Master's Thesis

A Business Process in Execution

- En investeringsbedömning av Business Process Execution Language med Business Process Management

Carl Orvinder IE05

Department of Computer Science
Faculty of Engineering LTH
Lund University, 2009

ISSN 1650-2884
LU-CS-EX: 2009-32

Business Process in Execution

En investeringsbedömning av Business Process Execution Language med Business Process Management Notation

LUNDS UNIVERSITET
Lunds Tekniska Högskola

Examensarbete i datavetenskap

Institutionen för datavetenskap

Carl Orvinder

Handledare:

Gösta Yllner (Sigma Exallon AB)

Examinator

Lars Bendix (LTH)

Sammanfattning

BPEL är ett programmeringsspråk framtagt av IBM, BEA och Microsoft för att implementera affärsprocesser. Programmeringsspråket bygger på XML och är tänkt att koppla ihop olika web services. BPEL har tillsammans med BPMN, som är en kartläggningsstandard utvärderats och jämförts med motsvarande process med implementeringen i Java. Detta har gjorts genom att ett affärsscenario skapats och implementerats i både BPEL och Java med utgångspunkt från en kartläggning av processen i BPMN. Utvärderingen har gjorts med fokus på den totala lönsamheten en investering i BPEL/BPMN skulle rendera. Detta görs mot bakgrund att BPEL/BPMN idag inte har någon större spridning när det gäller användning och därför är det i de flesta fall en investering. Java har valts som referenspunkt och representerar ett allmänt och etablerat programmeringsspråk som programvaruutvecklingsverksamheter redan har goda kunskaper i. Att utvärderingen görs av BPEL och BPMN samfällt har sin bakgrund i att det ofta framhålls att detta skulle vara ett sätt att överbygga gapet mellan programvaruutveckling och verksamhet. Något som renderar stora kostnader. Fokus har legat på investeringen som helhet men har delats i fyra områden som har utvärderats och vägts ihop, anskaffning, arbete, teknik och strategi.

Slutsatsen är att det idag finns fler nackdelar än fördelar med att satsa på BPEL/BPMN. Slutsatsen bygger på flera olika faktorer.

Den största fördelen med tekniken är att den kan föra verksamhet och systemutveckling närmare varandra genom att tillhandahålla verktyg som båda parterna kan använda. Detta görs dock inte bäst genom konvertering mellan BPMN och BPEL utan genom de grafiska utvecklingsverktyg som tillhandahålls för BPEL. Vissa av dessa är gjorda med BPMN-notation vilket är ännu mer fördelaktigt.

Den näst största fördelen är att tekniken är specialanpassad till web services. Det gör att det är smidigare att arbeta med samordning av web services än i ett konventionellt språk. Denna fördel dras dock ner av att språket med sin specialisering har stora begränsningar. Denna nackdel är i dagsläget i princip lika stor som fördelarna med specialiseringen.

En annan faktor som är till nackdel för BPEL är att prestandan på web services byggda i BPEL verkar vara lägre än hos web services byggda i Java, när det kommer till exekveringshastighet.

Den största nackdelen som idag är helt övervägande är dock att tekniken inte har mognat. Samtliga av de verktyg och servermiljöer som provats innehåller en mängd kända och okända buggar, och flera är inte färdigutvecklade. Detta gäller framförallt verktyg som relaterar till BPEL v 2.0. De verktyg som relaterar till v 1.1 av BPEL är visserligen stabilare, men har nackdelen att de riskerar att bli förlegade. System byggda i dessa verktyg riskerar därmed att sakna stöd i framtida programvaror.

Till dessa nackdelar kommer också att en investering innehåller ett anskaffningsmoment som i det här fallet innebär vissa inköp av programvara men framförallt att lära sig ett nytt programmeringsspråk.

Hur mycket olika faktorer värderas beror på företagets övergripande strategi, marknadsposition och verksamhetsinriktning och därför måste slutsatsen självklart sättas i relation till detta. I allmänhet består dock slutsatsen, såvida man inte redan investerat i tekniken eller profilerar sig som föregångare inom BPM och SOA områdena.

Förord

Denna rapport är resultatet av mitt examensarbete på civilingenjörsprogrammet i Industriell ekonomi vid Lund Tekniska Högskola som motsvarar 30 hp eller en termins heltidsstudier. Examensarbetet är skrivet på institutionen för datavetenskap och är gjort på initiativ av Sigma Exallon AB och dess Malmökontor.

Jag vill tacka Sigma Exallon AB och dess personal för att jag fick möjlighet att göra mitt examensarbete hos er och nyttja era lokaler och resurser. Särskilt vill jag tacka min handledare på Sigma Exallon AB, Gösta Yllner, för ditt stöd och den feedback jag har fått under resans gång. Jag vill också tacka Lars Bendix för hjälpen i skrivprocessen med denna rapport, trots att du ”bara gör ditt jobb”. Jag vill även tacka Mikael, Mikael och Henrik, som hjälpt mig att korrekturläsa rapporten och gett mig många goda förslag till förbättringar. Slutligen vill jag tacka min familj Melina, som har stöttat mig under hela processen och som korrekturläst rapporten, och Teodor, som alltid håller pappa på gott humör.

Lund, oktober 2009

Carl Orvinder

Innehållsförteckning

1. Inledning

Informationsteknologins utveckling under de senaste 20 åren har gjort att IT inte bara är ett stöd till vår vardagliga verksamhet utan en naturlig del av den. Affärshändelser och affärsprocesser är alltså i allt större grad är kopplade till eller utförs helt eller delvis av olika IT-system. Detta gör att det blir allt viktigare att systemen kan förändras snabbt och i harmoni med verksamheten. Det finns två nyckelfaktorer för att detta skall lyckas. Den första faktorn är att det behövs en programvaruarkitektur som ordnar systemens tjänster mer i enlighet med processerna och där dessa tjänster är mer självständiga och löst kopplade delar. Detta så att de kan ändras och flyttas utan att andra delar av systemet påverkas. Denna arkitektur kallas Service Oriented Architecture (SOA). Den andra faktorn är ett närmande mellan systemutveckling och verksamhetsutveckling.

Business Process Execution Language (BPEL) är ett programmeringsspråk som bygger på XML och är ett språk för samordning av web services. Tillsammans med en rad andra protokoll i web service-stacken kan BPEL implementera SOA. BPEL är textbaserat men bär spår av flödesbaserade språk från sina föregångare och är också strukturerat så att det lätt kan beskrivas i flödesdiagram. Detta gör att det i de flesta verktyg har ett flödesbaserat grafiskt gränssnitt att arbeta i. Ofta nämns också släktskapet med Business Process Management Notation (BPMN), som är en grafisk kartläggningsnotation för processkartläggning som en möjlig bro mellan systemutveckling och verksamhet. Följaktligen är BPEL tillsammans BPMN av intresse för företag som arbetar med IT-system som snabbare kan anpassa sig och samverka med verksamheten.

Det är därför intressant att utvärdera BPEL tillsammans med BPMN för att undersöka om och på vilket sätt BPEL/BPMN kan bidra till verksamheten och öka lönsamheten, dvs. huruvida det är något att investera tid, pengar och andra resurser i. Utvärderingen av BPEL/BPMN kommer att utgå från ett investeringsperspektiv Detta eftersom språket är av specialiserad karaktär och relativt nytt vilket innebär att det rent allmänt borde vara den vanligaste utgångspunkten bland företag som är intresserade av teknologin. Följaktligen kommer BPEL med BPMN att utvärderas som en investering dvs. en teknik som skall anskaffas. Syftet med examensarbetet är därför att:

Utröna huruvida en investering i BPEL med BPMN ger något mervärde och därmed utgör ett investeringsalternativ som förbättrar arbetet med implementering av affärsprocesser i datorsystem.

Utvecklar man programvara på något plan är det ytterst troligt att man i företaget har goda kunskaper om ett eller oftast flera mer allmänna och traditionella programmeringsspråk och plattformer som Java, .NET, C# etc. Alltså är det naturligt att se en investering i BPEL/BPMN i förhållande till hur man kan göra motsvarande i dag, i ett traditionellt programmeringsspråk. Java har valts för att representera ett sådant språk. BPEL med BPMN kommer således att utvärderas i förhållande till Java.

Den övergripande frågeställningen som skall besvaras är:

Vilka mervärden och merkostnader finns kopplade till en investering i BPEL/BPMN i förhållande till ett allmänt programmeringsspråk, specifikt Java?

Investeringen kommer att utvärderas utifrån fyra huvudområden som tar upp olika delar av en investering:

1. *Anskaffning,* dvs. de faktorer som är kopplade till själva införskaffandet av tekniken, exempelvis direkta utbildningskostnader, minskad effektivitet i den initiala användningen och investeringar i verktyg.
2. *Arbete,* som består av faktorer som påverkar själva arbetet med utvecklingen, exempelvis ökad/minskad arbetstid för utveckling och samordning med verksamhet/kund och ökade/minskade kostnader vid förändringar i systemet för snabbare/långsammare responstid.
3. T*ekniska aspekter*, vilket består av aspekter kopplade till den teknik som används som exempelvis prestanda -ökning/-minskning, säkerhet i tekniken och tekniska risker.
4. *Strategiska aspekter,* handlar om att förstå investeraren och hur olika faktorer påverkar olika företag beroende på deras affärsidéer och strategier. Detta kan vara exempelvis kopplat till image och risker för konkurrensnackdelar respektive möjligheter till konkurrensfördelar.

Huvudområdena ger tillsammans en helhetsbild av investeringen samtidigt som de konkretiserar de olika momenten. För att angripa frågeställningen och göra en så bra utvärdering som möjligt inom de givna resursramarna har en metod i tre steg använts. Varje steg bygger på analys från det föregående och ger en fokusering av detsamma. Om detta kan du läsa i kapitel 2.

I det första steget gjordes en bred litteraturstudie. Studien sammanfattar utvecklingen och forskning på området och ger oss en utgångspunkt för att både designa den egna studien och tolka dess resultat. Resultaten från litteraturstudien kan du läsa i kapitel 3. I kapitlet presenteras de övergripande skolbildningarna som ligger till grund för att BPEL och BPMN alls är intressanta, nämligen Business Process Management (BPM) och Service Oriented Architecture (SOA). I avsnittet om BPM presenteras de övergripande modellerna och grundläggande begreppen inom skolbildningen och efter detta fördjupar vi oss i BPMN. Därefter presenteras SOA övergripligt på samma sätt för att sedan fokusera på web services, där BPEL är en del och som är ett sätt att implementera SOA. Avsnittet avslutas med en fördjupning på BPEL. Därpå kommer ett avsnitt om kopplingen mellan affärer och IT som fokuserar på kopplingen mellan BPMN och BPEL. Kapitlet avslutas med ett antal övergripande modeller som behövs för att tolka resultatet av studien.

I kapitel 4, presenteras den praktiska studien, implementering av en affärsprocess, som utgjort studiens andra steg. Kapitlet beskriver arbetsprocessen från framtagning av scenario vidare till kartläggning, kravhantering, implementering, testning och slutligen vad arbetet resulterade i. I kapitlet beskrivs även steg tre, kompletterande studier, där en mindre studie av prestanda gjordes. Avsnittet beskriver upplägg och resultat av den studien.

Det samlade resultatet analyseras sedan. Analysen kan du läsa i kapitel 5. Kapitlet är indelat i fyra delar utifrån de fyra huvudområdena *anskaffning, arbete, teknik* och *strategi*. På varje område analyseras resultatet av implementeringen i förhållande till litteraturstudien och fördelar respektive nackdelar från varje område konkluderas. Slutsatserna sammanfattas sedan i kapitel 6.

För den som vill fördjupa sig ytterligare finns avslutningsvis ett antal appendix. Appendix A och B, beskriver uppbyggnaden av, och elementen i BPEL respektive BPMN. I appendix C finns instruktioner för installation av BPEL i Oraclemiljö om man själv vill prova tekniken. I appendix D och E, finns krav- respektive testspecifikationerna som användes i implementeringen. I det sista appendixet, F, finns en listning över produkter för BPEL, BPMN och konvertering dessa emellan.

2. Metod

I detta avsnitt beskrivs hur studien är upplagd. Den övergripande strukturen beskrivs samt dess bakgrund och dess syfte. Att förstå och problematisera de metoder som används är centralt för att tolka studiens resultat och resultatets begränsningar. Först kommer frågeställningens natur och studiens omfattning att diskuteras i förhållande till olika metodologiska ansatser. Utifrån detta resonemang kommer studiens design att presenteras och problematiseras. Till sist kommer de olika metoderna i respektive moment att beskrivas och diskuteras.

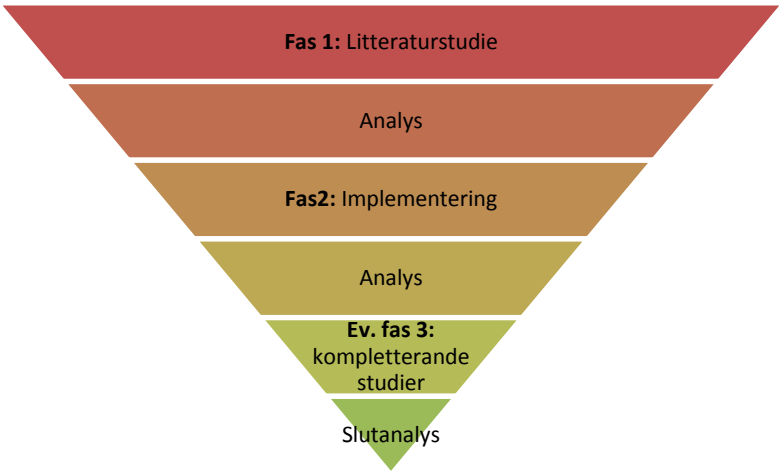
Studiens övergripande frågeställning, *"Vilka mervärden och merkostnader finns kopplade till en investering i BPEL/BPMN i förhållande till ett allmänt programmeringsspråk, specifikt Java?"* handlar om att förstå, värdera och förutsäga hur en investering i en teknik påverkar den som gör investeringen. Det innebär att vi måste förstå tekniken, dess utveckling, hur den skiljer sig från och liknar andra tekniker samt hur vi anskaffar tekniken. Dessutom behöver vi förstå hur dessa aspekter påverkar olika investerare. Det är alltså viktigt att göra en så bra helhetsbedömning som möjligt. Problemet är att studien är begränsad, både resursmässigt och tidsmässigt till en person à 20 veckor. Därför måste studien avgränsas och fokuseras men samtidigt ta upp så pass mycket att det går att bedöma investeringen som helhet. För att konkretisera syftet delades utvärderingen in i fyra aspekter, anskaffning, arbete, teknik och strategi. Dessa aspekter är valda utifrån den övergripande frågeställningens olika element. Anskaffning fokuserar på inhämtandet av själva tekniken i form av investeringar i teknik, men framför allt utbildning. Arbete och teknik fokuserar på själva tekniken och hur den liknar och skiljer sig från andra tekniker. Strategiaspekten fokuserar på hur investeringen påverkar olika investerare. Frågeställningen får med denna indelning ett tydligare fokus men är fortfarande bred. Att göra ytterligare begränsningar av frågan skulle göra att vi tappar fokus på investeringen som helhet, och därmed utvärderar en viss aspekt av investeringen istället. Detta innebär att vi måste göra andra begränsningar, eftersom det är omöjligt att göra en fullständig utvärdering av ens en av aspekterna på den givna tiden. Ytterligare begränsningar får istället göras med hjälp av utredningsmetodiken.

Den övergripande metoden för datainsamling kan antingen vara kvantitativ eller kvalitativ. Ett kvantitativt resultat har den stora fördelen att man, för den population undersökningen gäller, kan få fram generaliserbara resultat. Det går alltså att uttala sig om hur något är i en viss population. Nackdelen är att datainsamlingen måste vara relativt omfattande och dessutom att det inte alltid går att kvantifiera eller med kvantitativa metoder fånga alla aspekter. I detta fall är det enkelt att inse att en kvantitativ metod snabbt skulle leda till ett arbete som skulle bli allt för omfattade, alternativt begränsat till mycket få faktorer och därför inte särdeles aktuell. Istället kommer tillämpning av kvalitativa metoder för datainsamling och tolkning att tillämpas.

Problemet med att välja kvalitativa metoder som utredningsmetodik är att datainsamlingen i huvudsak bygger på tolkningar och att det därför kan vara svårt att garantera vetenskaplighet (Hartman, 2004, s. 275). Kunskapen är därför inte objektiv, i positivistisk mening, dvs. vi kan inte bygga den på observationer av verkligheten. Kunskapen bygger istället på tolkningar av verkligheten i enlighet med den så kallad hermeneutiska vetenskapsteorin. Den innebär allmänt att vi metodologiskt successivt bygger en kunskap om ämnet där varje ny input av data tolkas i förhållande till helhetsuppfattningen och därefter bidrar till en ny helhetsuppfattning. Dock behövs det en

systematik där utredaren kan följa vissa steg för att säkerställa vetenskaplighet i datainsamlingen, genom att i minsta möjliga mån påverka den data som samlas in. Den metod som kommer att användas heter interaktiv induktion. I denna metod samlas data in, analyserar och med analysen som grund samlas sedan ny data in. Under själva insamlandet undviker man att teoretisera över materialet för att inte påverka det. Sammanställning och analys görs efter att all data för den givna iterationen samlats in.

Med denna utgångspunkt har arbetet delats upp i två huvudsakliga faser eller iterationer med mellanliggande analyser, med möjlighet för en tredje kompletterande runda. Analyserna utgör basen för nästa varv. Metoden gör att vi successivt fördjupar oss och slutligen kan besvara vår frågeställning. Samtidigt har vi möjlighet att under resans gång korrigera riktningen så att vi verkligen utvärderar det vi skall och på så sätt få ett valitt resultat.



Figur 1 - Övergripande beskrivning av studiens design

Den första fasen består av litteraturstudier som huvudsakligen syftar till att ge en bred teoretisk bas och utgångspunkt i tidigare studier, expertkunskap och forskning. Man kan såklart tänka sig att man inhämtade denna kunskap på annat sätt eller att man valde att satsa bredare på implementering. Här valdes litteraturstudie eftersom vi på det sättet kan få en bredd i vårt material till en låg tidskostnad. En bredd som vi med ytterligare praktiska studier inte hade kunnat få med den resursbegränsning som vi har. Med detta som grund designas den andra fasen, implementering, för att dels fånga och ytterligare fördjupa den teoretiska kunskapen, dels för att kritiskt granska den. Fasen inriktar sig på att praktiskt använda tekniken och innebär kartläggning och implementering av ett affärsscenario. Processen och resultatet analyseras sedan. Med utgångspunkt från denna analys finns möjlighet att göra mindre kompletterande studier för att fånga viktiga aspekter som inte fångats i tidigare faser eller som ytterligare behöver belysas. I slutanalysen görs en helhetsvärdering av det samlade materialet och en total sammanvägning av olika aspekter görs för att till sist besvara den övergripande frågan. Nu skall vi diskutera de två huvudsakliga metoderna för insamling av data, litteraturstudien och den praktiska tillämpningen, lite djupare.

Den första metoden som används är litteraturstudie. Detta görs, som beskrivits ovan, i syfte att bygga upp en teoretisk referensram, med hjälp av vilken vidare fokusering kan göras. Som regel bör primärkällor användas för litteraturinsamling (Hartman, 2004). Detta eftersom sekundärlitteratur i allmänhet inte ger det djup som primärlitteraturen kan ge. Detta måste dock ställas i förhållande till hur lätt det är att få tag i dessa primärkällor, eftersom vi har en tidsmässig begränsning, och vilket

djup aktuell data behöver ha. I detta fall är dock inte tillgängligheten ett problem eftersom den mesta forskning och information om de aktuella teknologierna finns tillgängligt via ”nätet”, då teknologierna är utvecklat just för och även över nätet. Huvudsakligen kommer alltså primärkällor att användas men i de fall det handlar om att översiktligt beskriva områden kommer sekundärkällor att användas eftersom litteraturstudiens syfte i det fallet är att framför allt ge en översikt av ämnet. Materialet består delvis av böcker (både fysiska och nätbaserade), vetenskapliga artiklar och rapporter, delvis av beskrivningar av teknologin från dess upphovsmän som standarder och arbetsdokument, delvis av data från tillverkare och delvis av artiklar, inlägg och kommentarer från olika auktoriteter och debattörer i branschen. Materialets pålitlighet, reliabilitet, blir förstås aktuellt i samma sekund som man använder ordet ”nätet”. För att säkerställa pålitligheten har i huvudsak dokument som bygger på akademisk forskning (exempelvis forskningsartiklar) samt dokument från dem som tagit fram standarden använts (exempelvis de antagna standarderna eller måldokument för standardarbetet). Visserligen står det kommersiella intressen bakom de organisationer som utvecklar standarderna fastän standardorganisationerna själva inte är vinstdrivande, varför objektiviteten hos dessa data självklart kan ifrågasättas. Dock handlar det i första hand om beskrivningar och bakgrunder till standarderna där detta knappast är ett problem. En del av det material som samlas in kommer dock inte från officiella källor utan från olika mer osäkra källor på ”nätet”. Detta material ger i många fall en djupare förståelse av olika fenomen och är därför viktiga. Problemet är att dessa källor har en låg pålitlighet eftersom de är svåra att verifiera och i många fall kraftigt vinklade till förmån för en viss intressent. För att hantera detta har det i dessa fall använts flera källor som beskriver samma fakta oberoende av varandra, i alla fall observerbart oberoende. Alternativt har olika åsikter om samma sak beskrivits i kontrast till varandra. I samtliga fall har data använts som kan spåras tillbaka till en fysisk person eller till ett företag. Sammantaget uppnår vi därmed en acceptabel nivå för reliabilitet och objektivitet i materialet. Materialet har sedan med ledning av frågeställningen sammanställts och beskrivs i kapitel 3.

Utifrån materialet i litteraturstudien tas sedan ett affärsscenario fram. Dess design bygger på den analys som görs av litteraturstudien. Designen beskrivs i kapitel 4. Dock är det redan från början viktigt att bestämma en metod för själva implementeringen. Detta eftersom implementeringsmetoden i ”verkligheten” som vi strävar efter att efterlikna redan är bestämd innan själva implementeringsuppgiften fås. Implementeringen består av två delar; en kartläggningsdel och en programvaruutvecklingsdel. Kartläggningen görs i BPMN och programvaruutvecklingen görs i BPEL respektive Java.

Eftersom kartläggningen behöver göras i båda fallen och BPMNs duglighet som kartläggningsnotation inte är föremål för utredningen görs bara en kartläggning som båda implementeringarna utgår ifrån. Detta är praktiskt då det både sparar tid och minskar komplexiteten. Dessutom hamnar inte BPMN utan endast kopplingen till BPEL i fokus helt i enlighet med syftet. Detta är också fördelaktigt eftersom kartläggningen då inte påverkar själva resultatet, annat än i det fall kartan måste anpassas för konvertering till BPEL. Eftersom scenariot är fiktivt, kommer affärsprocessen vara tillrättalagd, och då samma person som skrivit affärsscenariot kommer att kartlägga processen, kommer processkartan bara vara en fysisk representation av den idé som personen hade. Därför behövs ingen speciell metod för kartläggning och analys av den kartlagda verkligheten. Däremot behöver vi ha en metod för att göra själva implementeringen av processen.

Denna metod måste överrensstämma med en verklig programvaruutvecklingsprocess och se lika ut för båda implementeringarna. En programvaruutvecklingsprocess innehåller fyra fundamentala delar (Sommerville, 2007):

1. **Specifikation**, där man specificerar den produkt som skall produceras och dess begränsningar.
2. **Utveckling**, där man bygger produkten.
3. **Testning/utvärdering**, där man testar produkten och utvärderar den utifrån de krav som ställts.
4. **Vidareutveckling**, där man vidareutvecklar och förändrar produkten utifrån nya krav.

Specificering av krav och tester behöver bara göras en gång i detta fall, eftersom båda implementeringarna skall vara likvärdiga. Specifikationerna görs alltså först och är variantsoberoende. Därefter görs utvecklingen av respektive variant och slutligen kan båda implementeringarna testas och utvärderas emot specifikationen och tillslut gentemot varandra (någon vidareutveckling gjordes inte p.g.a. de begränsningar i tidsmässiga resurser som studien har, se också kapitel 4). Processen måste således vara linjär och varje steg påbörjas när steget innan är färdigt. Detta för att i så liten utsträckning som möjligt påverka resultatet. Exempelvis kunde kraven påverkas genom att man börjar designa BPEL-processen samtidigt som man sätter upp kraven. Alltså kommer vi att arbeta med den klassiska vattenfallsmodellen (Royce, 1970) fast utan att programvaran är en del av ett system eller att den körs och underhålls (dvs. utan det första och sista steget). Detta upplägg gör att resultatet i så liten mån som möjligt påverkas av att samma person gör alla stegen och ökar därmed resultatets pålitlighet. Denna person har dock fortfarande stor påverkan på studiens resultat. Därför måste vi noga underbygga och påvisa varje resultat och beskriva dess begränsningar. För att analysera slutresultatet kommer det förutom resultatet från litteraturstudien och implementeringen att behöva vissa analysverktyg, framför allt när det gäller det strategiska perspektivet. Därför har ett antal allmänna modeller använts. Dessa presenteras i sista avsnittet i kapitel 3 (3.4). Modellerna hjälper oss att förstå resultaten och förklara dem och kan därmed underbygga dess trovärdighet ytterligare.

3. Teori

I detta kapitel beskrivs de teoretiska modeller och begrepp som behövs för att utvärdera och förstå den empiriska studien. Kapitlet ger en introduktion till de övergripande skolbildningarna som utgör den idémässiga bakgrunden till BPMN och BPEL, nämligen Business Process Management (BPM) och Service Oriented Architecture (SOA). Hur dessa ser ut och är kopplade till varandra utgör själva motivet till att man alls vill använda BPEL, BPMN och dessa tillsammans. Kapitlet ger också en övergripande beskrivning av web service-teknologin, som BPEL ingår i, och hur den implementerar SOA. Detta är viktigt för att förstå BPELs funktion och hur språket relaterar till andra teknologier. Vidare fördjupar vi oss i BPEL och BPMN och hur dessa är kopplade till varandra. Avslutningsvis går vi igenom några övergripande modeller som vi kommer behöva i vår utvärdering.

3.1 Business Process Management (BPM) – Processbaserad verksamhetsutveckling

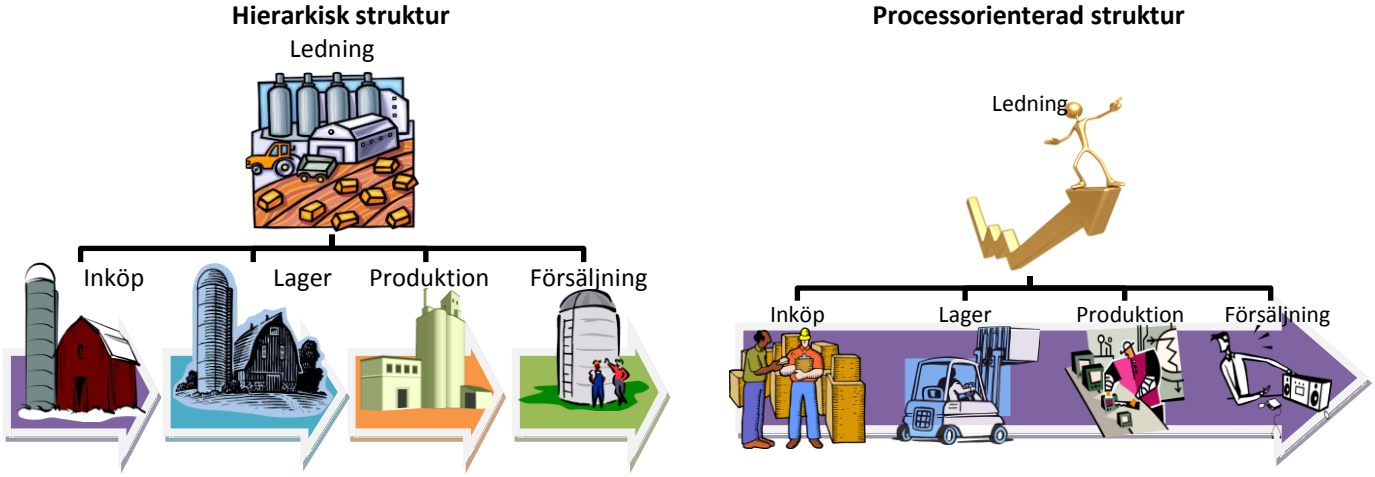
I detta avsnitt ges en introduktion till BPM och en fördjupning i BPMN.

3.1.1 BPM en introduktion

Process är ett sådant ord som alla vet vad det betyder men som inte är riktigt enkelt att sätta fingret på. Det handlar om rörelse och länkade händelser. I (Ljungberg & Larsson, 2001, s. 44) finns flera olika definitioner av processer i affärssammanhang den mest heltäckande lyder:

> *En process är ett repetitivt använt nätverk av i ordning länkade aktiviteter som använder information och resurser för att transformera objekt in" till "objekt ut", från identifiering till tillfredställelse av kundens behov.*

I ett företag pågår ständigt en massa sådana här processer, med olika objekt, kunder och intressenter. Alla dessa processer skall förhoppningsvis göra att företaget uppfyller de affärsmål som finns för verksamheten och tillfredställer de krav som ägare och andra intressenter har på organisationen. För att organisera arbetet i företag behöver de som leder dessa en strategi för att samordna och övervaka arbetet. I årtusenden har människor löst detta genom att bygga hierarkiska strukturer med vertikal indelning och där olika grupper specialiserar sig på vissa typer av uppgifter. I Business Process Management (BPM) sätter man istället de vertikala värdeskapande processerna, som går tvärs genom hela företaget från behov till kundtillfredställelse, i fokus och organiserar företagen efter dem (Ljungberg & Larsson, 2001).



Figur 2 - Hierarkiskt företag jämfört med ett processorienterat

Varje komponent i processen skall addera så mycket värde som möjligt till slutkunden. Kundens behov sätts alltså i fokus. BPM definieras ständigt om idag kan man sammanfatta det som:

> *Aligning processes with the organization's strategic goals, designing and implementing process architectures, establishing process measurement systems that align with organizational goals, and educating and organizing managers so that they will manage processes effective (Harmon, 2003).*

Denna sammanfattning innehåller flera komponenter som talar om hur man skall arbeta med BPM. Processen skall utifrån de strategiska målen för företaget designas och implementeras. Den implementerade processen skall mätas i enlighet med målen och chefer skall utbildas så att arbetet organiseras så effektivt som möjligt. I denna uppsats kommer framför allt två komponenter att fokuseras på och därför beskrivs de närmare. Den första handlar om design, mer specifikt kartläggning av både är- och bör-design. Den andra handlar om implementering men med fokus på processer som är helt eller delvis automatiserade med hjälp av mjukvarutjänster. För att förstå hur vi skall göra detta skall vi först gå igenom olika typer av och nivåer på, processer och också vissa grundläggande begrepp.

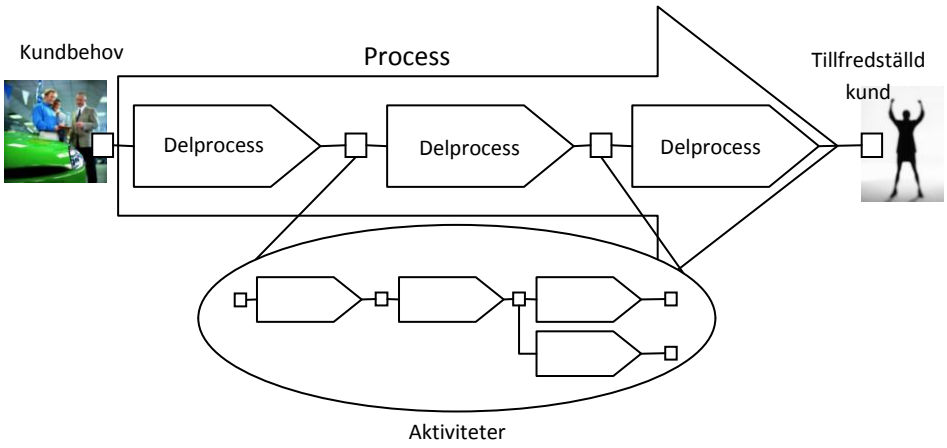
Processer är som vi redan konstaterat ett ganska svårfångat begrepp. Att ha en teoretisk definition hjälper oss visserligen en bit på vägen eftersom den talar om vad vi skall fokusera på men inte hur vi praktiskt skall gå tillväga. Det är inte helt enkelt att se vilka repetitiva länkade mönster som finns i en organisation, var varje länk startar och slutar och inte minst vilka som är viktiga och vilka som inte är viktiga. Ofta börjar man därför med att dela in processer i tre övergripande kategorier (Ljungberg & Larsson, 2001):

1. **Huvudprocesser**, som på ett övergripande plan är de processer som realiserar verksamhetens syfte, exempelvis en produktionsprocess hos ett producerande företag.
2. **Stödprocesser**, är processer som behövs för att organisationen skall fungera i sin helhet men som inte direkt realiserar organisationens syfte, exempelvis underhåll eller bokföring.
3. **Ledningsprocesser**, är den tredje sortens processer och som behövs för att koordinera verksamheten. Ledningsprocesserna går liksom de andra kategorierna att kartlägga och arbeta med men detta görs oftast inte utan fokuset ligger allt som oftast på huvud- och stödprocesserna.

När man arbetar med processer på en övergripande nivå är det viktigt att skilja på dem eftersom de olika kategorierna fungerar på olika sätt och/eller är olika centrala i förhållande till organisationens syfte. Exempelvis är det i ett programvaruutvecklande företag produktion (programvaruutveckling och systemunderhåll), försäljning och produktutveckling som är huvudprocesserna och kanske därför de processer man främst skall prioritera. Bokföringen är exempelvis en stödprocess i samma företag, som måste finnas där för att företaget skall fungera, men kanske inte lika central för företaget och därför mindre prioriterad. På en revisionsfirma är å andra sidan bokföring en huvudprocess, eftersom det är företagets huvudsakliga affärsidé.

