inte punkt till punkt utan det finns en mellanhand inblandad. Samtidigt måste det garanteras att endast den part som verkligen skall ha betalningsinformation skall kunna läsa denna. Informationen måste således skyddas då den passerar en mellanhand.

2.2 24-timmarsmyndighet med Web Services

Myndigheter och verk handlägger ärenden där det, i de allra flesta fall, krävs omfattande informationsinhämtning från andra myndigheter, register och andra externa aktörer. En tjänstebaserad arkitektur kan underlätta och effektivisera handläggningen då uppdaterad information finns ständigt tillgänglig och inhämtningen kan automatiseras oavsett systemplattform och datainnehåll.

Ett typiskt fall kan beskrivas på följande sätt: en person vill anmäla sig som sjukskriven samt ansöka om sjukpenning hos försäkringskassan. På försäkringskassans hemsida kan han/hon fylla i en ansökan som sedan via en Web Service skickas direkt in i ärendesystemet och sammankopplas med eventuella tidigare ansökningar eller annan viktig information. Andra uppgifter som behövs för att ärendet ska kunna handläggas, så som läkarintyg och uppgifter om inkomst kan också hämtas via Web Services hos de berörda parterna. Tjänsten innebär dels att den sökande kan initiera ett ärende så fort det är nödvändigt utan att behöva vänta på/hämta pappers blanketter, dels avlastas personalen genom att registreringsprocessen avkortas då all nödvändig information samlas in och registreras automatiskt.

En exponering av systemet hos försäkringskassan gör det också möjligt för andra myndigheter att hämta och uppdatera information direkt i det interna systemet.



Figur 2: Försäkringskassan

Juridiska aspekter

I och med att Försäkringskassan är en offentlig myndighet finns det ett antal särskilda juridiska omständigheter att ta hänsyn till. Bland annat finns en lag om offentlig upphandling, och det är oklart hur offentlighetsprincipen påverkar användningen av Web Services när det gäller filtrering, förvaring och lagring av information hos annan part. Detta behöver utredas innan Web Services skapas och används. I vårt exempel skulle filtrering av information kunna bli aktuellt, till exempel, eftersom andra myndigheter eller register kan ges tillgång till Försäkringskassans system (se figur 2).

Lagen om offentlig upphandling kan också påverka hur Försäkringskassan köper in sina Web Services. Idag är det oklart om statliga myndigheter affärsmässigt kan kräva att alla som vill deltaga i offentlig upphandling ska använda Web Services i sin arkitektur. Det betyder att Försäkringskassan skulle behöva gå igenom lagen med sina jurister innan upphandling sker.

Relaterade mönster

De mönster som är centrala i det beskrivna fallet är de som syftar till att skydda känsliga personuppgifter. Det handlar delvis om rättsliga aspekter och delvis om tekniska – de juridiska kan sägas påverka vilka tekniska lösningar bör implementeras för att garantera skyddet för personlig integritet. Observera att mönsterhänvisningarna inte är uttömmande utan pekar ut endast några typiska mönster för scenariot.

- *Att hantera personlig integritet* (se avsnitt 3.1)
  Förutsättningen för att tjänsten ska fungera är att myndighetsregister exponeras och informationen skickas över Internet. Det skapar problem då informationen som hanteras i beskrivningen ovan är personuppgifter, i vissa fall känsliga personuppgifter så som läkarintyg som beskriver patientens sjukdomstillstånd, eller inkomstuppgifter. Därför måste de lagar som skyddar den personliga integriteten tas hänsyn till vid utvecklingen av Web Services.

- *Att hantera elektroniska signaturer och Web Services* (se avsnitt 3.7)
  Om myndigheter exponerar sina register för att externa parter ska kunna hämta eller uppdatera information måste det finnas sätt att garantera att det endast är betrodda parter som ges åtkomst. Exempelvis ska ingen annan part än försäkringskassan kunna hämta information om läkarintyg hos en vårdcentral. Men för att veta om det verkligen är försäkringskassan som knackar på dörren måste en identifiering ske. I sådana fall kan något som kallas för Elektroniska signaturer användas men även de omfattas av vissa legala ramverk.

- *Att motverka obehörig åtkomst* (se avsnitt 5.4)
  Om en vårdcentral bestämmer sig för att den myndighet som handlägger sjukskrivningar ska kunna komma åt det interna journalsystemet för att hämta vissa utvalda läkarutlåtanden så måste det finnas tekniska lösningar som säkerställer att det är en Web Service från just den myndigheten som skickar en förfrågan till vårdcentralen och inte någon som utger sig för att vara det. De tekniska lösningar som rekommenderas i detta fall finns beskrivna under *Säkerhet* i katalogen och behandlar obehörig åtkomst.

- *Att uppnå meddelandesäkerhet - utan mellanhänder* (se avsnitt 5.4)
  Om informationen om en patients sjukdomstillstånd (som skickas mellan en myndighet och en annan) skulle fångas av en obehörig, läsas och manipuleras, vilket inte är så svårt, skulle det innebära stort intrång i den personliga integriteten. Myndigheter måste kunna garantera att sådant inte sker. En osäkerhet hos dem som tjänsten riktar sig till medför en minskad användning och syftet med en implementation av en Web Service går förlorad.

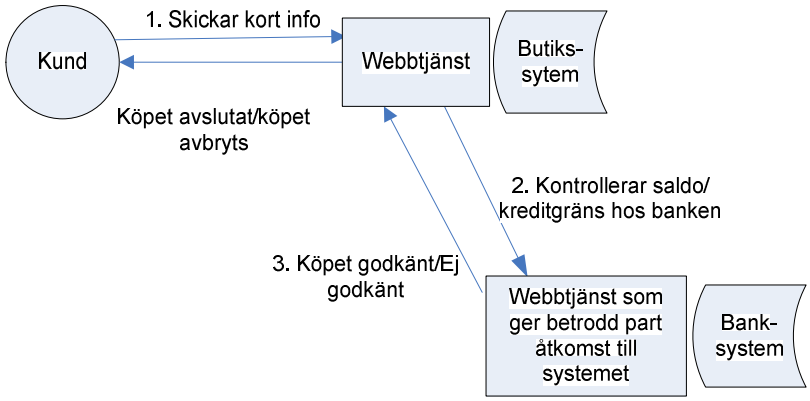
2.3 Bank – kortkontroll och betaltjänst

Ett tillämpningsområde för tjänsteorienterade arkitekturer och Web Services är de transaktionstjänster som erbjuds av banker och andra finansiella institut. Via en Web Service kan interna system exponeras för extern åtkomst och därmed uppdateras i realtid.

Banken kan exempelvis erbjuda en betaltjänst där samarbetspartners så som butiker eller andra försäljningsställen får en säker åtkomst via Web Services till bankens system så att genomförda transaktioner kan registreras utan fördröjning. Framför allt är det en mycket passande lösning för e-handeln.

En bankkund gör flera avslut per dag med sitt betal- eller kreditkort. Systemen är dock sällan integrerade med banksystemet och olika systemplattformar, tekniska lösningar och interna rutiner innebär att registreringen fördröjs, eventuellt icke godkända transaktioner inte blir kända direkt och betalningen går igenom trots brist på täckning. En saldo- eller kreditkontroll direkt i bankens system skulle ge både butiken och kunden ett omedelbart svar om köpet går att genomföra eller inte samt uppdatera saldot/kreditgränsen i realtid. Kunden behöver inte oroa sig för att övertrassera sitt konto och får alltid den aktuella och senast uppdaterade informationen om sina utgifter och tillgångar.

För butikens del innebär det säkrare avslut och mindre jobb med att administrera alla kortköp. Flera helt från varandra oberoende system kommer att upplevas som ett enhetligt och integrerat.



Figur 3: Betaltjänst

Juridiska aspekter

Liksom offentliga myndigheter så är bankväsendet omgivet av ett antal speciella lagar som reglerar verksamheten. Det skulle kunna bli aktuellt när vi i vårt exempel (se figur 3) talar om banksystemets del av betaltjänsten. Om vi ser exemplet från butiksystemets synvinkel, så är problemen snarlika dem för resebyrån. Personlig information om kunden – i detta fall kortnumret m.m. – måste överföras säkert, särskilt som att media ständigt rapporterar om stulna kortnummer.

Dessutom kommer avtalsrätten in, eftersom banken och butiken behöver avtal som reglerar deras relation. Detta nämndes också i exemplet med resebyrån, där problemet med att många parter som samverkar saknar eller brister i sina juridiskt bindande avtal togs upp.

Relaterade mönster

Informationen som skickas över Internet och innehåller bland annat kontokortsnummer kan missbrukas och bör därför skyddas från både avlyssning och manipulering. Vidare måste problemet med autentisering lösas så att obehöriga ej ges åtkomst till systemen. Banken bör också fundera kring juridiska frågor som rör avtal och kostnader för användning av betaltjänsten. Observera att mönsterhänvisningarna inte är uttömmande utan pekar ut endast några typiska mönster för scenariot.

- *Att hantera elektroniska signaturer och Web Services* (se avsnitt 3.7)
  Mönstret tar upp den rättsliga aspekten av tekniska lösningen på problemet med autentisering. Elektroniska signaturer behövs för att kunna säkerställa att det är endast en tillförlitlig part, i det här fallet en butik, som kommer åt personliga konton i bankens system.

- *Att hantera avtalsrätt* (se avsnitt 3.3)
  Precis som icke elektroniska tjänster bör betaltjänster via Web Services inte ske i ett avtalslöst tillstånd. Banken och deras samarbetspartner måste vara överens om vilka avtalsvillkor som gäller vid användningen av betaltjänsten.

- *Att motverka obehörig åtkomst* (se avsnitt 5.4)
  En betaltjänst hos en bank med många kunder kommer att bli nyttjad ofta och av många användare samtidigt. Ett tekniskt mönster som beskriver olika lösningar för autentisering, det vill säga mekanismer som skyddar mot intrång av obehörig part, bör kollas upp.

- *Att uppnå meddelandesäkerhet - utan mellanhänder* (se avsnitt 5.4)
  Den informationen som skickas mellan butiken och banken (punkt-till-punkt förbindelse), bland annat kontokortsnummer, måste skyddas med hjälp av kryptering så att ingen kan fånga upp, läsa och förändra innehållet. Ett kontokortsnummer med eventuella personuppgifter så som namn eller personnummer som fångas upp kan orsaka stor skada för den enskilda personen vilket dessutom skulle innebära ett minskat förtroendet för banken. Säkerhet är i dessa sammanhang en mycket kritisk faktor för verksamheten och måste tryggas till varje pris.

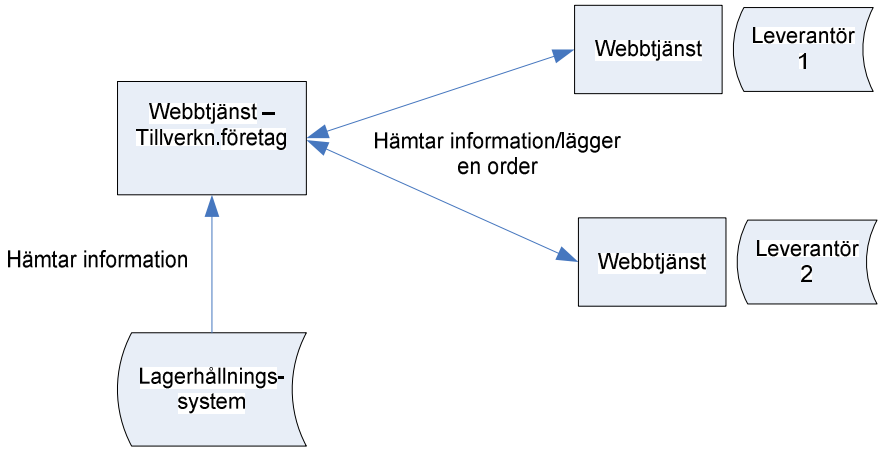
- *Att utnyttja orkestrering (se avsnitt 5.3)*
  En tjänst som implementeras som en Web Service kan utföra olika uppgifter. Mönstret beskriver hur det går att lösa problemet med hur ordningen mellan aktuella uppgifter ska utföras så att det motsvarar den affärsprocess som skall genomföras. I exemplet med bankens betaltjänst ovan skulle detta innebära att en Web Service först kontrollerar att kortnumret stämmer, därefter kontrolleras saldo och kreditlimit och slutligen genomförs själva penningtransaktionen.

2.4 Beställningstjänst – Låt en Web Service göra jobbet

Ett tillverkningsföretag måste se till att lagret alltid har de produkter och komponenter som tillverkningen kräver så att produktionen inte avstannar. Det fordrar delvis ett ganska stort lager och delvis en resurskrävande inköpsprocess - hitta rätt leverantör, skriva kontrakt och vänta på leverans. Företagen tvingas till att göra stora beställningar mer sällan och oftare hos en och samma leverantör.