Att praktiskt arbeta med processerna på en sådan här övergripande nivå är i princip omöjligt. Istället måste man bryta ner de övergripande processerna, oavsett om den är huvud-, stöd- eller ledningsprocess, i delprocesser. Dessa delprocesser kan i sin tur av brytas ner till ytterligare delprocesser eller aktiviteter. Vilken nivå man arbetar med beror på syftet med att studera

processen, allmänt bör man dock börja på en så hög nivå som möjligt för att förstå helheten. För att implementera processer i mjukvarutjänster måste man dock ner på en ganska låg nivå med aktiviteter (Ljungberg & Larsson, 2001).



Figur 3 - Huvudprocesser, delprocesser och aktiviteter

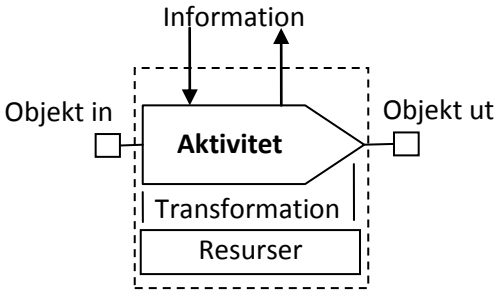
När man hittat en lämplig nivå att arbeta på kan man börja analysera och förbättra processerna. Det handlar om att förstå vad man gör och vad som är värdeskapande och vad som inte är det. Vad som är värdeskapande måste förstås utifrån kunden och dess behov. Kunden och dess behov är dock svårfångade och förändas ofta över tid. Ofta består kunden av många individer med olika behov som inte är helt enkel att identifiera. Det är därför inte helt lätt att veta vad som är värdeskapande och vad som inte är det. Utifrån den uppfattning man ändå har om kundens behov kan man värdera de olika processerna och dess aktiviteter och dela in dem i tre kategorier (Ljungberg & Larsson, 2001):

1. **Värdeadderande aktiviteter** är de aktiviteterna man vill ha så mycket av som möjligt, dessa adderar direkt till värdet, dvs. hjälper till att uppfylla kundens krav. Detta är alltså aktiviteter som kunden är beredd att betala för.
2. **Icke värdeadderande aktiviteter**, är aktiviteter som inte skapar värde för kunden men som behövs för att verksamheten skall fungera eller som skapar värde för andra intressenter. Detta kan vara processövervakning, personalvård eller bokslut. Dessa aktiviteter måste finnas men skall hållas på en så låg nivå som möjligt.
3. **Spill**, är aktiviteter som inte skapar värde för någon intressent eller för kunden. Det kan vara exempelvis överproduktion, väntetider, onödiga transporter etc. Spill skall alltid elimineras.

Man kan såklart dela in aktiviteter i dessa kategorier även om man inte arbetar med processorientering men det innebär att man ofta missar saker som ligger på gränserna mellan organisationers funktioner.

Nu har vi en grundläggande bild av hur vi skall förstå processer i ett företag, dess koppling till företagets syfte, till kundens behov och till varandra. För att kunna arbeta med processerna måste man på något sätt beskriva och kartlägga dem. Kartläggning av de nuvarande processerna är grunden för all förbättring. Förstår vi hur vår verksamhet ser ut idag, kan visualisera och kommunicera det, kan vi arbeta med att förbättra den. Detta görs ofta med olika typer av kartor, exempelvis organisationsdiagram eller processkartor. Det finns en hel uppsjö med olika kartläggningsnotationer. Utgår man från definitionen av affärsprocesser ovan kan man identifiera fem nyckelbegrepp som bör finnas med (Ljungberg & Larsson, 2001):

1. **Objekt in**, är det som startar processen/aktiviteten.
2. **Aktivitet**, är den verksamhet som förädlar objekt in och/eller annan input till objekt ut.
3. **Resurser**, består av de som behövs för att aktiviteten skall kunna utföras, exempelvis personal och maskiner.
4. **Information**, stödjer eller styr processen.
5. **Objekt ut**, är transformationens resultat och blir objekt in för nästkommande aktivitet.



Figur 4 - Processen och dess element

Det finns inte bara en massa notationer för processkartläggning utan också en massa metoder. I detta arbete kommer ingen sådan metod att tillämpas och därför beskrivs inte kartläggningsprocessen närmare. Istället skall vi nu fördjupa oss i den i detta arbete centrala kartläggningsnotationen Business Process Management Notation (BPMN).

3.1.2 Business Process Management Notation (BPMN)

Business Process Management Notation är en standardiserad grafisk notation för kartläggning av affärsprocesser. Den ägs och handhas idag av The Object Management Group (OMG), men togs från början fram av Business Process Management Initiative (BPMI). BPMI publicerade den första versionen av BPMN, version 1.0 (White, 2004), i maj 2004. Standardens huvudsakliga mål var och är att tillhandahålla en notation som både affärsanalytiker, tekniker, ledare och arbetare skall kunna förstå.

> *The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes* (White, 2004).

Ett annat mål som redan år 2004 nämns som viktigt är att tillhandahålla en affärsorienterad visualisering av XML-språk som är designade för att implementera affärsprocesser. Här nämns BPEL som det enda exemplet. Man poängterar visserligen att det kan finnas andra språk men än så länge finns det inga andra alternativ upptagna. Samtliga versioner av BPMN innehåller ett appendix om hur ”mappning” till BPEL kan ske.

I juni 2005 annonserade BPMI och OMG att de skulle slå samman sina arbeten kring BPM (OMG & BPMI, 2005). År 2006 övergick BPMN-rättigheterna till OMG. I februari 2008 släppte OMG version 1.1 (White, 2008) av BPMN. Skillnaderna mellan v 1.0 och v 1.1 är mest kosmetiska. Huvudsakligen bestod de av nya symboler för vissa händelser, nya signalhändelser och en separation mellan *catch* och *throw* händelser (se appendix B). Mappningen till BPEL ändrades också från BPEL4WS version 1.0 till WSBPEL version 1.1. I januari 2009 släptes version 1.2 (White, 2009) av BPMN. En version där man inte gjort några förändringar utan bara gjort mindre formateringar och buggfixar i själva dokumentet (Taylor, 2009), (Decker, 2009).

I dag arbetar OMG med en ny version, 2.0. Version 2.0 innehåller betydligt större förändringar. Först och främst fokuserar OMG på att göra BPMN exekverbar och integrerbar med processexekveringsplattformar. Exempelvis har man inkluderat Business Process Definition Metamodel (BPDM), en XML-baserad metamodell som standardiserar schematiken för diagrammen.

Tidigare har bara själva figurerna varit standardiserade vilket lett till problem när man vill flytta dem mellan olika verktyg. Detta eftersom varje verktyg har haft olika underliggande beskrivningar av diagrammen. Man har också bl.a. definierat mänsklig interaktion, koreografimodell och stöd för affärsregler (business rules) samt fixat en del kända fel i v 1.1. BPMN 2.0 innehåller också mappning till den senast BPEL-standarden WS-BPEL v 2.0. (Axway, o.a., 2009) (Silver, 2009)

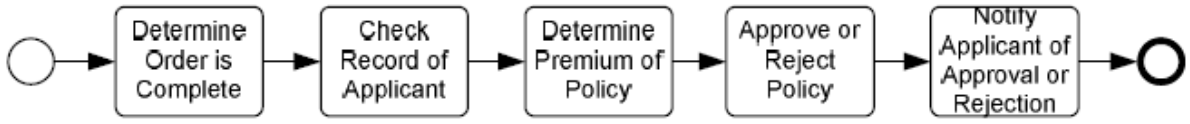
BPMN är alltså en standardnotation för kartläggning av processer. Man har i BPMN valt att dela in notationen i tre underkategorier utifrån olika sätt samordna och beskriva processerna (Axway, o.a., 2009):

1. *Processer* (Orkestrering), som inkluderar:
    - *Privata icke exekverbara* (interna) affärsprocesser
    - *Privata exekverbara* (interna) affärsprocesser
    - *Publika* processer
2. *Koreografier*
3. *Samarbeten* (Collaborations), som kan inkludera processer och/eller koreografier
    - Konversationer

Det är framför allt de så kallade *processerna* och främst då de exekverbara som är intressanta för vår del, eftersom dessa kan mappas till BPEL, då de beskriver orkestreringar. Därför skall vi beskriva dessa lite närmare. Även de andra två kategorierna beskrivs kort, för att det skall bli tydligt varför dessa inte kan mappas till BPEL.

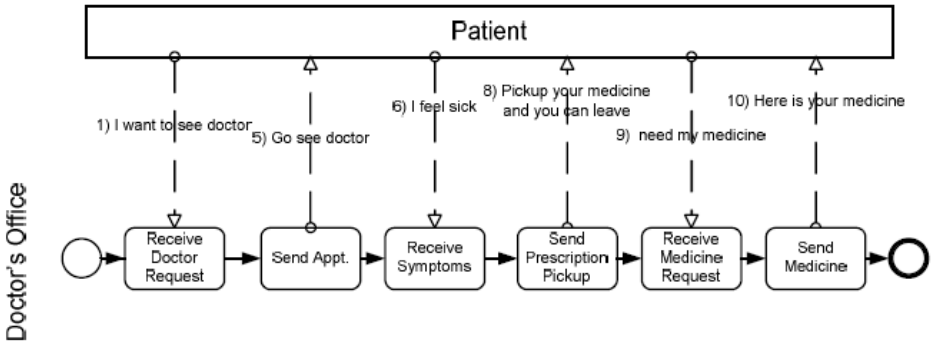
Processer (Orkestrering)

Det finns tre former av *processer,* privata exekverbara, privata icke-exekverbara och publika processer. De privata behandlar interna processflöden medan de publika processerna inkluderar externa partners. De privata processerna behandlar båda **ett** internt flöde. Flödet finns då bara i **en** så kallad *pool* och kan inte korsa dess gränser in till andra *pooler*, in i andra affärsprocesser. Meddelanden kan dock få korsa gränserna för att visa hur processen interagerar med andra processer. Skillnaden mellan exekverbara och icke exekverbara processer är syftet men dem, dvs. huruvida processdiagrammet görs med avsikten att det exekveras i en någon form av automation (exekverbara) eller för dokumentation (icke-exekverbara) (Axway, o.a., 2009). Ett exempel på en privat process finns i figur 5.



Figur 5 – Exempel på en privat process i BPMN (Axway, o.a., 2009)

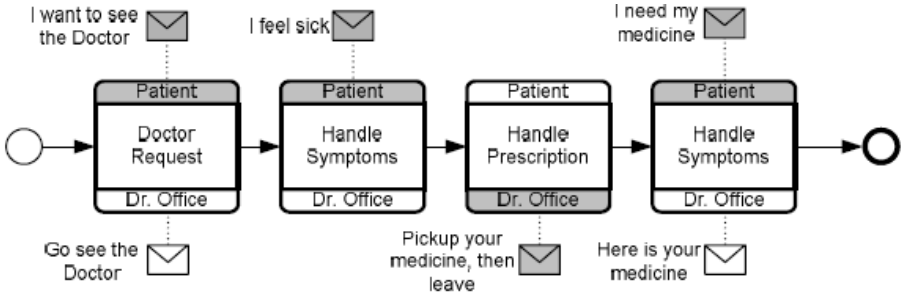
En publik process (i tidigare versioner av BPMN kallad abstrakt) modellerar utbytet mellan en privat process och en annan process eller deltagare (se figur6). Den inkluderar bara de element som kommunicerar med den andra/ de andra deltagarna och visar inget om hur den interna processen ser ut. Den publika processen visar alltså bara informationsflödet och hur sekvensen för meddelandeutbytet (Axway, o.a., 2009).



Figur 6 - Ett exempel på en publik process (Axway, o.a., 2009)

Koreografier och Samarbeten

Koreografier har stora likheter utseendemässigt med processerna. De modelleras som ett flöde med olika aktiviteter och val. Skillnaden är att aktiviteterna i en koreografi består av interaktioner som representerar en mängd (minst ett) meddelandeutbyten mellan en mängd (minst två) parter. Det finns alltså ingen central styrning eller övervakning utan är en slags överrenskommen procedurordning mellan de ingående parterna. Koreografierna existerar i BPMN mellan pooler (Axway, o.a., 2009).



Figur 7 - Ett exempel på en koreografi (Axway, o.a., 2009)

Samarbeten består av flera affärsentiteter, på BPMN-språk minst två *pooler*. Samarbeten är alltså modeller som innehåller två eller flera privata processer. Det finns också en typ av samarbeten kallat konversationer, där processer och koreografier inte får finnas med. Konversationerna består bara av *pooler* och meddelanden och visualiserar endast affärslogiken mellan olika parter. Både koreografier och samarbeten handlar alltså om att modellera logiken mellan affärspartners och inte mellan processerna och är därför inte möjliga att koppla till BPEL (Axway, o.a., 2009).

BPMN har implemnterats i ett antal verktyg, en lista över de flesta sådanna återfinns i appendix F. Listan utgår från den lista som OMG har på den offeciella BPMN-hemsidan och har utökats med ett antal verktyg som hittats. Totalt har 65 verktyg hittats, många av dessa är dock endast byggda för kartläggning och annat affärsstöd och har ingen koppling till programvaruutveckling.

3.2 Service Oriented Architecture (SOA) - Tjänsteorienterad arkitektur

Informationsteknologins utveckling under de senaste 20 åren har gjort att IT inte bara är ett stöd i vår vardag utan en del av den. Den ökade tillgången till information har också gjort att marknaden idag är mer lättrörlig och föränderlig (agile) med allt kortare produktlivscykler. De flesta affärsprocesser involverar idag datorsystem inte bara som stöd utan som en del av själva processen. Detta gör att det idag finns större potential än någonsin att bygga effektiva processer, koppla ihop längre kedjor och upprätthålla större flexibilitet. Samtidigt ökar komplexiteten lavinartat och det som en gång var relativt tydliga kedjor med få varianter, idag är nät med närmast oändligt antal noder. Alltså måste vi idag bygga system som arbetar utifrån våra processer. System med möjlighet att förändras när processerna förändras. Att göra en förändring i ett IT-system idag tar ofta lång tid och utgör en allt större flaskhals i förändrings och förbättringsarbetet. Problemet är att processer och processkartor idag mest är ”fina bilder”, när man kommer till implementering av dem i IT-system. IT-system är ofta komplexa och hierarkiskt uppbyggda med starka beroenden sinsemellan, vilket gör att de är svåra att ändra. Förändringar i IT-arkitekturen blir därför ofta gjorda i klump. Något som medför stora omvälvande förändringar för dem som använder systemet, ofta med stort förändringsmotstånd som konsekvens. Så frågan kvarstår, kan man bygga IT-system som inte bara följer den underliggande processen utan också kan utvecklas tillsammans med den? Tjänsteorienterad arkitektur, *Service Oriented Architecture* (SOA) kanske är svaret (Juric & Pant, 2008).

3.2.1 Övergripande koncept

Vad är då SOA? Det finns flera olika definitioner. I OASIS standarden *Reference Model for Service Oriented Architecture 1.0* (MacKenzie, Laskey, McCabe, Brown, & Metz, 2006), från 2006 definieras begreppet som:

> *Tjänsteorienterad arkitektur är ett paradigm för organisering och utnyttjande av spridda kompetenser som kan vara kontrollerade och ägda av olika domäner.*
> *Fritt översatt från (MacKenzie, Laskey, McCabe, Brown, & Metz, 2006, s. 8)*

The Open Group definierar begreppet på ett annat sätt i *SOA Source Book, 2009*:

> *Tjänsteorienterad arkitektur är en arkitektisk stil som stödjer tjänsteorientering.*
> *Tjänsteorientering är ett sätt att tänka när det gäller tjänster, tjänstebaserad utveckling och resultatet av tjänster.*
> *En tjänst:*
> - *Är en logisk representation av repeterbara affärsaktiviteter som har ett specificerat resultat (ex. undersöker kundkredit, tillhandahåller väderdata, sammanför borrapporter).*
> - *Är autonom.*
> - *Kan bestå av andra tjänster.*
> - *Är en ”black box” för tjänstens konsument.*
>
> *Fritt översatt från (The Open Group, 2009)*

OASIS definition tar sin utgångspunkt i implementeringen av det arkitektoniska mönstret. The Open Group, å andra sidan, definierar begreppet utifrån begreppet tjänst och tjänsteorientering och har en mer konceptuell approach. Tillsammans ger dock definitionerna en ganska heltäckande bild av vad det faktiskt handlar om, nämligen om organisering av IT-tjänster där varje tjänst direkt skall bidra med ett mervärde, dvs. vara värdeadderande. Dessa tjänster kan dessutom ägas av olika företag och vara implementerade på olika ställen. Alltså en slags processorientering av datatjänster. Begreppet kan också ställas i kontrast till objektorienterad arkitektur där objektorientering fokuserar på att

paketera data med olika operationer och där tjänsteorienterad arkitektur fokuserar på själva uppgiften/tjänsten.
SOA-arkitekturen bygger utifrån den allmänna definitionen på fem centrala koncept (MacKenzie, Laskey, McCabe, Brown, & Metz, 2006, ss. 8-9) (Juric, Mathew, & Sarang, 2006, s. 13):

1. **Synlighet och gränssnitt**, är en av arkitekturens första grundsten och innebär att de som har tjänster att tillhandahålla och de som behöver tjänster kan se varandra. Detta görs vanligen genom beskrivningar av tjänsternas olika aspekter. För att det skall bli effektivt måste dessa beskrivningar göras med, eller översättas till, en syntax och semantik som har stor spridning i fråga om tillgänglig och förståelighet. För att underlätta matchningen av kompetenser och behov krävs någon typ av register som håller ordning på de olika tjänsterna.

2. När tjänstekompetenser och behov kan matchas med varandra behövs någon form av **interaktion och meddelandeutbyte**, för att kompetenserna skall kunna utnyttjas. Meddelanden definierar den data som kan utbytas. Detta görs i SOA plattforms- och språkoberoende med mallar. I SOA utbyts bara data vilket är en stor skillnad mot objektorientering då även beteende kan utbytas. Kommunikationen mellan tjänsterna skall kunna ske både **synkront** och **asynkront** beroende på hur processen ser ut.

3. Syftet med att utnyttja kompetenserna skall vara att **skapa reella effekter.** Interaktionen kan därför ses som en handling och resultatet av den som handlingens effekt. Tjänsterna skall alltså återspegla verkliga affärsfunktioner som ger ett reellt mervärde till användaren.

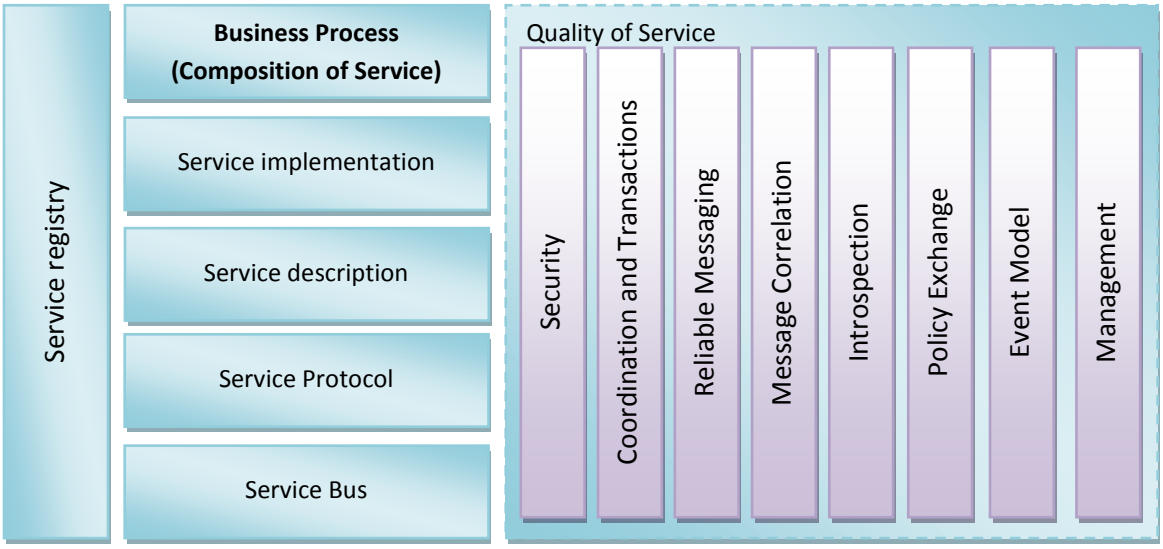
4. En tjänst i en SOA-arkitektur skall alltså vara relativt självständig med **lösa kopplingar** till andra tjänster. Dock måste en infrastruktur kring tjänsterna finnas som garanterar att de fungerar med hög kvalitet. Därför behövs vissa **kvalitetsattribut**, som säkerhet, pålitliga meddelanden, policys och styrning.

5. Det sista konceptet är **organisering av tjänster**. Med löst kopplade tjänster som avspeglar reella ”affärer” kan tjänsterna organiseras i kedjor till processer. Processer som då kan spegla och följa företags och försörjningskedjors affärsprocesser. De digitala processerna skall, då tjänsterna är löst kopplade, enkelt kunna organiseras om, då processerna ändras, för att skapa flexibilitet i kedjan. Inom SOA talar man om två mönster för organisation av tjänster, orkestreringar och koreograferingar. Eftersom BPEL är ett språk för just organisering av tjänster skall vi fördjupa oss lite i dessa mönster innan vi går vidare.

   I en orkestreringsprocess tar en central process kontroll över de involverade tjänsterna och koordinerar utförandet av de olika operationerna. Tjänsterna behöver inte veta att de är en del av en högre affärsprocess de agerar bara på de anrop de får. Denna vetskap ”besitter” endast den centrala processen. En koreografi har ingen central process som orkestrering utan varje tjänst vet när den skall utföras och med vem den skall kommunicera. Alla tjänsterna måste därför förstå processen och sin plats i kedjan. Koreografimönstret är mer i enlighet med processtänkandet men skapar i realiteten mindre flexibilitet än orkestreringen i en implementering.

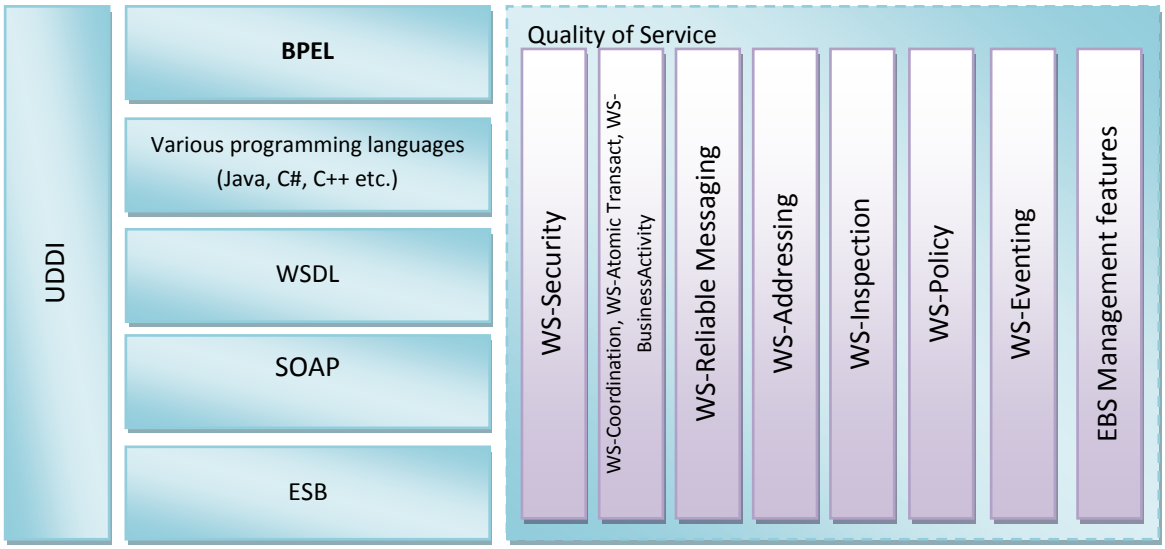
Figur 8 - Orkestrering och Koreografi

Sammanfattningsvis kan vi konstatera att de olika delkoncepten ställer rad krav på vad en implementering av SOA skall innehålla. Vi sammanfattar dessa delar i figur 9.



Figur 9 - Sammanfattning av SOA och dess koncept (Juric, Mathew, & Sarang, 2006)

De olika delarna i figuren (figur 9) kan implementeras på olika sätt. En teknologi som implementerar de flesta av dessa delar är web service-teknologin, där bl.a. BPEL ingår. I figur 10 finns olika web service-teknologier som relaterar till de olika delarna insatta på respektive plats. I nästa avsnitt beskrivs dessa teknologier översiktligt och vi fördjupar oss sedan i BPEL som tillhandahåller ett språk för sammordning av SOA-tjänster för web services.



Figur 10 - Web serviceteknologier för implementering av SOA

3.2.2 Web services

Web services är en samling teknologier för affärstjänster med nätet som plattform. World Wide Web Consortium (W3C) som har hand om de stora standarderna på nätet definierar web services som:

> *En Web service är ett mjukvarusystem avsett att stödja informationsutbyte i maskin-till-maskin-interaktion över ett nätverk. Den har ett gränssnitt beskrivet i en maskinprocesserbart format (framför allt WSDL). Andra system interagerar med Web servicen på sätt som föreskrivs av dess beskrivning genom att använda SOAP-meddelanden. Normalt skickas dessa med HTTP i XML-format i kombination med andra webbrelaterade standarder.*
> *Fritt översatt från (Haas & Brown, 2004)*

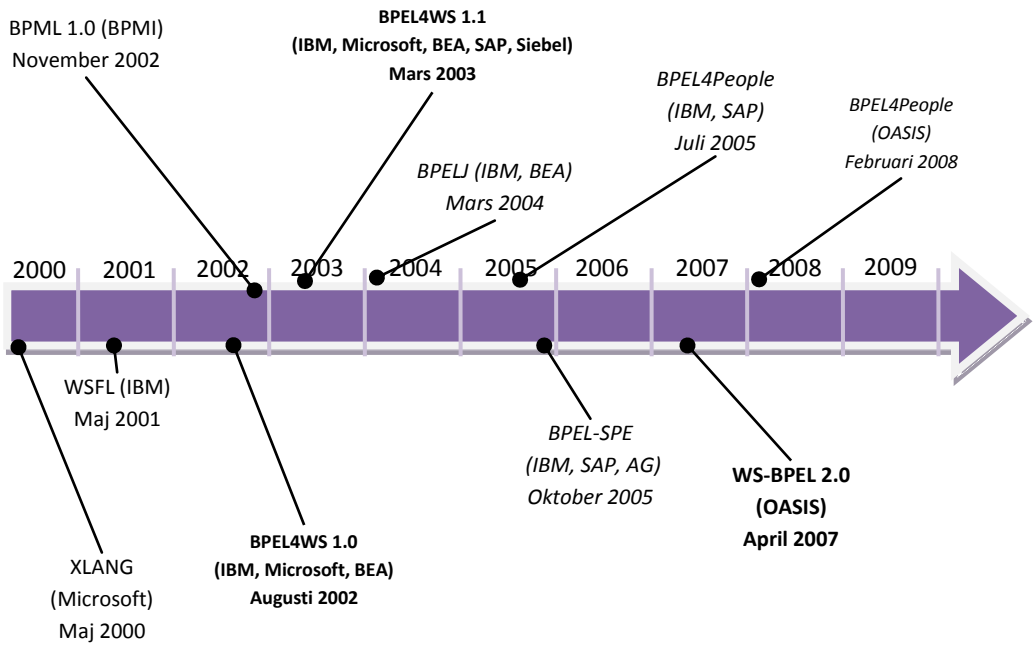
Definitionen innehåller egentligen de flesta komponenterna i SOA, **synliga gränssnitt** framförallt i WDSL, **meddelandeutbyte och interaktion** som definieras av SOAP-meddelanden. Interaktionen sker över nätverk med **lösa kopplingar**. Detta kan då användas för att skapa **reella effekter** genom att organisera web services utifrån affärsfunktioner. Standarder för web services utvecklas framför allt av W3C och Organization for the Advancement of Structured Information Standards (OASIS). Inom begreppet web services ryms en mängd protokoll. Utifrån samordning av web services med hjälp av BPEL kommer här översiktligt de teknologier som krävs, för att detta skall kunna göras på ett bra sätt, att beskrivas. Teknologierna sammanfattas ovan i figur 10.

Längst ner i figur 10 finns en Enterprise Service Bus (ESB) som inte är en web service teknologi men som skapar en infrastruktur för meddelandeutbyte och kommunikation mellan tjänster. Nästa lager, lager två består av SOAP, Simple Object Access Protocol, som specificerar strukturen för meddelandeutbytena. Tredje lagret i figuren (figur 10) är WSDL, Web Service Description Language som beskriva tjänsters gränssnitt på ett enhetligt sätt. Detta gör att själva tjänsten kan vara implementerad i vilket språk som helst exempelvis C# eller Java (lager fyra). Till vänster i figuren (figur 10) återfinns UDDI, Universal Description Discovery and Integration, som är ett register genom vilket web services och användare av dessa kan hitta varandra.

Till höger finns ett antal specifikationer för att säkerställa kvalitet och säkerhet, de så kallade kvalitetsattributen. Längst till vänster bland dessa återfinns WS-Security. Med hjälp av WS-Security, kan man skapa en miljö för säkra transporter. Därefter återfinns tre protokoll som hanterat koordineringar och transaktioner mellan tjänster, WS-Atomic Transaction, WS-Business Activity och WS-Coordination. WS-Coordination definierar ett skelett (framework) för koordinering mellan parter. WS-Atomic Transaction och WS-Business Activity specificerar båda hur distribuerade transaktioner skall gå till, WS-Atomic Transaction för transaktioner med kort varaktighet och WS-Business Activity för transaktioner med lång varaktighet. I nästa ruta finns WS-Reliable Messaging, som definierar en standard för pålitlig leverans av meddelanden mellan parterna. WS-Addressing är nästa protokoll och beskriver kommunikationens ändar. Det kan ses som ett komplement till WSDL beskrivningar av sådana slutstationer. De tre sista protokollen hjälper kommunikationen mellan tjänster. WS-Inspection hjälper kunder att hitta de tjänster den söker i exempelvis ett UDDI-register. WS-Policy hjälper tjänsterna att kommunicera sin interna policys till varandra, exempelvis kan en tjänst kräva en viss säkerhetstyp. WS-Evening finns till för att de involverade tjänsterna skall kunna registrera händelser som de är speciellt intresserade av i relationen. Managementrutan längst till höger implementeras genom ESBs management funktioner. Kvar högst upp i mitten finns BPEL för samordning av tjänster. I nästa stycke kommer vi beskriva BPEL närmare eftersom det är den teknologi som skall utvärderas i detta arbete.

3.2.3 Business Process Execution Language (BPEL)

Business Process Execution Language (BPEL), eller BPEL4WS (BPEL for Web Services) eller WS-BPEL som det numera formellt heter, är en standard framtagen av IBM, BEA och Microsoft för att implementera affärsprocesser i datorsystem. Numera handhas den av OASIS och är sedan år 2003 upptagen som OASIS-standard. Språket är byggt på XML och är skapat för att hantera samordning av web services. BPEL är alltså en standard för att implementera själva affärsprocessen. Vi skall först kort beskriva hur standarden växte fram och därefter gå igenom dess centrala begrepp.



Figur 11 - BPEL och andra språk för samordning av web services, en historisk översikt

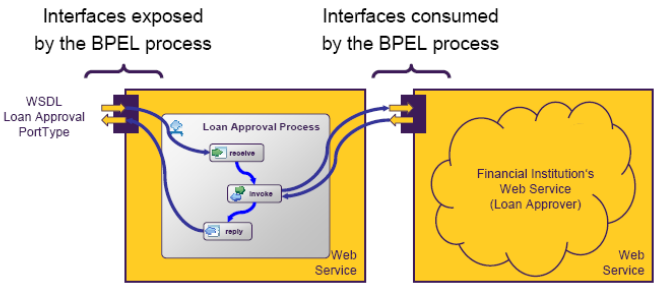
I augusti 2002 publicerar IBM, Microsoft och BEA den första versionen av BPEL, BPEL for Web Services (BPEL4WS) v 1.0 (Thatte, o.a., 2002). BPEL4WS utgör en sammanslagning av XLANG från Microsoft och Web Services Flow Language (WSFL) från IBM. Senare samma år publicerade BPMI en konkurrerad specifikation, Business Process Modelling Language (BPML) som var tänkt att utgöra en implementering av BPMN. Men eftersom IBM, Microsoft och BEA och senare även SAP och Siebel satsade på BPEL övergav BPMI BPML till förmån för BPEL och det arbete man gjort överlämnades till dem som arbetade med BPEL. I maj 2003 släptes BPEL4WS version 1.1 (Andrews, o.a., 2003) och efter det togs arbetet över av OASIS och Web Service Business Process Execution Language Technical Committee (WSBPEL TC) tillsattes (Juric, Mathew, & Sarang, 2006, s. 22). I april 2007 ansåg sig WSBPEL TC klara med sitt arbete och Web Service-Business Process Execution Language (WS-BPEL) v 2.0 (Jordan & Evdemon, 2007) släpptes. I version 2.0 finns en hel del förändringar gentemot tidigare version och versionen är inte heller bakåtkompatibel (Jordan & Evdemon, 2007).

När man började implementera BPEL märkte man snart att språket inte alltid var tillräckligt för att klara av alla uppgifter. Olika intressenter har därför föreslagit olika påbyggnader av BPEL. I mars 2004 föreslog BEA tillsammans med IBM, WS-BPEL extension for Java (BPELJ) (Blow, o.a., 2004), där BPELs styrka att programmera i ”det stora”, kombineras med Javas styrka att programmera i ”det lilla”. Tanken var att snuttar av javakod skall kunna bäddas in i BPEL-koden. BPELJ har ännu inte resulterat i något förslag till standard utan finns än så länge som idé (Juric, Mathew, & Sarang, 2006), men exempelvis har Oracle med tillägget i sina produkter.

I juli 2005 publicerade IBM och SAP whitepaperet *WS-BPEL Extension for People – BPEL4People* (Koppmann, o.a., 2005). Två år senare i juni 2007 publicerade Active Endpoints, Adobe, BEA, IBM, Oracle och SAP två specifikationer BPEL4People v 1.0 (Agrawal, o.a., 2007) och Web Service Human Task (WS-Human Task) v 1.0 (Amend, o.a., 2007) utifrån whitepaperet. BPEL4People utökar BPEL för interaktion med människor och manuellt utförande av olika tjänster (Human tasks). Själva beskrivningen av de mänskliga uppgifterna specificeras i WS-Human task. I februari 2008 tillsatte OASIS en teknisk kommitté, OASIS WS-BPEL Extension for People (BPEL4People) TC, för att arbeta med att standardisera tillägget. Alla större aktörer, även Microsoft som inte stod bakom själva specifikationerna är idag involverade i arbetet och man verkar vara angelägna om att BPEL4People skall bli en allmän standard (Gardner, 2009).

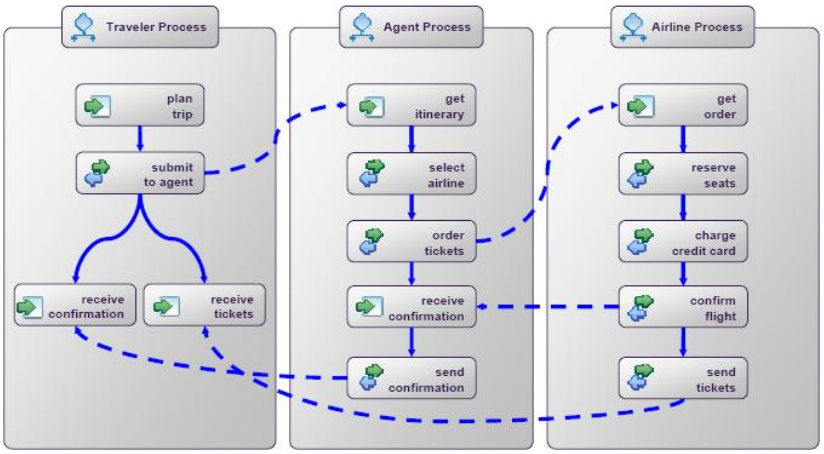
Under 2005, i september, publicerade också IBM tillsammans med SAP WS-BPEL extension for Sub-processes (BPEL-SPE) som whitepaper (König, o.a., 2005). BPEL-SPE låter processer vara subprocesser till andra processer. Enda sättet att uppnå detta beteende i WS-BPEL är att definiera en process (barn) som en service och sedan anropa den från en förälderprocess med <invoke>. Med BPEL-SPE kan man koppla ihop föräldrar och barn direkt och låta dem följa samma livscykel och dela kontext. BPEL-SPE har likt BPELJ ännu inte resulterat i något förslag till standard.

När vi nu har gått igenom historien bakom BPEL skall vi se på själva standarden och dess uppbyggnad. BPEL är alltså ett XML-baserat språk för främst orkestrering men även koreografering av web services. Språket saknar helt grafisk notation men lämpar sig egentligen ganska bra för att ha en sådan och i princip alla editorer har också grafiska utvecklingsverktyg för språket. Detta eftersom det är ett blockbaserat språk, vilket kommer från dess föregångare XLANG, och samtidigt innehåller händelser och kontrollänkar från WSFL som var just grafbaserat. Vid en första anblick verkar BPEL var ett strukturerat språk, som exempelvis Java eller C. Det är dock att betrakta som ostrukturerat pga. att det innehåller inslag från WSFL (Vigneras, 2008). Alla gränssnitt mot andra tjänster beskrivs i WSDL. Detta innebär egentligen två saker, dels att processen interagerar med web services genom gränssnitt beskrivna i WSDL och dels att processen själv beskriver sig genom ett WSDL-gränssnitt. Interaktionen är dock inte beroende av en viss port, som är vanligt för web services, utan endast av porttyp och ger på det sättet utrymme för en högre abstraktion. WS-BPEL 2.0 är kopplat till WSDL 1.1, men WSBPEL TC anser att det är kritiskt att WS-BPEL följer utvecklingen av WSDL och är kompatibelt med kommande versioner (Leymann, Roller, & Thatte, 2003).



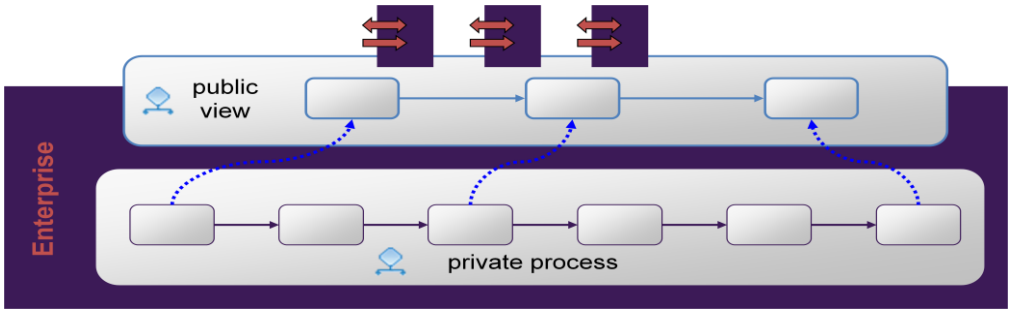
Figur 12 - Gränssnitt i BPEL (König, 2007)

BPEL är byggt både för att hantera processer från en intern och en extern synvinkel. I BPEL kan man implementera det exakta processflödet, vilket kallas för exekverbar process (Executable processes). De exekverbara processerna följer orkestreringsmönstret (se 3.2.1).



Figur 13 - Exempel på exekverbara processer (König, 2007)

Det går också att specificera affärslogiken och meddelandeutbytet mellan olika affärsprocesser utan att specificera det interna processflödet. Detta kallas för abstrakta processer (Abstract processes). Detta kallas också för affärsprotokoll (Business Protocol). Ett affärsprotokoll består egentligen av två delar, dels specificering av beteende mellan multipla partners och dels externt observerbara beteenden av en process. BPEL har inget formellt stöd för den första aspekten utan bara för den andra delen. Man kan alltså endast specificera en process externa behov utan att behöva ”avslöja” dess interna komponenter. Detta kan användas exempelvis mellan företag i en försörjningskedja. De abstrakta processerna följer följaktligen ett koreograferingsmönster (se 3.2.1) men är alltså inte komplett varför det inte kan matchas mot koreografin i BPMN. I appendix A hittar du en mer detaljerad genomgång av BPELs struktur och element.



Figur 14 – Exempel på abstrakt process (König, 2007)

Det finns ett antal verktyg som implementerar BPEL, dels BPEL-serverprogramvara och dels av BPEL-editorer. En sammanställning av dessa finns i appendix F. Lista över serverprogramvaror utgår från (Wikipedia, 2009) med flera tillägg och uppdateringar. Produktdata är hämtade från respektive tillverkares hemsida. Totalt hittades 14 serverprogramvaror och 10 BPEL-editorer.

3.3 BPMN och BPEL - Gapet mellan IT och affärer

Kopplingen mellan verksamhet och IT har i takt med affärssystemens utveckling blivit ett allt större problem- och diskussionsområde. Ofta talas det om ”the IT-Business-gap”, gapet mellan IT och affärer. Problemet är att de som förstår affärerna inte förstår systemen och vice versa. Det finns alltså ett allt större behov av att koppla ihop verktyg för beskrivning av affärsprocesser och implementeringen av dessa affärsprocesser i IT-system. Att koppla ihop BPMN och BPEL är ett försök att göra just detta. I specifikationen till BPMN (v 1.0–1.2) finns ett appendix som föreslår en sådan mappning mellan BPMN (beskrivning av affärsprocesserna) och BPEL (implementeringen av affärsprocesserna) . I kommande v 2.0 av BPMN har mappningen blivit en del av själva specifikationen. Översättningen i samtliga versioner är dock bristfällig och begränsad vilket också konstateras i specifikationerna till BPMN v 1.0–1.2. Detta beror främst på att BPMN inte i sin schematik är speciellt strikt, något som BPEL är eftersom det krävs hos ett exekverbart språk. Främst finns det möjligheter att i BPMN skapa konstruktioner som ger upphov till deadlocks[1] och livelocks[2]. En mer komplett mappning har föreslagits av (Ouyang, Dumas, van der Aalst, & ter Hofstede, 2006). Denna mappning har dock ett stort problem och det är att koden som genereras är svårläslig. I (Ouyang, Dumas, van der Aalst, & ter Hofstede, 2008) föreslår de därför en kompletterande metod för att skapa bättre kod. Denna metod bygger på igenkänning av mönster. Författarna menar att denna metod skall vara den huvudsakliga och att den tidigare metoden skall användas i de fall som den senare inte täcker.

Ett problem som inte (Ouyang, Dumas, van der Aalst, & ter Hofstede, 2008) eller specifikationerna av BPEL och BPMN tar upp är kopplingen tillbaka från BPEL till BPMN. För att standarderna skall bli riktigt användbara behöver visualisering av BPEL-kod i BPMN också vara möjlig. (Schumm, Karastoyanova, Leymann, & Nitzsche, 2009) föreslår en metod som bygger på att varje objekt i BPEL mappas till ett objekt i BPMN. Eftersom BPEL har en strikt schematik menar författarna att denna approach genererar ett bra resultat.

Det har från flera håll lagts mycket tid och resurser på skapa en mappning mellan BPMN och BPEL. Många debattörer, forskare och andra inblandade intressenter ifrågasätter dock om standarderna alls är tillräckligt kompatibla för att en bra mappning faktiskt skall vara möjlig. En av dessa är (Dubray, 2007). Han summerar de flesta invändningar som finns mot mappningen i sju punkter som han kallar sju villfarelser eller osanningar som finns kring exekverbara affärsprocesser.

Den första villfarelsen är *”Business analysts model their processes from a systems' point of view.”* Med detta menar Dubray att ett problem som alltid kommer finnas är att affärsanalytiker inte tänker i termer av exekvering utan bara följer affärslogik. Något som gör att affärsprocesserna de bygger i BPMN inte utgör en bra grund för program. Andra som (Juric & Pant, 2008) argumenterar att detta kan överbyggas genom att programmerare och affärsanalytiker gör kartor tillsammans.

En annan villfarelse handlar om BPMN; ”*Business users can easily learn BPMN and use all its features.”* Affärsanvändare kan lätt lära sig BPMN och använda alla dess finesser. Han menar att

[1] Deadlock är ett tillstånd där två händelser båda väntar på att den andra skall bli färdig och ett dödläge skapas (WebMediaBrands Inc., 2009).

[2] Livelock är ett tillstånd då två processer hela tiden byter tillstånd som reaktion på varandra. Resultatet blir att ingen av dem blir klar. Det kan liknas med två personer som möts i en korridor och försöker gå förbi varandra men hamnar i ett läge där de går från fot till fot och blockerar varandra (WebMediaBrands Inc., 2009).

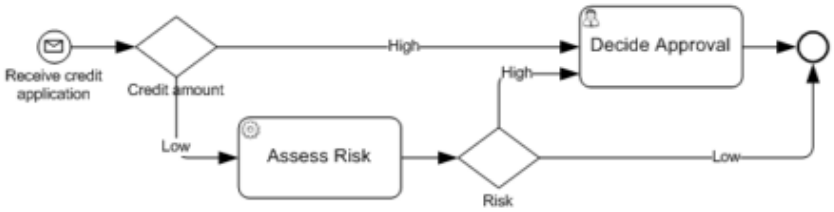
många finesser i BPMN bara finns till för att göra det kompatibelt med BPEL och att dessa ofta ignoreras. Dubray hänvisar exempelvis till (zur Muelhlen, 2007) som i en studie kommit fram till att det i princip bara är 25 BPMN-konstrukter som används.

Två villfarelser handlar om att affärsanalytiker skall kunna ta över byggandet av tjänster,"*Business analysts should be able to create executable solutions from process models."* och *If we add a magical BPMS that create solutions directly from business analysts inputs we would not need to develop any of integration with existing systems nor to change existing systems of record nor to do any QA.* Att affärsanalytiker skulle kunna producera exekverbara lösningar från processmodeller är bara en dröm, argumenterar Dubray. Han menar att ingen tillverkare idag kan presentera en sådan lösning. Ingen affärsanalytiker skulle heller vilja skriva "*något som ens liknar ett Xpath-uttryck*".

Dubray menar också att det finns villfarelser om att "*Business Process Execution must be centralized".* Han argumenterar för att centraliserade affärsprocessexekveringar (orkestreringar) egentligen strider mot processens natur som handlar om att *"tillhandahålla en värdeadderande organisering genom transformation av resurser"*.

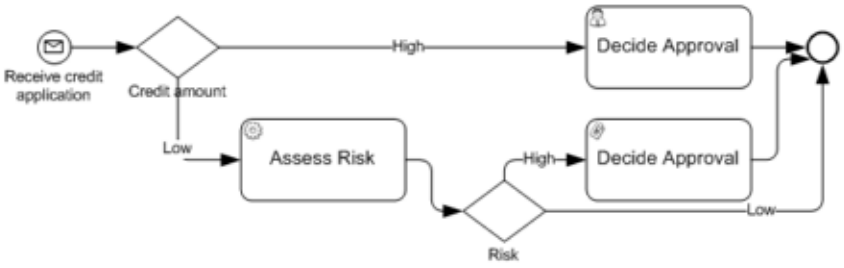
Den näst sista villfarelsen handlar om schematik, "*Business Process Execution semantics can be derived easily from existing programming concepts."* Dubray menar att de nuvarande programspråken för affärsexekvering är byggda av systemarkitekter och programmerare och baserade på enkla användarscenarion. Det är inte helt trivialt att bygga schematik för mer komplicerade affärsflöden som innehåller större abstraktion.

I den sista villfarelsen menar Dubray att det inte bara går att implementera affärsprocesser direkt från affärsmodeller eftersom han menar att vi inte kan modellera arbetsflödet och resurser livscykel tillsammans. Dubrays slutsats är att vi idag har tekniken men att vi använder den på fel sätt, han tycker sig dock se en förändring i detta. Dubrays kritik handlar huvudsakligen om gapet mellan IT och affärer i allmänhet. Andra har fokuserat på svårigheter med just kopplingen mellan BPMN och BPEL. Vambenepe, 2008 drar slutsatsen att automatisk översättning tenderar att överanvändas. Han visar med exempel på dålig kod som han genererat med Oracles Business Process Architect, när han försöker översätta en OR-split och join. Silver, 2008 diskuterar (Vambenepe, 2008) och menar istället att de två största problemen är *"Interleaved flows"* och "*attached events"*. Med *interleaved flows* eller interfolierade flöden menar han mellanliggande flöden. I figur 15 finns en enkel process för hantering av lån, som ett exempel på detta.



Figur 15 - Exempel på interfolierade flöden i BPMN (Silver, 2008)

Problemet är att det går två flödespilar till *Decide Approval.* Detta kan inte hanteras utan måste byggas som i figur 16, där *Decide Approval* måste dupliceras.



Figur 16 - Exempel på hur man löser problemet med interfolierade flöden för översättning från BPMN till BPEL (Silver, 2008)

Med *attached events* menar Silver, 2008, händelser som avbryter ett flöde och omdirigerar till ett undantagsflöde. I BPMN kan denna omdirigering göras valfritt inom poolen eller subprocessen. I BPEL måste undantagsflödet alltid återvända till slutet av händelsen för undantaget.

Det finns flera återstående problem med översättningen från BPMN till BPEL, där split, join, interfolierade aktiviteter och attached events är de största problemen. Andra som (Vigneras, 2008) har också pointerat dessa svårigheter. Dessutom finns det som vi tidigare konstaterat lite forskning på översättning åt andra hållet. Dock är kanske den största begränsningen, som man ofta inte upptäcker förrän man använder verktygen och som Silver tar upp i ett svar till (Silver, 2008), att översättningen i olika verktyg oftast bara kan användas för att översätta det övergripande flödet och att det mesta ändå måste implementeras manuellt. Det finns i vilket fall som helst ett fåtal verktyg där man göra konvertering från BPMN till BPEL Det finns en sammanställning av dessa i appendix F. Totalt hittades fem verktyg.

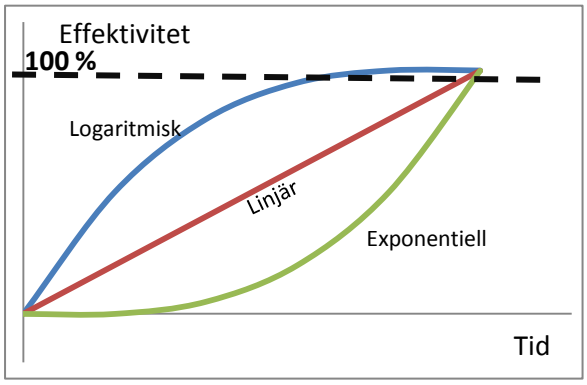
Vilka slutsatser kan man då dra utifrån den litteratur som sammanställts i detta kapitel? Det vi kan konstatera är att SOA och BPM är idéer som gör att vi kan organisera vår verksamhet respektive våra system på ett flexibelt och kostnadseffektivt sätt. Vi kan också konstatera att web services är ett bra sätt att implementera SOA och att BPEL därför har en central roll eftersom den står för själva samordningen av web services. Vi anar också att både BPMN och BPEL fortfarande har en hel del problem och inte riktigt har mognat. De har dock båda brett stöd i industrin. Antalet produkter är än så länge relativt få, även om BPMN är implementerad i betydligt fler. Att kunna koppla kartläggning till implementering är centralt för att överbygga gapet mellan affärer och IT. Vi kan konstatera att möjligheter finns att göra detta med BPMN och BPEL. Fortfarande finns det dock flera problem med översättningen, både i hur översättningen skall göras och hur vi nyttjar tekniken. Dessa slutsatser kommer i nästa kapitel att kompletteras och ifrågasättas genom att vi använder tekniken. Innan vi går vidare och beskriver den implementering som gjordes skall vi i det sista avsnittet i detta kapitel presentera tre övergripande modeller som behövs i vår slutanalys av teknologierna.

3.4 Analysmodeller

I detta avslutande avsnitt i teorikapitlet beskrivs tre modeller som vi behöver för att analysera vårt empiriska resultat, inlärningskurvan, produktlivscykeln och innovationsadoptionsmodellen.

3.4.1 Inlärningskurva

Att lära sig är en process. Ju bättre man blir på något desto effektivare kan man utföra detta. Detta får i ett företag ekonomiska konsekvenser eftersom effektiviteten hos de som arbetar ökar med antalet upprepningar. En erfaren arbetare är således bättre än en ny. Den förste som uppmärksammade detta var (Wright, 1936), inom flygplansindustrin. Denna process brukar kallas inlärningsprocessen och kan beskrivas med en kurva som brukar kallas inlärningskurvan. Inlärningskurvan beskriver sambandet mellan effektivitet och nedlagd tid. Kurvan kan ha olika utseenden den kan vara logaritmisk, linjär eller exponentiell. Ett logaritmiskt utseende innebär att vi redan efter kort tid uppnår en effektivitet som är tillräcklig för att vi skall kunna producera. Ett exponentiellt utseende innebär motsatsen, dvs. att vi först måste lägga ner mycket tid på inlärning innan vi uppnår en effektivitet där vi kan börja producera. Idéalt är alltså en logaritmisk kurva med brant lutning.

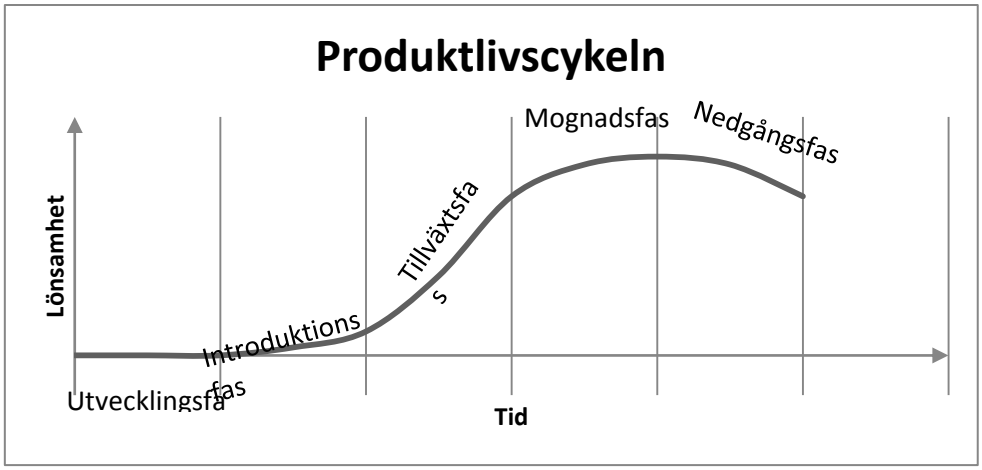


Figur 17 - Exempel på inlärningskurvor

Inlärningskurvan kommer vi att använda för att diskutera den kunskapsanskaffning som en investering i BPEL med BPMN innebär.

3.4.2 Produktlivscykeln

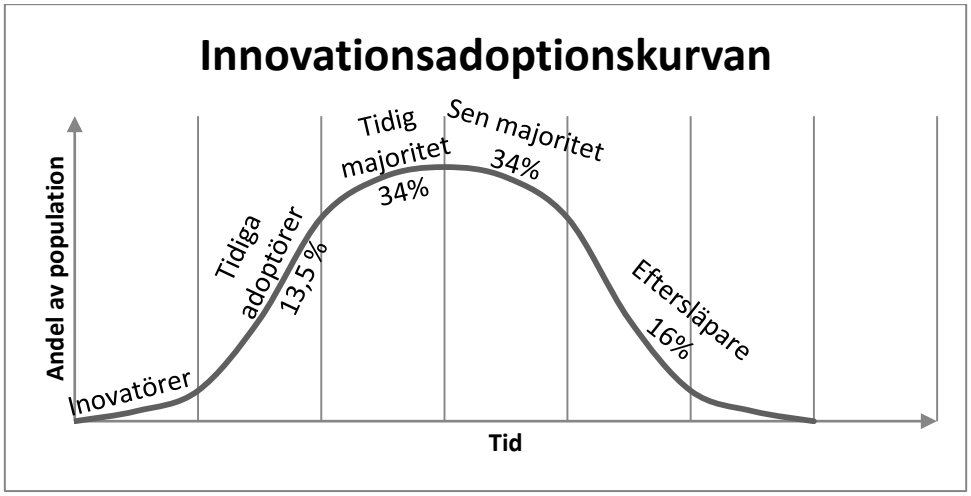
Produktlivscykeln är en modell som beskriver en produkt eller teknologis liv i förhållande till produktens eller teknologins lönsamhet. Modellen har sitt ursprung i en artikel i Harvard Business Review, vol. 43 från år 1965, *Exploit the product life cycle*, av Theodore Levitt (Levitt, 1965). Idén är att produkten föds, växer, når sin topp och tillsist dör. Cykeln är indelad i fem stadier eller faser. Den första fasen är utvecklingsfasen då produkten eller teknologin utvecklas och lönsamheten är negativ. När produkten är färdigutvecklad introduceras den på ”marknaden”, lönsamheten är låg eftersom det finns få köpare och kostnader för att marknadsföra den är stora. I det tredje stadiet tillväxt- eller utvecklingsstadiet ökar lönsamheten snabbt, antalet intressenter ökar eftersom teknologin eller produkten har blivit känd. När tillväxten börjar avta och efterfrågetoppen kommer närmare går produkten in i mognadsfasen. Detta är produktens mest lönsamma fas, med stor efterfrågan. När efterfrågan och lönsamheten börjar avta går produkten in i nedgångsfasen. Slutligen avvecklas teknologin eller produkten och ersätts med någon annan teknologi eller dör helt enkelt eftersom behovet inte längre finns på ”marknaden”. Utifrån modellen har olika strategier för de olika stadierna utvecklats. Beroende på hur produktägarna lyckas med olika strategier blir de olika faserna olika långa och lönsamma både i förhållande till varandra och totalt. Detta beror också på hur marknaden ser ut, modekläder har exempelvis en mycket kortare livscykel än flygplan. Den allmänna utvecklingen i alla branscher är att cyklerna blivit kortare och kortare för varje år. (Levitt, 2006)



Figur 18 - Produktlivscykeln

3.4.3 Innovationsadoptionskurvan

Konceptet bakom innovationsadoptionskurvan publicerades av (Bohlen & Beal, 1957) och generaliserades senare av (Rogers, 2003). Kurvan hänger ihop med produktlivscykeln och förklarar den delvis. Innovationsadoptionskurvan beskriver hur en population tar åt sig ny teknik eller nya trender. Kurvan beskriver andelen av populationen som adopterar tekniken vid en viss tid och populationen är indelad i fem grupper analogt med produktlivscykeln. De som först adopterar tekniken är innovatörerna, dvs. de som utvecklar tekniken, dessa är relativt få. Tidiga adoptörer är nästa grupp, vanligtvis ”prylnissar” eller ”trendsättare” och utgör en mindre del av befolkningen. Den stora majoriteten adopterar trenden eller teknologin då den börjar mogna. Den tidiga majoriteten är något mer mode eller teknik progressiva och adopterar trenden när den är på uppgång (tillväxtfasen se 3.4.2), medan den sena majoriteten är mer konservativa och väntar tills trenden är riktigt mogen (mognadsfasen se 3.4.2). Den sista gruppen är eftersläparna som inte adopterar tekniken förrän den är på utgång. Modellen kan användas för att analysera kunder för en viss produkt eller teknologi och för att avgöra vilken strategi man i ett visst skede skall använda sig av för att nå rätt målgrupp.



Figur 19 – Innovationsadoptionskurvan

Med hjälp av produktlivscykeln och innovationsadoptionskurvan kommer vi att diskutera BPEL- och BPMN-teknologiernas marknadsutveckling, var de idag befinner sig och hur detta påverkar vårt investeringsbeslut.

4. Empiri

I detta kapitel beskrivs det praktiska arbete som utgör huvuddelen av denna studie. Kapitlet beskriver affärsscenariot som togs fram och implementerades och de olika arbetsstegen, samt de resultat som arbetet utmynnade i. Med hjälp av denna praktiska implementering undersöks och värderas de teoretiska slutsatserna i kapitel 3. Kapitlet beskriver också de överväganden som gjordes vid val av implementeringsmiljö och konstruktionen av affärsscenariot.

Kapitlet är uppdelat i 7 delar. Den första handlar om implementeringsmiljön. Del två handlar om scenariot och varför det ser ut som det gör. Del tre beskriver kartläggningsarbetet och i fjärde delen beskrivs hur krav- och testspecifikationerna togs fram samt hur affärsmiljön specificerades. I del fem beskrivs själva implementeringen i BPEL och Java. I den sjätte delen beskrivs resultatet och i sjunde och sista delen beskrivs de kompletterande studier som gjordes.

4.1 Implementeringsmiljö

I detta avsnitt beskrivs val av verktyg för utvecklings- och servermiljöer för kartläggning och programvaruutveckling. Utgångspunkten är den sammanställning av verktyg som återfinns i 3.3.

För att kunna utvärdera BPMN/BPEL praktiskt behövs en implementeringsmiljö som har stöd för både BPEL och BPMN samt konvertering dessa emellan. Detta eftersom en av de stora fördelarna med BPMN/BPEL just skall vara möjligheten att konvertera. Dessutom är det bra om miljön har stöd för Java eftersom det gör att utvärderingen som görs inte blir beroende av skillnader vad gäller miljö. Dels behövs en utvecklingsmiljö och dels en severmiljö. Med tanke på tid och resurser är det bra om det är en miljö som är enkel att få tag på och som är gratis att använda eller åtminstone prova under ett par månader.

Det naturliga valet är att använda Eclipse som editor tillsammans med det BPMN till BPEL plugin (http://code.google.com/p/bpmn2bpel/) som korresponderar med metoden som beskrivs i Ouyang, Dumas, van der Aalst, & ter Hofstede, 2008 som är den senaste forskningen på området (se 3.3.3). Pluginet bygger dock på ett antal andra plugin och vissa specifika versioner av dessa. Efter att noga ha följt installationsinstruktionerna (Igbanuelos, 2008) och dessutom provat olika varianter i både Windows XP och Windows Vista utan framgång avskrevs Eclipse som editoralternativ. Istället undersöktes andra möjligheter.

Den första möjligheten som undersöktes var att använda Oracles produkter, Oracle BPA Suite för BPMN och konvertering till BPEL, Jdeveloper för BPEL- och Javautveckling och Oracle SOA Suite som servermiljö. De senaste versionerna var vid tillfället Oracle SOA Suite 10.1.3.1.0 respektive Jdeveloper Studio Edition 10.1.3.4.0.4270 samt Oracle BPA Suite 10.1.3.4. Instruktioner för installation av BPEL för Oracle SOA Suite och Jdeveloper återfinns i appendix C, BPA Suite installeras enkelt genom att följa instruktioner i installationen. Det finns dock en stor nackdel med Oraclemiljön och det är att Oracle ännu inte utvecklat sin miljö för BPEL v 2.0. Alla verktygen relaterar till BPEL v 1.1 med egna utökningar (varav många finns med i BPEL v 2.0). Eftersom det är mest intressant att prova den senaste versionen av BPEL (skillnaden är ganska stor mellan v 1.1 och v 2.0 och dessutom är v 2.0 inte bakåtkompatibelt), undersöktes andra möjligheter. En förhoppning var att Oracle i sitt

paket Oracle Fusion Middleware 11g, med SOA Suite 11g samt BPEL för Jdeveloper 11g, som båda släpptes under sommaren, skulle ha uppdaterat till v 2.0 av BPEL. Så var dock inte fallet. Vissa förbättringar hade gjorts och det var egentligen att föredra att arbeta i Jdeveloper 11g, som släpptes tidigt på sommaren, istället för Jdeveloper v 10.1.3.4. Problemet var att Jdeveloper 11g inte var kompatibelt med SOA Suite 10.1.3.1.0 utan endast med SOA Suite 11g som inte släpptes förrän ganska långt in i implementeringsfasen och dessutom var för resurskrävande för att på ett bra sätt fungera på den tillgängliga utrustningen.

Sun erbjuder en integrerad utvecklings och servermiljö för BPEL v 2.0, Netbeans tillsammans med en Glassfishserver. Något verktyg eller plugin för BPMN eller konvertering mellan BPMN och BPEL fanns dock inte. Dessutom visade sig Sunmiljön snart ha stora problem eftersom den inte var färdigutvecklad. Försök att göra implementeringen i Sunmiljön gjordes i ca en vecka. Under försöket stöttes flera kända buggar på, exempelvis att en BPEL-process som konsumerar en annan slutar fungera om man ändrar den konsumerande processen efter att man ”deployat” eller diverse ”null pointer exceptions” i BPEL-kompilatorn. Detta alternativ fick till slut avskrivas eftersom processerna helt slutade fungera och måste göras om från scratch gång på gång.

En produkt som kan vara värd att nämna, men som av tidsmässiga skäl inte provades, är Intalios Business Process Management plattform. Den är gratis att prova under ett år och har till skillnad från de flesta andra verktyg valt att bygga ett gränssnitt i BPMN som man utvecklar i och som sedan körs som BPEL-processer på serverprogramvaran.

Valet föll tillslut på Oraclemiljö trots att den inte implementerade BPEL v 2.0. Kartläggningen och konverteringen från BPMN till BPEL gjordes i Oracle BPA Suite 10.1.3.4. Som servermiljö valdes Oracle SOA Suite 10.1.3.1.0 och utvecklingen i BPEL gjordes Jdeveloper Studio Edition 10.1.3.4.0.4270. Utvecklingen i Java kunde göras i den nyare versionen Jdeveloper Studio Edition 11.1.1.1.0.

4.2 Scenario

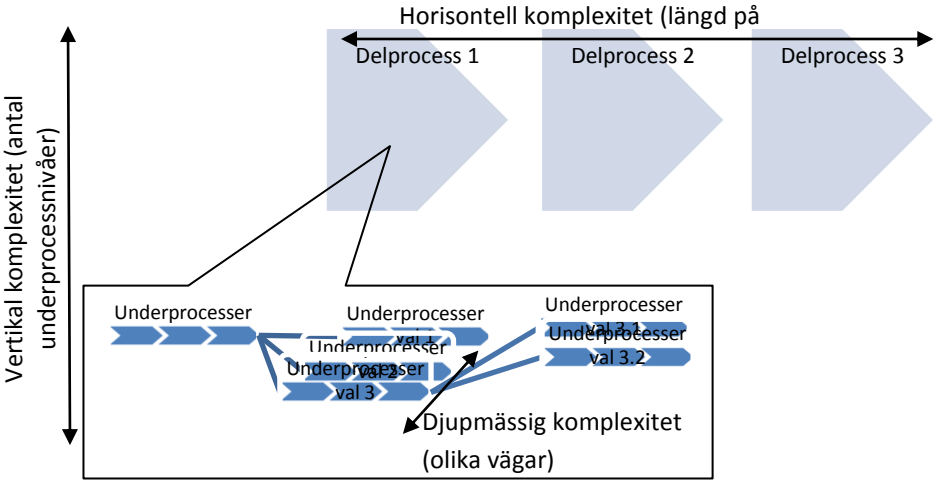
I detta avsnitt beskrivs det affärsscenario som skapats för att utvärdera BPEL och BPMN. I avsnittet diskuteras också bakgrunden till att scenariot ser ut som det gör.

4.2.1 Konstruktion av scenario

För att kunna utvärdera BPEL och BPMN på ett så heltäckande sätt som möjligt har en metod valts där både teoretiska och praktiska studier genomförts (mer om detta kan du läsa i kapitel 2). Efter den inledande fasen, litteraturstudierna (se kapitel 3), inleddes andra fasen, implementering, med att ett scenario för implementering konstruerades. Scenariots konstruktion avgör vilka aspekter som kan och vilka aspekter som inte kan utvärderas. Självklart vill man fånga så många aspekter som möjligt, men med tanke på att projektet är begränsat tidsmässigt gjordes en värdering av vilka aspekter som var mest intressanta och realistiska att fånga. Värderingen gjordes med ledning av litteraturstudierna.

Det första kravet på scenariot hittar man redan i namnen BPEL och BPMN, nämligen att scenariot skall vara en affärsprocess, *business process*. Det framhålls att det är när komplexiteten är stor som BPEL och BPMN ger störst fördelar. Därför bör affärsprocessen vara så komplex som möjligt. Det innebär att den bör ha så många processteg som möjligt (horisontell komplexitet), att den är så pass

omfattande att det krävs att processerna struktureras hierarkiskt i nivåer (vertikal komplexitet)och att den har många olika val och vägar (”djupmässig” komplexitet).