En tjänsteorienterad arkitektur kan effektivisera inköpsprocessen och framför allt göra den mer flexibel. Det som menas med det är att om olika leverantörer erbjuder beställningstjänster via Web Services så kan alltså ett företag, förutom att ha en konstant överblick över leverantörernas produkter, låta sitt lagerhållningssystem anropa leverantörernas interna system så snart lagret behöver fyllas på. Tjänsterna kan utnyttjas oavsett teknisk plattform, vilket skapar en känsla av ett integrerade system.

För att konkretisera detta kan ett företag som tillverkar bilar tas som ett exempel: Företaget köper exempelvis in däck från andra företag. Om olika leverantörer exponerar sina system kan biltillverkaren via Web Services söka efter aktuell information om vilka typer av däck som finns, antal i lager hos olika leverantörer, jämföra deras priser och leveranstider och utifrån denna information lägga en order hos den leverantören som har det bästa erbjudandet det vill säga där leveranstiden är kortast, produkten finns tillgänglig och priset är lägst. Resultatet av den typen av beställningstjänster är att inköpsprocessen automatiseras då det egna lagerhållningssystemet anropar berörd Web Service hos leverantören, lägger en order direkt i leverantörernas system och sätter igång deras försäljningsprocess. Ett annat tänkbart resultat av sådana tjänster är en minimering av lagerutrymmet utan att leveransen till de egna kunderna riskeras.



Figur 4: Beställningstjänst via Web Service

Juridiska aspekter

Liksom för flera av de tidigare diskuterade mönstren är det aktuellt att diskutera avtalsrätten när fler än en part är inblandade. Diskussionen liknar det som redan tagits

upp och kommer därför inte att upprepas igen. Detsamma gäller för diskussionen kring elektronisk handel och informationssamhällets tjänster, där det är oklart hur erbjudandet av en Web Services ska behandlas.

Något som inte berörts i de tidigare beskrivningarna är immaterialrätt och hur det kan påverka utformningen av Web Services. Man kan anta att tillverkningsföretaget i vårt exempel (se figur 4), kanske också dess leverantörer, har ett visst antal patent och upphovsrätter för sina produkter. Sådant material betraktas vanligen som skyddat och det är oklart var ansvaret ligger om just skyddat material inkluderas i Web Services. Exempelvis finns licensproblem mellan olika patentsystem i EU och USA, vilket kan vara aktuellt för vårt exempel om t.ex. leverantör 1 finns i USA medan det tillverkande företaget finns i ett europeiskt land.

Relaterade mönster

- *Att hantera betalning för Web Services* (se avsnitt 4.1)
  I vissa fall är det lämpligt att ta betalt för sina tjänster. Man hamnar som leverantör i ett kanske mer kritiskt läge då det inte går att förutse sin omsättning som det går om man har långsiktiga avtal. Nu finns det risker för att tillverkningsföretaget väljer andra leverantörer. Å andra sidan öppnas det fler möjligheter att få nya kunder i och med en publicerad Web Service. Om användningen av Web Services ska kosta eller ej måste vägas mot de fördelar som finns med att den är gratis och på så sätt tillgänglig för fler.

- *Att göra en Web Service tillgänglig* (se avsnitt 5.2)
  För att en sökning av produkter och orderläggning ska fungera mellan olika plattformar måste Web Services implementeras enligt vissa standarder så att kommunikationen fungerar. De data som hämtas från olika system ska bearbetas och presenteras i det egna systemet. Därför måste klienten, det vill säga den applikationen hos biltillverkaren som anropar berörd Web Service hos leverantörer veta hur meddelandet ska utformas för att tas emot av mottagande gränssnitt. På samma sätt måste klienten kunna tolka och bearbeta svaret från en Web Service. Mönstret beskriver problemet och ger svar på hur det kan lösas.

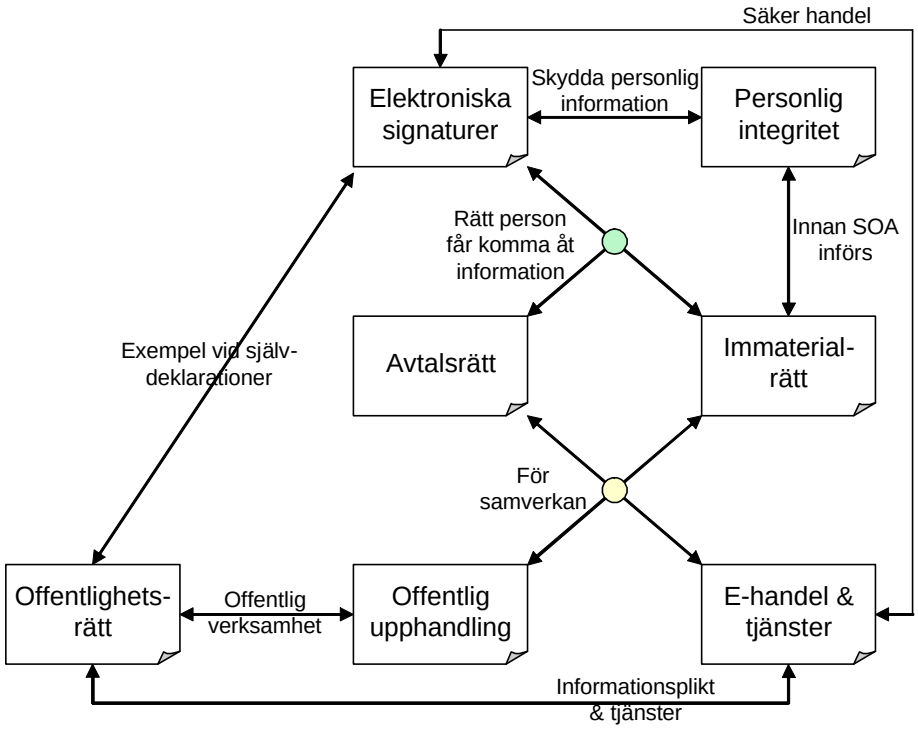
- *Att utnyttja koreografi* (se avsnitt 5.3)
  Om en order ska läggas av tillverkningsföretaget så sker hela processen i ett antal steg. Hur dessa ska koordineras, det vill säga i vilken ordning delprocesserna ska utföras måste specificeras. Beställande företaget, i detta fall biltillverkaren och dess Web Service lägger en beställning, väntar på ett svar från leverantörens Web Service, när svaret i form av en bekräftelse kommer är nästa steg att en betalning ska ske och sedan får köparen information om hur leveransen kommer att ske. Det kan självklart finnas olika aktiviteter som ska utföras med tanken med mönstret är att peka på det problemet som måste lösas då ett antal steg ska utföras och det är viktigt att ordningen bestäms.

- *Att förhindra kringgående av brandväggar* (se avsnitt 5.4)
  I fallet med tillverkningsföretaget och orderläggningen så är säkerheten viktig dock handlar det inte om skydd av känsliga personuppgifter och därför är det inte de juridiska aspekterna som styr hur hårt informationen måste skyddas utan snarare företagens egna och interna säkerhetskrav. Framför allt är det skydd mot intrång i systemen som är central i utformningen av Web Services. Brandväggar

är ett sätt att skydda sig men även där finns det kryphål och mönstret tar upp problematiken specifikt för SOAP meddelanden som skickas mellan företagen.

3 Juridiska mönster

Detta avsnitt fokuserar på att beskriva några av de vanligaste juridiska aspekterna på publika Web Services. Informationen är beskriven i form av mönster på en affärsnivå/strategisk nivå, och lösningarna bör läsas som rekommendationer. Avsnitten 3.1 till 3.7 är därmed tänkta att vara en guide för hur juridiska aspekter för Web Services bör hanteras av organisationer som funderar på att anamma publika Web Services. Mönstren för juridiska aspekter på SOA relaterar till varandra, vilket visas i figur 5. Kopplingarna mellan dem kommer att beskrivas med utgångspunkt från relationerna mellan dem.



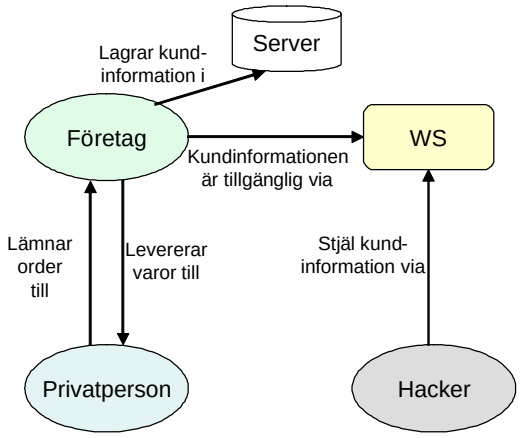
Figur 5: Illustration av relationen mellan de juridiska mönstren

De dubbelriktade pilarna i figur 5 visar att mönstren har något gemensamt. I några fall finns relationer som berör fler än två mönster. I dessa fall går de dubbelriktade pilarna genom en rund ring. Immaterialrätt och Personlig integritet behöver båda hanteras innan SOA införs. Personlig integritet och Elektroniska signaturer handlar båda om att information om privatpersoner ska skyddas. Avtalsrätt, Immaterialrätt och Elektroniska signaturer involverar att endast personer med uttryckligt tillstånd eller auktoritet ska kunna komma åt viss information. Avtalsrätt, Immaterialrätt, Offentlig upphandling och E-handel & tjänster syftar alla på att möjliggöra och reglera samverkan mellan parter, vare sig dessa tillhör offentlig eller privat sektor. Egenskaperna och de speciella lagar som omger offentlig verksamhet är också den gemensamma nämnaren mellan Offentlighetsrätten och Offentlig upphandling. Offentlighetsrätt och E-handel & tjänster berör båda informationskrav och informationsplikter som den nya SOA-arkitekturen kan medföra. Offentlighetsrätten och Elektroniska signaturer har det gemensamt att de berör deklarationer som hanteras via nätet. Till sist kopplas E-handel & tjänster samman med Elektroniska signaturer via ett säkerhetsperspektiv. Här finns en skillnad i typ av säkerhet, dock, i och med att det förstnämnda handlar om generell säkerhet i transaktioner, medan det senare har större fokus på teknisk säkerhet.

3.1 Att hantera personlig integritet

Nedanstående mönster handlar om hur personlig integritet, med avseende på juridiska aspekter, kan hanteras i SOA.

Namn och källa	Personlig integritet, från rapporten SERVIAM-LIT-20, avsnitt 1.3.
Även känt som	-
Typ	Juridiskt mönster.
Syfte	Att ge anvisningar om hur den personliga integriteten bör hanteras i en Web Services-arkitektur.
Krafter	Relevanta lagar.
Problem	Om information om enskilda individer missbrukas eller används på ett därför icke avsett sätt på Internet riskerar dessa individer att kränkas i och med att deras personliga integritet hotas. Detta gäller oavsett om det felaktiga bruket är avsiktligt eller oavsiktligt, och oavsett om förövaren är en privatperson eller ett företag.
Lösning	Koppla in jurister vid skapandet av en Web Services-arkitektur för att gå igenom relevanta lagar *innan* systemdesignen startar. Detta behövs för att skydda och säkra att personlig information används på ett korrekt och lagenligt sätt.
Konsekvenser	Den resulterande Web Services-arkitekturen är anpassad efter gällande lagar och skyddar användarens personliga integritet.
Relaterade mönster	Immaterialrätt, Elektroniska signaturer

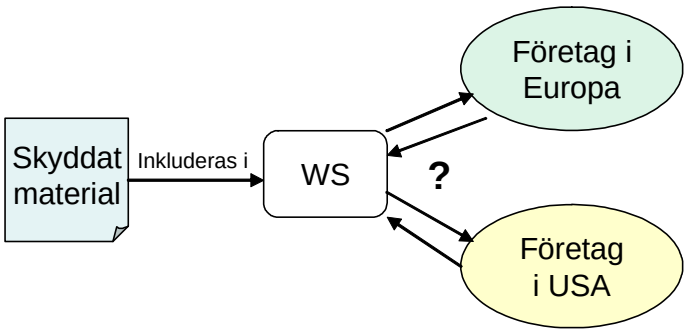


Figur 6: Illustration av problemet för mönstret Personlig integritet.

3.2 Att hantera immaterialrätt

Nedanstående mönster handlar om hur immaterialrätt påverkar utformning av SOA.

Namn och källa	Immaterialrätt, från rapporten: SERVIAM-LIT-20, avsnitt 1.3.
Även känt som	-
Typ	Juridiskt mönster
Syfte	Att ge anvisningar om vilka rättsliga skyddsformer Web Services omfattas av.
Krafter	Relevanta lagar och avtal inom exempelvis olika geografiska regioner.
Problem	Om skyddat material inkluderas i en Web Service idag är det oklart var ansvaret ligger. Detta är ett problem t.ex. eftersom olika patentsystem existerar i EU och USA, vilket kan medföra problem vid oenighet. Om olika nivåer i en Web Services-arkitektur omfattas av olika rättigheter kan immaterialrätten bli fokus för rättsliga tvister. Se figur 7.
Lösning	Företagen bör koppla in jurister för att utreda vilka rättsliga skyddsformer Web Services omfattas av innan arkitekturen införs, *innan* Web Services börjar användas och *innan* visst innehåll tillhandahålles.
Konsekvenser	Den resulterande Web Services-arkitekturen är anpassad efter gällande lagar, samt tar hänsyn till eventuell immaterialrätt på flera olika nivåer.
Relaterade mönster	Avtalsrätt, Personlig integritet, Elektroniska signaturer och Web Services.