Figur 20 - Dimensioner av komplexitet

Eftersom BPEL är byggt för att samordna web services är det alltså naturligt att scenariot skall innehålla ett antal tjänster i form av web services som skall samordnas. En grund för bra hantering och samordning är att de interna strukturerna och datamanipulationerna i själva programmet fungerar bra. Därför bör scenariot även vara byggt så att olika vanliga strukturer som loopar och val prövas. Scenariot bör också vara sådant att implementeringen behöver ha en bredd i fråga om olika dataobjekt. Olika manipulationer bör göras både med ”enkla” objekt, som strängar och tal, och med listor och andra objekt som består av samlingar av ”enkla” objekt. Det är också bra om BPELs förmåga att hantera fel prövas, och därför bör även felhantering inkluderas i uppgiften.

Det är också viktigt att scenariot är så verklighetstroget som möjligt, eftersom inte bara tekniska utan även arbetsmässiga aspekter skall utvärderas. Då de arbetsmässiga aspekterna relaterar till ”verkligheten” är detta alltså centralt. Ett verklighetstroget scenario gör det också lättare att relatera till, diskutera och presentera och är därmed mer relevant för någon som är intresserad av tekniken.

En aspekt som framhålls som en fördel med BPEL- och BPMN-tekniken är att det är lätt att göra förändringar av den implementerade processen. Utifrån detta är det lämpligt att scenariots krav förändras över tid, så att ändringar behöver göras. Det finns dock två problem med detta som gör att denna aspekt inte ingår i scenariot. Det första och största problemet är att implementeringsfasen är begränsad tidsmässigt vilket innebär att tiden för att implementera och testa två program samt att sedan ändra och testa dessa igen är knapp. Det skulle innebära att programmen måste vara relativt enkla om det alls skall gå. Det skulle betyda att det ändå inte skulle testa förändringar särskilt bra eftersom enkla program är enkla att ändra och dessutom skulle många andra aspekter få stryka på foten. Det andra problemet är att scenariot skapas och implementeras av samma person. Det innebär att de tänkta ändringarna inte kan skapas innan implementeringen gjorts eftersom det då är troligt att implementeringen blir påverkad av att dessa ändringar skall göras. Skapas ändringarna efter att den första implementeringen gjorts kommer de troligtvis att påverkas av hur implementeringen gjorts. Ändringarna kan visserligen skrivas av någon annan, för att undvika detta. Det kräver dock att det finns sådana resurser. Således faller denna aspekt utanför scenariot p.g.a. framförallt tidsmässiga men också genomförandemässiga orsaker.

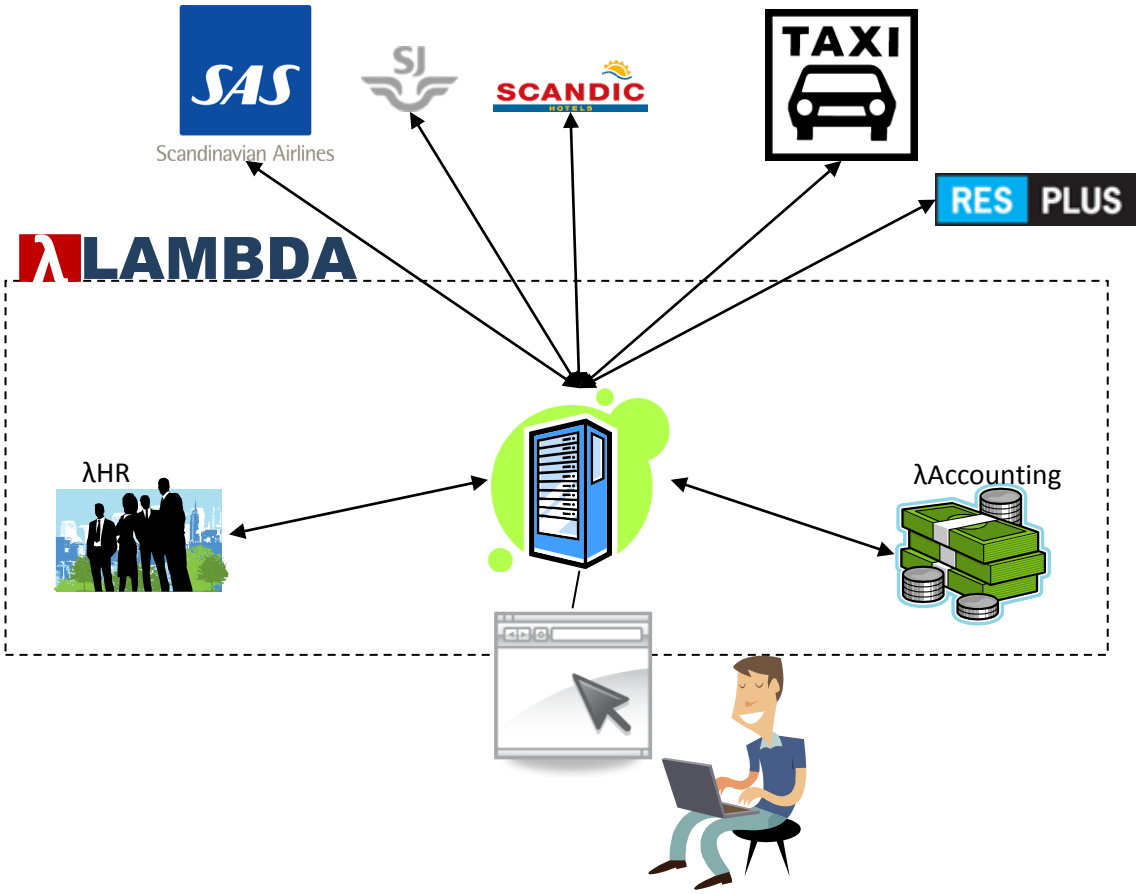
En annan begränsning som gjordes redan tidigt är att i minsta möjliga mån lägga tid på ett grafiskt användargränssnitt. Detta innebär att inget sådant gränssnitt inkluderas i scenariot utan inmatning och utmatning representeras bara av enkla objekt som går in i, och ut ur processerna.

Med utgångspunkt i dessa aspekter och med inspiration från exempel i (Juric, Mathew, & Sarang, Business Process Execution Language for Web Services, 2006) skapades ett scenario som handlar om tjänsteresor. Uppgiften blev att skapa ett system som tar fram och bokar tjänsteresor för det fiktiva företaget Lambda Omikron. Exemplet fick inkludera tre reseslag: tåg, flyg och taxi. För att göra det verklighetstroget är tjänsterna för dessa inte helt identiskt implementerade. Scenariot inkluderar också en tjänst för att hitta resor med kombinerade resslag i stil med ResPlus. Det innebär att en relativt komplex struktur för att hitta och boka den bästa resan måste skapas, en process i flera steg och med olika underprocesser beroende på hur varje resförslag ser ut. För att göra processen något längre finns dessutom möjlighet att boka hotell på resans mål. För att ytterligare bygga ut exemplet en dimension och också göra det mer realistiskt relaterar tjänsten till andra interna system på Lambda Omikron, dels ett personalsystem med olika personalkategorier och dels till ett bokföringssystem. Till varje personalkategori finns olika affärsregler kopplade i tjänsten, bokföringssystemet finns mest för att det i en verklig situation borde vara smart att koppla en sådan här tjänst till det. En process för hur bokningen skall gå till måste följaktligen kartläggas, en relativt komplex process måste implementeras som dels kräver att många tjänster tas i anspråk och orkestreras och som dels innehåller en struktur som kräver att olika struktur- och dataobjekt används och manipuleras.

Scenariot är slutligen nedskrivet i berättelseform och återfinns nedan (4.2.2). Utifrån scenariot har först en processkartläggning gjorts. Utifrån kartläggningen och scenariobeskrivningen har ett kravdokument tagits fram. Kraven har skrivits med utgångspunkt från den ovanstående diskussionen för att aspekter så tydligt som möjligt skall kunna utvärderas då dessa av naturliga skäl inte ingår i affärsscenariot. Utifrån kraven har en testspecifikation gjorts och därefter har implementering och test genomförts.

4.2.2 Scenariots slutgiltiga utseende

Inom Lambdakoncernen reser anställda ofta mellan kontor och kunder som finns på olika ställen i landet. Det finns behov av ett internt system som hanterar anställdas resor så att de hanteras på ett enhetligt sätt. En testversion för Lambda Omikron, skall byggas. Systemet skall interagera med Lambda Omikrons interna system för hantering av anställda, λHR, och dess ekonomisystem, λAcconting. Vidare har man inom Lambda upphandlat tjänsteresor och övernattningar inom Sverige med olika externa företag. Detta scenario inkluderar fiktiva upphandlingar med en aktör för varje transportslag SAS (flyg), SJ (tåg) och lokala taxibolag på respektive ort, genom det fiktiva taxisamarbetet TaxiSverige samt Scandic hotell för övernattning. Samtliga företag har i scenariot system i vilka man direkt kan boka biljetter. Resebolagen har dessutom tillsammans det så kallade Resplussystemet som tillhandahåller reserutter med kombinerade fordonsslag och som liknar det riktiga Resplus. Till denna tjänst kan man ställa frågor om resor mellan två platser vid viss tid och få tillbaka ruttförslag. Samtliga interagerande system använder web servicegränssnitt för att kommunicera med andra system. Tjänsten skall inte bara spara pengar indirekt utan även direkt, därför skall konsulter alltid resa så billigt som möjligt. Chefer har dock ett mer pressat schema varför de alltid skall få den resa som erbjuder kortast restid. Övriga skall inte ha någon behörighet att själva boka resor. Både konsulter och chefer bor billigast möjligt på Scandic Hotels.

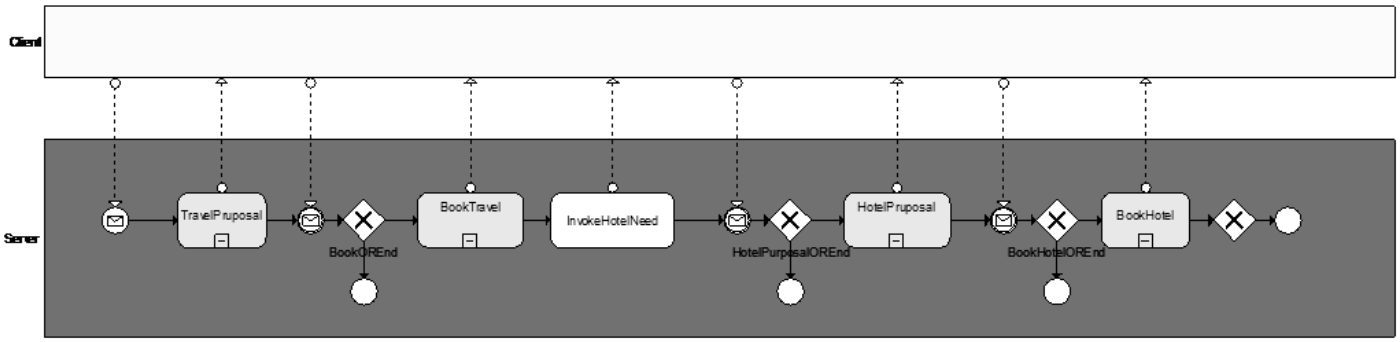


Figur 21 – λTraveler, översikt av affärsmiljön

4.3 Kartläggning

I detta avsnitt beskrivs processkartläggningen av scenariot som utgör grunden för implementeringen.

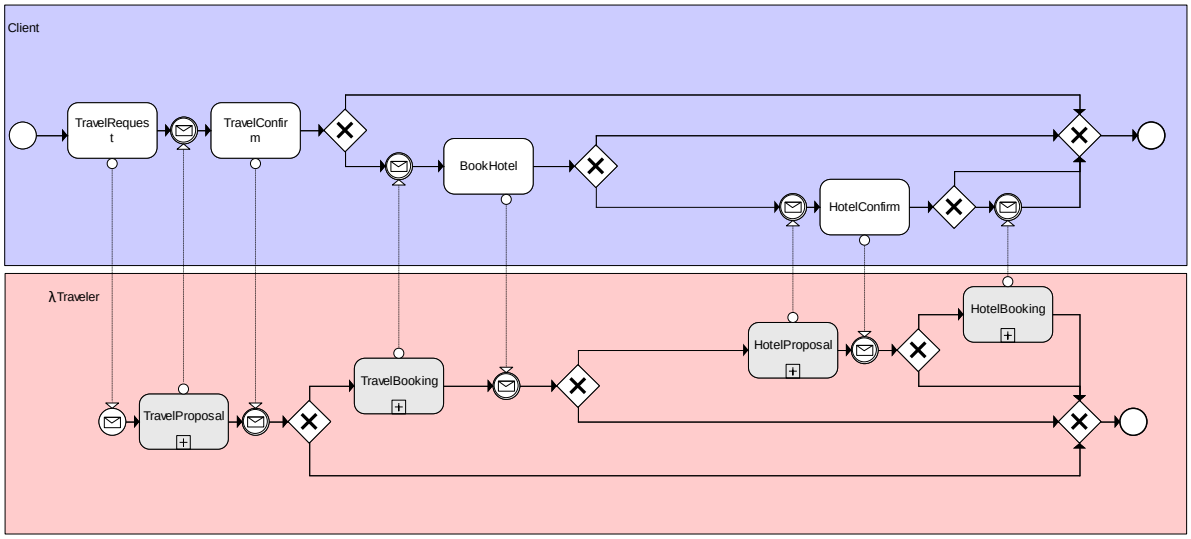
Utifrån det beskrivna scenariot identifierades ett antal delprocesser som tillsammans utgör den övergripande processkedjan. De fyra stegen är framtagning av reseförslag, bokning av resa, framtagning av hotellförslag och hotellbokning. Delprocess tre och fyra är bara aktuella i det fall klienten är intresserad av att boka hotell.



Figur 22 – Processbeskrivning, linjär process, BPMN gjord i Oracle BPA

Processen initieras med att klienten efterfrågar en resa. ”Objekt in” i först delprocessen är således förfrågan. Den första delprocessen tar fram ett reseförslag och skickar detta som objekt ut. Därefter kan klienten välja att boka eller avsluta. Väljer klienten att boka så startas hotellbokningsprocessen

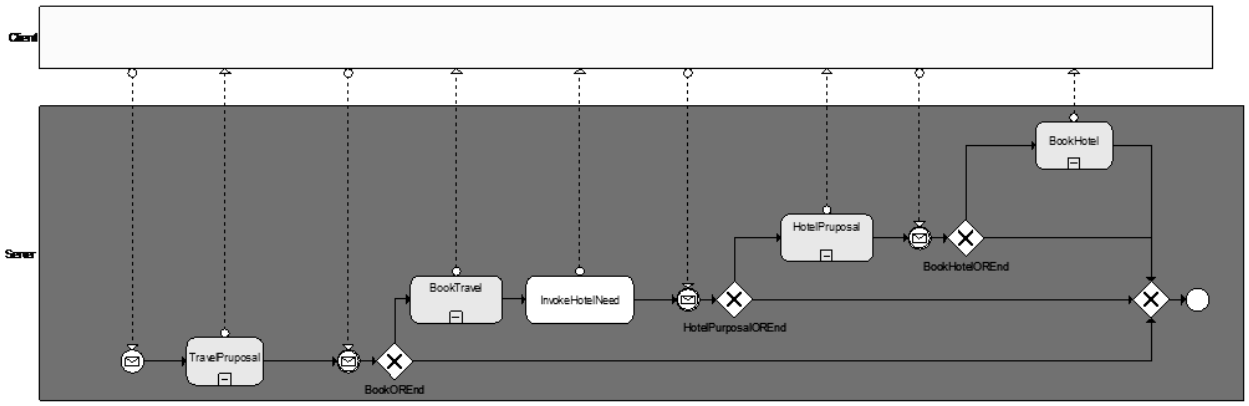
och en bekräftelse på bokningen retuneras till kunden. I första varianten (se figur 23) bestäms om kunden vill boka hotell redan i början av processen. Detta beslut flyttades sedan till efter bokningen eftersom det var naturligare (se figur 24). På samma sätt som med resan tas ett hotellförslag fram och beroende på klienten bokas det eller processen avslutas.



Figur 23 – Processbeskrivning, första varianten, med både klientens och programmets processer, BPMN gjord i Oracle BPA

Värt att notera är att det redan i första versionen (se figur 23) tagits fasta på att bara ha ett end event i varje process, intiutivt borde nog processen snarare se ut som i figur 22, dvs. en möjlighet att avsluta vid varje val och det är också så som den faktiska BPEL-implementationen sedan ser ut. Detta blir framförallt ett problem om man skall bygga väldigt stora modeller då alla pilar till end eventet kommer att göra modellen mycket rörig.

Dock kunde inte verktyget hantera konvertering av flera paralella processer, *swimlanes*, med egen intern logik, som modellen i figur 23. Verktyget accepterade bara att en process behandlades åt gången och att alla andra är *abstrakta.* Därför fick klientprocessen bli *abstrakt* och resultatet återfinns i figur 24 som är den slutgiltiga processmodellen. Kartläggningen och scenariot har sedan legat till grund för den kravspecifikation (se Appendix D) som togs fram.



Figur 24 - Slutgiltig processkarta, BPMN gjord i Oracle BPA

4.4 Krav- och testspecifikationer

I detta avsnitt beskrivs hur krav och testspecifikationer togs fram.

Nästa steg i implementeringsprocessen var att ta fram en kravspecifikation för systemet. Specifikationen utgick huvudsakligen från scenariobeskrivningen och processkartläggningen. Vissa krav på exempelvis felhantering, omfattades dock inte av någon av dessa utan har skrivits med utgångspunkt i aspekter som skall testas. Ett ytterligare problem identifierades ganska omgående. Den affärsmiljö dvs. de web services som skall användas, finns i normala fall innan man specificerar en tjänst. Dessa skapades därför omgående som ”mock-ups”.

Från λHR behöver man veta vilken bokningsbehörighet olika personer har, därför skapades en tjänst där man skickar namn och får tillbaka bokningsbehörighet (eller att personen inte finns). För att kunna testa olika varianter laddades λHR med tre olika personer, en utan behörighet, en konsult och en chef. Till λAcconting skall fakturor kunna skickas. Tjänsten som skapades har därför möjlighet att ta emot fakturor med data.

För att simulera ResPlus skapades en tjänst som på basis av start och slutstation samt tidigaste avgång och senaste ankomsttid returnerar en lista med reseförslag. För att kunna testa olika varianter laddades tjänsten med en resa med tre ruttförslag, taxi-tåg-taxi, taxi-flyg-taxi och taxi-tåg-flyg-taxi. Samtliga delsträckor korresponderade till delsträckor som laddats in i tjänsterna för respektive fordonsslag.

Tjänsterna för respektive fordonsslag tåg, flyg och taxi behövde ha tjänster för prisuppgifter på en viss resa och möjlighet att boka den resan. Tåg och flyg behöver dessutom en tjänst för att se antal platser som är kvar, en begränsning som taxi rimligtvis saknar. Flyg beställs på basis av flightnummer eftersom varje avgång har ett unikt nummer. Tåg beställs också med hjälp av unika nummer som motsvara tåg och från och till station. Taxi beställs i form av från och till.

Tjänsten för hotellbokning behöver en funktion för att hitta hotell på den valda orten, eftersom det ofta finns flera hotell på större orter. Hotellbokningstjänsten behöver också tjänster för pris på hotellnatt och lediga bäddar under vistelsen och dessutom en funktion för att boka hotell. För att kunna testa resetjänsten laddades hotelltjänst med en stad, innehållande tre hotell med olika pris och olika tillgång på bäddar. Den fullständiga specifikationen av affärsmiljön finns i kravspecifikationen (se appendix D).

När affärsmiljön fanns på plats gjordes kravspecifikationen. Domänkraven utgörs av specifikationen av affärsmiljön och en beskrivning av arbetsuppgiften med de olika möjliga felen och valen. Utifrån detta skapades detaljerade krav på produkt och designnivå för varje steg i processen. Den fullständiga specifikationen finns i appendix D.

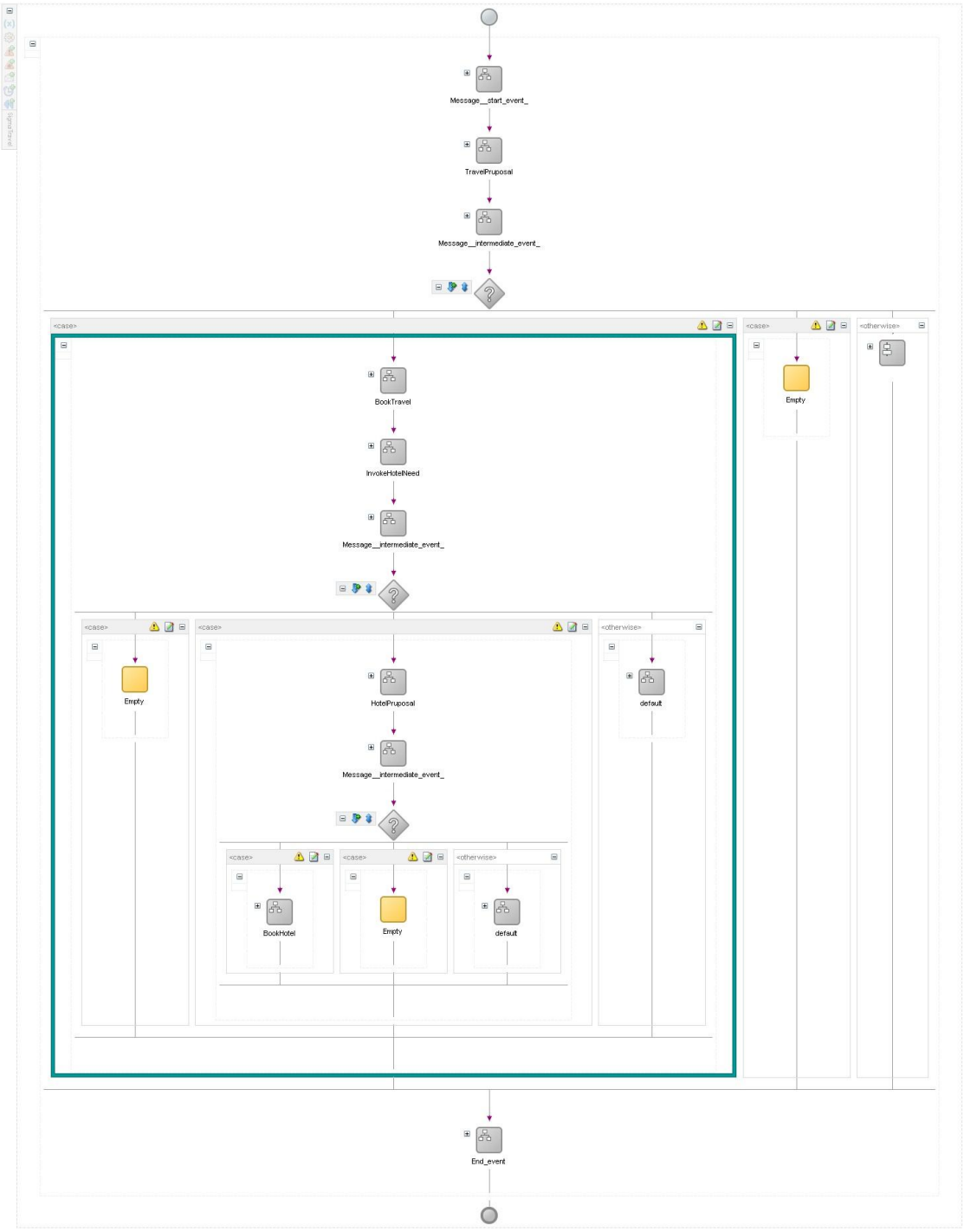
Utifrån kravspecifikationen skapades en testspecifikation med 24 testfall som täcker in de olika möjligheterna som systemet skall hantera. Testspecifikationen återfinns i Appendix E.

4.5 Implementering

I detta avsnitt beskrivs arbetet med implementeringen av scenariot, dels i BPEL och dels i Java.

4.5.1 BPEL

En av de stora fördelarna som framhålls med kopplingen mellan BPEL och BPMN är att det går att göra program för konvertering mellan dem. I Oracle BPA finns en funktion för detta. Man kan utifrån BPMN-kartan generera en så kallad BPEL-blueprint. Dock kan en BPEL-process inte återföras till BPMN-format eftersom det inte finns något stöd för detta. Implementeringen i BPEL inleddes med att göra en sådan konvertering. Resultatet återfinns i figur 25. Man kan konstatera att den är ganska lik processkartan (se figur 24). Denna karta är dock anpassad för att konverteringen överhuvudtaget skall vara möjlig. I den slutliga versionen av programmet ordnades strukturen mer linjärt i linje med figur 22.



Figur 25 - BPEL-process genererad från BPMN-processkartan

Att de olika delprocesserna ligger i varandra, som i figur 25, gör att delprocesser som är omslutna av andra delprocesser kan använda och påverka den omslutande delprocessens variabler och data. Det uppstår lätt bindningar som gör att systemet blir svårare att ändra särskilt i större konstruktioner. Detta ställer till problem om man vill ändra processordningen eller förändra processerna. Därför kan det vara bra att bygga applikationerna så likt delprocesserna som möjligt. Således ändrades strukturen så att den blev mer linjär.

De två största delprocesserna, ta fram reseförslag och boka resa, bröts ut och fick bli egna underprocesser med egna BPEL-processer för att göra programmet mer överskådligt, eftersom de var så pass stora. De övriga fick ligga kvar som *scopes,* egna undermiljöer, eftersom de var förhållandevis små, men kunde lika gärna blivit egna applikationer. Varje delprocess implementerades sedan. Felhantering hängdes dessutom på hela processen.

I figur 26 återfinns en översikt av det slutgiltiga programmet. Dess olika delar kommer nu att beskrivas och olika problem och finesser kommer att kommenteras.

Till vänster i figur 26 återfinns ett antal boxar som representerar de olika tjänsterna som anropas. Ett problem som ganska snabbt uppstod i samband med dessa är att Oracle utanför BPEL-specifikationen har en fil som heter bpel.xml där bindningarna till de olika tjänsterna som används specificeras. Denna fil uppdateras dynamiskt av editorn. Adressen till vissa tjänster går inte att spara utan försvinner automatiskt och applikationen går inte att kompilera. Detta är kopplat till att tjänstebeskrivningarna, i form av WSDL-filer, ligger externt. Därför måste dessa adresser skrivas in manuellt varje gång innan kompilering skall göras.

Till höger i figur 26 hänger en ”felfångare”. Den är inställd så att den fångar alla fel som kastas i processen. Dessa behandlas på samma sätt, dvs. de skickar felbeskrivningen till användaren. Detta kan givetvis varieras och felhantering kan hängas på delar av processen och/eller ställas in så att en viss felhanterare bara fångar vissa fel. I detta fall är det tillräckligt att alla fel fångas av en felhanterare.

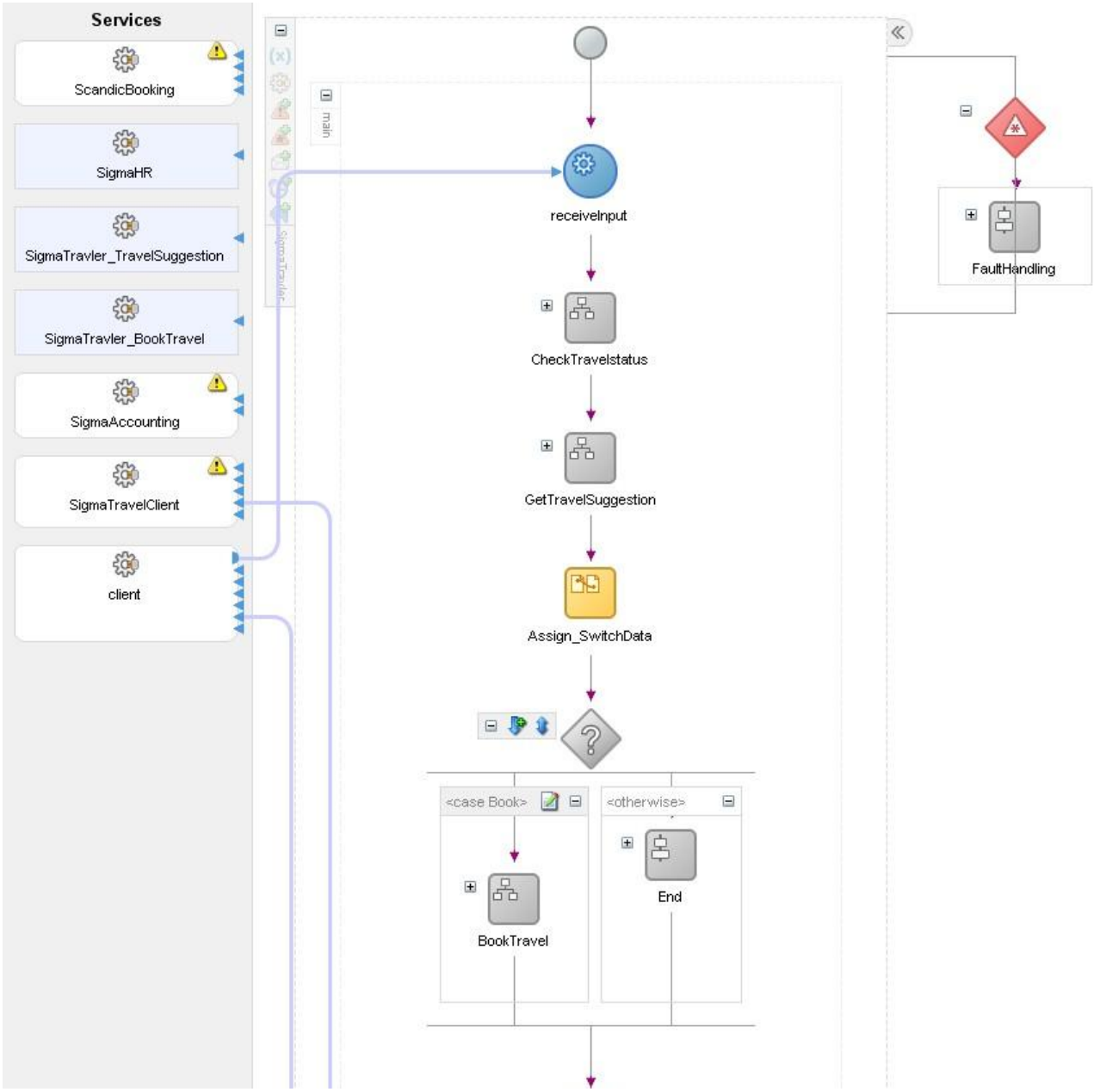
När det sedan gäller själva processen initieras den genom att den anropas genom WSDL-gränssnittet. I första steget kontrolleras bokningsbehörigheten genom att Sigma HR anropas. Är den godkänd fortsätter processen vidare, i annat fall avbryts den här.

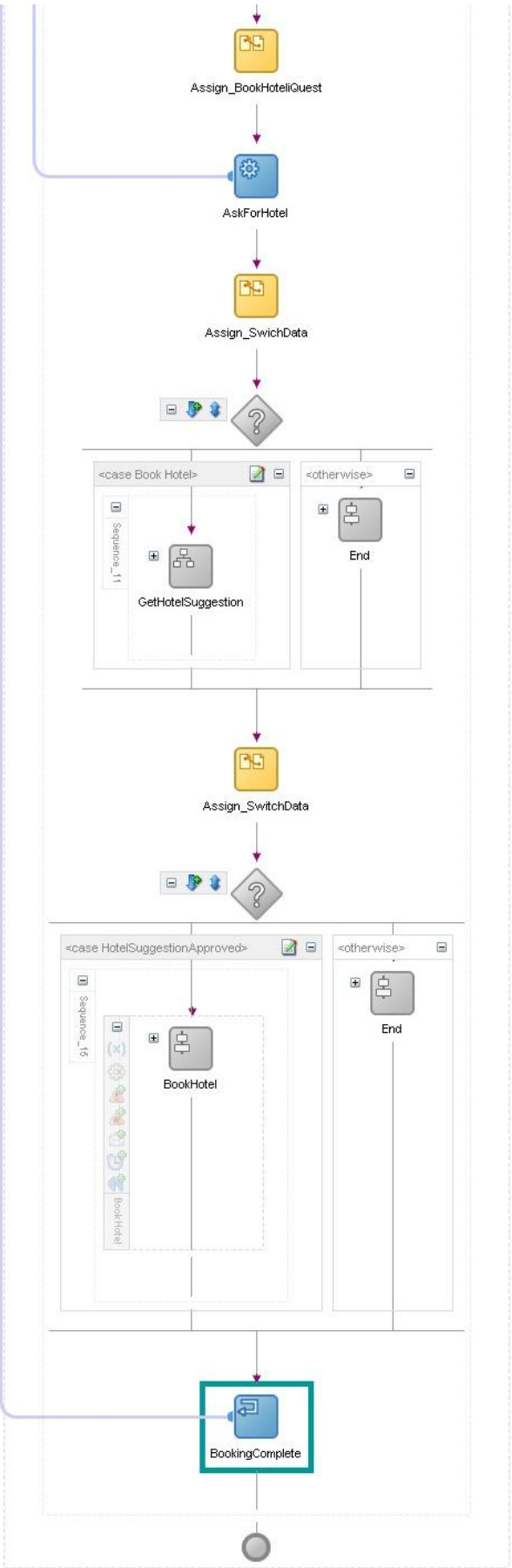
I steg två, som egentligen är den första delprocessen, anropas underprogrammet TravelSuggestion som tar fram ett reseförslag. Underprogrammet anropar ResPlus och får tillbaka en lista med resförslag, listan gås igenom och den bästa resan väljs. Vilken resa som är bäst beror på om den som bokar är konsult, då det är den billigaste resan som är bäst, eller chef, då det är den resan som tar kortast tid som är bäst. Varje förslag i listan gås igenom. Förslagen inehåller ett antal delsträckor. För varje delsträcka kontrolleras att den är möjlig att boka. Om någon delsträcka inte går att boka väljs resan bort. Kostnaden för varje delsträcka hämtas och den totala kostnaden beräknas i det fall resenären är konsult. I annat fall hämtas inte priset utan den totala tiden för resan beräknas. Att beräkna tiden mellan två *datetime*[3] var dock inte så lätt i BPEL utan fick göras med att en liten Javaapplikation. Slutligen retuneras den bästa resan till huvudapplikationen.

[3] Datetime är en enkel XML-klass för datum och tid på formen åååå**:**mm**:**dd**T**tt**:**mm**:**ss.

Huvudprogrammet skickar förslaget till klienten. Om klienten väljer att boka resan startar nästa delprocess, bokning, genom att underprogrammet BookTravel anropas. Underprogrammet går igenom den valda resan och bokar samtliga delsträckor. Om någon delstäcka inte kan bokas, för att biljetterna tagit slut exempelvis, signalerar underprogrammet detta och processen avbryts. Lyckas bokningen returnerar underprogrammet bokningsbekräftelsen till huvudprogrammet. Bokningsbekräftelsen skickas sedan till bokföringssystemet och till klienten av huvudprogrammet.

När bokningen av resan returnerats skickas en fråga om klienten vill boka hotell. Vill klienten detta tas ett hotellförslag fram. Detta görs genom att programmet först anropar hotelltjänsten och frågar vilka hotell som finns på den aktuella orten. Sedan itereras listan igenom och tillgänglighet och pris kontrolleras för varje hotell. Det billigaste alternativet av de hotell som har bäddar tillgängliga väljs. Förslaget returneras till klienten. Vill klienten boka detta alternativ bokas detta och bokningskvitto returneras till klienten och skickas också till bokföringssystemet. Sedan avslutas processen.

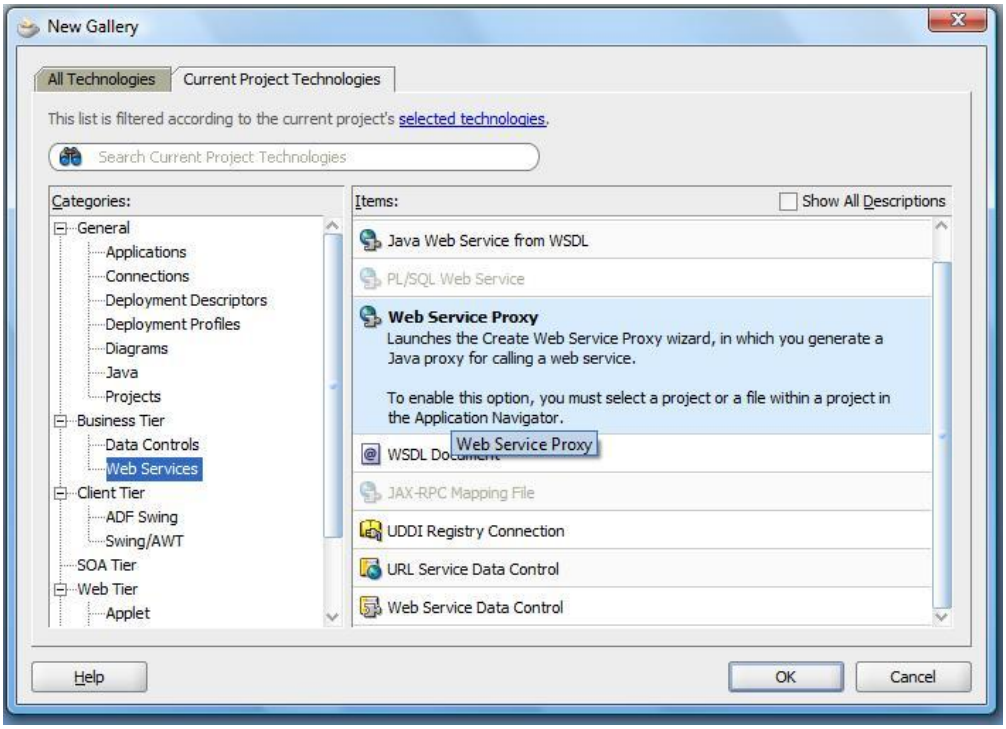




Figur 26 - Översikt av slutgiltig BPEL-process

4.5.2 Java

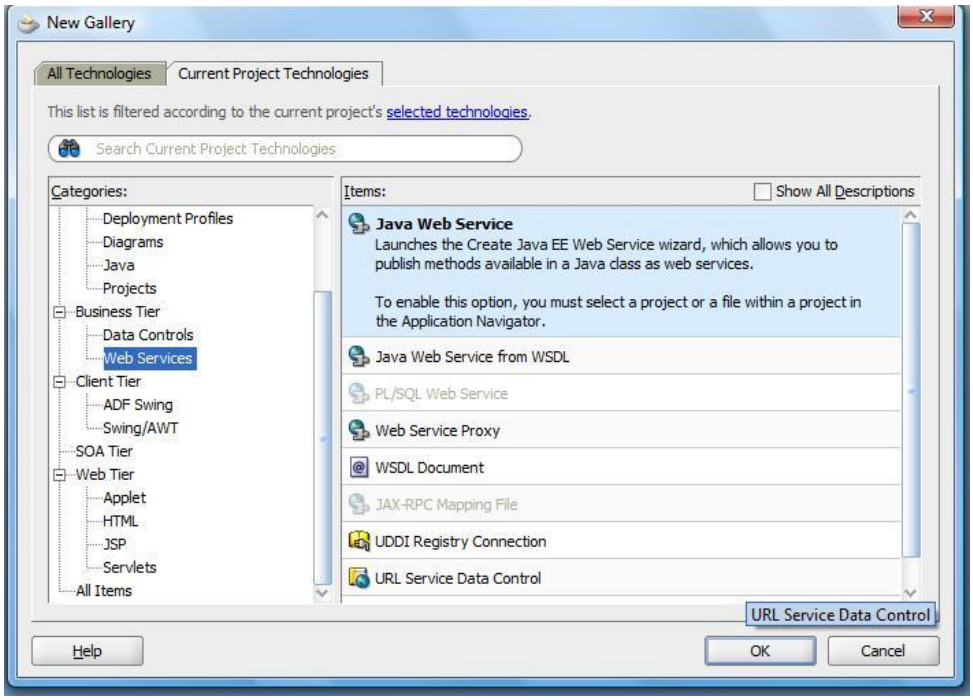
Javaimplementeringen liknar till sin struktur BPEL-varianten mycket. Den övergripande processen (Kontrollera bokningsbehörighet) > Hämta reseförslag > Boka resa > Hämta hotellförslag > Boka hotell är implementerad övergripande och varje delprocess är implementerad för sig. Processen följer samma logik som beskrivs i BPEL-avsnittet ovan, dvs. först kontrolleras bokningsbehörigheten. Är den okej hämtas reseförslag. Varje alternativ kontrolleras så att alla delsträckor har tillgängliga biljetter och det bästa alternativet av de som har tillgängliga biljetter på varje delsträcka returneras till klienten (bästa alternativ beror på bokningsbehörighet). Om klienten vill boka detta startar bokningsprocessen som går igenom och bokar varje delsträcka. Bokningsbekräftelse returneras till klienten om bokningen går bra samt till bokföringssystemet. Programmet frågar sedan klienten om denne vill boka hotell. Om klienten vill boka hotell tas det billigaste hotellet på orten, som har bäddar lediga för den aktuella perioden, fram. Förslag returneras till klienten. Om klienten vill boka görs en bokning på det aktuella hotellet och bokningsbekräftelse returneras till klienten och till bokföringssystemet. Sedan avslutas processen. På varje del finns också felhantering med ”try och catch”-satser. Skillnaderna är egentligen ganska små. Bredden i Java och metodutbudet hos de klasser som används gör att logiken på många ställen kan göras lite smidigare än i BPEL. Det enda som egentligen utmärker sig när man arbetar med web services i Java är kopplingen till själva web servicegränssnitten.



Figur 27 - Web Service Proxy i JDeveloper

Att arbeta med web servicegränssnitt i Java är inte lika ”straight forward” som i BPEL. Det kräver klasser som motsvarar XML-objekten, klasser som översätter mellan XML och Java, samt klasser som kommunicerar med web servicerna, skapas. Detta kan dock enkelt genereras i Jdeveloper genom att använda ”funktionen” Web Service Proxy som skapar en proxy för att kommunicera med web services.

Vill man publicera sitt Javaprogram som en web service kan man göra på lite olika sätt, det enklaste är att skapa en Java Web Service. Den skapar ett web servicegränssnitt utifrån en eller flera publika metoder i en redan skapad Javaklass.



Figur 28 - Java Web Service i JDeveloper

4.6 Resultat

Här beskrivs resultatet, dvs. hur de olika alternativen hanterar scenariot och klarar specifikationens krav.

De båda applikationerna testades sedan emot kravspecifikationen utifrån de test som återfinns i testspecifikationen (appendix E). Resultatet återfinns i tabell 1.

Tabell 1 - Testresultat

Test	BPEL	Java
Test 1: Normalfallet	👍	👍
Test 2: Felaktig indata	👍	👍
Test 3: Ej behörig klient	👍	👍
Test 4: Användaren finns ej i databas	👍	👍
Test 5: Ingen kontakt med λHR (λHR avstängd)	👍	👍
Test 6: Ingen rutt tillgänglig	👍	👍
Test7: Ingen kontakt med ResplusTravelRoute (vid hämtning av reseförslag)	👍	👍
Test8: Ingen kontakt med SJBooking (vid hämtning av reseförslag)	👍	👍
Test9: Ingen kontakt med SASBooking (vid hämtning av reseförslag)	👍	👍
Test10: Ingen kontakt med TaxiSverigeBooking (vid hämtning av reseförslag)	👍	👍
Test11: Klienten avböjer reseförslag	👍	👍

Test12: Ingen kontakt med SJBooking (vid bokning av resa)	👍[1]	👍
Test13: Ingen kontakt med SASBooking (vid bokning av resa)	👍[1]	👍
Test14: Ingen kontakt med TaxiSverigeBooking (vid bokning av resa)	👍[1]	👍
Test15: Inga tågbiljetter tillgängliga vid bokning	-[3]	-[3]
Test16: Inga flygbiljetter tillgängliga vid bokning	-[3]	-[3]
Test17: Ingen kontakt med λAcconting (efter bokning av resa)	👍	👍
Test18: Klienten avböjer boende på destination	👍	👍
Test19: Inga boenden tillgängliga på destination (resa till plats utan boende)	👍	👍
Test20: Ingen kontakt med ScandicBooking (vid hämtning av boendeförslag)	👍	👍
Test21: Klienten avböjer föreslaget boende	👍	👍
Test22: Ingen kontakt med ScandicBooking (vid bokning av boende)	-[2]	👍
Test23: Ingen Inga boenden tillgängliga vid bokning	-[3]	-[3]
Test24: Ingen kontakt med λAcconting (efter bokning av boende)	-[2]	👍
Total status:	**19/24**	**21/24**

[1] *Serveruppsättningen gör det svårt att hinna stänga av biljettjänsterna mellan första och andra gången de används i processen. Dock görs dessa hämtningar i en subBPELprocess som kan testas separat. Därför har dessa testats endast direkt mot subprocessen och inte via huvudprocessen.*

[2] *Ej testat. Serveruppsättningen gör det svårt att hinna stänga av ScandicBooking och λAcconting mellan första och andra gången de används. ScandicBooking och λAcconting anropas direkt från huvudprocessen och kan därför inte testas mot någon subprocess.*

[3] *Test 15,16 och 23 har inte att testats eftersom mock-up versionerna av SAS-, SJ- och ScandicBooking inte byggdes för testen och att det av tidsmässiga skäl inte fanns tid till att modifiera dessa.*

Som framgår av tabellen klarade Javaapplikationen alla test som gjordes. För BPEL-applikationen gick det inte att testa hur processen hanterar att tjänster inte är tillgängliga mitt i applikationen eftersom testmiljön inte tillät att man pausade processen annat än genom att lägga in timers i koden. Med största sannolikhet skulle dessa test gå igenom eftersom de är analoga med testerna där tjänsterna stängs av tidigare i processen. Båda teknikerna fungerade i alla fall bra för att implementera uppgiften.

Implementeringen i Java tog betydligt kortare tid, ca 25 % av tidsåtgången jämfört med BPEL-implementeringen. Detta berodde till största del på att det var få saker som var nya i Java (vissa klasser samt web service proxy och publicering). En del av skillnaden hänger antagligen ihop med att Javaimplementeringen startade senare än BPELvarianten (som dock var klar senare), då mindre tid behövdes för att strukturera upp designen. Kopplingen till BPMN gav inga större fördelar eftersom det i realiteten endast kan användas för de övergripande processflödena och dessa går ganska snabbt att skapa från scratch. Dessutom måste den genererade koden gås igenom och kontrolleras och i detta fall även ändras.

Java har som bredare språk större utbud av strukturer och objekt för datamanipulation än BPEL. BPEL var bättre när det gäller web servicegränssnitt och har ett stort plus i det grafiska gränssnittet i utvecklingsmiljön. I Java stöttes inga buggar eller konstigheter på, medan båda stöttes på i BPEL.

Testmiljö för BPEL kräver betydligt mer serverprogramvara och serverkapacitet (särskilt för Oracle 11g som inte användes). För Java och Java web services finns i Jdeveloper 11g en integrerad testmiljö, vilket är praktiskt. Testmiljön för BPEL var dock precis som editorn grafisk vilket gjorde att den var enkel att felsöka.

Eftersom implementeringarna gjordes i olika versioner av Jdeveloper har de testats på olika servrar (Jdeveloper 11g kan inte kopplas till SOA Suite 10g) och därför har ingen prestandatestning gjorts. För att få en uppfattning har kompletterande studier (se 4.7) gjorts .

Sammanfattningsvis kan man säga att BPEL var mer visuellt och bättre anpassat för web services medan Java har större bredd, är mer stabilt och hade en betydligt kortare implementeringstid.

4.7 Kompletterande ”studier”

I detta avsnitt kompletteras den huvudsakliga studien med ytterligare några små implementeringar och tester för att kasta ljus över aspekter som huvudstudien inte riktigt fångar.

4.7.1 Beskrivning

En fråga som inte kunde besvaras p.g.a. att huvudapplikationen implementerades i olika versioner av Jdeveloper, med följd att de inte kunde köras i samma server, var hur BPEL står sig prestandamässigt gentemot Java web services. För att testa detta och i alla fall få en uppfattning i frågan skapades två extremt enkla program (ett i BPEL och ett i Java) som endast returnerar en textsträng som man skickar till dem. Båda lades upp på samma server (Oracle SOA Suite 10.1.3.1.0). Ett test med en kort textsträng gjordes. Strängen skickades 10 gånger till respektive web service med 1000 ms fördröjning.

4.7.2 Resultat

Resultatet var rätt nedslående för BPEL (se tabell 2). I genomsnitt var Javaapplikationen ca 50 gånger snabbare. Omgång 1 och 9 för BPEL respektive 1, 5 och 9 för Java gav extrema värden som antagligen beror på något annat än själva exekveringen. Därför beräknades även genomsnittet utan dessa värden med resultatet att Javaapplikationen var ännu snabbare i förhållande till BPEL-applikationen ca 60 gånger snabbare. Resultaten ger en känsla för skillnaden i prestanda men säger inget om hur denna förändras med komplexiteten.

Tabell 2 - Svarstider för BPEL och Java web services

Omgång	BPEL	Java
1	*1803 ms*	*80ms*
2	611 ms	10ms
3	621 ms	10ms
4	591 ms	10ms
5	591 ms	*0ms (<1 ms)*
6	711 ms	10ms
7	621 ms	10ms
8	650 ms	10ms
9	*1803 ms*	*0ms (<1 ms)*
10	611 ms	10ms
Genomsnitt	**745.08 ms**	**15 ms**
Genomsnitt utan de kursiverade värdena	**625,88 ms**	**10 ms**

5. Analys

I detta kapitel diskuteras de olika för- och nackdelarna som BPEL med BPMN har och som framkommit i arbetet. Resultatet av den empiriska studien sätts i förhållande till den teoretiska bakgrunden och problematiseras och diskuteras. Analysen är uppdelad i fyra avsnitt utifrån arbetets fyra huvudsakliga fokus. I avsnitt ett till tre diskuteras anskaffning, arbete och teknik och de kostnader, intäkter och kostnadsreduceringar som är kopplade till respektive område. I det sista avsnittet lyfts strategiska aspekter, utifrån de tre ovanstående avsnitten, fram och diskuteras.

5.1 Anskaffningssaspekter

I detta avsnitt diskuteras anskaffningsaspekter av en eventuell investering.

Alla investeringar innehåller som första moment (efter att investeringsbeslutet tagits), en anskaffningsprocess. Beroende på investeringens natur är anskaffningarna olika kostsamma och tidskrävande. När det gäller anskaffning av en teknologi som BPEL med BPMN, består anskaffningen av dels *direkt teknologianskaffning,* exempelvis programvara, och dels *kunskapsanskaffning,* exempelvis utbildning i språket/språken. Till dessa moment är olika kostnader och förtjänster kopplade (huvudsakligen kostnader).

Om vi först fokuserar på de *direkta investeringarna* i form av teknik, såsom programvara, kan vi konstatera att kostnaden för detta är låg. Det finns flera alternativa programvaror både för server och för editor som är gratis. De flesta större tillverkare har dessutom BPEL-tillägg till sina serverprogramvaror och till sina editorer och därför är det stor chans att några av de verktyg man redan använder kan utökas utan extra kostnad. De kostnader som finns kopplade till teknologiinvesteringar utgörs därför huvudsakligen av omkostnader i form av tid för val, hämtning och installation av produkterna, kostnader som sammantaget torde vara relativt små i sammanhanget.

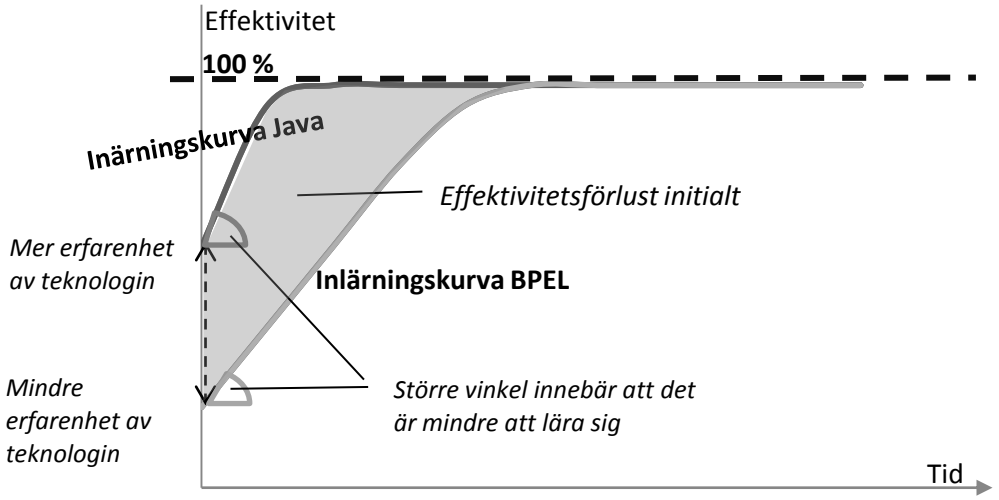
Fokuserar vi på det andra elementet i anskaffningen, *kunskapsanskaffningen,* kan vi ana att kostnaden för detta är något större. Kostnaden kan egentligen delas upp i två delar, dels *kostnader för utbildning,* såväl direkta, exempelvis kurser, böcker som indirekta, exempelvis inlärningstid och administration, och dels *introduktionskostnader,* dvs. kostnader för effektivitetsminskning i samband med att man börjar använda teknologierna. Hur stora dessa är och hur förhållandet mellan dem ser ut beror till viss del på hur man väljer att lägga upp kunskapsanskaffningen, till viss del på kunskapens beskaffenhet och till viss del på tidigare kunskap och erfarenhet hos dem som deltar i anskaffningen.

Om vi fokuserar på kunskapens beskaffenhet är det troligt att kunskapsanskaffningen i detta fall huvudsakligen består av att lära sig att göra saker man redan kan, fast i ett nytt språk och i en ny miljö, dvs. man kan bygga program men skall lära sig att göra det i ett nytt programmeringsspråk och kanske i en ny eller delvis ny utvecklingsmiljö. Således borde inlärningskurvan vara logaritmisk där vi redan tidigt kommer ha en viss effektivitet. Detta är bra och innebär bästa tänkbara utseende på inlärningen. Dock är det ännu mindre att lära sig om man använder Java som således kommer ha en ännu brantare kurva. Kunskapen är också i huvudsak praktisk. Det handlar om att lära sig språket och verktygen genom att arbeta med dem. I den implementering som gjordes tog BPEL-varianten

betydligt längre tid att bygga än Javavarianten, ca fyra gånger så lång tid. Implementeringen gjordes efter ca fem veckors teoretiska studier av BPEL. Den teoretiska kunskapen gav alltså inte speciellt mycket hjälp i det praktiska arbetet. Den största delen av lärandet låg alltså i det praktiska användandet av språket, vilket ledde till en effektivitetsminskning i förhållande till implementeringen i Java. Effektivitetsminskningskostnaden var därför, i detta fall, i förhållande till utbildningskostnaderna hög. Det skall dock påpekas att hela effektivitetsminskningen inte torde bero på att BPEL var nytt. Det beror också på upplägget av uppgiften och på att implementeringsmiljön fick bytas under resans gång.

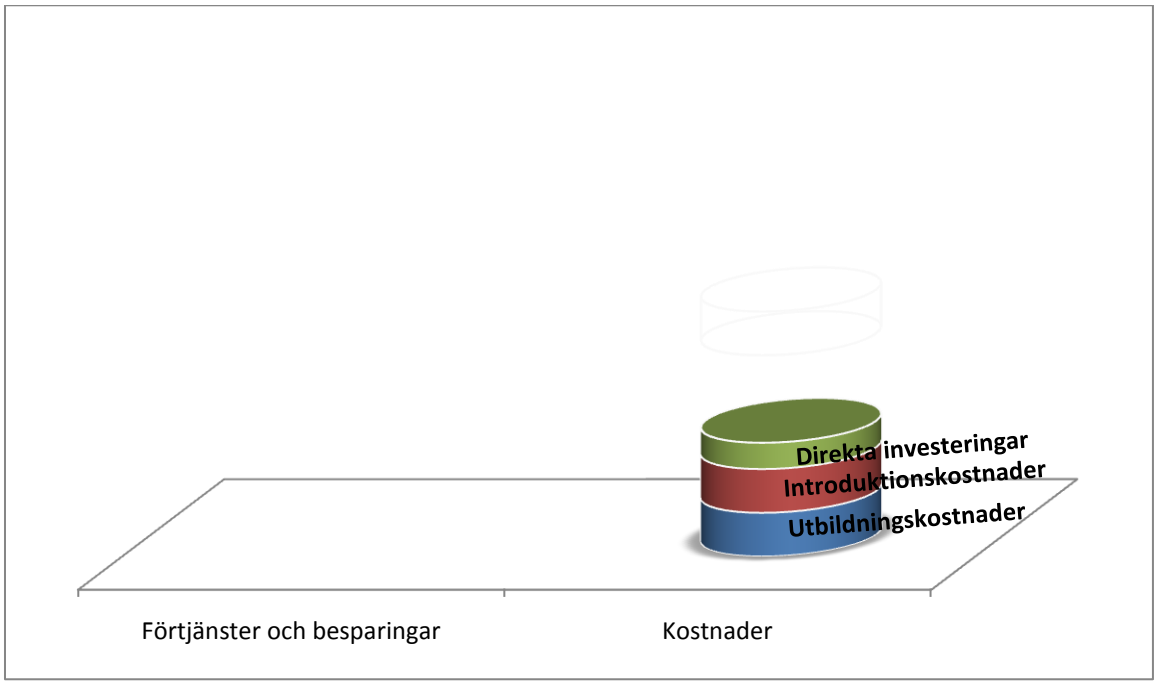
Genom upplägget av anskaffningen av kunskap kan man givetvis i stor omfattning påverka förhållandet mellan kostanden för utbildning och för effektivitetsminskning. I allmänhet leder mer utbildning till ökad effektivitet initialt (dock är effektiviteten i produktion noll under utbildningstiden). Utbildningens upplägg är också viktigt för att så mycket kunskap som möjligt skall inhämtas på så kort tid som möjligt, exempelvis kan man välja kurser med mycket praktisk träning. Det gäller alltså kort och gott att hitta en balans där utbildningen är lagom omfattande och har en lagom hög kostnad. I detta skall också vägas in att det kan finnas både dolda vinster och kostnader kopplade till att skicka personal på utbildning eftersom det både kan uppfattas som positivt och därmed höjer intern goodwill och arbetsmoral men också utpekande, dvs. du behöver utbildning och skapa sänkt arbetsmoral. Normalt anser man dock att utbildning är berikande och höjer arbetsmoral och lojalitet men att effekterna inte är entydiga och är svåra att fånga. Utbildning kan också vara ett bra sätt att hantera övertalighet vid dåligt konjunkturläge och/eller produktionsbottnar.

Den tredje faktorn, erfarenheter hos de som deltar i kunskapsinhämtningen, påverkar självklart också den kostnaden. I den studie som gjordes var inte bara språket nytt utan även att arbeta med web services och utvecklingsmiljön. De flesta företag som är intresserade av teknologin har antagligen medarbetare som har större erfarenhet av programmering i allmänhet, av att använda web services och av olika implementeringsmiljöer, än som var fallet i implementeringen som gjordes i denna studie. Detta medför att man börjar på en högre nivå än vad som var utgångsläget i studien. Något som leder till att det är mindre att lära sig (kortare inlärningstid) och mindre effektivitetsförluster dvs. lägre kostnader.



Figur 29 - Inlärningskurva BPEL i förhållande till Java

Sammanfattningsvis består kostnaden för anskaffning av BPEL med BPMN av en ganska liten andel för anskaffande av programvara och en större andel för anskaffande av kunskap. Totalt är kostnaden för anskaffning är ändå inte speciellt stor såvida man inte låter utbildningskostnaderna svälla.



Figur 30 - Förtjänster och besparingar jämfört med kostnader för investering i BPEL med BPMN 1, anskaffningsaspekter

5.2 Arbetesaspekter

I detta avsnitt diskuteras hur en investering i BPEL påverkar arbetet med programvaruutveckling.

När teknologin är anskaffad kan man börja arbeta med den. Olika teknologier påverkar arbetet med en viss uppgift på olika sätt. Teknologin har så att säga en rad särdrag i förhållande till konkurrerande teknologier. Vissa av dessa särdrag gör arbetet enklare och vissa gör det svårare och resulterar i kostnadsminskningar respektive kostnadsökningar. Studien pekar på ett antal sådana särdrag som BPEL med BPMN har i förhållande till Java. Detta innebär självklart inte att studien tar upp eller har identifierat alla särdrag, men att flera viktiga särdrag har uppmärksammats. De två absolut största fördelarna med BPEL och BPMN, som identifierats i studien, är sådana särdrag. Det är dels det grafiska möjligheterna och dels de specialanpassningar som finns till web services som bredare språk saknar.

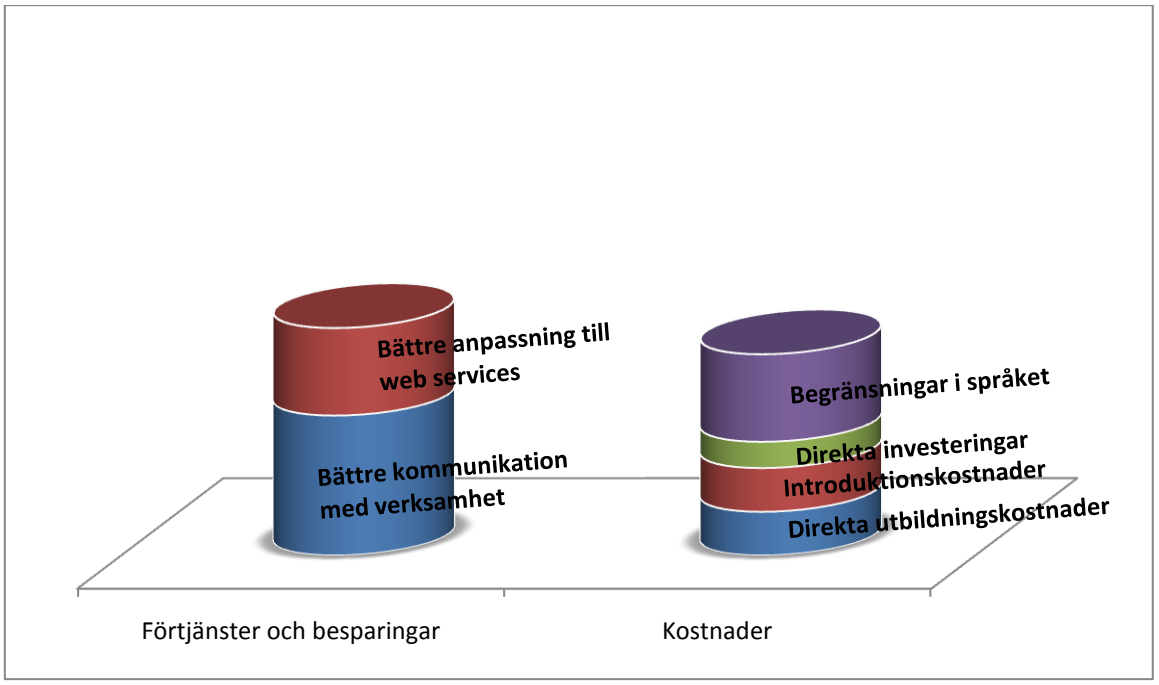
I litteraturen framhålls ofta kopplingen mellan BPEL och BPMN som den stora behållningen med teknologin eftersom den skulle kunna överbygga gapet mellan verksamhet och IT. Denna koppling har dock sina begränsningar och är i praktiken ganska överflödig just nu. Någon teknik där man kan konvertera fram och tillbaka mellan BPMN och BPEL finns inte. En sådan ”roundtripping” är kanske inte så meningsfull och verkar inte heller kunna göras på ett bra sätt. Den teknik som användes var konvertering från BPMN till BPEL där en så kallad ”blueprint” skapades som utgångspunkt. Vinsten var dock liten och det var svårt att se att det kan användas till mer än övergripande kartbilder. Schematiken skiljer sig och den tid det tar att gå igenom den genererade koden är inte mycket kortare än att skapa den från scratch. Dessutom innebär detta arbetssätt att man när man gör implementeringen efter att man konverterat processkartan slänger processdesignen, vilket innebär att det färdiga systemet inte nödvändigtvis stämmer överens med processkartan av det. Detta kommer troligen att skapa problem när man vill ändra på designen, eftersom den som designat eller kartlagt processen kommer göra förändringar utifrån processkartan och inte utifrån systemet. Skall man använda teknikerna tillsammans bör man nog skippa konverteringssteget och använda BPMN som ett grafiskt gränssnitt direkt på BPEL. Något som vissa tillverkare, exempelvis Intalio, har tagit fasta på. Det gör att de som arbetar med processerna och de som implementerar dem kan arbeta i samma verktyg och hela tiden se samma bild. Därmed undviks problemet ovan, där systemet och kartan inte stämmer överens. BPELs struktur lämpar sig i allmänhet bra för att ha ett grafiskt gränssnitt för utveckling, oavsett om detta bygger på BPMN, är inspirerat av BPMN eller om det är en helt fristående representation. Samtliga editorer som undersökts har också haft ett grafiskt gränssnitt för utveckling av något slag. Gränssnitten varierar i kvalitet och vissa, exempelvis Netbeans, är mer likt BPMN medan andra, exempelvis Jdevelopers, är mindre likt. Dock är de alla tillräckligt simpla och begripliga för att någon som är van vid att läsa processkartor skall kunna förstå och följa dem. Med hjälp av de grafiska gränssnitten finns det goda möjligheter att föra verksamheten närmare IT och därmed spara kostnader kopplade till både samarbete och till eftersläpning i förändrings- och förbättringsarbetet. Ännu bättre är det såklart om gränssnittet består av en kartläggningsnotation, som BPMN.

Den andra stora fördelen med BPEL gentemot Java, nämligen att språket är anpassat speciellt för web services, gör att det är enkelt att arbeta med kopplingar mellan olika web services. Det finns gott stöd för generering av liknande kopplingar i Java men detta innebär ändå betydligt större kodmängd, som dessutom är standardgenererad och alltså inte lika smidig att använda. Varje gång en web service ändras måste dessutom en ny proxy genereras i Java. BPEL har här dessutom möjlighet till

högre abstraktion eftersom man använder sig av *PartnerLinks* och *PartnerLinkTypes* dvs. porttyp istället för port. Baksidan av specialiseringen är såklart att språket är smalare och inte alltid har stöd för alla typer av operationer. Ett sådant exempel finns i implementeringen, nämligen att BPEL inte på ett bra sätt kunde beräkna tiden mellan två *datetime*. Detta innebar i detta fall en del merarbete för att skapa sådana funktioner. Fördelarna och nackdelarna med specialiseringen är ungefär lika stora i allmänhet. Dock beror förhållandet på vad man skall använda BPEL-processen till. Ju mindre datamanipulation och ju fler web servicer som skall samordnas desto bättre är BPEL för uppgiften.

En fördel med BPEL som ofta framhålls är att det skall vara lätt att designa om och ändra i processflödena. Detta är en faktor som inte ingick i den praktiska implementeringen eftersom tiden för studien var begränsad. Efter att ha arbetat med verktygen är min känsla i alla fall att det visserligen är en faktor som talar till BPELs fördel, men att fördelen är mindre än vad som görs gällande. Det som gör det enklare är framför allt möjligheten att dra, släppa och flytta objekt i det grafiska gränssnittet. Det känns också lite naturligare att organisera strukturen efter processbeskrivningen i BPEL, som är mer processorienterad, än i Java, som är mer objektorienterad.

Sammanfattningsvis kan man säga att när det gäller arbetsaspekter så överväger fördelarna med specialiseringen och det grafiska gränssnittet gentemot nackdelarna med begränsningarna.