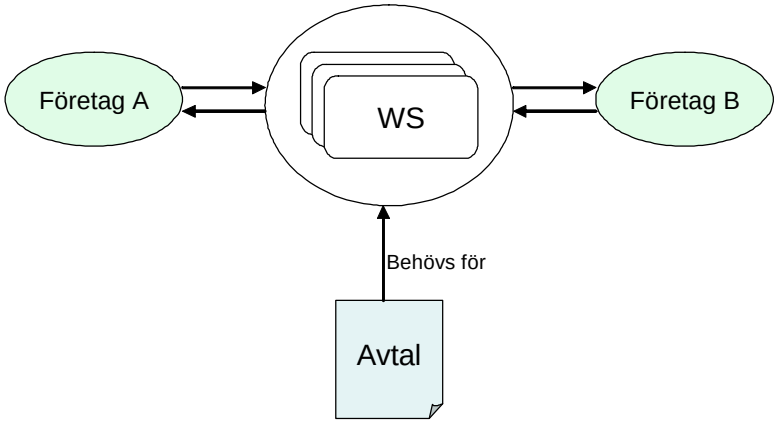


Figur 7: Illustration av problemet för mönstret Immaterialrätt

3.3 Att hantera avtalsrätt

Nedanstående mönster handlar om hur avtalsrätt påverkar SOA.

Namn och källa	Avtalsrätt, från rapporten SERVIAM-LIT-20, avsnitt 1.3.
Även känt som	-
Typ	Juridiskt mönster.
Syfte	Att ge anvisningar om hur avtal kan bli aktuella vid användningen av Web Services.
Krafter	Avtalsrätten.
Problem	Om parter som samverkar via SOA och Web Services saknar eller har brister i sina juridiskt bindande avtal, t.ex. genom avsaknad av giltighetsperiod, kan problem uppkomma vid oenigheter. Brister i avtalen är i dagsläget vanliga. Se figur 8.
Lösning	Branschorganisationer bör fokusera på att skapa standardavtal för samarbete via SOA och Web Services. Företag bör fastställa vilka avtal som kan bli aktuella, hur de sluts och med vilket innehåll *innan* Web Services tas i bruk.
Konsekvenser	Den resulterande Web Services-arkitekturen har inbyggda rättsliga lösningar.
Relaterade mönster	Immaterialrätt, Elektronisk handel och informationssamhällets tjänster, Elektroniska signaturer.

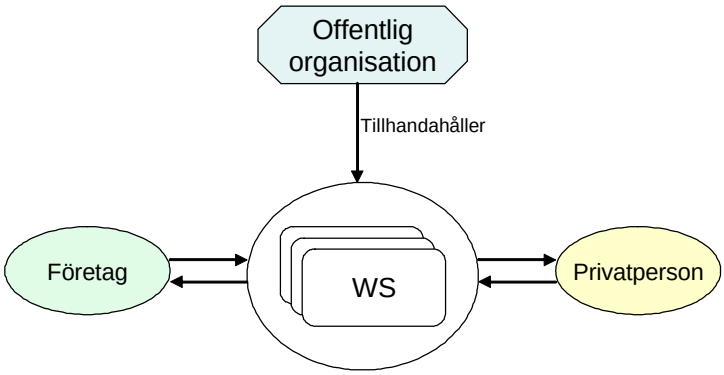


Figur 8: Illustration av problemet för mönstret Avtalsrätt

3.4 Att hantera offentlighetsrätt

Nedanstående mönster handlar om hur offentlighetsrätten kan påverka SOA.

Namn och källa	Offentlighetsrätt, från rapporten SERVIAM-LIT-20, avsnitt 1.3.
Även känt som	-
Typ	Juridiskt mönster.
Syfte	Att belysa hur Web Service-arkitekturer kan påverkas av offentlighetsrätten.
Krafter	Offentlighetsprincipen.
Problem	Om statliga myndigheter lägger ut sin verksamhet elektroniskt på Internet kan de påverkas av offentlighetsprincipen i fråga om t.ex. filtrering, förvaring och lagring hos annan part i samband med elektronisk upphandling. Se figur 9.
Lösning	Statliga myndigheter måste tillsammans med jurister utreda hur offentlighetsprincipen påverkar Web Service-arkitekturen då de är avsändare eller mottagare i densamma.
Konsekvenser	Den resulterande Web Services-arkitekturen är anpassad efter gällande lagar och tar hänsyn till offentlighetsprincipen.
Relaterade mönster	Offentlig upphandling, Elektronisk handel och informationssamhällets tjänster, Elektroniska signaturer.

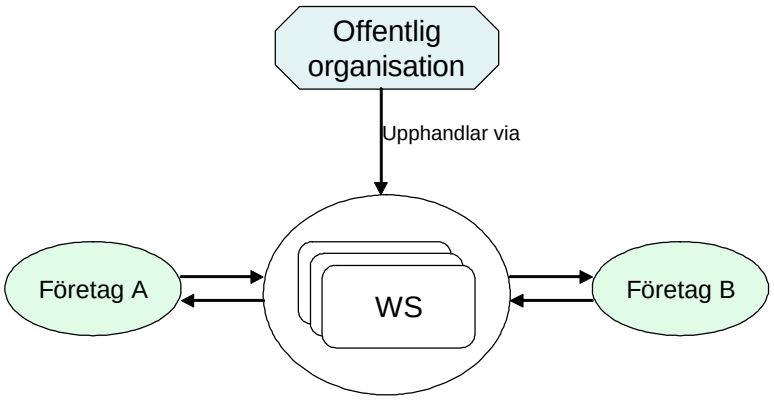


Figur 9: Illustration av problemet för mönstret Offentlighetsrätt

3.5 Att hantera offentlig upphandling

Nedanstående mönster handlar om hur offentlig upphandling påverkas juridiskt av SOA.

Namn och källa	Offentlig upphandling, från rapporten SERVIAM-LIT-20, avsnitt 1.3.
Även känt som	-
Typ	Juridiskt mönster.
Syfte	Att ge anvisningar om hur lagen om offentlig upphandling påverkar användning av SOA.
Krafter	Relevanta lagar.
Problem	På grund av lagen om offentlig upphandling är det oklar om statliga myndigheter affärsmässigt kan kräva att alla som vill deltaga i offentlig upphandling elektroniskt använder t.ex. en specifik Web Services-arkitektur. Ett exempel är om anbud kan komma myndigheten tillhanda i förtid och hur de kan förvaras säkert. Se figur 10.
Lösning	Statliga myndigheter behöver, tillsammans med jurister och de företag som de samarbetar med, utreda hur frågorna kring offentlig upphandling skall/kan behandlas i deras Web Services-arkitekturen. Detsamma gäller för företag som har statliga myndigheter som mottagare.
Konsekvenser	Web Services-arkitekturen är förberedd för att korrekt och säkert kunna hantera problem och frågor som kan uppstå kring offentlig upphandling.
Relaterade mönster	Offentlighetsrätt, Elektronisk handel och informationssamhällets tjänster.

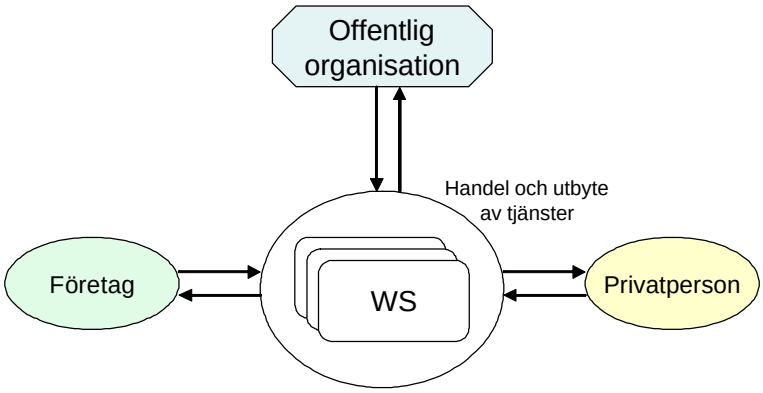


Figur 10: Illustration av problemet för mönstret Offentlig upphandling.

3.6 Att hantera elektronisk handel och informationssamhällets tjänster

Nedanstående mönster handlar om hur SOA juridiskt påverkar elektronisk handel och informationssamhällets tjänster.

Namn och källa	Elektronisk handel och informationssamhällets tjänster, från rapporten SERVIAM-LIT-20, avsnitt 1.3.
Även känt som	-
Typ	Juridiskt mönster.
Syfte	Att ge anvisningar om hur marknadsförändringar kan ställa nya informationskrav på Web Services-arkitekturer.
Krafter	Lagen om elektronisk handel, lagen om distansavtal, m.m.
Problem	Om befintliga gränssnittsstandarder inte uppfylls och eftersom Web Services omfattas av lagen om elektronisk handels definitioner är det oklart hur erbjudandet av en Web Service ska behandlas. Det är även oklart hur dels lagens krav på informationsplikt och dels olika lagtolkningar påverkar Web Services. Se figur 11.
Lösning	Företag och myndigheter behöver analysera lagen om elektronisk handel och fastställa exakt på vilket sätt deras SOA påverkas av den. Detta ska även innefatta i vilken mån lagen om distansavtal gäller.
Konsekvenser	Den resulterande Web Services-arkitekturen tar hänsyn till eventuella informationskrav, samt är anpassad efter gällande lagar.
Relaterade mönster	Avtalsrätt, Offentlig upphandling, Offentlighetsrätt, Elektroniska signaturer och Web Services.

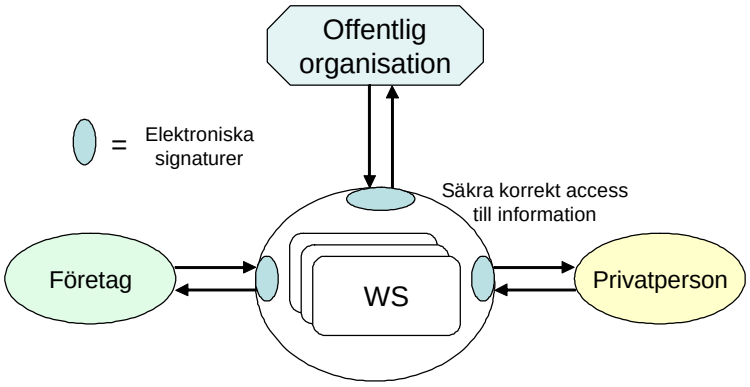


Figur 11: Illustration av problemet för mönstret Elektronisk handel och informationssamhällets tjänster

3.7 Att hantera elektroniska signaturer och Web Services

Nedanstående mönster handlar om hur Web Services relaterar till elektroniska signaturer.

Namn och källa	Elektroniska signaturer och Web Services, från rapporten SERVIAM-LIT-20, avsnitt 1.3.
Även känt som	-
Typ	Juridiskt mönster.
Syfte	Att ge information om hur Web Services-arkitekturer påverkas av lagen om elektroniska signaturer.
Krafter	Lagen om elektroniska signaturer.
Problem	Om befintliga säkerhetsramverk inte tar tillräcklig hänsyn till juridiska aspekter från krav på personlig integritet och inte inkluderar elektroniska signaturer i tillräcklig grad riskerar användningen av Web Services att öka juridiska problem och minska skyddet för enskilda individer. Se figur 12.
Lösning	Företag och myndigheter behöver tillsammans med jurister klargöra hur lagen om elektroniska signaturer påverkar utformningen av SOA, med tanke på säker kommunikation med certifikat och elektroniska signaturer.
Konsekvenser	Den resulterande Web Services-arkitekturen är anpassad efter gällande lagar, samt bidrar till en säkrare kommunikation i en Web Services-arkitektur.
Relaterade mönster	Elektronisk handel och informationssamhällets tjänster, Personlig integritet, Avtalsrätt, Offentlighetsrätt, Immaterialrätt.



Figur 12: Illustration av problemet för mönstret Elektroniska signaturer och Web Services.

4 Ekonomiska mönster

Detta avsnitt fokuserar på att beskriva en vanlig ekonomisk faktor för SOA i form av ett mönster. All information är beskriven på en affärsnivå/strategisk nivå, och lösningen bör läsas som en rekommendation.