Figur 31 - Förtjänster och besparingar jämfört med kostnader för investering i BPEL med BPMN 2, anskaffnings- och arbetsaspekter

5.3 Tekniska aspekter

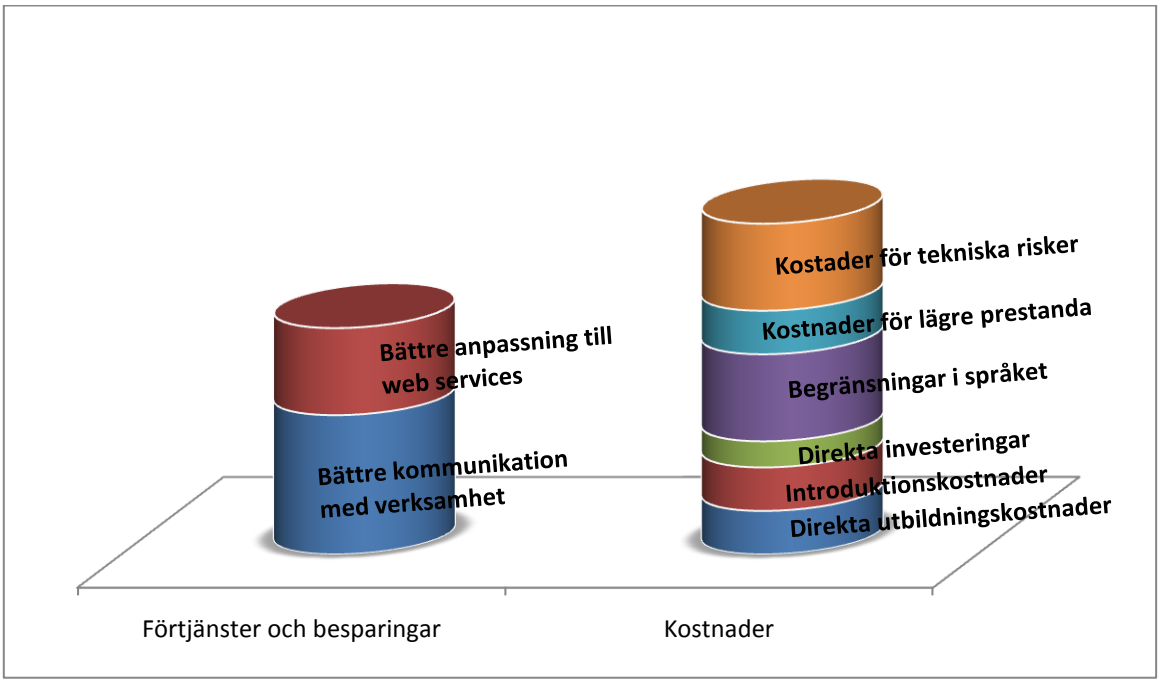
I avsnittet diskuteras olika aspekter av själva tekniken som kommit fram i studien och hur dessa påverkaren en eventuell investering.

Vissa av de kostnader och vinster som beskrivits ovan under rubrik 5.2 arbetsaspekter hänger självklart ihop med teknik. De är dock de arbetsmässiga aspekterna som genererar den största delen av kostnaderna. I det här avsnittet presenteras två aspekter som visserligen får arbetsmässiga konsekvenser men som till sin natur i huvudsak är tekniska, den första aspekten är prestanda och den andra är tekniska risker.

I det praktiska arbetet gjordes en mindre utvärdering av prestandaskillnader mellan BPEL och Java. Resultatet av denna utvärdering var att Java var överlägset snabbare gentemot BPEL. Utvärderingen är dock mycket begränsad eftersom den endast testade hur snabbt serven kunde returnera en textsträng som man skickade till den. Testet indikerar i alla fall att Java web services verkar vara betydligt snabbare än BPEL. Hur hastigheten ökar med komplexiteten vet vi dock inte. Det skall också påpekas att resultatet endast är giltigt för den valda testmiljön och kan inte därför generaliseras. Resultatet känns dock logiskt eftersom Java funnits mycket längre och utvecklingen av kompilatorer och serverprogramvara således borde kommit längre och därmed borde det vara effektivare. Är det så att BPEL i allmänhet är långsammare än Java och andra allmänna programmeringsspråk innebär det en viss merkostnad att använda tekniken eftersom man behöver bättre hårdvara om man skall få jämförlig prestanda.

Till varje teknik finns alltid risker kopplade. Ju nyare och mer oprövad en teknik är desto större risker finns det i allmänhet med den. Som specifikation har BPEL funnits ganska länge (sedan mitten av 2002). Detta borde i allmänhet betyda att tekniken är väl utvecklad och att riskerna är låga. Men med BPEL är detta inte riktigt fallet, i alla fall inte när det gäller de verktyg som testas här. Detta är kanske inte så konstigt. Utvecklingen av verktyg torde ha startat i slutet av 2002 eller början 2003. De första versionerna kom troligtvis ut under 2004 (de är i alla fall de tidigaste som jag har hittat). Samtidigt arbetar OASIS med en ny version av BPEL, version 2.0. Mellan 2004 och 2006 fortsätter arbetet att utveckla verktygen. År 2007 kommer dock den nya versionen av BPEL, version 2.0, som är en mer komplett och mogen specifikation. Den var inte bakåtkompatibel med tidigare versioner och innehåller ganska stora förändringar. Detta har gjort att vissa tillverkare exempelvis Oracle och Microsoft, valt att inte utveckla sin programvara nämnvärt sedan 2006, medan andra valt att arbeta mot den nya specifikationen exempelvis Sun och IBM. Att version 2.0 av BPEL inte är bakåtkompatibel har alltså bidragit till att utvecklingen på området har stannat av. Detta innebär att det finns risker kopplade till både version 1.1 och 2.0 men att dessa är av lite olika karaktär. Den stora risken med att använda verktyg som korresponderar med v 1.1 är att dessa troligtvis inte kommer att utvecklas mer och tillslut inte stödjas av framtida programvara. Dock har de funnits längre och är därför stabilare. De verktyg som provats här och som korresponderar med v 2.0 är å andra sidan så pass färska att de fortfarande innehåller väldigt många buggar, exempelvis Suns verktyg i Netbeans/Open ESB för Glassfishserver som provades. Alltså har man att välja på att ta kostnader nu och använda ostabila verktyg för v 2.0 som kanske utvecklas och blir bättre i framtiden eller att använda relativt stabila verktyg för v 1.1 men senare riskera stora kostnader för migrering. Oavsett alternativ innebär detta stora riskkostnader.

Sammanfattningsvis kan man säga att de tekniska aspekterna idag är till nackdel för BPEL, både kostnader för prestanda (om den är lägre vilket verkar troligt) och tekniska risker är relativt stora. Värt att notera är dock att dessa kostnader med största sannolikhet är kostnader som minskar med tiden till skillnad från många av de tidigare aspekterna vi tagit upp.



Figur 32 - Förtjänster och besparingar jämfört med kostnader för investering i BPEL med BPMN 3, anskaffnings-, arbets- och tekniska aspekter

5.4 Strategiska aspekter

Teknik-, anskaffnings- och arbetsaspekter är alla i någon mening praktiska aspekter som direkt påverkar arbetet och renderar kostnader och intäkter i den vardagliga verksamheten. I detta avsnitt diskuteras mer strategiska och övergripande problem och möjligheter kopplat till en eventuell investering.

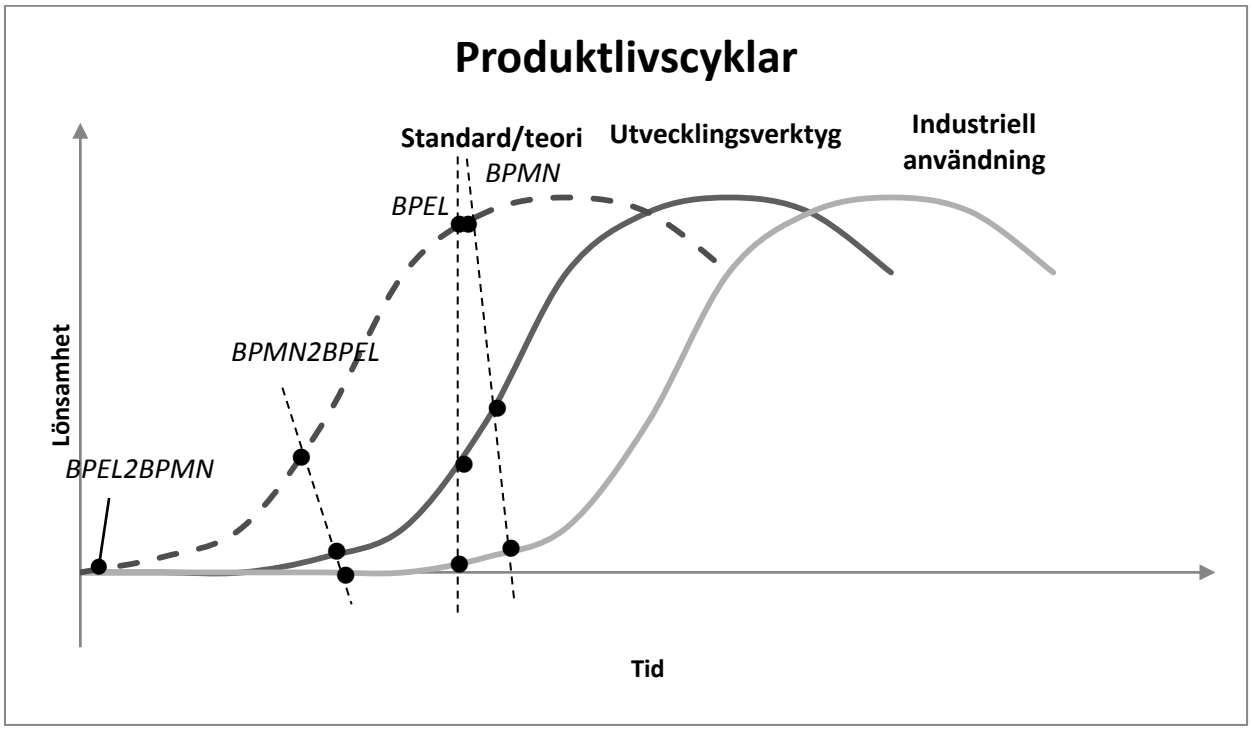
I teoriavsnittet (3.4 Analysmodeller) presenterades två stycken modeller, produktlivscykeln och innovationsadoptionskurvan, som kan användas för att förstå hur produkter och teknologier utvecklas och används. Utifrån modellerna kan vi diskutera var de olika teknologierna är i sin utveckling, vilka typer av intressenter som rimligtvis har intresse av teknologierna och vilka fördelar och nackdelar deras nuvarande läge innebär.

Det vi först kan konstatera är att dessa teknologier finns på tre nivåer eller tre marknader:

1. *Standard eller teoretisk modell*, dvs. de standardokument som OASIS respektive OMG har tagit fram och de teoretiska modeller för översättning mellan BPMN och BPEL som finns exempelvis (Ouyang, Dumas, van der Aalst, & ter Hofstede, 2008). Producenter på denna marknad är alltså i huvudsak OASIS och OMG men också de forskare och utvecklare som tar fram de teoretiska modellerna. Kunder är de företag som bygger utvecklingsverktyg, som Microsoft, IBM, Oracle m.fl.
2. *Utvecklingsverktyg*, är marknaden för själva de verktyg som utvecklas som servrar, editorer och affärssystem. Producenter på den marknaden är kunderna på nivån ovan, dvs. företag som bygger dessa verktyg. Kunderna på marknaden är företag som utvecklar system som exempelvis Sigma, Cybercom eller Tieto eller företag som använder verktygen i sina tjänster exempelvis managementbolag.
3. *Industriell användning,* är den sista nivån eller marknaden på vilken implementeringar i BPEL och/eller BPMN handlas. Producenter är kunderna från nivå två, dvs. systemutvecklarna m.fl. och kunderna är verksamheter som använder system eller som äger den verksamhet de används i, exempelvis industriföretag eller offentlig sektor.

Var och en av dessa delar har en egen utvecklingskurva, men kurvorna är beroende av varandra (se figur 33). Standarden genererar visserligen ingen lönsamhet, men utvecklingen styr de andra två och följer någon slags mognadskurva därför är den streckad i figur 33. Standardutvecklingen påverkar utvecklingen av verktyg som implementerar den och verktygsutbudet påverkar utvecklingen av program och system. Självklart påverkar en massa andra faktorer, som intresse för tekniken, utvecklingen på hela web serviceområdet etc., utvecklingen. Dessa faktorer torde dock påverka alla aspekter av teknologin. Exempelvis leder ett ökat intresse till högre efterfrågan på BPEL-system som leder till högre efterfrågan på utvecklingsverktyg och i slutändan även till högre intresse och nya förslag till utökningar och förbättringar av standarden. Det som är intressant är att lönsamheten i de olika segmenten, (standard), utvecklingsverktyg och industriell användning är olika stor för olika tidpunkter men att utvecklingen i ett segment påverkar de andra. Segmentens livscyklar är så att säga horisontellt förskjutna i förhållande till varandra (se figur 33). Detta innebär att standarden har kommit längre i sin utveckling än vad marknaden för utvecklingsverktyg har, som i sin tur har kommit längre än marknaden för industriell användning.

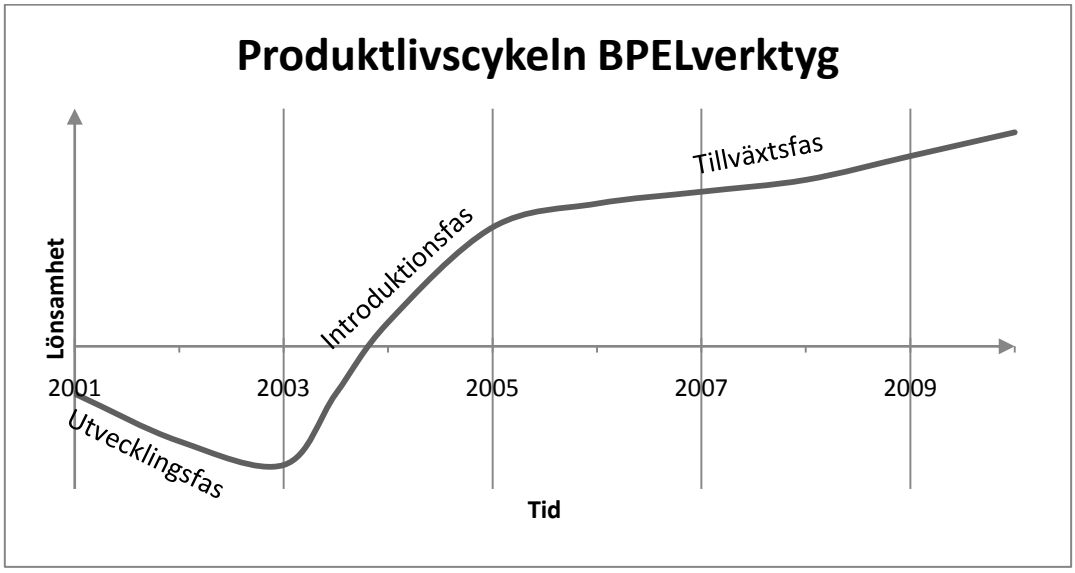
Börjar vi att se på teknologin kan vi konstatera att BPEL med BPMN egentligen består av fyra delteknologier, BPEL, BPMN, BPMN2BPEL och BPEL2BPMN. Som vi har konstaterat finns det inte några produkter som konverterar från BPEL till BPMN. Det enda som egentligen finns är idéer. Denna konvertering skulle behövas för att skapa en sluten cirkel, ”roundtripping”. Idén med ”roundtripping” verkar dock vara övergiven eftersom BPEL och BPMN är för olika. Följaktligen kom denna delteknologi aldrig längre än till tidig utvecklingsfas. Teknologin för konvertering åt andra hållet har kommit längre och det finns ett antal verktyg. Verktygen fungerar, som vi konstaterat, sådär och är just nu ganska överflödiga. Just nu befinner sig verktygen tidigt i introduktionsfasen med låg lönsamhet. Implementeringar byggda på tekniken är vad det verkar få. Den enda variant där BPEL och BPMN används tillsammans som har tillväxtpotential är BPMN som grafiskt gränssnitt direkt på BPEL. Denna variant betraktar jag som en BPEL-teknologi som tillsammans med andra BPEL-varianter med andra interface utgör verktygsutbudet för BPEL. BPMN och verktyg för detta lever sitt eget liv eftersom teknologin i första hand är till för kartläggning och inte för exekverbara processer. Utvecklingsverktygen för båda teknologierna, BPMN och BPEL, har funnits ett tag och befinner sig idag i en tillväxtfas där fler och fler börjar intressera sig för teknikerna. Den industriella användningen är dock lägre och befinner sig fortfarande i introduktionsfasen för båda verktygen. BPMN har en större spridning med ca 65 implementeringar i utvecklingsverktyg jämfört med BPELs ca 10-15, varför BPMN ligger något före BPEL på både nivå två, utvecklingsverktyg och tre, industriell användning. Standarderna är ungefär jämbördiga. Fortsättningsvis skall vi fokusera på BPEL som egen teknologi eftersom vi har konstaterat att ”roundtripping” inte är speciellt intressant i sammanhanget och kopplingen till BPMN bara är intressant då BPMN finns som grafiskt gränssnitt.



Figur 33 – Produktlivscykler för tre marknader, jämförelse mellan de olika delteknologierna

BPEL har alltså funnits några år och det finns ett utbud av ett flertal verktyg. Många av dessa är dock inte helt stabila. Den största anledningen till att utvecklingen inte riktigt tagit fart är antagligen att utvecklingen planade ut i samband med att v 2.0 av standarden skulle komma. Att kopplingen till BPMN inte blev så bra som man hade tänkt sig kan också vara en bidragande orsak. Den verkliga

lönsamhetsutvecklingen för utvecklingsverktygen ser nog snarare ut som i figur 34, dvs. att lönsamheten till en början var negativ och att den steg när verktygen började komma ut på marknaden. Utvecklingen dämpades dock när v 2.0 var på väg och börjar nu öka igen i takt med att nya verktygsversioner kommer ut. Jämför man med den allmänna grafen i figur 33 kan vi konstatera att kurvan inte riktigt följer typkurvan utan att det skett viss fördröjning i utvecklingen.

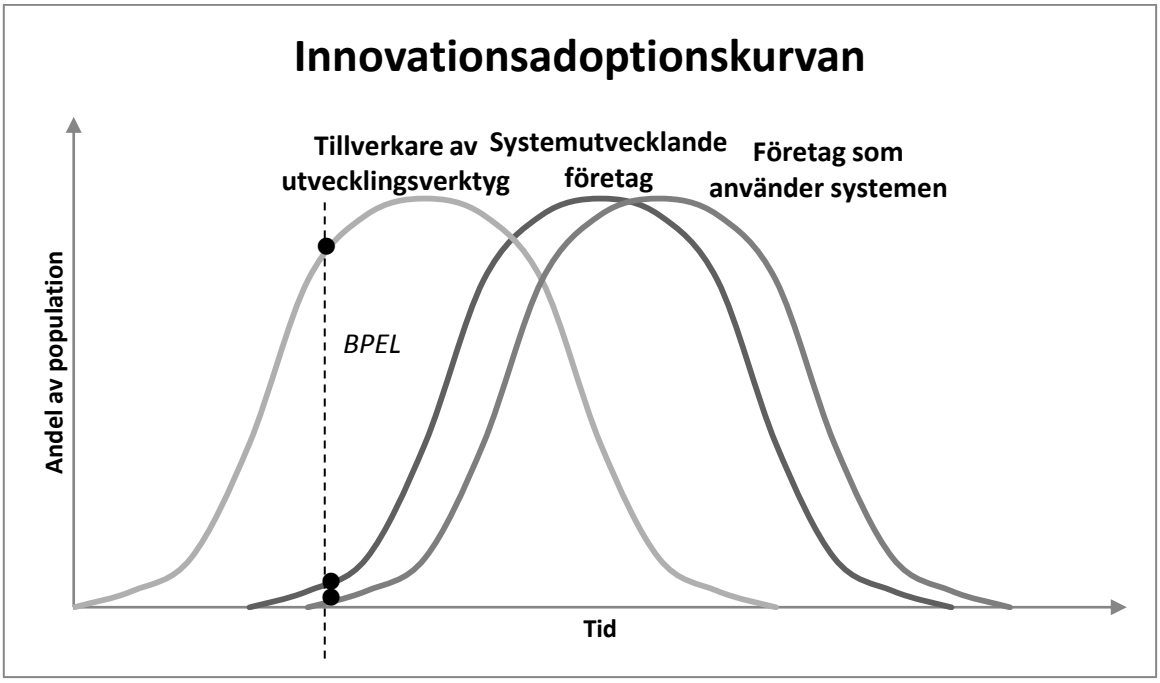


Figur 34 - Produktlivscykeln, utvecklingen för BPEL-verktyg

Att vara med tidigt i en produkt eller teknologis utveckling har både för- och nackdelar. Fördelarna är just att man tidigt kan profitera och ta marknadsandelar när teknologin sprids samt att man kan vara med och påverka och styra teknologins utveckling. Nackdelarna ligger i risken med att investera i en teknologi som inte lyfter. Hur stora fördelarna och nackdelarna är i förhållande till varandra beror på vilken position och profil man har på marknaden. Med hjälp av innovationsadoptionskurvan kan vi diskutera vilka som adopterar BPEL vid olika tidpunkter. Innovationsadoptionsprocessen hänger ihop med produktlivscykeln och förklarar den delvis. Med bakgrund i diskussionen ovan, om olika marknader bör adoptionen av BPEL bland företag som bygger utvecklingsverktyg ses mot utvecklingen av standarden, adoptionen bland företag som utvecklar programvara mot utvecklingen av verktyg och adoptionen hos företag som använder systemen mot utvecklingen av den industriella användningen av BPEL-baserade system.

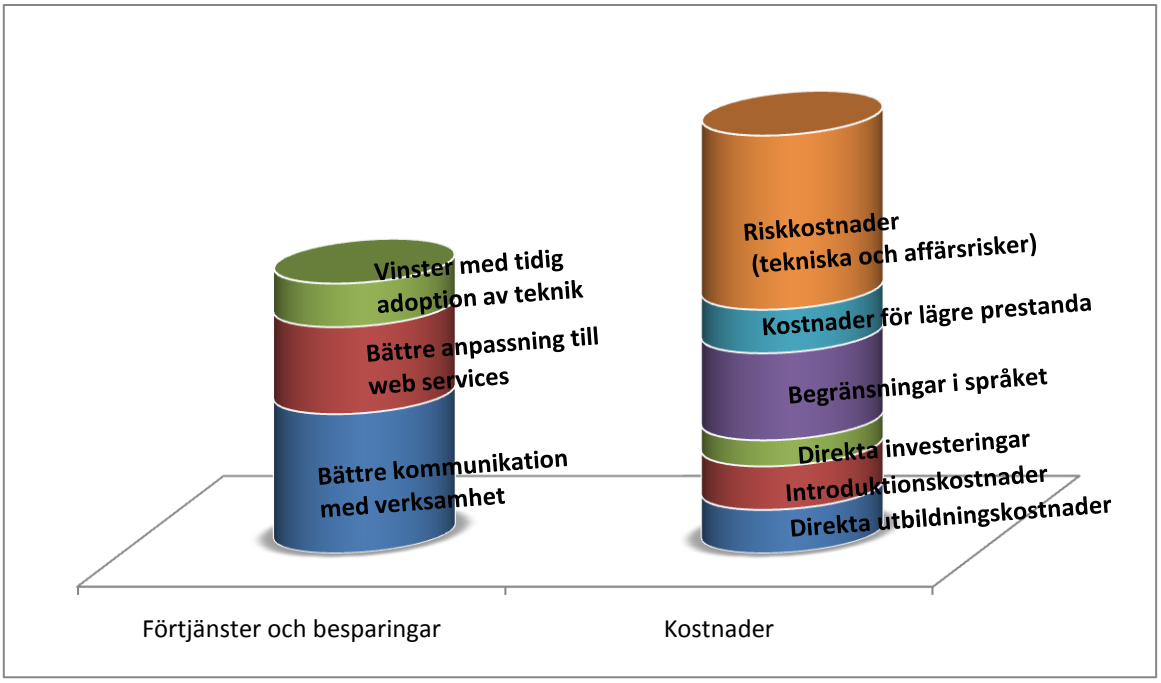
I BPELs fall är innovatörerna de ”stora drakarna”, Microsoft, Oracle (BEA & Siebel), IBM, SAP m.fl. De finns tidigt med för att kunna styra marknaden och bevaka sina positioner. Dessa är också de som tidigt börjar implementera standarden i sina utvecklingsverktyg. Detta gör att adoptionen bland verktygstillverkarna är relativt stor eftersom dessa utgör en stor del av marknaden. Alltså ligger adoptionen här på gränsen till tidig majoritet. Dock är lönsamheten som vi konstaterat fortfarande låg eftersom verktygen inte är färdiga, eller nyligen blivit färdiga, för version 2.0 av BPEL samt att adoptionen bland programvaruutvecklande företag inte verkar vara så stor. Min uppfattning är att det bland de programvaruutvecklande företagen finns ganska få som ännu arbetar med tekniken, men det finns ingen sammanställning av detta. Min uppfattning bygger på att det bland allt det material jag gått igenom under arbetet funnits få exempel på faktiska implementeringar i företag och att utvecklingen av verktyg för implementering inte är helt klara eller i alla fall inte helt stabila. Utifrån generella antagande om adoption och med bakgrund mot resonemanget kring

produktlivscykeln kan man dock tänka sig att de som idag arbetar med BPEL är företag som dels varit inblandade i utvecklandet av BPEL, samt företag vars profil är att de alltid ligger i teknisk framkant eller är nischade på BPM/SOA-teknologi med fokus på BPEL. Dessa företag har en profil och en strategi som gör att deras födkrok hänger på att de adopterar ny teknik. Företag som exempelvis har en strategi som bygger på att de har hög kvalitet eller låga priser bör kanske avvakta eftersom riskerna för dem är betydligt större än förtjänsterna.



Figur 35 - Innovationsadoptionskurvan för BPEL

Sammanfattningsvis kan vi konstatera att lönsamheten utifrån vårt resonemang ovan fortfarande är låg för BPEL både för verktygstillverkare och för systemutvecklare. Många utvecklingstillverkare har trots detta adopterat tekniken men utvecklingen hålls tillbaka av att implementeringen av verktygen inte är helt klara eller stabila. Adoptionen bland verktygstillverkarna är stor men mest p.g.a. att ”de stora jättarna” adopterat tekniken, antalet verktyg är dock fortfarande få, 10-15 stycken. Är man verktygstillverkare måste man fråga sig om BPEL tillför något till ens verktyg, dvs. om man tillhandahåller verktyg för utveckling av SOA och affärsprocesser. Adoptionen bland programvaruutvecklande företag är vad det verkar låg. Är man ett systemutvecklande företag innebär en investering i en teknik som fortfarande är så pass tidig i sin livscykel därför stora affärsrisker men det innebär också betydande fördelar eftersom man tidigt kan positionera sig och ta marknadsandelar när marknaden växer. Vid vilken tidpunkt man investerar bör hänga ihop med positionen på marknaden och företagets övergripande strategi, där vissa företag är mer beroende av att tidigt adoptera ny teknik än andra. Huruvida lönsamheten lyfter är svårt att säga. Ser man på tekniken verkar det som att den under 2009 och 2010 kommer att utvecklas och bli mer mogen. När det gäller standarden har arbetet med standardiseringen av BPEL4People just startat, något som kommer addera mer värde till tekniken. Känslan är också att fokuset lämnat ”roundtripping” och att tillverkarna nu fokuserar på BPEL med egna grafiska gränssnitt eller med BPMN som gränssnitt, alternativt på en exekverbar variant av BPMN.



Figur 36 - Förtjänster och besparingar jämfört med kostnader för investering i BPEL med BPMN 4, anskaffnings-, arbets-, tekniska och strategiska aspekter

Avslutningsvis kan man konstatera att de totala kostnaderna enligt min uppfattning i dagsläget överstiger de totala förtjänsterna men att de största kostnaderna är kopplade till teknikutvecklingen och således minskar de med tiden. Med en positiv utveckling på området, framför allt stabila verktyg kopplade till BPEL v 2.0, standardisering av tillägg och kartläggningsbaserade utvecklingsverktyg där både process- och systemutvecklare kan använda samma verktyg, kan tekniken således vara mer mogen och lönsam inom ett par år.

6. Slutsats

Slutsatsen är att det idag finns fler nackdelar än fördelar, dvs. att det i allmänhet innebär en merkostnad att satsa på BPEL/BPMN. Slutsatsen bygger på en sammanvägning av flera olika faktorer, vissa till fördel för tekniken och andra till nackdel. En sammanvägning där alltså nackdelarna överväger.

Den största fördelen med tekniken är att den kan föra verksamhet och systemutveckling närmare varandra genom att tillhandahålla verktyg som båda parterna kan använda. Detta görs dock inte bäst genom konvertering mellan BPMN och BPEL. Konvertering kan visserligen ske från BPMN till BPEL (dock inte åt andra hållet), men för att resultatet skall bli bra måste man när man kartlägger i BPMN tänka som en programmerare och ta hänsyn till BPELs begränsningar. Dessutom kommer troligen skillnader mellan processkartan och det färdiga systemet att uppstå vilket kan leda till problem när processen förändras. Istället kan detta närmande göras med hjälp av de grafiska gränssnitt som de flesta utvecklingsverktyg erbjuder. Dessa är relativt lätta att förstå och följa för någon som är van vid att läsa processkartor och därför kan man skapa en gemensam bild av systemet och processen. Vissa tillverkare erbjuder verktyg som använder BPMN-notationen i det grafiska gränssnittet vilket såklart gör tekniken ännu mer fördelaktig.

Den näst största fördelen är att tekniken är specialanpassad till web services och orkestrering av dessa. Det gör att det är smidigt att arbeta med samordning av web servicerna utan att behöva skapa proxies för översättning till och från web servicestandarderna. Man har också möjlighet att skapa högre abstraktion genom att BPEL använder sig av porttypen istället för de faktiska portarna. Det finns såklart en baksida av att vara specialiserad, nämligen att man har stora begränsningar. BPEL är begränsat i förhållande till exempelvis Java och inte lika smidigt att hantera och manipulera data i. Begränsningarna väger i dagsläget upp fördelarna med specialiseringen men med fler tillägg och ökad utveckling på området som exempelvis tillägg för mänsklig interaktion och inbakad Javakod kommer denna nackdel minska.

En annan faktor som är till nackdel för BPEL är att prestandan på web services byggda i BPEL verkar vara lägre än hos web services byggda i Java, när det kommer till exekveringshastighet. Detta är kanske inte så konstigt eftersom BPEL har avsevärt färre år på nacken än Java och således inte utvecklats lika långt.

Den största nackdelen, som idag är helt övervägande, är att tekniken inte har mognat. Samtliga av de verktyg och servermiljöer som provats innehåller en mängd kända och okända buggar och flera är inte färdigutvecklade. Detta gäller framförallt verktyg som relaterar till BPEL v 2.0, som är den senaste versionen. De verktyg som relaterar till v 1.1 av BPEL är visserligen stabilare, men har inte uppdaterats eller utvecklats på flera år. Detta gör att de riskerar att inte inkluderas i kommande versioner av utvecklingsverktyg och serverprogramvara, eftersom det inte finns någon kompabilitet mellan BPEL v 1.1 och BPEL v 2.0,. Något som resulterar i stora kostnader för migrering av system byggda i dessa miljöer. Dock är detta en nackdel som kommer bli mindre med fortsatt utveckling.

Till dessa nackdelar kommer också att en investering innehåller ett anskaffningsmoment som i det här fallet innebär vissa inköp av programvara men framförallt att lära sig ett nytt

programmeringsspråk. Denna nackdel är i sammanhanget ganska liten då språket är relativt lätt att lära sig samtidigt som utvecklingsmiljöerna antingen är gratis eller finns som tillägg hos de flesta större tillverkare och kan användas på redan anskaffade licenser.

Hur dessa faktorer väger mot varandra beror förstås på vad man som företag har för affärsidé och strategi. Tekniken är spridd och stöds av alla större tillverkare och standarden är upptagen av en större standardiseringsorganisation. Detta innebär att adoptionen av BPEL bland tillverkarna är stor. Dock finns det få exempel på implementeringar och användning av tekniken eftersom många verktyg fortfarande är lite omogna och bland systemutvecklande företag är det kommersiella värdet högt endast om man har en företagsidé och en strategi där tidig adoption av nya tekniker inom BPM/SOA-området är central.

Hur mycket varje fördel respektive nackdel är värd är såklart svårt att uppskatta och beror på hur verksamheten ser ut, rent allmänt är min slutsats att nackdelarna överväger fördelarna i dagsläget för de allra flesta företag som inte redan adopterat tekniken, men att många av dessa nackdelar med vidare utveckling kan komma att försvinna och att tekniken är värd att bevaka och utvärdera om ett par år igen.

7. Källförteckning

Agrawal, A., Amend, M., Das, M., Ford, M., Keller, C., Koppmann, M., o.a. (Juni 2007). *WS-BPEL Extension for People (BPEL4People), Version 1.0.* Hämtat från IBM: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/BPEL4People_v1.pdf den 2 Juni 2009

Amend, M., Agrawal, A., Das, M., Ford, M., Keller, C., Koppmann, M., o.a. (Juni 2007). *Web Services Human Task (WS-HumanTask), Version 1.0.* Hämtat från IBM: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/WS-HumanTask_v1.pdf den 2 Juni 2009

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., o.a. (den 5 Maj 2003). *Business Process Execution Language for Web Services Version 1.1* . Hämtat från IBM: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf den 10 Oktober 2009

Axway, IBM, Internationals, M., Oracle, AG, S., Unisys, o.a. (den 22 Maj 2009). *Proposal for: Business Process Model and Notation (BPMN) Specification 2.0 V0.9.14 (revised submission draft).* Hämtat från OMG: http://www.omg.org/docs/bmi/09-05-03.pdf den 29 Maj 2009

Blow, M., Goland, Y., Koppmann, M., Leymann, F., Pfuh, G., Roller, D., o.a. (Mars 2004). *BPELJ: BPEL for Java.* Hämtat från IBM: http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-bpelj/ws-bpelj.pdf den 2 Juni 2009

Bohlen, J. M., & Beal, G. M. (1957). *"The Diffusion Process" Special Report No. 18.* Iowa State College, Agriculture Extension Service.

Decker, G. (den 28 Januari 2009). *BPMN 1.2 published - nothing changed*. Hämtat från BPMN.info: http://www.bpmn.info/ den 29 Maj 2009

Dubray, J.-J. (den 4 December 2007). *The Seven Fallacies of Business Process Execution.* Hämtat från InfoQ: http://www.infoq.com/articles/seven-fallacies-of-bpm den 11 Juni 2009

Gardner, D. (den 16 Mars 2009). *BPEL4People: Remaining Human While Automating Processes*. Hämtat från E-commerce Times: http://www.ecommercetimes.com/story/66503.html den 2 Juni 2009

Haas, H., & Brown, A. (den 11 Februari 2004). *Web Services Glossary.* (H. Haas, & A. Brown, Red.) Hämtat från W3C Working Group Note: http://www.w3.org/TR/ws-gloss/ den 22 April 2009

Harmon, P. (2003). *Business process change.* Morgan Kaufmann.

Hartman, J. (2004). *Vetenskapligt tänkande – Från kunskapsteori till metodteori* (Andra upplagan uppl.). Lund: Studentlitteratur.

Igbanuelos. (Juni 2008). *bpmn2bpel*. Hämtat från Google Code: http://bpmn2bpel.googlecode.com/files/Installation.pdf Juni 2009

Jordan, D., & Evdemon, J. (den 11 April 2007). *Web Services Business Process Execution Language Version 2.0.* Hämtat från OASIS: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf den 4 Juni 2009

Juric, M. B., & Pant, K. (2008). *Business Process Driven SOA using BPMN and BPEL.* Birmingham: Packt Publishing Ltd.

Juric, M. B., Mathew, B., & Sarang, P. (2006). *Business Process Execution Language for Web Services* (2:a upplagan uppl.). Birmingham: Packt Publishing Ltd.

Koppmann, M., König, D., Leymann, F., Pfuh, G., Rickayzen, A., von Riegen, C., o.a. (Juli 2005). *WS-BPEL Extension for People – BPEL4People.* Hämtat från IBM: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/BPEL4People_white_paper.pdf den 2 Juni 2009

König, D. (den 29 Oktober 2007). *Web Services Business Process Execution Language (WS-BPEL 2.0) The Standards Landscape.* Hämtat från Open Standards Forum 2007: http://events.oasis-open.org/home/sites/events.oasis-open.org.home/files/Koenig.ppt den 10 Oktober 2009

König, D., Koppmann, M., Leymann, F., Pfuh, G., Rickayzen, A., von Riegen, C., o.a. (September 2005). *WS-BPEL Extension for Sub-processes – BPEL-SPE*. (IBM och SAP) Hämtat från IBM: http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-bpelsubproc/ws-bpelsubproc.pdf den 2 Juni 2009

Lauesen, S. (2002). *Software Requirements - Styles and Techniques.* Harlow: Pearson Education Ltd.

Levitt, T. (1965). Exploit the product life cycle. *Harvard Business Review , 43*.

Levitt, T. (2006). *Ted Levitt on Marketing.* Boston: Harvard Business School Publishing Corporation.

Leymann, F., Roller, D., & Thatte, S. (Agusti 2003). *Goals of the BPEL4WS Specification.* Hämtat från OASIS: http://xml.coverpages.org/BPEL4WS-DesignGoals.pdf den 2 Juni 2009

Ljungberg, A., & Larsson, E. (2001). *Processbaserad verksamhetsutveckling.* Lund: Studentlitteratur AB.

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., & Metz, R. (den 26 Oktober 2006). *Reference Model for Service Oriented Architecture 1.0.* Hämtat från OASIS SOA Reference Model TC: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf den 11 Maj 2009

OMG & BPMI. (den 29 Juni 2005). *BPMI.org and OMG Announce Strategic Merger of Business Process Management Activities.* Hämtat från OMG: http://www.omg.org/news/releases/pr2005/06-29-05.htm den 28 Maj 2009

OMG. (den 24 September 2009). *BPMN Supporters*. (OMG) Hämtat från BPMN.org: http://www.bpmn.org/BPMN_Supporters.htm den 7 Oktober 2009

Ouyang, C., Dumas, M., van der Aalst, W. M., & ter Hofstede, A. H. (2006). From BPMN Process Models to BPEL Web Services. *Proceedings of the 4th International Conference on Web Services* , 285–292.

Ouyang, C., Dumas, M., van der Aalst, W. M., & ter Hofstede, A. H. (Januari-Mars 2008). Pattern-based translation of BPMN Process Models to BPEL Web Services. *International Journal of Web Services Research , 5*, ss. 42-62.

Rogers, E. M. (2003). *Diffusion of Innovations, Fifth Edition.* New York: Free Press.

Royce, W. W. (Augusti 1970). Managing the Development of Large Software Systems: Concepts and Techniques. *Technical Papers of Western Electronic Show and Convention (WesCon)* , ss. 25-28.

Schumm, D., Karastoyanova, D., Leymann, F., & Nitzsche, J. (2009). On Visualizing and Modelling BPEL with BPMN. *2009 Workshops at the Grid and Pervasive Computing Conference* , ss. 80-87.

Silver, B. (den 18 Mars 2009). *BPMS Watch: Five Things to Love About BPMN 2.0*. Hämtat från BPMInstitute.org: http://www.bpminstitute.org/articles/article/article/bpms-watch-five-things-to-love-about-bpmn-2-0.html den 29 Maj 2009

Silver, B. (den 4 April 2008). *More on BPMN-to-BPEL.* Hämtat från BPMS Watch - Bruce Silver's blog on business process management: http://www.brsilver.com/wordpress/2008/04/04/more-on-bpmn-to-bpel/ den 11 Juni 2009

Sommerville, I. (2007). *Software engineering* (8 upplagan uppl.). Essex: Pearson Education Ltd.

Taylor, P. (den 29 Januari 2009). *BPMN Version 1.2*. Hämtat från BPM in the Land of OZ: http://homepage.mac.com/paul_taylor/blog/bpm-blog.html den 29 Maj 2009

Thatte, S., Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., o.a. (den 21 Juli 2002). *Business Process Execution Language for Web Services, Version 1.0.* Hämtat från IMB: WS-Reliable Messaging, definierar en standard för pålitlig leverans av meddelanden mellan parterna. den 9 Oktober 2009

The Open Group. (den 29 April 2009). *SOA Scource Book.* Hämtat från theopengroup.com: http://www.opengroup.org/projects/soa-book/ den 11 Maj 2009

Vambenepe, W. (den 11 Mars 2008). *BPMN to BPEL: going to battle with one hand tied?* Hämtat från William Vambenepe's blog: http://stage.vambenepe.com/archives/177 den 11 Juni 32009

WebMediaBrands Inc. (2009). *What is livelock? - A Word Definition from the Webopedia Computer Dictionary*. Hämtat från internet.com: http://www.webopedia.com/TERM/L/livelock.html den 5 Juni 2009

WebMediaBrands Inc. (2009). *What is deadlock? - A Word Definition from the Webopedia Computer Dictionary*. Hämtat från internet.com: http://www.webopedia.com/TERM/D/deadlock.html den 5 Juni 2009

White, S. A. (den 3 Maj 2004). *Business Process Modeling Notation (BPMN) v 1.0.* Hämtat från OMG: http://www.bpmn.org/Documents/BPMN%20V1-0%20May%203%202004.pdf den 28 Maj 2009

White, S. A. (den 3 Januari 2009). *Business Process Modeling Notation (BPMN) Version 1.2.* Hämtat från OMG: http://www.omg.org/docs/formal/09-01-03.pdf den 29 Maj 2009

White, S. A. (den 17 Januari 2008). *Business Process Modeling Notation, V1.1.* Hämtat från OMG: http://www.bpmn.org/Documents/BPMN%201-1%20Specification.pdf den 29 Maj 2009

Vigneras, P. (den 21 Oktober 2008). *Why BPEL is not the holy grail for BPM.* Hämtat från InfoQ: http://www.infoq.com/articles/bpelbpm den 11 Juni 2009

Wikipedia. (den 3 September 2009). *Comparison of BPEL engines*. Hämtat från Wikipedia: http://en.wikipedia.org/wiki/Comparison_of_BPEL_engines den 5 Oktober 2009

Wright, T. P. (1936). Factors Affecting the Cost of Airplanes. *Journal of Aeronautical Sciences , 3* (4).

zur Muelhlen, M. (2007). *Business Process Management Standards Tutorial.* Hämtat från SlideShare: http://www.slideshare.net/mzurmuehlen/business-process-management-standards-tutorial den 11 Juni 2009

Appendix

A. Beskrivning av syntax, BPEL

Element

I figuren till höger finns den grundläggande strukturen. Denna struktur gäller för exekverbara processer, all exempelkod är hämtade från (Jordan & Evdemon, 2007). De abstrakta processerna har en något annorlunda uppbyggnad, se Jordan & Evdemon, 2007, s. 147.

Processblocket**, <process>** utgör rotelementet i BPEL, nedan återfinns strukturen för processblocket, den exekverbara varianten.

```
<process name="NCName"
   targetNamespace="anyURI"
   queryLanguage="anyURI"?
   expressionLanguage="anyURI"?
   suppressJoinFailure="yes|no"?
   exitOnStandardFault="yes|no"?
   xmlns="http://docs.oasis-
          open.org/wsbpel/2.0/process/exec
utable">
…
</process>
```

I processblocket finns åtta block med olika typer av deklarationer, dessa återföljs sedan av själva processlayouten. De åtta blocken är:

<extension>, här finns det möjlighet att utöka BPEL med tillägg exempelvis BPEL4People eller BPELJ.

```
   <extensions>?
      <extension namespace="anyURI"

mustUnderstand="yes|no" />+
   </extensions>
```

I **<import>**, finns det möjlighet att importera dokument exempelvis WDSL och XML schema.

```
<import namespace="anyURI"?
      location="anyURI"?
      importType="anyURI" />*
```

<partnerLinks>, är ett av de centrala blocken i BPEL, här specificeras hur olika parter interagerar och vad varje part erbjuder. En ”partnerlänk” innehåller bara en referens till typen av ”partnerlänk” som avses. Partnerlänkstyperna <partnerLinkType> specificerar en eller flera affärsrelationer mellan två parter, så kallade roller, Dessa

<partnerLinkType> specificeras i WSDL. Varje aktivitet i BPEL kräver en ”partnerlänk”.

```
<partnerLinks>?
   <!-- Note: At least one role must be specified. -->
   <partnerLink name="NCName"
      partnerLinkType="QName"
      myRole="NCName"?
      partnerRole="NCName"?
      initializePartnerRole="yes|no"?>+
   </partnerLink>
</partnerLinks>
```

<messageExchange>, definierar olika meddelandeutbyten. Ett definierat meddelandeutbyte kan sedan koppla ett ”receive”-aktivitet med en ”reply”-aktivitet.

```
<messageExchanges>?
      <messageExchange name="NCName" />+
   </messageExchanges>
```

<variables>, är variabler för att spara data som skickas mellan olika affärspartner, den kan innehålla messageType, ett WSDL meddelande; type, ett XML Schema simpel type och element, ett XML-Schemaelement.

```
<variables>?
      <variable name="BPELVariableName"
         messageType="QName"?
         type="QName"?
         element="QName"?>+
         from-spec?
      </variable>
   </variables>
```

<correlationSets>, är en uppsättning av properties som används för att korrelera och hålla reda på meddelanden och vart de skall.

```
<correlationSets>?
      <correlationSet name="NCName" properties="QName-list" />+
   </correlationSets>
```

<faultHandlers>, här definieras de olika fel som kan fångas och vad som händer när de fångas.

```
<faultHandlers>?
   <!-- Note: There must be at least one faultHandler -->
   <catch faultName="QName"?
      faultVariable="BPELVariableName"?
       ( faultMessageType="QName" | faultElement="QName" )?
>*
      activity
   </catch>
   <catchAll>?
      activity
   </catchAll>
</faultHandlers>
```

<eventHandlers>, hanterar händelser som kan löpa parallellt med de huvudsakliga flödena som svar på olika händelser eller när en viss tid har gått. Det finns två händelser, <onEvent>, som hanterar händelser och <onAlarm>, som hanterar tid.

```
           <eventHandlers>?
                 <!-- Note: There must be at least one onEvent or onAlarm.
-->
                 <onEvent partnerLink="NCName"
                    portType="QName"?
                    operation="NCName"
                     ( messageType="QName" | element="QName" )?
                    variable="BPELVariableName"?
                    messageExchange="NCName"?>*
                    <correlations>?
                       <correlation set="NCName" initiate="yes|join|no"?
/>+
                    </correlations>
                    <fromParts>?
                       <fromPart part="NCName"
toVariable="BPELVariableName" />+
                    </fromParts>
                    <scope ...>...</scope>
                 </onEvent>
                 <onAlarm>*
                    <!-- Note: There must be at least one expression. -->
                     (
                    <for expressionLanguage="anyURI"?>duration-expr</for>
                    |
                    <until expressionLanguage="anyURI"?>deadline-
expr</until>
                    )?
                    <repeatEvery expressionLanguage="anyURI"?>
                       duration-expr
                    </repeatEvery>?
                    <scope ...>...</scope>
                 </onAlarm>
              </eventHandlers>
```

Strukturen för dokumentet ser därför sammanfattningsvis ut som följer:

```
           <process name="NCName" targetNamespace="anyURI"
              queryLanguage="anyURI"?
              expressionLanguage="anyURI"?
              suppressJoinFailure="yes|no"?
              exitOnStandardFault="yes|no"?
              xmlns="http://docs.oasis-
open.org/wsbpel/2.0/process/executable">

              <extensions>?
                 <extension namespace="anyURI" mustUnderstand="yes|no" />+
              </extensions>

              <import namespace="anyURI"?
```

```
                location="anyURI"?
                importType="anyURI" />*

             <partnerLinks>?
                <!-- Note: At least one role must be specified. -->
                <partnerLink name="NCName"
                   partnerLinkType="QName"
                   myRole="NCName"?
                   partnerRole="NCName"?
                   initializePartnerRole="yes|no"?>+
                </partnerLink>
             </partnerLinks>

             <messageExchanges>?
                <messageExchange name="NCName" />+
             </messageExchanges>

             <variables>?
                <variable name="BPELVariableName"
                   messageType="QName"?
                   type="QName"?
                   element="QName"?>+
                   from-spec?
                </variable>
             </variables>

             <correlationSets>?
                <correlationSet name="NCName" properties="QName-list" />+
             </correlationSets>

             <faultHandlers>?
                <!-- Note: There must be at least one faultHandler -->
                <catch faultName="QName"?
                   faultVariable="BPELVariableName"?
                    ( faultMessageType="QName" | faultElement="QName" )?
>*
                   activity
                </catch>
                <catchAll>?
                   activity
                </catchAll>
             </faultHandlers>

             <eventHandlers>?
                <!-- Note: There must be at least one onEvent or onAlarm.
-->
                <onEvent partnerLink="NCName"
                   portType="QName"?
                   operation="NCName"
                    ( messageType="QName" | element="QName" )?
                   variable="BPELVariableName"?
                   messageExchange="NCName"?>*
                   <correlations>?
```

```
                        <correlation set="NCName" initiate="yes|join|no"?
/>+
                    </correlations>
                    <fromParts>?
                        <fromPart part="NCName"
toVariable="BPELVariableName" />+
                    </fromParts>
                    <scope ...>...</scope>
                </onEvent>
                <onAlarm>*
                    <!-- Note: There must be at least one expression. -->
                     (
                    <for expressionLanguage="anyURI"?>duration-expr</for>
                    |
                    <until expressionLanguage="anyURI"?>deadline-
expr</until>
                    )?
                    <repeatEvery expressionLanguage="anyURI"?>
                        duration-expr
                    </repeatEvery>?
                    <scope ...>...</scope>
                </onAlarm>
            </eventHandlers>
            activity
        </process>
```

Efter alla deklarationer kommer själva aktiviteterna, kroppen i BPEL-dokumentet.

Aktiviteter

Det finns 21 aktiviteter i BPEL. De kan delas in i strukturerade och enkla aktiviteter.

Enkla aktiviteter

<assign>, kopiera ett eller flera värden till variabler och ”partnerlänkar”.

```
<assign validate="yes|no"? standard-attributes>
   standard-elements
    (
   <copy keepSrcElementName="yes|no"? ignoreMissingFromData="yes|no"?>
      from-spec
      to-spec
   </copy>
   |
   <extensionAssignOperation>
      assign-element-of-other-namespace
   </extensionAssignOperation>
   )+
</assign>
```

<compensate>, åberopar (invoke) en uppsättning kompensationshanterare, används vid fel för att starta kompenserade åtgärder.

```
<compensate standard-attributes>
```

```
   standard-elements
</compensate>
```

<compensateScope>, samma som <compensate> fast åberopar kompensationshanterare i ett visst <scope>.

```
<compensateScope target="NCName" standard-attributes>
   standard-elements
</compensateScope>
```

<empty>, representerar en tom aktivitet.

```
<empty standard-attributes>
   standard-elements
</empty>
```

<exit>, avslutar processen omedelbart, hette tidigare <terminate>.

```
<exit standard-attributes>
   standard-elements
</exit>
```

<extensionActivity>, aktiviteter som definieras av utökningar/tillägg till BPEL.

```
<extensionActivity>
   <anyElementQName standard-attributes>
      standard-elements
   </anyElementQName>
</extensionActivity>
```

<invoke>, åberopar en partnerservice.

```
<invoke partnerLink="NCName"
   portType="QName"?
   operation="NCName"
   inputVariable="BPELVariableName"?
   outputVariable="BPELVariableName"?
   standard-attributes>
   standard-elements
   <correlations>?
      <correlation set="NCName" initiate="yes|join|no"?
         pattern="request|response|request-response"? />+
   </correlations>
   <catch faultName="QName"?
      faultVariable="BPELVariableName"?
      faultMessageType="QName"?
      faultElement="QName"?>*
      activity
   </catch>
   <catchAll>?
      activity
   </catchAll>
   <compensationHandler>?
      activity
   </compensationHandler>
```

```
   <toParts>?
      <toPart part="NCName" fromVariable="BPELVariableName" />+
   </toParts>
   <fromParts>?
      <fromPart part="NCName" toVariable="BPELVariableName" />+
   </fromParts>
</invoke>
```

<receive>, låter processen vänta på ett visst meddelande.

```
<receive partnerLink="NCName"
   portType="QName"?
   operation="NCName"
   variable="BPELVariableName"?
   createInstance="yes|no"?
   messageExchange="NCName"?
   standard-attributes>
   standard-elements
   <correlations>?
      <correlation set="NCName" initiate="yes|join|no"? />+
   </correlations>
   <fromParts>?
      <fromPart part="NCName" toVariable="BPELVariableName" />+
   </fromParts>
</receive>
```

<reply>, låter processen svara på ett meddelande.

```
<reply partnerLink="NCName"
   portType="QName"?
   operation="NCName"
   variable="BPELVariableName"?
   faultName="QName"?
   messageExchange="NCName"?
   standard-attributes>
   standard-elements
   <correlations>?
      <correlation set="NCName" initiate="yes|join|no"? />+
   </correlations>
   <toParts>?
      <toPart part="NCName" fromVariable="BPELVariableName" />+
   </toParts>
</reply>
```

<rethrow>, kastar vidare ett fel från en felhanterare till en annan.

```
<rethrow standard-attributes>
   standard-elements
</rethrow>
```

<throw>, kastar fel till en felhanterare.

```
<throw faultName="QName"
   faultVariable="BPELVariableName"?
```

```
   standard-attributes>
   standard-elements
</throw>
```

<validate>, validerar en eller flera variabler mot deras datatyp.

```
<validate variables="BPELVariableNames" standard-attributes>
   standard-elements
</validate>
```

<wait>, är en aktivitet som väntar en viss tid eller till in viss tidsangivelse.

```
<wait standard-attributes>
   standard-elements
    (
   <for expressionLanguage="anyURI"?>duration-expr</for>
   |
   <until expressionLanguage="anyURI"?>deadline-expr</until>
   )
</wait>
```

Strukturerade aktiviteter

<extensionActivity> aktiviteter som definieras av utökningar/tillägg till BPEL, kan vara både strukturerad och ostrukturerad.

<if>, logiskt om-villkor.

```
<if standard-attributes>
   standard-elements
   <condition expressionLanguage="anyURI"?>bool-expr</condition>
   activity
   <elseif>*
      <condition expressionLanguage="anyURI"?>bool-expr</condition>
      activity
   </elseif>
   <else>?
      activity
   </else>
</if>
```

<forEach>, gör något N+1 gånger.

```
<forEach counterName="BPELVariableName" parallel="yes|no"
   standard-attributes>
   standard-elements
   <startCounterValue expressionLanguage="anyURI"?>
      unsigned-integer-expression
   </startCounterValue>
   <finalCounterValue expressionLanguage="anyURI"?>
      unsigned-integer-expression
   </finalCounterValue>
   <completionCondition>?
      <branches expressionLanguage="anyURI"?
         successfulBranchesOnly="yes|no"?>?
```

```
         unsigned-integer-expression
      </branches>
   </completionCondition>
   <scope ...>...</scope>
</forEach>
```

<flow>, definierar parallella flöden av aktiviteter.

```
<flow standard-attributes>
   standard-elements
   <links>?
      <link name="NCName" />+
   </links>
   activity+
</flow>
```

<pick>, används för att vänta på att meddelanden kommer eller tidsvillkor uppfylls.

```
<pick createInstance="yes|no"? standard-attributes>
   standard-elements
   <onMessage partnerLink="NCName"
      portType="QName"?
      operation="NCName"
      variable="BPELVariableName"?
      messageExchange="NCName"?>+
      <correlations>?
         <correlation set="NCName" initiate="yes|join|no"? />+
      </correlations>
      <fromParts>?
         <fromPart part="NCName" toVariable="BPELVariableName" />+
      </fromParts>
      activity
   </onMessage>
   <onAlarm>*
       (
      <for expressionLanguage="anyURI"?>duration-expr</for>
      |
      <until expressionLanguage="anyURI"?>deadline-expr</until>
      )
      activity
   </onAlarm>
</pick>
```

<repeatUntil>, repetera tills ett visst villkor är uppfyllt.

```
<repeatUntil standard-attributes>
   standard-elements
   activity
   <condition expressionLanguage="anyURI"?>bool-expr</condition>
</repeatUntil>
```

<scope>, används för att skapa “nästade” (nested) aktiviteter med egna associerade <partnerLinks>, <messageExchanges>, <variables>, <correlationSets>, <faultHandlers>, <compensationHandler>, <terminationHandler>, och <eventHandlers>.

```
<scope isolated="yes|no"? exitOnStandardFault="yes|no"?
   standard-attributes>
   standard-elements
   <partnerLinks>?
      ... see above under <process> for syntax ...
   </partnerLinks>
   <messageExchanges>?
      ... see above under <process> for syntax ...
   </messageExchanges>
   <variables>?
      ... see above under <process> for syntax ...
   </variables>
   <correlationSets>?
      ... see above under <process> for syntax ...
   </correlationSets>
   <faultHandlers>?
      ... see above under <process> for syntax ...
   </faultHandlers>
   <compensationHandler>?
      ...
   </compensationHandler>
   <terminationHandler>?
      ...
   </terminationHandler>
   <eventHandlers>?
      ... see above under <process> for syntax ...
   </eventHandlers>
   activity
</scope>
```

<sequence>, en sekventiell exekvering av aktiviteter.

```
<sequence standard-attributes>
   standard-elements
   activity+
</sequence>
```

<while>, loopar tills ett visst villkor är uppfyllt.

```
<while standard-attributes>
   standard-elements
   <condition expressionLanguage="anyURI"?>bool-expr</condition>
   activity
</while>
```

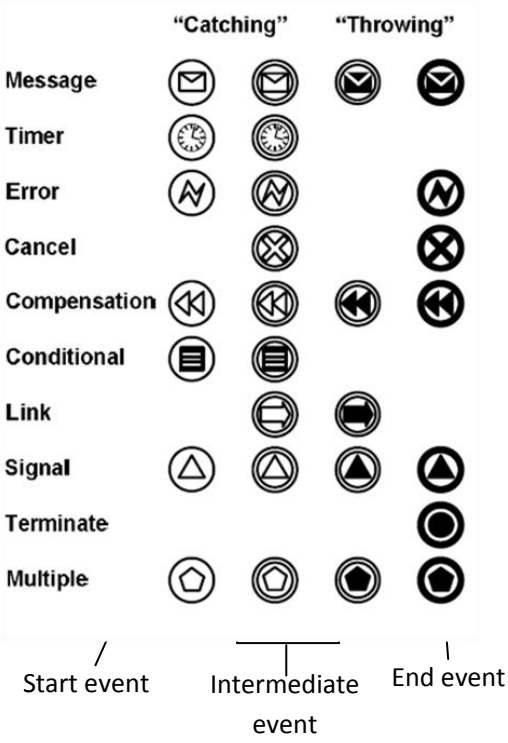
B. Beskrivning av syntax, BPMN

Element

BPMN är en grafisk notation och består därför av olika grafiska element. Dessa element är indelade i fem huvudkategorier:

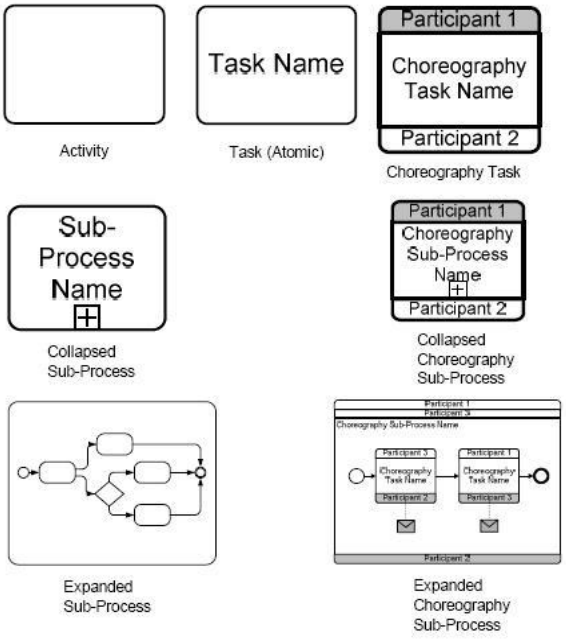
1. Flow Objects (flödesobjekt),

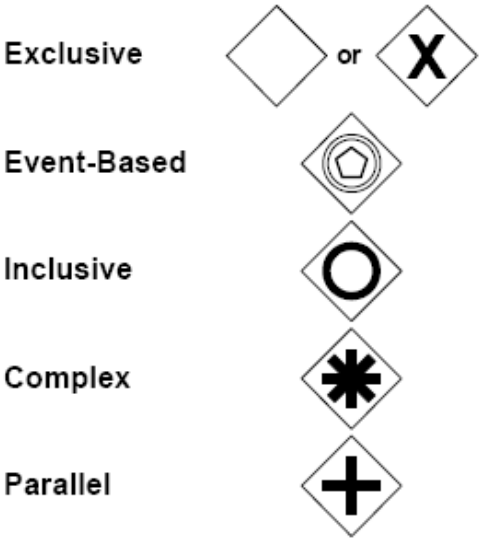
Vilka kan delas in i tre underkategorier:



Activities, aktiviteter som består av själva uppgifterna vara både enkla och sammansatta. Det finns också olika undertyper beroende på om de exempelvis skall loopas eller vem som skall utföra dem exempelvis människa eller maskin.

Events, händelser av olika typ som inträffar när en process exempelvis meddelanden, fel, timers och signaler. Det finns fyra olika typer av händelser indelande i om händelsen skall fångas eller kastas. Starthändelser, startar en ny processinstans när de fångar en händelse. Mellanliggande händelser finns som både kastande och fångande, den kastande varianten kastar en händelser och fortsätter processen medan den fångande fortsätter processen först när en händelse fångats. Sluthändelser kastar en händelse i slutet av en process.

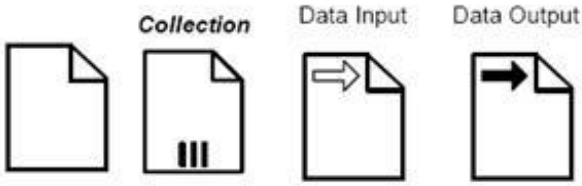




Gateways, portar är objekt som kontrollerar konvergens och divergens i flödesdiagrammet, dvs. hur olika flöden delar på sig och slås ihop.

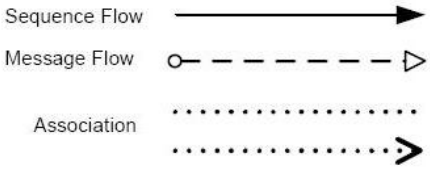
2. *Data (dataobjekt)*

Innehåller information om aktiviteter och vad de gör. De kan vara enkla eller sammanbuntade.



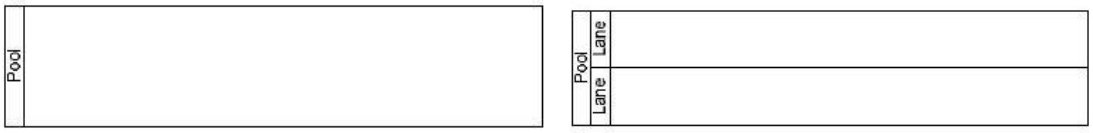
3. *Connecting Objects (kopplingsobjekt)*

Kopplar ihop och visar ordningen för aktiviteter (sequence ***flow***), meddelandeutbyte (message ***flow***) och artefakter (***association***) och data (***data association***) med aktiviteter.



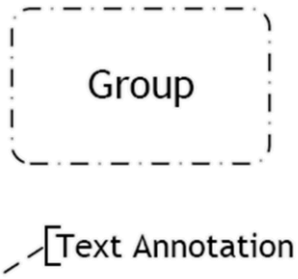
4. *Swimlanes (simbanor)*

Som består av ***pools (poler)*** med ***lanes (banor)***. En Pool representerar en deltagare i ett samarbete och är också en grafisk container för ett antal aktiviteter (en process). En bana är en delmängd av en pool och kan dela poolen antingen vertikalt eller horisontellt i hela bolens längd. Den används för att organisera aktiviteter i poolen.



5. *Artefacts, (artefakter)*

Som består av ***groups (grupper)*** och ***text annotation (textkommentarer)***. Artefakterna är främst till för att det skall bli enklare att förstå diagrammen. Med grupper kan man gruppera liknande aktiviteter och textkommentarer är kommentarer som man kan koppla till andra objekt i diagrammet för ytterligare beskrivningar.



C. Instruktioner för installation av Oracle SOA Suite 10g för BPEL (av Gösta Yllner)

Här följer en beskrivning av hur man sätter upp och testkör en BPEL miljö av Oracle version 10.

Börja med att ladda ner Oracle SOA suite. Denna innehåller en applikationsserver och BPEL exekveringsmiljö. Vidare finns webbaserade konsoler för applikationsservern och BPEL processen.

Hämta installationsfiler för Oracle SOA suite 10g härifrån:

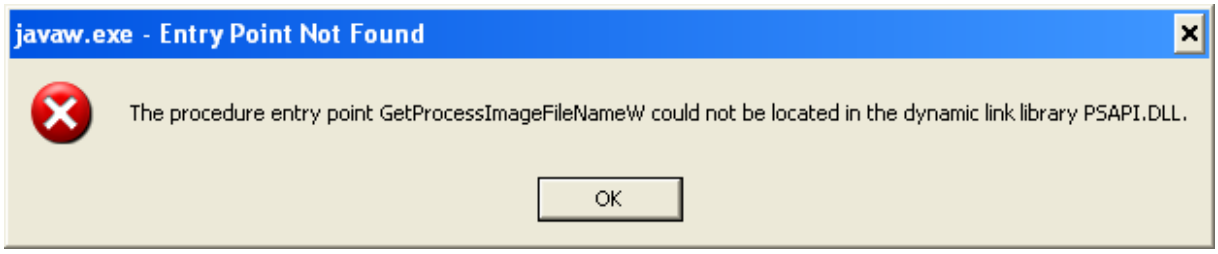
http://www.oracle.com/technology/software/products/ias/htdocs/101310.html

Skulle du promptas på inloggning till OTN (Oracle technical network) får du registrera dig om du inte redan är medlem.

Starta installationen och följ menyerna.

OBS! Glöm inte att anteckna lösenordet till admin användaren för applikationsservern, som du skall ange i en av menyerna. Detta måste du ha tillgång när du använder systemet senare.

När jag körde installationen fick jag ett felmeddelande enligt:



Jag bara klickade OK och har inte märkt av några problem.

Tanka ner zip fil för JDeveloper 10.1.3.4. Denna finns här:

http://www.oracle.com/technology/software/products/jdev/htdocs/soft10134.html

Välj Studio Edition

Zippa upp filen till någon lämplig folder

Starta applikationsserven i SOA suite m.h.a. Start/All Programs/Oracle-xxx/Start SOA suite

För att JDeveloper skall hitta till servern i SOA suite måste men skapa en koppling i JDeveloper till servern.

Starta JDeveloper och välj View/Connection Navigator. Connection Navigator dyker nu upp i den vänstra ramen. Högerklicka på Application Server och välj ”New application server connection” . En wizard öppnar sig nu. Ange följande data:

Connection name: Hitta på ett namn
Connection type: Oracle Application Server 10g 10.1.3
Username: oc4jadmin
Password: Det lösenord du angav under installationen av SOA suite

Connect to: Single instance
Host name: Namnet på din dator
OC4J Instance name: home (ja just det, det skall vara “home” och inget annat)
OPMN port: 6003

När man startar JDeveloper första gången kommer man till en ”Start page” med tips and tricks. Denna sida kan man också komma åt senare via Help/Start page. Här finns en guide för hur man skapar och deployar en ”Hello World” applikation med hjälp av Visual Designer .

Klicka på ”Define a BPEL process” och följ anvisningarna steg för steg.

När man har gått igenom guiden, har man en ”Hello World” applikation deployad på Applikationsservern i SOA suite. Det som återstår nu är att testköra den. Starta BEPEL konsolen via Start/All Programs/Oracle-xxx/Oracle BPEL Process manager/BPEL Control och logga in med användarnamn och password från serverinstallationen.