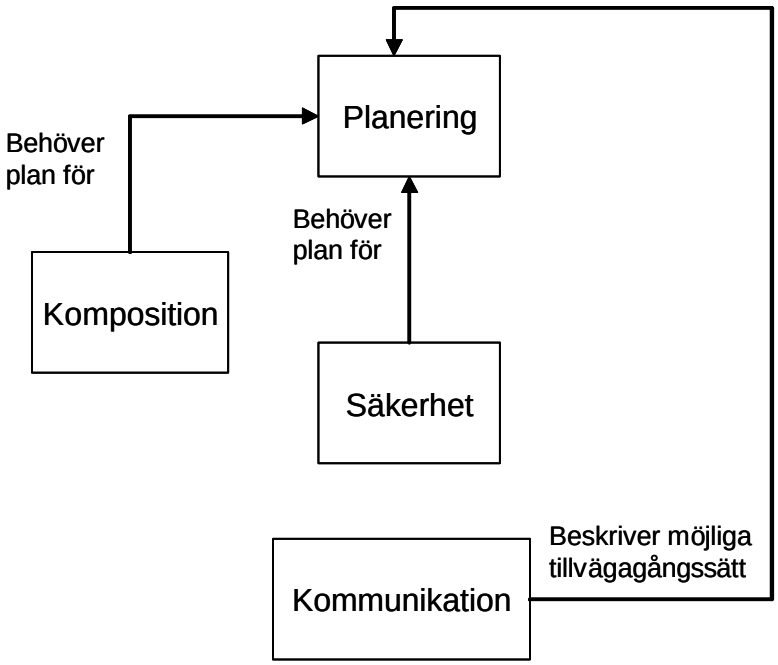
4.1 Att hantera betalning för Web Services

Nedanstående mönster visar olika betalningssätt för Web Services som har framkommit under SERVIAM-projektet.

Namn och källa	Betalning för Web Services, från rapporten SERVIAM-LIT-10, avsnitt 6.2.
Även känt som	-
Typ	Ekonomiskt mönster.
Syfte	Att belysa några olika sätt att ta betalt för Web Services som säljes till andra parter.
Krafter	Företagspartners finansiella situation, samt art av samarbete.
Problem	Det är idag oklart vilka sätt att ta betalt för Web Services som är rättvisa och effektiva, eftersom SOA t.ex. kräver resurser både inledningsvis och i form av underhåll och förändring av tjänsterna.
Lösning	Tre sätt att ta betalt för Web Services nämndes under Serviam-projektet: 1) Auktoriseringstjänst: främst för interna projekt. Kostnaden är något lägre än det skulle kosta för den andra parten att utveckla tjänsten själv. 2) Engångsbetalningar: detta är en vanlig approach, där betalning krävs endast en gång per Web Service. 3) Årlig avgift: detta är en mer realistisk approach än engångsbetalningar, där betalning sker en gång per år efter en slags prenumerationstanke.
Konsekvenser	Användning av Web Services med för få parter kan innebära att önskvärda nyttor inte uppnås som tänkt. Den årliga avgiften är realistisk, men kräver extra funktionalitet och service på grund av det högre priset. De tre exemplen behöver inte vara de enda betalningssätten, men är viktiga alternativ.
Relaterade mönster	Exempel på relaterade mönster: *Avtalsrätt* – eftersom avtal kan reglera hur betalning för en Web Services sker och med vilken periodicitet. *Elektronisk handel och informationssamhällets tjänster* – eftersom Web Services omfattas av dessa koncept i gällande lag.

5 Tekniska mönster

Tekniska mönster utgör förslag till hur Web Services kan realiseras i enlighet med en tjänstebaserad arkitektur och beskriver mönster med en teknisk inriktning mot förslag till implementeringar av lösningar, det vill säga basmodeller på realiseringsnivå. Tekniska mönster är, i detta dokument grupperade inom olika kategorier som behandlar olika aspekter av tekniska mönster. En översikt över kapitlet ges i figur 13. Utgångspunkten är alltid att planeringen av att införa ett SOA-tänkande innefattar att andra tekniska aspekter måste beaktas för att nå ett fungerande resultat.



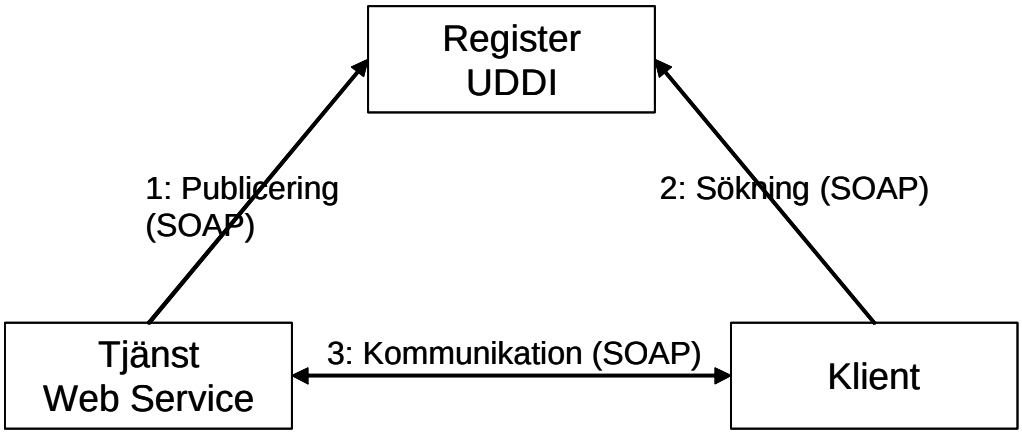
Figur 13: Översikt över tekniska mönster

5.1 Planering

SOA är i grunden ett koncept som bygger på löst kopplade tjänster, implementerade i olika utvecklingsmiljö och på olika plattformar, som kan söka efter och anropa varandra utan att någon bakomliggande programkod skall behöva ändras, det vill säga lokaliseringstransparens och implementeringstransparens. För att åstadkomma detta behövs någon form av miljö att implementera SOA tänkandet i. Detta kapitel ger förslag på hur SOA tänkande kan realiseras mer konkret.

Att realisera SOA med Web Services

Namn och källa	Att realisera SOA med Web Services. Sidorna 35-56 i boken ”Web Service Patterns: Java Edition”.
Även känt som	-
Typ	Tekniskt mönster; Planering
Syfte	Att kunna uppnå implementeringstransparens och lokaliseringstransparens med hjälp av Web Services, det vill säga att oberoende av plattform och programspråk (implementeringstransparens) dynamiskt kunna upptäcka och anropa tjänster oavsett var dessa finns lokaliserade (lokaliseringstransparens).
Krafter	Det är vanligt att det finns olika mjukvarusystem, utvecklade i olika programspråk och på olika plattformar, inom företag. För att kunna utväxla data mellan system, både inom ett företag och mellan företag, görs mer eller mindre situationsanpassade lösningar som behöver justeras om nya system skall involveras i datautbytet. En bättre och mer effektiv lösning på datautbyte, både inom och mellan företag, vore att integrera berörda system med hjälp av Web Services.
Problem	Behov av integrering mellan olika mjukvarusystem kan finnas både internt inom en verksamhet samt externt, via Internet, mellan verksamheter. Gemensamt för båda fallen är att åstadkomma interaktion mellan system utvecklade i olika programspråk/plattformar, det vill säga att uppnå implementeringstransparens. När applikationer interagerar med andra applikationer vore det fördelaktigt om dessa kunde upptäcka andra applikationer som implementerar samma interface. Exempelvis ett företag som har ett system som automatiskt beställer böcker från olika återförsäljare, där beställningen görs hos den återförsäljare som har det lägsta priset för en aktuell titel. Om en ny återförsäljare startar upp en försäljningstjänst, vore det fördelaktigt om det beställande företaget direkt skulle kunna inkludera även denna återförsäljare i sitt beställningssystem. Att dynamiskt kunna lokalisera och använda nya komponenter utan att behöva ändra någon kod för att tala om för applikationen att det finns en ny komponent kallas för lokaliseringstransparens.
Lösning	Implementeringstransparens, det vill säga oberoende av plattform/programspråk kan uppnås genom att använda SOAP, som är ett XML-baserat kommunikationsprotokoll som stöds av de flesta plattformar och programspråk. Lokaliseringstransparens kan uppnås genom att låta tjänster publicera sin existens och de interface som stöds i ett sökbart register. Potentiella användare kan därefter söka i registret efter passande tjänster. Det övergripande skeendet och flödet av data kan utläsas ur figur 14. Den aktör som håller implementeringen av aktuell Web Service publicerar denna i ett register (1). Klienten som efterfrågar en tjänst kan söka efter passande tjänster (2) och när en passande tjänst hittats kan klienten, förutsatt att en proxyklass för gällande interface har genererats, börja kommunicera med tjänsten (3).
Konsekvenser	Prestanda kan komma att påverkas då SOAP naturligt innehåller en del overhead eftersom SOAP-meddelanden översätts till och från XML och vidare till de programspråk som ligger bakom tjänsteinterfacet. Dessutom finns en mängd faktorer som fortfarande är osäkra när användningen av Web Services sker publikt över Internet, exempelvis säkerhetsfrågor, avtalsrätt samt QoS.
Relaterade mönster	Samtliga tekniska mönster i denna katalog.



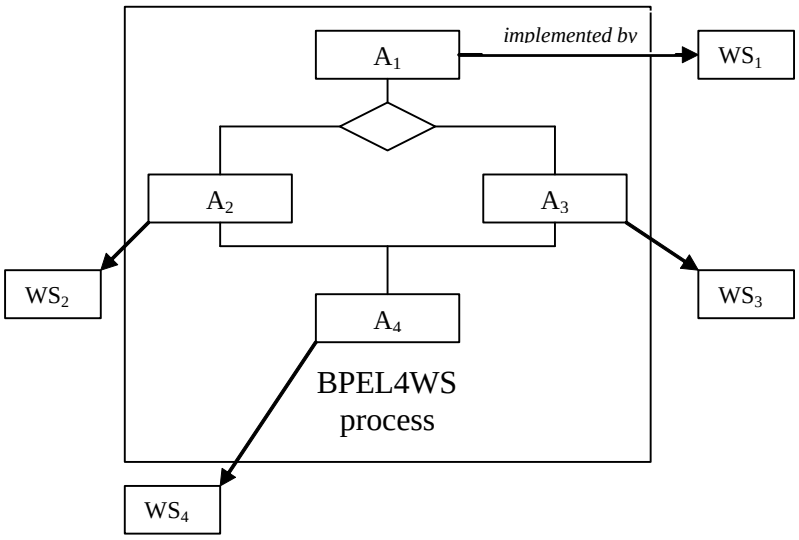
Figur 14: Grundkomponenter och grundaktiviteter för SOA.

5.2 Komposition

Storskaliga tjänsteorienterade system är baserade på komplexa meddelandeutbyten mellan fristående tjänster. När komplexiteten och antalet interaktioner ökar finns det ett behov av att explicit kunna kontrollera och implementera tjänsteinteraktioner. Två sätt att göra detta på är att utnyttja orkestrering och koreografi.

Att utnyttja Orkestrering

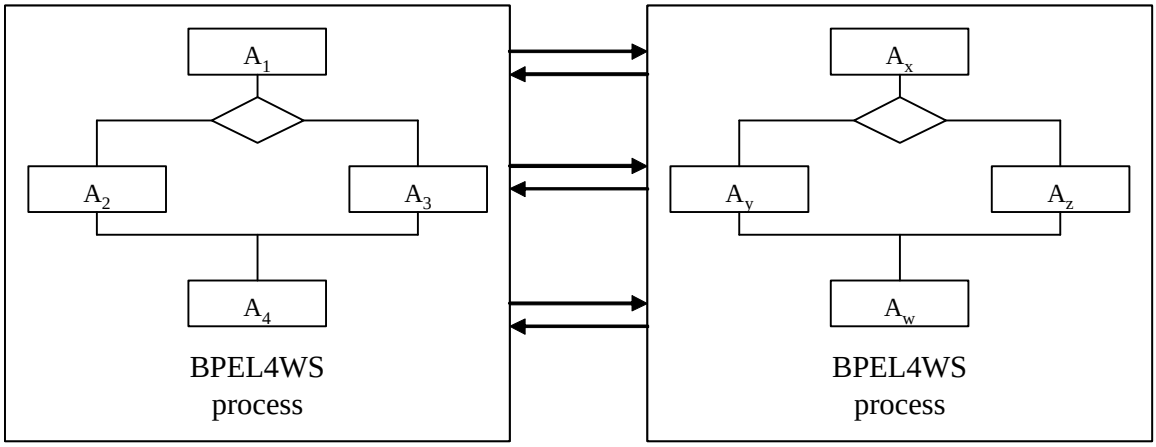
Namn och källa	Att genomföra orkestrering av en Web Service, från rapporten SERVIAM-LIT-04.
Även känt som	-
Typ	Tekniskt mönster; Komposition
Syfte	Att komponera en Web Service i form av en process.
Krafter	Web Services är individuella komponenter. För att stödja komplexa affärstransaktioner som kräver exekvering av flera Web Services i en viss ordning, måste berörda Web Services samordnas och koordineras på något sätt.
Problem	I många olika affärssituationer är det nödvändigt att utföra flera uppgifter, där varje uppgift implementeras i en Web Service, för att slutföra en viss affärsaktivitet. Ett enkelt exempel för att illustrera ovanstående resonemang skulle kunna vara ett företag som säljer böcker online. Det första en kund gör är att välja ut den bok denne vill ha genom att söka rätt på ett visst bokexemplar genom en Web Service: ”ChooseBook”. När kunden kunnat konstatera vilka fakta som hör till boken och bestämmer sig för att köpa boken anropas en annan Web Service: ”BuyBook”. När en eller flera böcker valts ut för att köpas lägger kunden sålunda en order genom att anropa ytterligare en Web Service:”MakeOrder”. När väl ordern har lagts kan kunden dessutom bestämma på vilket sätt denne vill betala genom att anropa nästa Web Service: ”ChoosePayment”. Kunden kan här välja mellan att betala via kreditkort, vilket innebär att kortuppgifter måste lämnas, eller att betala via faktura, vilket renderar i att kunden måste bifoga uppgifter om adress etc. När ordern är konfirmerad kan bokhandeln kolla upp när de beställda böckerna kommer att kunna levereras genom att anropa återförsäljaren för böckerna. Samtidigt kan kunden anropa en annan Web Service: ”WhereAreMyBooks” för att kontrollera statusen på aktuell order. Innan böckerna anländer kan även kunden anropa ytterligare en tjänst för att avbryta ordern varpå bokhandeln måste annullera ordern hos återförsäljaren. Problemet med allt detta är hur alla medverkande Web Services skall kunna koordineras för att en order ska kunna behandlas korrekt?
Lösning	Använd en Web Service baserad teknik för att koordinera Web Services i en viss given ordning, exempelvis BPEL4WS. BPEL är ett skript- och XML baserat språk som gör det möjligt att samordna Web Services som medverkar i komplexa situationer, exempelvis sekvensering, parallellism, synkronisering, loopning och villkorsbaserade beslut. (Se figur 18)
Konsekvenser	Web Services är individuella komponenter och kan anpassas för varandra när det behövs. Dagens skriptbaserade språk för orkestrering, exempelvis BPEL4WS, kräver en komplett specifikation när ett orkestrerat system av Web Services ska designas.
Relaterade mönster	Koreografi. Orkestrering fokuserar på att koordinera Web Services som styrs av en viss part medan koreografi fokuserar på att hantera problematiken runt hur Web Services som styrs av mer än en part skall samarbeta.