Längst till vänster finns en lista över deployade applikationer. Klicka på din ”Hello World” applikation.

Du kommer nu till en sida med många flikar, där varje flik innehåller olika typer av information om din applikation. Välj fliken ”Initiate”.

Välj ”HTML formulär” och fyll i ”World” (blanktecken följt av World utan citationstecken) i input fältet och klicka på knappen ”Lägg in XML meddelande”. BPEL applikationen körs nu och den returnerade XMLen visas på en ny sida. Om det blir rätt skall det se ut ungefär så här:

```
<BPELTestProcess1ProcessResponsehttp://xmlns.oracle.com/BPELTestProcess1>
  <resulthttp://xmlns.oracle.com/BPELTestProcess1>Hello World</result>
</BPELTestProcess1ProcessResponse>
```

D. Kravdokument – λTraveler

I detta appendix beskrivs den fiktiva produkten λTraveler som skall implementeras och de krav som finns på implementeringen av den. Kravhanteringen utgår, i form och metod, utifrån (Lauesen, 2002).

Elicitering

Eliciteringen utgår ifrån den beskrivning av affärsscenariot som finns ovan (se 4.2.2). Utifrån denna beskrivning har krav på mål och domännivå preciserats. Domänkraven har sedan legat till grund för produkt och designkrav.

För att ställa upp krav på produktnivå användes metoden domänkravsanalys.

Begränsningar

B1: λTraveler skall som en test endast implementeras för Lambda Omikrons resor.

B2: B1 innebär att de enda interna systemen som λTraveler använder är Omikrons interna system.

Krav på målnivå

B3: Ett antal krav som har med klienten att göra har satts inom parentes och kommer inte att implementeras eller testas.

Affärsmål

~~**M1:** λTraveler skall på ett enhetligt sätt automatisera processen för bokning av biljetter.~~

Struket då det är överflödigt och snarare medel för att uppnå M2 och M3.

M2: λTraveler skall minska direkta kostnaderna för resor inom koncernen.

M3: λTraveler skall skapa en konformitet för resebokning inom koncernen, som minskar komplexiteten och minskar administrationskostnader.

Krav på domännivå

Beskrivning av relaterade tjänster

I detta avsnitt beskrivs de relaterade tjänsterna och de gränssnitt som är relevanta för interaktionen med λTraveler.

D1: λTraveler skall stödja de gränssnitt som beskrivs nedan i detta stycke samt i blid 1.

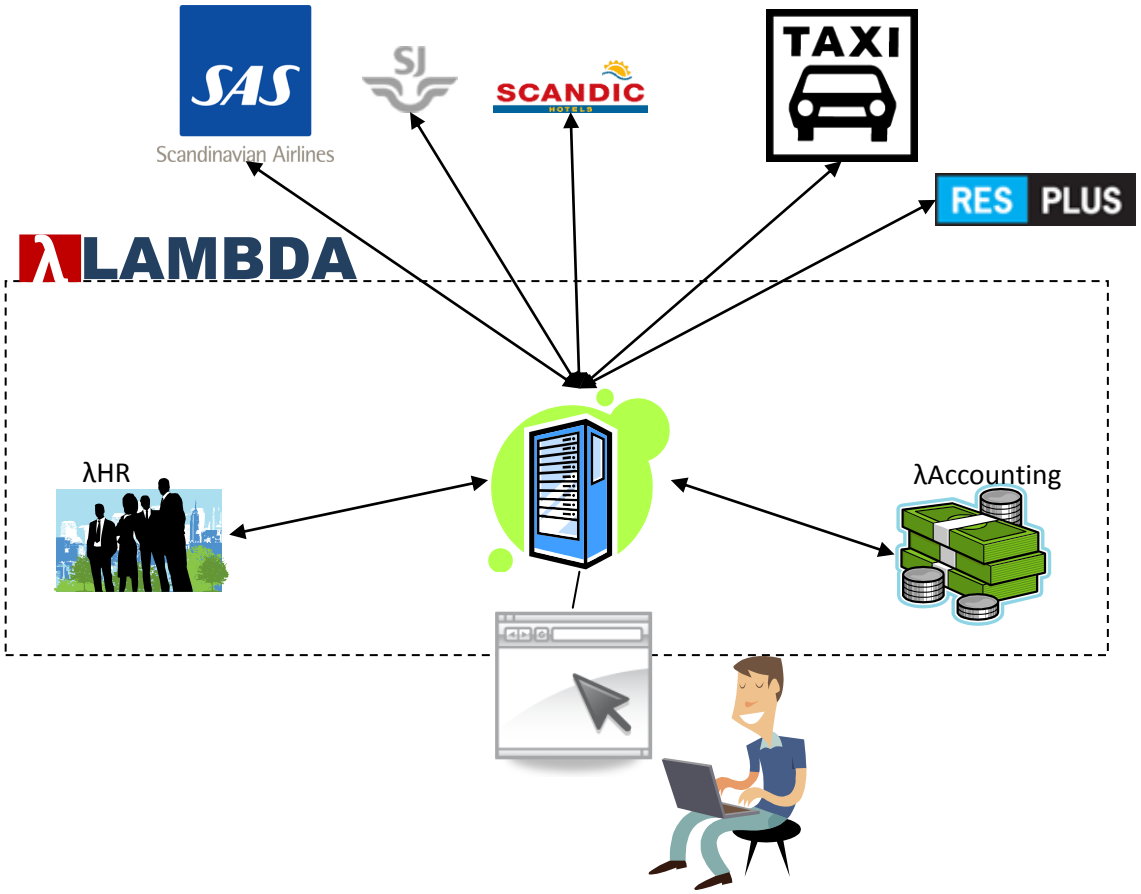


Bild 1 - Beskrivning av affärsmiljö för λTraveler

λHR är Lambda Omikrons personalsystem. Det håller reda på anställda och deras uppgifter. λTraveler hämtar och bekräftar personuppgifter och bokningsbehörighet härifrån.

getTravelerStatus:

Input: firstName *type: string*

lastName *type: string*

Output: getTravelStatusResponseElement

(travelStatus) Result *type: string (none, consultant, manager)*

λAccounting

*λ*Accounting är Lambda Omikrons bokföringssystem. Hit skickas fakturor på bokningar.

InvoiceBill:

Input: person:

firstName *type: string*

lastName *type: string*

travelStatus *type: string* (möjliga värden none; consultant; manager)

bookingReceipt *type: string* [1..*]

Output: LambdaAccountingProcessResponset

(Receipt) Result *type: string*

ResplusTravelRoute

ResplusTravelRoute tillhandahåller resruttsförslag mellan två destinationer ett visst tidsintervall för kombinerade resor.

getRoute:

Input: from *type: string*

to *type: string*

earliestStart *type: dateTime*

latestArrival *type: dateTime*

Output: ResplusTravelResponset (From *type: string*; To type*: string*; DepartureTime *type: dateTime*; ArrivalTime *type: dateTime*)

TravelSuggestion [1..*]

Route (tID *type: string*) [1..*]

From *type: string*

DepartureTime *type: dateTime*

To *type: string*

ArrivalTime *type: dateTime*

SASBooking

SASBooking är SAS bokningssystem, här kan tillgängliga biljetter och resedata på en viss flygning hämtas och bokningar göras.

getPrice:

Input: tID *type: string*

Output: getPriceResponseElement

(Price [-1=flyg ej tillgängligt]) Result *type: int*

getStatus:

Input: tID *type: string*

Output: getStatusResponseElement

(nbrOfSeats [-1=flyg ej tillgängligt]) Result *type: int*

bookTickets:

Input: tID *type: string*

nbrOfSeats *type: int*

Output: bookTicketsResponseElement
result *type: arrayList*

(meddelande) item *type: string*

(nbrOfSeats [-1=ej bokat]) item *type: int*

(totalPrice [-1=ej bokat])*)* item *type: string*

SJBooking

SJBooking är SJ bokningssystem, här kan tillgängliga biljetter och resedata på en viss avgång hämtas och bokningar göras.

getPrice:

Input: tID *type: string*

Output: getPriceResponseElement

(Price [-1=tåg ej tillgängligt]) Result *type: int*

getStatus:

Input: tID *type: string*

Output: getStatusResponseElement

(nbrOfSeats [-1=tåg ej tillgängligt]) Result *type: int*

bookTickets:

Input: tID *type: string*

nbrOfSeats *type: int*

Output: bookTicketsResponseElement
result *type: arrayList*

(meddelande) item *type: string*

(nbrOfSeats [-1=ej bokat]) item *type: int*

(totalPrice [-1=ej bokat])*)* item *type: string*

TaxiSverigeBooking

TaxiSverigeBooking är bokningssystem för taxiresor inom Sverige, här kan bokningar av taxiresor göras.

getPrice:

Input: from *type: string*

to *type: string*

Output: getPriceResponseElement

(Price [-1=taxi ej tillgängligt]) Result *type: int*

bookTaxi

Input: from *type: string*

to *type: string*

date *type: dateTime*

Output: bookTicketsResponseElement
(bokningsmeddelande) result *type: string*

ScandicBooking

Med hjälp av ScandicBooking kan bokningar av hotellrum göras på Scandic Hotels runt om i Sverige. Tjänsten kan även svara på om det finns tillgängliga rum på en viss ort.

hotelsIn:

Input: town *type: string*

Output: getHotelsInResponseElement
result *type: arrayList*

(meddelande/hotellnamn) item *type: string[1..*]*

getPrice:

Input: hotel *type: string*

Output: getPriceResponseElement

(Price [-1=hotel ej tillgängligt]) Result *type: int*

getStatus:

Input: hotel *type: string*

date *type: dateTime*

nbrOfDays *type: int*

Output: getStatusResponseElement

(nbrOfRooms [-1=hotel ej tillgängligt]) Result *type: int*

bookRooms:

Input: hotel *type: string*

date *type: dateTime*

nbrOfDBeds *type: int*

nbrOfDays *type: int*

Output: bookTicketsResponseElement
result *type: arrayList*

(meddelande) item *type: string*

(nbrOfBeds [-1=ej bokat]) item *type: int*

(nbrOfDays[-1=ej bokat])	item *type: int*
(totalPrice [-1=ej bokat])*)*	item *type: string*

Arbetsuppgiftsbeskrivningar

D2: λTraveler skall stödja följande arbetsuppgift:

Arbetsuppgift 1, Beställning av resa och boende

Uppgift:	Beställning av resa och boende för en användare.
Syfte:	Att tillhandahålla billigaste rese- och boendealternativet för den givna tidsramen.
Utlösare/Grundstatus:	Användaren skickar en förfrågan via sin klient.
Frekvens:	Ofta.
Kritiskt:	Den förfrågade resan startar samtidigt som förfrågan görs. Biljetter tar slut under exekvering.

Underuppgifter:

1	Ta emot förfrågan.
2	Kontrollera användarens resebehörighet med λHR.
3	Ta fram reseförslag.
4	Skicka reseförslag till klient.
5	Ta emot bekräftelse.
6	Boka resa.
7	Skicka bekräftelse till klient.
8	Förfrågan om boende på destinationen.
9	Ta emot bekräftelse.
10	Ta fram boendeförslag.
11	Skicka boendeförslag.
12	Ta emot bekräftelse.

13	Boka resa.
14	Skicka bekräftelse.

Varianter:

1a	Förfrågan är inkorrekt.
2a	Ej behörig.
2b	Användaren finns ej i databas.
2c	Ingen kontakt med λHR
3a	Ingen rutt tillgänglig.
3b	Ingen kontakt med ResplusTravelRoute.
3c	Inga tågbiljetter tillgängliga.
3d	Ingen kontakt med SJBooking
3e	Inga flygbiljetter tillgängliga.
3f	Ingen kontakt med SASBooking.
3g	Ingen kontakt med TaxiSverigeBooking.
(4a	*Ingen kontakt med klient.)*
(5a	*Ingen bekräftelse mottaget.)*
5b	Användare avböjer förslag.
6a	Inga tågbiljetter tillgängliga.
6b	Ingen kontakt med SJBooking
6c	Inga flygbiljetter tillgängliga.
6d	Ingen kontakt med SASBooking.

6e	Ingen kontakt med TaxiSverigeBooking.
6f	Ingen kontakt med λAcconting.
(7a	*Ingen kontakt med klient.)*
(8a	*Ingen kontakt med klient.)*
(9a	*Ingen bekräftelse mottaget.)*
9b	Användare avböjer förslag.
10a	Inga boenden tillgängliga.
10b	Ingen kontakt med ScandicBooking.
(11a	*Ingen kontakt med klient.)*
(12a	*Ingen bekräftelse mottaget.)*
12b	Användare avböjer förslag.
13a	Inga boenden tillgängliga.
13b	Ingen kontakt med ScandicBooking.
13c	Ingen kontakt med λAcconting.
14a	Ingen kontakt med klient.

Krav på produktnivå och designkrav

Tillstånd

P1: λTraveler skall ha tillstånd som definieras i bild 2.

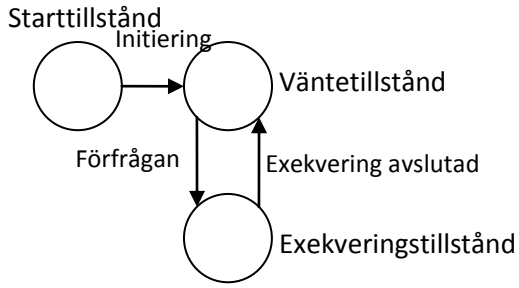


Bild 2 - Tillståndsgraf

Övergripande affärsflöde

I bild 3 finns en övergripande skiss på det övergripande affärsflödet mellan λTraveler och klienten, nummer som motsvarar nummer på underuppgifterna (se *D2*) är insatta för att koppla ihop flödet med arbetsuppgiften.

P3: Det övergripande processflödet för λTraveler skall löpa enligt bild 3.

P4: Processflödet motsvarar händelse inom Exekveringstillståndet.

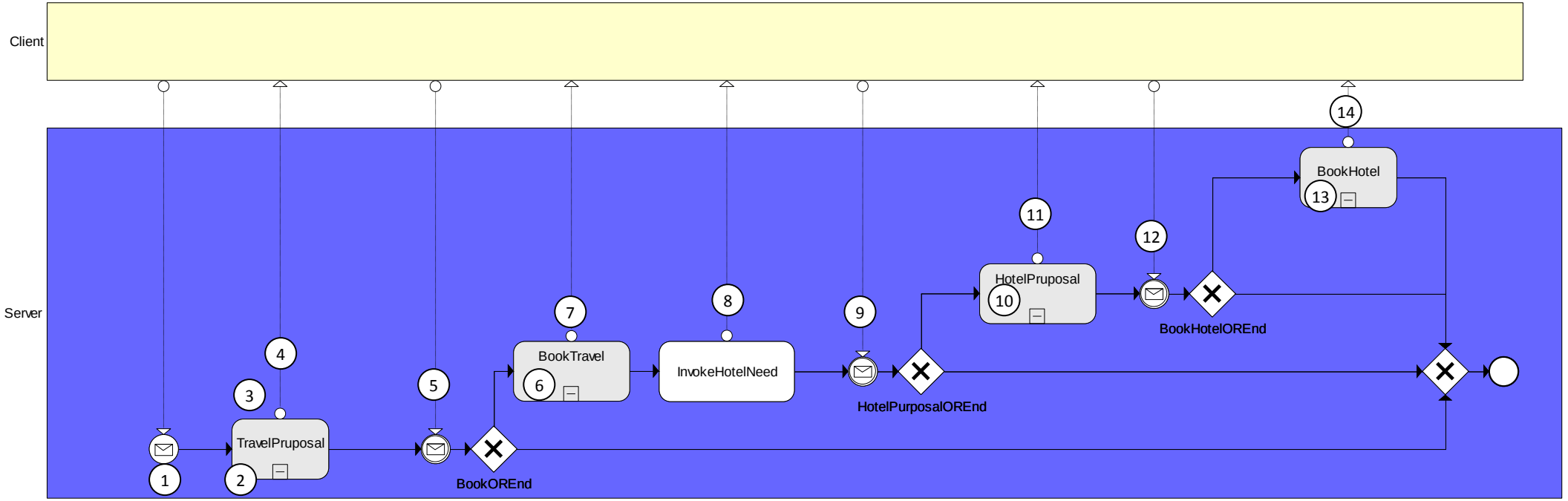


Bild 3 - Processflöde

"Feature" lista

1. Ta emot förfrågan

F1: Frågor som skickas till λTraveler skall innehålla, från och till adress, datum och tid för tidigaste och senaste avgång samt användarens för- och efternamn.

F2: Vid inkorrekt förfrågan skall felmeddelande E1 skickas till klienten och processen avbrytas.

2. Kontrollera användarens resebehörighet med λHR

F3: Har användaren resebehörighet skall reseförslag tas fram.

F4: Har användaren inte behörighet skall processen avbrytas och felmeddelande E2 skickas till klienten.

F5: Finns inte användaren i databasen skall processen avbrytas och felmeddelande E3 skickas till klienten.

F6: Fås ingen kontakt med λHR skall processen avbrytas och felmeddelande E4 skickas till klienten.

3. Ta fram reseförslag

F7: Har användaren consultant som resestatus skall det billigaste resealternativet tas fram.

F8: Har användaren manager som resestatus skall det resealternativ som har kortast restid tas fram.

F9: Tillgängligheten av biljetter på varje delsträcka skall kontrolleras.

F10: En rutt med minst en delsträcka utan tillgängliga biljetter är ett icke valbart alternativ.

F11: Finns ingen rutt med tillgängliga biljetter på alla delsträckor skall felmeddelande E5 skickas till klienten och processen avbrytas.

F12: Finns ingen rutt för de valda adresserna skall felmeddelande E5 skickas till klienten och processen avbrytas.

F13: Fås ingen kontakt med ResplusTravelRoute skall processen avbrytas och felmeddelande E6 skickas till klienten.

F14: Fås ingen kontakt med SJBooking skall processen avbrytas och felmeddelande E6 skickas till klienten.

F15: Fås ingen kontakt med SASBooking skall processen avbrytas och felmeddelande E6 skickas till klienten.

F16: Fås ingen kontakt med TaxiSverigeBooking skall processen avbrytas och felmeddelande E6 skickas till klienten.

4. Skicka reseförslag till klient

F17: Det valda resealternativet skall returneras till användaren.

***(F18:** Fås ingen kontakt med klienten skall processen avbrytas.)*

5. Ta emot bekräftelse

F19: En bekräftelse av föreslagen resa från klienten skall mottas.

***(F20:** Mottas ingen bekräftelse inom 30 sekunder skickas en påminnelse till klienten.)*

***(F21:** Mottas ingen bekräftelse inom 1 min avbryts processen.)*

F22: Är bekräftelsen negativ avbryts processen.

F23: Är bekräftelsen positiv bokas resan.

6. Boka resa

F24: Finns inga tillgängliga biljetter på någon av delsträckorna skall felmeddelande E7 skickas till klienten och processen avbrytas.

F25: Fås ingen kontakt med SJBooking skall processen avbrytas och felmeddelande E6 skickas till klienten.

F26: Fås ingen kontakt med SASBooking skall processen avbrytas och felmeddelande E6 skickas till klienten.

F27: Fås ingen kontakt med TaxiSverigeBooking skall processen avbrytas och felmeddelande E6 skickas till klienten.

F28: Samtliga bokningskvitton skall skickas till λAcconting som en räkning.

F29: Fås ingen kontakt med λAcconting skall felmeddelande E8 skickas till klienten.

7. Skicka bekräftelse till klient

F30: Bekräftelse på bokning skall skickas till klienten.

8. Förfrågan om boende på destinationen

F31: Förfrågan om användaren vill boka boende på destinationsorten skall skickas till klienten.

9. Ta emot bekräftelse

***(F32:** Mottas ingen bekräftelse inom 30 sekunder skickas en påminnelse till klienten.)*

***(F33:** Mottas ingen bekräftelse inom 1 min avbryts processen.)*

F34: Är bekräftelsen negativ avbryts processen.

F35: Är bekräftelsen positiv tas hotellförslag fram.

10. Ta fram boendeförslag

F36: Det billigaste boendealternativet skall tas fram.

F37: Tillgängligheten av rum på hotellen skall kontrolleras.

F38: Ett hotell utan tillgängliga rum är ett icke valbart alternativ.

F39: Finns inga hotell med tillgängliga rum på orten skall felmeddelande E9 skickas till klienten och processen avbrytas.

F40: Finns inga hotell på den valda orten skall felmeddelande E10 skickas till klienten och processen avbrytas.

F41: Fås ingen kontakt med ScandicBooking skall processen avbrytas och felmeddelande E6 skickas till klienten.

11. Skicka boendeförslag

F42: Boendeförslaget skall skickas till klienten.

12. Ta emot bekräftelse

***(F43:** Mottas ingen bekräftelse inom 30 sekunder skickas en påminnelse till klienten.)*

***(F44:** Mottas ingen bekräftelse inom 1 min avbryts processen.)*

F45: Är bekräftelsen negativ avbryts processen.

F46: Är bekräftelsen positiv bokas resan.

13. Boka resa

F47: Finns inga tillgängliga rum skall felmeddelande E11 skickas till klienten och processen skall gå till steg 8. Förfrågan om boende på destinationen.

F48: Fås ingen kontakt med ScandicBooking skall processen avbrytas och felmeddelande E6 skickas till klienten.

F49: Samtliga bokningskvitton skall skickas till λAcconting som en räkning.

F50: Fås ingen kontakt med λAcconting skall felmeddelande E8 skickas till klienten.

14. Skicka bekräftelse

F51: En bekräftelse på bokningen skickas till klienten.

F52: När bekräftelse skickats skall processen avslutas.

Felkoder

E1: Felaktig förfrågan.

E2: Personen har ingen behörighet att boka resa.

E3: Personen finns inte i HR-systemet, kontrollera att uppgifterna är rätt.

E4: Ingen kontakt med HR-systemet bokningsstatus kan ej bekräftas.

E5: Inget resealternativ finns för den aktuella sträckan.

E6: Ingen kontakt med server, processen avbröts.

E7: Någon del av resan är fullbokad försök igen.

E8: Biljetten inte bokförd rätt kontakta omedelbart ekonomienheten, för att göra detta manuellt.

E9: Inga tillgängliga hotellrum.

E10: Inga hotell på den valda orten.

E11: Tillgängliga rum på det valda hotellet är slut, försök igen.

E. Testspecifikation – λTraveler

Samtliga test utom 2-5, testas i en a och en b variant, i a varianten är bokaren konsult (travelStatus consultant) och i b är bokaren chef (travelStatus manager).

Test 1: Normalfallet

1	Skicka förfrågan.
2	Ta emot reseförslag.
3	Skicka bekräftelse (positiv).
4	Ta emot bekräftelse.
5	Ta emot förfrågan om boende på destinationen.
6	Skicka bekräftelse (positiv).
7	Ta emot boendeförslag.
8	Skicka bekräftelse (positiv).
9	Ta emot bekräftelse.

Test 2: Felaktig indata

1	Skicka förfrågan (felaktig).
2	Ta emot felmeddelande.

Test 3: Ej behörig användare

1	Skicka förfrågan (användare ej behörig).
2	Ta emot felmeddelande.

Test 4: Användaren finns ej i databas

1	Skicka förfrågan (användare finns ej i databas).
2	Ta emot felmeddelande.

Test 5: Ingen kontakt med λHR (λHR avstängd)

1	Skicka förfrågan.
2	Ta emot felmeddelande.

Test 6: Ingen rutt tillgänglig

1	Skicka förfrågan (resa som inte finns i resplus).
2	Ta emot felmeddelande.

Test7: Ingen kontakt med ResplusTravelRoute (ResplusTravelRoute avstängd)

1	Skicka förfrågan.
2	Ta emot felmeddelande.

Test8: Ingen kontakt med SJBooking (SJBooking avstängd)

1 Skicka förfrågan.

2 Ta emot felmeddelande.

Test9: Ingen kontakt med SASBooking (SASBooking avstängd)

1 Skicka förfrågan.

2 Ta emot felmeddelande.

Test10: Ingen kontakt med TaxiSverigeBooking (TaxiSverigeBooking avstängd)

1 Skicka förfrågan.

2 Ta emot felmeddelande.

Test11: Användare avböjer förslag.

1-2 se normalfallet.

3 Skicka bekräftelse (negativ).

Test12: Ingen kontakt med SJBooking (SJBooking avstängd efter bekräftelse (3))

1-3 se normalfallet.

4 Ta emot felmeddelande.

Test13: Ingen kontakt med SASBooking (SASBooking avstängd efter bekräftelse (3))

1-3 se normalfallet.

4 Ta emot felmeddelande.

Test14: Ingen kontakt med TaxiSverigeBooking (TaxiSverigeBooking avstängd)

1-3 se normalfallet.

4 Ta emot felmeddelande.

Test15: Inga tågbiljetter tillgängliga vid bokning

1-3 se normalfallet.

4 Ta emot felmeddelande.

Test16: Inga flygbiljetter tillgängliga vid bokning

1-3 se normalfallet.

4 Ta emot felmeddelande.

Test17: Ingen kontakt med λAcconting (efter bokning).

1-3 se normalfallet.

4 Ta emot felmeddelande.

Test18: Användare avböjer boende på destination.

1-5 se normalfallet.

6 Skicka bekräftelse (negativ).

Test19: Inga boenden tillgängliga på destination (resa till plats utan boende).

1-6 se normalfallet.

7 Ta emot felmeddelande.

Test20: Ingen kontakt med ScandicBooking (ScandicBooking avstängd)

1-6 se normalfallet.

7 Ta emot felmeddelande.

Test21: Användare avböjer boende på destination.

1-7 se normalfallet.

8 Skicka bekräftelse (negativ).

Test22: Ingen kontakt med ScandicBooking (ScandicBooking avstängd efter bekräftelse)

1-8 se normalfallet.

9 Ta emot felmeddelande.

Test23: Ingen Inga boenden tillgängliga vid bokning

1-8 se normalfallet.

9 Ta emot felmeddelande.

Test24: Ingen kontakt med λAcconting (efter bokning).

1-8 se normalfallet.

9 Ta emot felmeddelande.

F. Sammanställning av verktyg för BPMN, BPEL och BPMN2BPEL

Tabell 3 - Lista över produkter med BPMN kompabilitet (OMG, 2009)

Tillverkare/Produkt	Webbadress
ActiveVOS	http://www.activevos.com/products-features.php
Altova: UModel v2008r2	www.altova.com
Avolution: Abacus	www.avolution.com.au
Appian Enterprise 5 Business Process Management Suite	www.appian.com
aXway: Process Manager™	www.axway.com
Barium Live!	http://www.bariumlive.com
BizAgi:	www.bizagi.com
BOC Information Systems: ADONIS®	www.boc-eu.com
Borland® Together® Products: Together Architect® 2006 and Together Designer® 2006	www.borland.com
BPM-X GmbH: BPM-Xchange® Product Suite	http://www.bpm-x.com
Casewise: Corporate Modeler™	www.casewise.com
Cordys: Business Operations Platform	http://www.cordys.com/cordyscms_com/business_operations_platform.php
DAI-Laboratory: VSDT	http://energy.dai-labor.de/index.php?id=15
Ekuar	http://www.ekuar.com/
The Eclipse Foundation: Eclipse BPMN Modeler	http://www.eclipse.org/bpmn/
Elixer Intelligent Software: eliXir BPMN-MDA Framework	www.al-ixir.com
EMC: EMC Documentation Process Suite	software.emc.com
Embarcadero Technologies: EA/Studio	http://www.embarcadero.com/products/eastudio/
ENGINEERING Ingegneria Informatica - ISUFI University of Salento: bxModeller	bxmodeller.eng.it
Fujitsu: Interstage Business Process Manager 7.1	www.fujitsu.com/global/services/software/interstage/products/bpm/
Graham Technology: GT-X	www.gtnet.com
Global 360: Business Optimization Server - Process Sketchpad	www.global360.com/
HandySoft Global Corp: BizFlow® BPM	www.handysoft.com
IDS-Scheer: Aris™	www.ids-scheer.com
Corel: iGrafx™	www.igrafx.com
ILOG: JViews™	www.ilog.com
Image Technology	www.imagetec.com.br
Intalio: n³ Designer™	www.intalio.com
Intellior AG: AENEIS	www.intellior.ag
Interfacing Technologies: Enterprise Process Center; Free BPMN Modeler	www.interfacing.com
ITpearls: Process Modeler for Visio	www.itpearls.com
inubit AG: inubit BPM-Suite	inubit.com
Kaisha-Tec: ActiveModeler Avantage	www.activemodeler.com
KnowEnterprise	http://www.knowgravity.com/

Lanner: Witness™	www.lanner.com
Lombardi Software: TeamWorks®	www.lombardi.com
M1 Global: BPI Studio	www.m1global.com
Mega International: Mega Suite™	www.mega.com www.metastorm.com
No Magic: MagicDraw UML 10.0	www.nomagic.com
Orbus Software: iServer	www.orbussoftware.com
Oryx: Web-based Collaborative Process Modeling	www.oryx-editor.org
Pallas Athena: BPM|one®	www.pallas-athena.com
PNM Soft: SEQUENCE BPM	http://www.pnmsoft.com
Process Master: ProcessPad	www.ProcessMaster.com
Pegasystems: BPMSuite	www.pega.com
QPR Software: QPR BPM Suite	www.qpr.com
Seagull Software: LegaSuite BPM	www.seagullsoftware.org
Signavio GmbH: Signavio Process Editor	http://www.signavio.com
Software AG: Enterprise Business Process Manager (EBPM)	www.softwareagusa.com
Popkin: System Architect™	www.popkin.com
Santeon: XIP BPM Platform	www.santeon.com
SAP: SAP NetWeaver Composition Environment (CE), component SAP NetWeaver Business Process Management (BPM)	www.sap.com/platform/netweaver/ components/sapnetweaverbpm
Orbus Software: iServer	www.orbussoftware.com
Oracle BPM-suite	www.oracle.com
Savvion: Process Asset Management	www.savvion.com
Select Business Solutions: Select Component Factory	www.selectbs.com
Skelta: Skelta BPM.NET 2006	www.skelta.com
Soyatec: eBPMN Designer	www.soyatec.com
Sparx Systems: Enterprise Architect 7.1	www.sparxsystems.com
Sun Microsystems: Studio Enterprise Edition	www.sun.com
Sybase: PowerDesigner® 12	www.sybase.com/developmentintegration
Tibco: Business Studio ™	www.tibco.com
Troux™: Metis 3.6 Enterprise Architecture Suite™	www.troux.com
Whitestein Technologies: Living Systems® Autonomic Business Process Management (LS/ABPM)	www.whitestein.com/ls-abpm
Visual Paradigm: Visual Architect™	www.visual-paradigm.com

Tabell 4- BPEL-serverprogramvaror

Produkt	Tillverkare	Senaste Version	Relese-datum	Kompabilitet	Licens
ActiveVOS 7.0	Active Endpoints	7.0		V 2.0	Kommersiell
ActiveBPEL Engine	Active Endpoints	2.1		V 1.1	Open source
Apache ODE		1.3.2	2008-05-11	V 1.1 (Beta för v 2.0 finns)	Apache
BizTalk Server	Microsoft	BizTalk 2006 R2, 2009	2006-04-03	V 1.1	Kommersiell
iBolt Server	Magic Software Enterprises			Version okänt	Kommersiell
Intalio BPM	Intalio			V 2.0	Kommersiell, ”try out” i ett år
jBPM	jBoss	jBPM v 4.0 jBPM BPEL v 1.1.1	2009-01-12 2008-08-12	V 1.1 och 2.0	LGPL
Open ESB	Sun Microsystems	2.0	2009-02-10	v 2.0	Open Source, CDDL
Oracle BPEL Process Manager (tidigare Collaxa BPEL Orchestration Server)	Oracle Corporation	10.1.2.0.2 (11.0.0.1)	2006-01-28	v 1.1 med egna tillägg	Kommersiell gratis
Parasoft BPEL Maestro	Parasoft	4.1	2009-06-05	V 2.0	Kommersiell
SAP Netweaver	SAP AG	3.0		Version okänt	Kommersiell
SEQUENCE BPM	PNMSoft	?		?	Kommersiell
Virtuoso Universal Server	OpenLink Software	4.5	2006	V 1.1	GPL and Kommersiell
WebSphere Process Server	IBM	6.1 6.2	2008-01-18 2009-01-16	V 2.0	Kommersiell

Tabell 5 - BPEL-editorer

Produkt	Tillverkare	Senaste Version	Kompabilitet	Licens
ActiveVOS 7.0	Active Endpoints	7.0	V 2.0	Kommersiell
Eclipse BPEL (Plugin för Eclipse)	The Eclipse Foundation	Eclipse 3.4 BPEL 0.4.0 (2009-06-08)	V 2.0	Open Source
Intalio Designer	Intalio	?	V 2.0	Gratis
JDeveloper	Oracle	10g 11g	V 1.1 med egna tillägg	Kommersiell gratis
Netbeans	Sun	6.5	V 2.0	Open Source
Parasoft BPEL toolkit (plugin för Eclipse)	Parasoft	4.1	V 2.0	Kommersiell
SAP Netweaver	SAP AG	3.0		Version okänt
SEQUENCE BPM	PNMSoft	?	?	Kommersiell
Visual Studios	Microsoft	2005 och framåt	v.1.1	Kommersiell
WebSphere® Integration Developer	IBM	6.1 6.2	V 2.0	Kommersiell

Tabell 6 - Sammanställning av verktyg för BPMN till BPEL konvertering

Produkt	Tillverkare	Senaste version	Kompabilitet	Licens
ActiveVOS 7.0	Active Endpoints	7.0	BPMN V 2.0 BPMN v 2.0	Kommersiell
bpmn2bpel plugin för Eclipse	Open source	0.0.1.2	BPMN v 1.1 BPEL v 2.0	Open source
Intalio BPM	Intalio		BPMN v 2.0 (grafiskt gränssnitt) BPEL v 2.0	Kommersiell
Oracle BPA-suite	Oracle	10.1.3.4	BPMN v 1.1 BPEL v 1.1	Kommersiell
SEQUENCE BPM	PMNSoft	?	?	Kommersiell