Figur 15: Exempel på en lösning för orkestrering av Web Services i en BPEL process bestående av aktiviteterna A_{1-4}

Att utnyttja koreografi

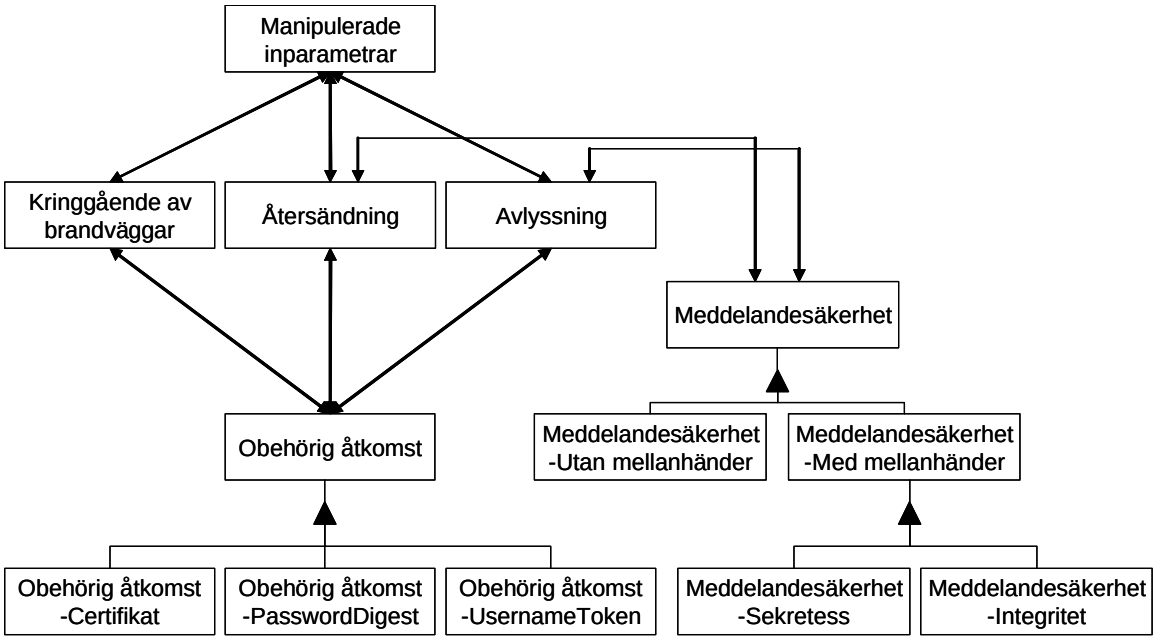
Namn och källa	Att skapa koreografi för en Web Service, från rapporten SERVIAM-LIT-04.
Även känt som	-
Typ	Tekniskt mönster; Komposition
Syfte	Att samordna interaktioner mellan Web Services.
Krafter	Web Service är individuella komponenter. För att stödja komplexa interaktioner mellan olika Web Services som tillhör olika affärspartners måste interaktioner mellan dessa Web Services hanteras på något sätt.
Problem	I ett B2B scenario är det nödvändigt att koordinera affärspartners uppgifter i en viss bestämd ordning. Exempelvis, i ett köpare – säljare sammanhang, beställer först en köpare vissa artiklar genom att anropa säljarens Web Service: ”OrderGoods”. Säljaren svarar på detta genom en konfirmering samt en faktura som skapas genom att anropa köparens Web Service: ”RecieveInvoice”. När köparen utför en betalning genom att anropa säljarens Web Service: ”Payment”, informerar säljaren köparen om hur orderna skall levereras genom att anropa köparens Web Service: ”GetDeliveryInformation”. Problemet är hur det är möjligt att koordinera användandet av berörda Web Services i interaktionerna mellan köpare och säljare så att allt sker i rätt ordning.
Lösning	Använd en Web Service baserad teknik för att koordinera Web Service-interaktioner mellan B2B partners. Ett exempel på en sådan teknik är WSCI som är ett skriptbaserat språk som möjliggör specificering av i vilken ordning olika samverkande Web Services, som ägs av olika parter, skall köras.
Konsekvenser	När en specifikation för ineragering mellan Web Services är gjord måste B2B partners implementera internt stöd för den bestämda interaktionsordningen, exempelvis med hjälp av orkestrering. (Se figur 19)
Relaterade mönster	Orkestrering. Där mönster för koreografi fokuserar på samarbete mellan Web Services ägda av olika parter fokuserar orkestrering istället på hur problematiken runt koordinering av Web Services ägda av en enda part skall hanteras. När koreografi väl etablerats i form av en WSCI specifikation bör varje deltagande part orkestrera sina interna Web Services.



Figur 16: Exempel på en lösning för Koreografi av Web Services från två BPEL processer i ett B2B scenario.

5.3 Säkerhet

Nedanstående mönster visar lösningar på grundläggande säkerhetsproblem som är behäftat med Web Services i allmänhet och publikt exponerade Web Services i synnerhet. Samband mellan redovisade säkerhetsmönster redovisas i figur 20.



Figur 17: Samband mellan säkerhetsmönster

Relationerna mellan ovan presenterade säkerhetsmönster är i regel tvåvägsriktade. Om ett mönster påverkar/relaterar till ett annat sker alltid detta i båda riktningar. Om ett mönster är relaterat till en superklass för ett mönster, exempelvis Obehörig åtkomst, antas implicit att även subklasserna till detta mönster är relaterade. Generellt påverkar redovisade mönster varandra och lösningen på ett problem kan även lösa ett annat, likaväl som att en lösning behöver en kompletterande lösning för att lösa ett problem.

En faktor att beakta är att SOAP, kommunikationsprotokollet för Web Services, är oberoende av transportprotokoll, såsom HTTP, FPT etc. Ofta används HTTP men i praktiken ska SOAP kunna användas även med andra transportprotokoll. Säkerhetsmönstren fokuserar främst på situationer där flerpartskommunikaiton (mer än två parter) råder och mekanismer, baserade på WS-Security, för att hantera säkerhet i dessa situationer baseras generellt på SOAP använt över HTTP. I de fall där endast två parter är inblandade används ofta andra transportprotokoll, exempelvis SSL via HTTPS, som bara fungerar så länge kommunikationen är mellan två punkter. Varje berört redovisat mönster kommer att ha en hänvisning till vilket transportprotokoll, det vill säga HTTP, HTTPS etc., som avses för respektive lösning beroende på om det är flerparts- eller tvåpartskommunikation som avses.

Att motverka obehörig åtkomst

Namn och källa	Att motverka obehörig åtkomst. Från SERVIAM-LIT-05, SERVIAM-SUF-02, WSSb, WSSc.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster
Syfte	Att motverka obehörig åtkomst, det vill säga att avgöra om en anropande entitet är den denne utger sig för att vara.
Krafter	Olika former av obehöriga entiteter som försöker få tillgång till en Web Service, trots att rättigheter saknas, exempelvis hackare, sabotörer och konkurrenter.
Problem	Varje gång en Web Service, som inte är allmänt tillgänglig, dvs. öppen, tar emot en förfrågan, måste det finnas någon form av mekanism som kan avgöra om den anropande entiteten har behörighet till aktuell Web Service.
Lösning	Använd föredefinierade mekanismer för autentisering. Det finns ett antal olika mekanismer för att hantera autentisering, allt ifrån enklare lösenordsbaserade till mer avancerade, som certifikat. För Web Services förordas ett antal specifika mekanismer som beskrivs mer ingående i separata mönster. En viktig poäng med autentiseringsmekanismer är att deras styrka, alltså hur svåra de är att forcera, bör vara ekvivalent med hur känslig natur aktuell Web Service har, det vill säga vilka interna källsystem som kan nås ifrån aktuell Web Service samt vilken natur data, som kan nås genom aktuell Web Service, har. En annan aspekt som är värd att beakta är hur många aktörer som kommer att kommunicera. Beroende på om det är tvåparts- eller flerpartskommunikation som avses finns olika föredefinierade autentiseringsmekanismer som kan användas.
Konsekvenser	Autentisering är ett nödvändigt ont om ett företag vill skydda sina system från obehöriga gäster. Autentisering konsumerar systemresurser och kataloger för legala entiteter måste upprätthållas i vissa fall av lösningar.
Relaterade mönster	Att motverka avlyssning, Att förhindra kringgående av brandväggar, Att förhindra återsändning av meddelanden, Att motverka obehörig åtkomst – UsernameToken, PasswordDigest och Certifikat..

Att motverka obehörig åtkomst - Certifikat

Namn och källa	Att motverka obehörig åtkomst – Certifikat. Från SERVIAM-LIT-05, SERVIAM-SUF-02, WSSc.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster
Syfte	Att motverka obehörig åtkomst, dvs. att fastställa att en anropande entitet är den denne utger sig för att vara för en viss Web Service som behandlar data som har ett kritiskt värde och där antalet medverkande entiteter är stort till antalet.
Krafter	Olika former av obehöriga entiteter som försöker få tillgång till en Web Service, trots att rättigheter saknas, exempelvis hackare, sabotörer och konkurrenter.
Problem	Varje gång en Web Service, som inte är allmänt tillgänglig, dvs. öppen, tar emot en förfrågan, måste det finnas någon form av mekanism som kan avgöra om den anropande entiteten är den denne utger sig för att vara och därigenom har behörighet till aktuell Web Service.
Lösning	Använd `BinarySecurityToken`, dvs. binära former av autentiseringsmekanismer. Eftersom `BinarySecurityTokens` är binära måste en viss kodningstyp specificeras för att representera dessa i XML. Exempelvis kan kodningstypen specificeras som ”Base-64” och därefter specificeras värdetypen, dvs. vilken typ av `BinarySecurityToken` som används. Specifikationen för WS-Security stödjer två värdetyper av `BinarySecurityTokens`; X.509 certifikat och Kerberos tokens. Se WSS SOAP Message Security och WSS X.509 Certificate Token Profile för mer information. `BinarysecurityToken` avser i detta fall flerpartskommunikation över HTTP specificerat i WS-Security. Dock finns i princip samma mekanismer (certifikat) även för tvåpartskommunikation över HTTPS/SSL.
Konsekvenser	En nackdel med att använda binära autentiseringsmekanismer är att dessa är betydligt mer komplexa och dessutom dyrare att implementera. En positiv bieffekt av att använda certifikat eller tokens är att signering av meddelanden blir mer tillförlitlig, då en signatur, skapad i XML-Signature, kan kompletteras av autentiseringsmekanismen vilket ger ett ökat skydd. Ytterligare en positiv bieffekt är att återsändning av meddelanden omöjliggörs vid användning av certifikat.
Relaterade mönster	Att motverka obehörig åtkomst, Att motverka avlyssning, Att förhindra kringgående av brandväggar, Att förhindra återsändning av meddelanden, Att motverka obehörig åtkomst – UsernameToken och PasswordDigest.

Att motverka obehörig åtkomst - PasswordDigest

Namn och källa	Att motverka obehörig åtkomst – PasswordDigest. Från SERVIAM-LIT-05, SERVIAM-SUF-02, WSSb.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster.
Syfte	Att motverka obehörig åtkomst, dvs. att fastställa att en anropande entitet är den denne utger sig för att vara för en viss Web Service som behandlar data där antalet medverkande entiteter är begränsat och där medverkande entiteter inte vill ha certifikat som autentiseringsmekanismer.
Krafter	Olika former av obehöriga entiteter som försöker få tillgång till en Web Service, trots att rättigheter saknas, exempelvis hackare, sabotörer och konkurrenter.
Problem	Varje gång en Web Service, som inte är allmänt tillgänglig, dvs. öppen, tar emot en förfrågan, måste det finnas någon form av mekanism som kan avgöra om den anropande entiteten är den denne utger sig för att vara och därigenom har behörighet till aktuell Web Service.
Lösning	Använd `UsernameToken` tillsammans med `PasswordDigest`. Elementen innebär att ett användarenamn skickas i klartext, men lösenordet som hör till användarenamnet manipuleras och går alltså inte att utläsa i klartext av en betraktare. Lösenordet manipuleras genom att kombinera en tidstämpel, ett slumpvärde och lösenordet i klartext. Denna kombination körs sedan genom algoritmen SHA1 och resultatet skickas som ett lösenord till mottagaren. Mottagaren kör sedan samma process baklänges och erhåller, det på förhand kända, lösenordet i klartext. Se WSS Token Profile 1.0 för mer information. `PasswordDigest` uttryckt i WS-Security berör i första hand flerpartskommunikation men kan naturligtvis även användas för tvåpartskommunikation, båda formerna över HTTP, alternativt HTTPS/SSL enbart för tvåvägskommunikation.
Konsekvenser	Ett måste för att kunna hantera `PasswordDigest` är att både avsändare och mottagare har vetskap om lösenordet i klartext, vilket kan kräva en del administration om det finns många användare till aktuell Web Service. Det kan dessutom bli svårt att distribuera lösenord om många användare ansluter sig. En positiv effekt av att använda `PasswordDigest` är att det går att skydda sig från återsändning av meddelanden. Detta kan göras genom att utnyttja tidstämpeln som kan ge ett meddelande ett ”bäst före datum”, Se WSS Token Profile 1.0 för mer information.
Relaterade mönster	Att motverka obehörig åtkomst, Att motverka avlyssning, Att förhindra kringgående av brandväggar, Att förhindra återsändning av meddelanden, Att motverka obehörig åtkomst – UsernameToken och Certifikat..

Att motverka obehörig åtkomst - UsernameToken

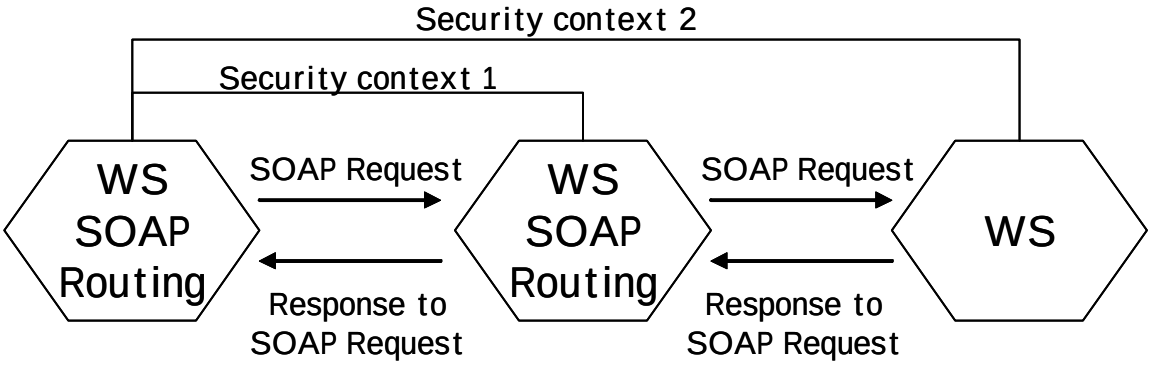
Namn och källa	Att motverka obehörig åtkomst – UsernameToken. Från SERVIAM-LIT-05, SERVIAM-SUF-02, WSSb.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster.
Syfte	Att motverka obehörig åtkomst, dvs. att fastställa att en anropande entitet är den denne utger sig för att vara för en viss Web Service som behandlar data som inte är av kritisk natur, antalet medverkande entiteter är begränsat och där medverkande entiteter inte vill ha mer avancerade autentiseringsmekanismer.
Krafter	Olika former av obehöriga entiteter som försöker få tillgång till en Web Service, trots att rättigheter saknas, exempelvis hackare, sabotörer och konkurrenter.
Problem	Varje gång en Web Service, som inte är allmänt tillgänglig, dvs. öppen, tar emot en förfrågan, måste det finnas någon form av mekanism som kan avgöra om den anropande entiteten är den denne utger sig för att vara och därigenom har behörighet till aktuell Web Service.
Lösning	Använd `UsernameToken` i specifikationen för WS-Security. Detta element utnyttjar den klassiska kombinationen av användarenamn och lösenord. `UsernameToken` implementeras direkt i huvudet på SOAP-meddelandet och valideras hos mottagaren av meddelandet. Se WSS Token Profile 1.0 för mer detaljerad information. `UsernameToken` uttryckt i WS-Security berör i första hand flerpartskommunikation men kan naturligtvis även användas för tvåpartskommunikation, båda formerna över HTTP, alternativt HTTPS/SSL enbart för tvåvägskommunikation.
Konsekvenser	En negativ effekt av att använda lösenord i klartext är att meddelandet som innehåller användarenamn och lösenord inte kan skydda sig självt. Ett uppfångat meddelande kan alltså ge värdefull information för en attackerande entitet med avseende på autentiseringsinformation. Ett annat problem av administrativ karaktär är att innehavaren av en Web Service måste ha kataloger över alla användarenamn och lösenord. En positiv aspekt av att utnyttja lösenord i klartext är att dessa lösningar är relativt enkla och billiga att implementera.
Relaterade mönster	Att motverka obehörig åtkomst, Att motverka avlyssning, Att förhindra kringgående av brandväggar, Att förhindra återsändning av meddelanden, Att motverka obehörig åtkomst – PasswordDigest och Certifikat.

Att uppnå meddelandesäkerhet - Utan mellanhänder

Namn och källa	Att garantera meddelandesäkerhet vid punkt till punkt förbindelse. Från SERVIAM-LIT-05, SERVIAM-SUF-02.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster.
Syfte	Att bevara integritet och sekretess för meddelanden mellan ändpunkterna i en Web Service-interaktion.
Krafter	Entiteter som, exempelvis genom att avlyssna vanliga HTTP-routrar, vill komma över information som kan användas för att skada både konsument och leverantör.
Problem	Om ett meddelande skickas över Internet, via HTTP, finns det goda möjligheter för den som vill och har kunnandet att fånga upp trafik. Om denna trafik skickas i klartext kan information som fångas upp utnyttjas för olika syften.
Lösning	Använd kryptering av meddelanden som skickas. Genom detta kan ingen obehörig läsa innehållet i ett meddelande. Exempel på säkerhetsmekanismer som hanterar skydd av meddelande mellan två punkter är SSL och TLS som, via HTTPS, båda kan hantera både kryptering och signering av meddelanden.
Konsekvenser	Punkt till punkt teknologier garanterar inte säkerhet mellan ändnoderna, endast mellan två punkter, se mönster om att garantera meddelandesäkerhet vid mellanhänder för mer information. Punkt till punkt teknologier erbjuder inte heller möjligheter till filtrering av SOAP-meddelanden vid brandväggar, se mönster om att undvika kringgående av brandväggar för mer information. Ytterligare en nackdel med punkt till punkt teknologier är att dessa inte erbjuder persistent säkerhet. Meddelanden lagras inte i ett krypterat tillstånd på respektive sida, vilket gör att intrång på, exempelvis en webbserver, kan rendera och att information i klartext kan läsas av obehöriga om dessa lyckas komma åt detta innehåll.
Relaterade mönster	Att motverka obehörig åtkomst, Att motverka avlyssning, Att förhindra återsändning av meddelanden, att skydda en Web Service mot manipulerade inparametrar, Att bevara ett meddelandes sekretess vid mellanhänder, Att bevara ett meddelandes integritet vid mellanhänder.

Att uppnå meddelandesäkerhet – Med mellanhänder

Namn och källa	Att garantera meddelandesäkerhet vid mellanhänder. Från SERVIAM-LIT-05, SERVIAM-SUF-02, WSSa.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster.
Syfte	Att bevara integritet och sekretess för meddelanden mellan ändnoderna i en Web Service-interaktion.
Krafter	Opålitliga eller dåligt implementerade former av mellanhänder som utnyttjas av hackare och annan typ av brottslighet som, av olika skäl, vill komma över information som kan användas för att skada både konsument och leverantör.
Problem	När någon form av applikation, dvs. en SOAP-mellanhand, befinner sig mellan ändnoderna i en WS-interaktion kan inte existerande krypteringstekniker, som SSL och TLS, hantera sekretess och integritet för meddelanden eftersom existerande säkerhetstekniker endast erbjuder säkerhet från en punkt till en annan, se figur 1 (security context 1).För att kunna garantera meddelandesäkerhet mellan ändpunkterna i interaktionen (security context 2) behövs ett annat sätt att hantera säkerheten. Ett enkelt exempel på denna problematik kan vara ett e-handelsföretag som agerar som en portal, skickandes meddelanden till olika Web Services som hanterar betalningsrutiner, leveranser av varor etc. Portalen skickar alltså samma meddelande till olika mottagare och varje mottagare skall endast kunna läsa ”sin” del av meddelandet, medan resten av meddelandet skall vara oläsligt, dvs. krypterat. (Se figur 21)
Lösning	Använd XML-Encryption och XML-Signature, via HTTP, enligt specifikationen för WS-Security. XML-Encryption hanterar sekretessen för meddelanden skickade via mellanhänder genom att kryptera valbara delar av meddelandet med olika nycklar, dvs. varje enskild mottagare kan endast läsa ”sin” del av meddelandet, resten är oläsligt. XML-Signature kan användas på, i princip samma sätt, som XML-Encryption. Olika delar av meddelandet kan signeras med olika signaturer. Om någon har manipulerat meddelandet under dess överföring kommer detta att märkas då signaturen för meddelandet, eller den del som är aktuell, inte stämmer överens med den ursprungliga.
Konsekvenser	De negativa effekter som kan härledas till kryptering är försämrad prestanda eftersom, jämfört med exempelvis SSL, ett större antal nycklar måste genereras. Dessutom är användning av XML-Encryption och XML-Signature relativt komplext jämfört med existerande säkerhetsteknologier som SSL och TLS. Om interaktionen mellan Web Services är direkt, dvs. punkt till punkt, fungerar fortfarande SSL och TLS bra. Dock erbjuder existerande teknologier inte persistent säkerhet, data som skickas lagras alltså i klartext på respektive serversida.
Relaterade mönster	Att motverka obehörig åtkomst, Att motverka avlyssning, Att förhindra återsändning av meddelanden, Att skydda en Web Service mot manipulerade inparametrar, Att bevara ett meddelandes sekretess vid mellanhänder, Att bevara ett meddelandes integritet vid mellanhänder.



Figur 18: Problematiken att garantera meddelandesäkerhet när en eller flera mellanhänder finns mellan ändnoderna.

Att uppnå meddelandesäkerhet - Sekretess

Namn och källa	Att bevara ett meddelandes sekretess vid mellanhänder. Från SERVIAM-LIT-05, SERVIAM-SUF-02, WSSa.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster.
Syfte	Att bevara sekretess för meddelanden mellan ändnoderna i en Web Service-interaktion.
Krafter	Existerande säkerhetstekniker som inte lever upp till den miljö publika Web Services arbetar i kan utnyttjas av attackerande entiteter för att komma över känslig information som kan användas för obehörig access eller andra syften.
Problem	När någon form av applikation, dvs. en SOAP-mellanhand, befinner sig mellan ändnoderna i en WS-interaktion kan inte existerande krypteringstekniker (som fungerar bra om inga mellanhänder finns), som SSL och TLS, hantera sekretess för meddelanden. Anledningen till detta problem är att SOAP och SSL arbetar på olika nivå, vilket medför att en mellanhand, arbetande på SOAP-nivå, måste dekryptera hela meddelandet för att hitta ”sin” information. Detta innebär att meddelandets sekretess inte kan upprätthållas.
Lösning	Använd XML-Encryption, via HTTP, som möjliggör att olika delar av ett meddelande kan krypteras med olika nycklar. Det finns olika alternativ till hur nycklar skall distribueras. Antingen kan en symmetrisk nyckel användas direkt men det vanligaste är att en symmetrisk nyckel genereras för att sedan utväxlas med en asymmetrisk nyckel. XML-Encryption stödjer flera krypteringsalgoritmer som 3DES och AES. All information om krypteringen inkluderas i huvudet på SOAP meddelandet. Den information som krypteras pekas ut via referenser i huvudet till resten av meddelandet där den verkliga krypteringen äger rum. Se WSS SOAP Message Security 1.0 för mer information.
Konsekvenser	En möjlig negativ bieffekt är att XML-Encryption, jämfört med mer etablerade säkerhetstekniker, som SSL, är att det, i dagsläget, inte finns speciellt mycket erfarenheter runt att säkra Web Services. Punkt till punkt tekniker, som SSL och TLS, är dokumenterat säkra och bör användas om det finns ett punkt till punkt förhållande mellan Web Services. Dock är XML-Encryption en mer framtidssäker approach med avseende på skalbarhet.
Relaterade mönster	Att motverka obehörig åtkomst, Att motverka avlyssning, Att förhindra återsändning av meddelanden, Att skydda en Web Service mot manipulerade inparametrar, Att garantera meddelandesäkerhet vid mellanhänder, Att bevara ett meddelandes integritet vid mellanhänder.

Att uppnå meddelandesäkerhet - Integritet

Namn och källa	Att bevara ett meddelandes integritet vid mellanhänder. Från SERVIAM-LIT-05, SERVIAM-SUF-02, WSSa.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster.
Syfte	Att bevara integritet för meddelanden mellan ändnoderna i en Web Service-interaktion.
Krafter	Existerande säkerhetstekniker som inte lever upp till den miljö publika Web Services arbetar i kan utnyttjas av attackerande entiteter för att komma över känslig information som kan manipuleras för olika destruktiva syften.
Problem	När någon form av applikation, dvs. en SOAP-mellanhand, befinner sig mellan ändnoderna i en Web Service-interaktion kan inte existerande krypteringstekniker (som fungerar bra om inga mellanhänder finns), som SSL och TLS, hantera sekretess för meddelanden. Anledningen till detta problem är att SOAP och SSL arbetar på olika nivå, vilket medför att en SOAP- mellanhand måste dekryptera hela meddelandet för att hitta ”sin” information. Detta innebär att meddelandets integritet inte kan upprätthållas eftersom att det, om hela meddelandet kan läsas, finns möjligheter att ändra information i meddelandet.
Lösning	Använd XML-Signature, via HTTP, som möjliggör att olika delar av ett meddelande kan signeras med olika signaturer. I och med att någon eller några delar av kroppen eller andra delar i ett SOAP meddelande signeras finns det en garanti för att alla försök till förändringar i meddelandet upptäcks i mottagarens ända av interaktionen. Signaturer i XML kan göras på olika sätt, med direkta eller indirekta referenser till vad som signeras, alltid pekande från huvudet i SOAP meddelandet. Se WSS SOAP Message Security 1.0 för mer information.
Konsekvenser	En negativ effekt av att utnyttja XML-Signature är att det kan bli relativt komplext att implementera mekanismer för att hantera meddelanden med många signaturer. En positiv bieffekt av att tillämpa XML-Signature är att autentiseringsmekanismer kan förstärkas. Detta sker genom att en avsändare av ett meddelande kan bevisa att dennes privata nyckel är just dennes då avsändaren kan kryptera ett litet textmeddelande med sin privata nyckel och därefter signera detta. Genom detta blir det än tydligare att avsändaren är den denne utger sig för att vara eftersom mottagaren kan erhålla avsändarens publika nyckel från certifieringsmyndigheten som visar vem som är innehavare av den privata nyckeln och dekryptera textmeddelandet som är signerat med avsändaren.
Relaterade mönster	Att motverka obehörig åtkomst, Att motverka avlyssning, Att förhindra återsändning av meddelanden, Att skydda en Web Service mot manipulerade inparametrar, Att garantera meddelandesäkerhet vid mellanhänder, Att bevara ett meddelandes sekretess vid mellanhänder.

Att förhindra kringgående av brandväggar

Namn och källa	Att förhindra kringgående av brandväggar. Från SERVIAM-LIT-05.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster.
Syfte	Att hindra manipulerade, till synes legitima meddelanden, att passera brandväggsmekanismer och komma åt bakomliggande källsystem.
Krafter	Attackerande entiteter, till synes legitima, som vill åsamka skada av olika art.
Problem	Attackerande entiteter skickar meddelanden som verkar legitima, dvs. passerar säkerhetsmekanismer som autentisering och kontroll av signatur. Den attackerande parten har sannolikt kommit över information som möjliggör detta genom att finnas på Web Service-konsumentens sida av interaktionen alternativt ifrån dåligt implementerade (sårbara) mellanhänder.
Lösning	Använd brandväggar som kan hantera SOAP-filtrering. Filtreringen bör göras mot ett XML Schema som matchar vad som får och inte får förekomma i ett meddelande.
Konsekvenser	Att filtrera SOAP-meddelanden konsumerar naturligtvis systemresurser. Dessutom krävs det att XML Schemat som används som filtreringsunderlag måste hållas konstant uppdaterat för nya former av manipulerade element. Ett problem med att utnyttja filtrering uppkommer om krypteringstekniker för punkt till punkt kommunikation används eftersom dessa tekniker ofta hindrar en brandvägg att se in i den dataström som flödar mellan sändare och mottagare.
Relaterade mönster	Att skydda en Web Service mot manipulerade inparametrar, Att motverka obehörig åtkomst, Att bevara ett meddelandes integritet vid mellanhänder.

Att förhindra återsändning

Namn och källa	Att förhindra återsändning av meddelanden. Från SERVIAM-LIT-05, WSSa, WSSb, WSSc.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster.
Syfte	Att förhindra obehörig åtkomst genom återsändning av meddelanden.
Krafter	Entiteter som av olika orsaker vill ha obehörig åtkomst till en Web Service.
Problem	En attackerande entitet kan använda uppfångade gamla legitima meddelanden och skicka dessa igen, tillsammans med samma autentiseringsinformation och genom detta få obehörig åtkomst till en Web Service. En sådan typ av attack kan ske i många former men basen är alltid att någon, på något sätt, fångar upp ett meddelande och skickar detta igen.
Lösning	Inom specifikationen för WS-Security finns det ett flertal sätt att hantera detta problem. Olika former av autentiseringsmekanismer som ”PasswordDigest” och certifikat kan användas, eventuellt i kombination med digitala signaturer. Se mer information om lösningar i mönster för att förhindra obehörig åtkomst.
Konsekvenser	Negativa effekter av lösningen är specifika för vilken form av autentiseringsmekanism som används. Se i mönster för att förhindra obehörig åtkomst för mer information.
Relaterade mönster	Att motverka obehörig åtkomst, Att skydda en Web Service mot manipulerade inparametrar, Att motverka obehörig åtkomst – PasswordDigest och Certifikat, Att bevara ett meddelandes integritet vid mellanhänder.

Att motverka avlyssning

Namn och källa	Att motverka avlyssning. Från SERVIAM-LIT-05, SERVIAM-SUF-02, WSSa.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster.
Syfte	Att bevara integritet och sekretess för meddelanden under transport.
Krafter	Attackerande entiteter som, för olika syften, vill komma åt information som skickas mellan sändare och mottagare i en Web Service-interaktion.
Problem	Om ett meddelande som är otillräckligt skyddat blir uppfångat kan en attackerande entitet komma över känslig information av olika art. Exempelvis kan autentiseringsmekanismer användas för obehörig access. Olika former av manipulerade inparametrar kan infogas och skickas vidare. Information kan även stjälas för att användas i andra syften, exempelvis kontokortsnummer som kan utnyttjas direkt olika e-handelsportaler etc.
Lösning	Använd XML-Encryption och XML-Signature enligt WS-Security specifikationen. XML-Encryption används för att bevara meddelandesekretess genom att kryptera olika delar av ett meddelande med olika nycklar. Genom detta kan en SOAP-mellanhand endast kryptera den information denna har rätt till vilket i sin tur gör att ett komplett meddelande aldrig kan läsas i klartext under dess vägg mellan ändnoderna i en Web Service-interaktion. XML-Signature används enligt samma princip och resulterar i att manipulerande med meddelanden inta kan göras utan att detta kan upptäckas då signaturen valideras.
Konsekvenser	De negativa effekter som kan härledas till kryptering är försämrad prestanda eftersom, jämfört med exempelvis SSL, ett större antal nycklar måste genereras. Dessutom är användning av XML-Encryption och XML-Signature relativt komplext jämfört med existerande säkerhetsteknologier som SSL och TLS. Om interaktionen mellan Web Services är direkt, dvs. punkt till punkt, fungerar fortfarande SSL och TLS bra.
Relaterade mönster	Att garantera meddelandesäkerhet vid mellanhänder, Att förhindra återsändning av meddelanden, Att skydda en Web Service mot manipulerade inparametrar, Att bevara ett meddelandes sekretess vid mellanhänder, Att bevara ett meddelandes integritet vid mellanhänder.

Att motverka manipulerade inparametrar

Namn och källa	Att skydda en Web Service mot manipulerade inparametrar. Från SERVIAM-LIT-05, WSSa.
Även känt som	-
Typ	Tekniskt mönster; Säkerhetsmönster
Syfte	Att bevara ett meddelandes integritet.
Krafter	Attackerande entiteter som manipulerar inparametrar genom att utnyttja olika svagheter i Web Services, exempelvis ”SQL-injection” eller ”buffer overflow”.
Problem	Attackerande entiteter kan, om möjlighet ges, skicka meddelanden till en Web Service med parametrar som på ett eller annat sätt försätter aktuell Web Service i ett oönskat tillstånd alternativt tvingar berörd Web Service att gå ned
Lösning	Använd XML-Signature enligt WS-Security specifikationen. Om någonting har ändrats i meddelandet kommer detta att kunna upptäckas då signaturen ändrats och det felaktiga meddelandet kan avslås.
Konsekvenser	Ett möjligt problem är om avsändaren av ett manipulerat meddelande är en betrodd part, dvs. att avsändaren tillhör en entitet som är betrodd och därför har behörighet till en viss Web Service. Om själva manipuleringen av meddelandet sker innan signering kommer naturligtvis inte detta att upptäckas vid en säkerhetskontroll hos mottagande Web Service. En möjlig lösning är att komplettera kontrollen av signaturer med en brandvägg som kan hantera filtrering av SOAP trafik och på så sätt komma åt de meddelanden som inte har det innehåll de borde ha. En positiv bieffekt av att använda signaturer i detta avseende är att avsändaren alltid kan spåras. Om en attack sker via en betrodd part finns det goda möjligheter att spåra upp förövaren av handlingen, vilket i sin tur borde fungera i ett preventivt syfte.
Relaterade mönster	Att garantera meddelandesäkerhet vid mellanhänder, Att motverka avlyssning, Att förhindra återsändning av meddelanden, Att förhindra kringgående av brandväggar, Att bevara ett meddelandes integritet vid mellanhänder.

5.4 Kommunikation

Att använda tjänster innefattar alltid att data med olika syfte kommuniceras mellan en eller flera aktörer via ett gemensamt gränssnitt. Det finns flera olika sätt att utforma kommunikation mellan två eller flera tjänster, beroende på antalet aktörer som medverkar, vilken art inblandade applikationer etc. I detta kapitel listas ett antal olika sätt att hantera kommunikation mellan tjänster beroende på vilken kommunikationsmiljö som eftersträvas.

Att kommunicera med hjälp av punkt-till-punkt lösningar

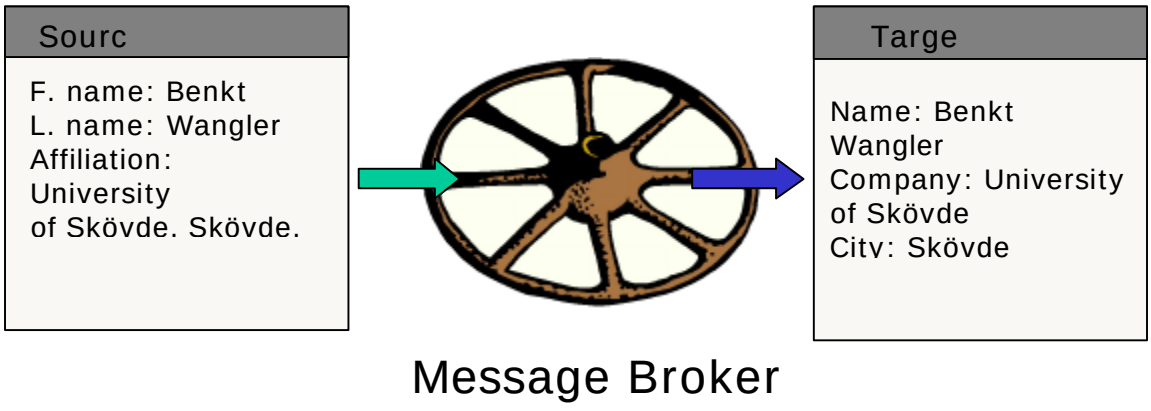
Namn och källa	Att kommunicera punkt till punkt mellan Web Services, från rapporten SERVIAM-LIT-05.
Även känt som	-
Typ	Tekniskt mönster; Kommunikationsmönster.
Syfte	Att erbjuda enkel, okomplicerad kommunikation mellan två aktörer.
Krafter	Företag som bara vill kommunicera direkt med en annan part och utan krav på skalbarhet och flexibilitet, med avseende på antalet kommunicerande noder i interaktionen samt möjligheter att utöka Web Service arkitekturen med kompletterande tjänster, det vill säga mellanhänder.
Problem	Användning och utnyttjande av publika Web Services kan lätt tendera till att bli komplex, då samverkande Web Services bildar mer eller mindre överblickbara nätverk av Web Services med beroenden mellan dessa. Vissa organisationer vill ha enklare och mer överblickbara lösningar som endast rör ett fåtal motparter. I detta fall kan punkt till punkt kommunikation, det vill säga direkt kommunikation mellan två Web Services, utan inblandning av mellanhänder, vara ett passande alternativ.
Lösning	Utnyttja XML, SOAP och WSDL enligt principen för SOA (Se mönstret ”Att realisera SOA med Web Services”, kap 5.1.1). Upprätta avtal etc. juridiska mönster (Se kap 3). Utnyttja beprövade säkerhetsteknologier som garanterar punkt till punkt säkerhet, exempelvis SSL och TLS.
Konsekvenser	Punkt till punkt lösningar är relativt enkla och ger en god överblick över hur Web Services interagerar inom och mellan företag. Dessvärre är en punkt till punkt lösning inte skalbar, då ett tillägg av nya användare, utnyttjande av flera tjänster etc. kan innebära att många ändringar i befintliga Web Services måste göras tillsammans med en större administrering av användare etc. Exempelvis måste säkerhetslösningar tänkas igenom noga med tanke på distribuering av lösenord och nycklar samt förekomster av mellanhänder vilket renderar i helt nya säkerhetsansatser för att kunna garantera säkerhet.
Relaterade mönster	Att garantera meddelandesäkerhet vid punkt till punkt förbindelse, Att realisera SOA med Web Services.

Att utnyttja en broker

Namn och källa	Att utnyttja en broker.
Även känt som	-
Typ	Tekniskt mönster; Kommunikationsmönster.
Syfte	Att göra det möjligt för aktörer och tjänster att samverka på ett effektivt sätt.
Krafter	Det är svårt för ett system att veta vilka tjänster som andra system kan erbjuda samt att veta på vilket sätt man kan kommunicera med dessa.
Problem	Att samordna ett sort antal tjänster av varierande komplexitet mellan ett stort antal aktörer.
Lösning	Att införa en broker, en mäklare mellan källsystem och målsystem som ansvarar för att koordinera kommunikationen mellan systemen. Ansvaret avser framför allt följande tre punkter: Routing. Att lokalisera ett målsystem Ändpunktsregistrering. En mekanism med vilken ett system kan registrera sig hos mäklaren så att systemet och dess tjänster kan hittas av andra. Transformering. En process då ett element konverteras från ett format till ett annat. Man kan skilja mellan två typer av mäklare, direkta och indirekta mäklare. En direkt mäklare etablerar kommunikation mellan ändpunkter, och efter denna initiala kommunikation så kommer ändpunkterna att kommunicera direkt med varandra. Man identifierar alltså först tjänsten via ett tjänsteregister (t.ex. UDDI) och därefter kommunicerar man punkt-till-punkt. En indirekt mäklare fungerar som en mellanhand, d.v.s. all kommunikation mellan ändpunkterna går genom mäklaren. Detta innebär att en sändare inte behöver veta något om målsystemet och dess tjänster. Indirekta mäklare ger också möjlighet till mer av central kontroll. Ett viktigt specialfall är message brokers, som beskrivs i nästa mönster. Till skillnad från en direkt mäklare har en indirekt mäklare ett begränsat ansvarsområde – den tar ej hand om ändpunktsregistrering, det vill säga ett tjänsteregister (t.ex. UDDI).
Konsekvenser	System blir beroende också av mäklare vilket kan öka sårbarheten.
Relaterade mönster	Att göra en Web Service tillgänglig för andra aktörer; Att kommunicera med hjälp av punkt-till-punkt lösningar; Att utnyttja en message broker

Att utnyttja en message broker

Namn och källa	Att utnyttja en message Broker.
Även känt som	-
Typ	Tekniskt mönster; Kommunikationsmönster.
Syfte	Att göra det möjligt för många aktörer och tjänster att samverka på ett effektivt sätt med central kontroll.
Krafter	Antalet tjänster och aktörer samt hur komplexa tjänsterna är kan göra lösningar av punkt-till-punkt modell mer eller mindre omöjliga att använda då varje aktör i detta fall behöver veta vart denne ska vända sig för varje specifik tjänsteinteraktion. Ju större komplexitet som råder desto svårare blir det att utnyttja punkt-till-punkt lösningar.
Problem	Att samordna ett sort antal tjänster av varierande komplexitet mellan ett stort antal aktörer.
Lösning	Inför ett nav, det vill säga en message broker, som hjälper en applikation att identifiera en tjänst som denna behöver. Navet fungerar som en mäklare som tar emot anrop från applikationer och skickar detta vidare till en passande tjänst. Svaret från tjänsten går även detta via navet tillbaka till den anropande applikationen. Navet hjälper framförallt till med två saker: Meddelandeöversättning samt intelligent routing. Med meddelandeöversättning menas att ett meddelande i ett anrop kan behöva översättas för att förstås av den anropade tjänsten, se figur 22. Intelligent routing innebär att navet kan gå in och titta på innehåller i meddelandet för att själv avgöra vilken tjänst som skall anropas.
Konsekvenser	Navet kan i sig bli ytterligare en tjänst där funktionalitet kan placeras vilket kan bidra till ökad komplexitet.
Relaterade mönster	Att göra en Web Service tillgänglig för andra aktörer; Att realisera SOA med Web Services; Att utnyttja en broker



Figur 19: Exempel på en lösning för Meddelandeöversättning med hjälp av ett nav.

Att förenkla filöverföring

Namn och källa	Att förenkla filöverföring.
Även känt som	-
Typ	Tekniskt mönster; Kommunikationsmönster.
Syfte	Att skicka olika typer av filer via en eller flera Web Services.
Krafter	Web Services utbyter information via SOAP/XML. För att möjliggöra filöverföringar av olika typ, exempelvis text, video etc., via Web Service måste dessa filer skickas som SOAP meddelanden.
Problem	Att använda XML för att skicka SOAP meddelanden innebär flera fördelar. Bland annat kan många olika API användas över olika plattformar. Dock finns det ett litet aber på grund av att XML är ett textbaserat format. Det är inte möjligt att skicka binär kod samt kod som inte innehåller XML strukturerad textdata med hjälp av SOAP meddelanden.
Lösning	Använd URL referenser, vilket innebär att data inte inkluderas i XML meddelandet. Istället ges bara en referens på Internet med hjälp av en URL, precis som vilken HTML sida som helst. URL referensen specificeras i XML texten i SOAP meddelandet . Denna lösningen fungera bra när data är tillgänglig för mottagaren av meddelandet. Dock kan problem uppstå om mottagaren av meddelandet av olika anledningar inte har tillgång till Internet. För att lösa dessa problem måste data färdas tillsammans med SOAP meddelandet. Använd kodning, vilket innebär att data som skickas kodas, exempelvis med hjälp av Base 64, vilket innebär att data konverteras till strömmar av symboler som kan överföras med XML. Nackdelen med konvertering är att det tar mycket plats, eftersom binär data alltid är mer kompakt än data representerad i Base 64. Dessutom konsumeras en hel del systemresurser när data skall konverteras fram och tillbaka. Ett annat alternativ är att använda SOAP meddelanden med tillägg (SWA, SOAP messages with attachments) som innebär att data skickas som ”MIME Attachments” till SOAP meddelandet. Detta gör att det går att undvika all konvertering fram och tillbaka som var fallet i stycket ovan. Dock finns det även vissa brister med detta förfarande eftersom vissa XML verktyg inte kan hantera innehåll som inte är i sin ursprungliga form av XML.
Konsekvenser	Alla lösningar utom den första (URL referenser) innebär någon form av prestandaförlust. Dock finns det i framtiden goda möjligheter att minimera prestandaåtgång samtidigt som fördelarna med dessa förfaranden kan utvecklas ytterligare.
Relaterade mönster	Att realisera SOA med Web Services.

Att utnyttja notifiering

Namn och källa	Att utnyttja notifiering.
Även känt som	-
Typ	Tekniskt mönster; Kommunikationsmönster.
Syfte	Att asynkront kunna ge en klient notifieringar på dennes förfrågningar mot en Web Service, det vill säga att undvika blockering av klienten medan denne väntar på ett resultat.
Krafter	När en Web Service, som tar lite tid på sig att utföra en serie operationer, anropas är det inte önskvärt att den anropande klienten blockeras under den tid denne väntar på ett svar, det vill säga notifiering, från den Web Service som anropas. För att stödja en ickeblockerande session skall svaren på anropen ges asynkront.
Problem	I många situationer är det inte önskvärt att låsa upp en anropande part under den tid denne väntar på ett svar. Ett exempel på detta skulle kunna vara ett säljande företag som anropar en Web Service, hanterande tidpunkter för försäljning, som tillhandahålls av detta företags leverantör. Leverantören kanske behöver lite tid, en till två dagar, för att hantera ett anrop. Det är naturligtvis inte möjligt för den anropande applikationen att vänta så länge utan att kunna arbeta vidare med annat.
Lösning	Leverantören av Web Services bör implementera notifieringar så att de kan svara på meddelanden asynkront. Detta är möjligt att åstadkomma genom att implementera en sammansatt Web Service med två asynkrona operationer (en för anrop och en för svar) som koordineras i en sekvens. Flödet mellan anrop och svar implementeras genom att utnyttja enkel orkestrering, implementerad med exempelvis BPEL4WS.
Konsekvenser	Genom att utnyttja detta mönster är det möjligt att upprätthålla en asynkron notifiering från en Web Service. Dock krävs det lite bakomliggande arbete då koordinering mellan anrop och svar behövs.
Relaterade mönster	Orkestrering.

Att skapa en gemensam ontologi

Namn och källa	Gemensam ontologi.
Även känt som	-
Typ	Tekniskt mönster; Kommunikationsmönster.
Syfte	Att överbrygga terminologiska skillnader mellan tjänster.
Krafter	Tjänster utvecklade av olika aktörer ökar risken för terminologiska skillnader, det vill säga att samma term (ord) kan ha olika innebörd.
Problem	Om två eller flera tjänster använder samma term men med olika innebörd, alternativt olika termer för samma sak, kommer dessa tjänster få svårt att samverka. Hur är det möjligt att komma till rätta med problematiken runt begreppsförvirring mellan samverkande tjänster?
Lösning	Man sätter upp en gemensam begreppsapparat, en s.k. ontologi, där man definierar vad som skall menas med olika termer inom ett visst område. I det enklaste fallet har man endast en thesaurus, d.v.s. en ordbok med förklaringar av termer i naturligt språk. Man kan också binda samman termerna i en taxonomi och ange vilka termer som är specialiseringar av andra termer, exempelvis att en hund är ett däggdjur. Slutligen kan man ha mer komplexa ontologier med regler som styr användningen av termerna. När en tjänst skall publicera vad den erbjuder så kan tjänsten använda sig av terminologi från en viss ontologi. Andra tjänster som använder samma ontologi kan då utnyttja denna tjänst på ett korrekt och säkert sätt. Det bör påpekas att dessa ansatser kring ontologier fortfarande är på visionsstadiet.
Konsekvenser	Den främsta svårigheten med denna lösning är att komma överens om ontologier. Det krävs ju att många aktörer använder samma ontologi för att lösningen skall vara meningsfull.
Relaterade mönster	Att göra en Web Service sökbar

6 Referenser

SERVIAM-LIT-05, Toms, A., (2004) Serviam Literature Survey, Part V, Web Services Security.

SERVIAM-SUF-02, Toms, A., (2004) WS-Security – An overview.

SERVIAM-LIT-10, Söderström, E. & Söderström, P. (2004) Serviam Company Visists, Part II, Business Value.

SERVIAM-LIT-20, Lundblad, N. (2004) Rättsliga frågor och Web Services – en probleninventering.

SERVIAM-LIT-04, Zdravkovic, J. (2004) Serviam Literature Survey, Part IV, Service Based Processes.

Monday, P, B. (2003) Web Service Patterns: Java Edition. ISBN: 1-59059-084-8.

WSSa: Web Service Security-SOAP Message Security 1.0 (WS-Security 2004). Available on the Internet (050406): http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

WSSb: Web Service Security-UsernameToken Profile 1.0. Available in the Internet (050406): http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

WSSc: Web Service Security-X.509 Certificate Token profile. Available on the Internet (050406): http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf