Avdelning för datavetenskap

Martin Bood och Karl-Johan Fisk

Tjänsteorienterad Integration, ESB

Service Oriented Integration, ESB

Examensarbete 10 poäng

Datum:	07-10-12
Handledare:	Donald Ross
Examinator:	Martin Blom
Löpnummer:	C-2007:08

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Martin Bood

Karl-Johan Fisk

Godkänd, 2007-10-12

Handledare: Donald Ross

Examinator: Martin Blom

Sammanfattning

För att dagens system och deras allt mer komplexa applikationer ska kunna integreras med varandra krävs det att de kommunicerar via tjänster. Denna tjänsteorienterade integration uppnås genom att man använder sig av Service Oriented Architecture (SOA) som bygger på löst kopplade tjänster som kommunicerar med varandra på ett standardiserat sätt. En viktig del i en integrationslösning är Enterprise Service Bus (ESB). I denna rapport kommer vi förklara grunderna i tjänsteorienterad integration och sedan fördjupa oss i ESB. Då ESB är ett luddigt begrepp ska vi på ett enkelt och lättbegripligt sätt ge vår syn på begreppet, samt dess fördelar och nackdelar. Vi kommer även att ge marknadens syn på ESB genom en enkätundersökning som innefattar både leverantörer, konsulter och kunder.

Nyckelord: Service Oriented Architecture, Web Services, Enterprise Service bus.

Service Oriented Integration, ESB

Abstract

If today's software systems and their complex applications shall be able to integrate with each other, they have to communicate through services. This service oriented integration can be accomplished by using Service Oriented Architecture (SOA) where all components are loosely coupled and communicate in a standardized way. An important part when building an integrated solution is the Enterprise Service Bus (ESB). In this report we will explain the basics of SOA and take a more detailed look at the world of ESB. The concept of ESB is not well defined and hence means different things to different people. We are going to present an interpretation of the ESB and its benefits and disadvantages. To find out what the market thinks about ESB we have been talking to producers, consultants and customers.

Keyword: Service Oriented Architecture, Web Services, Enterprise Service bus.

Innehållsförteckning

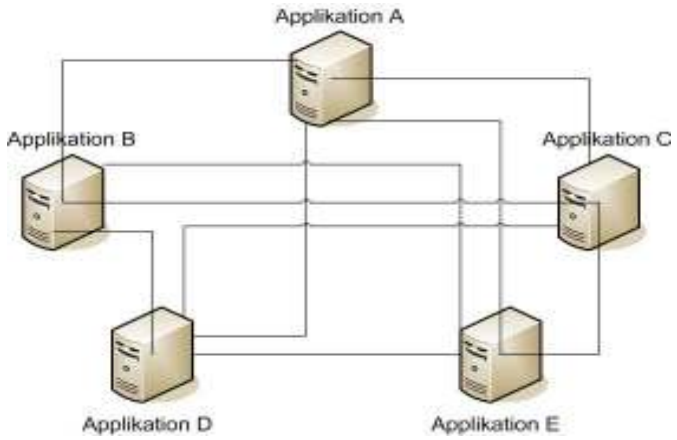
Figurförteckning

1 Inledning

1.1 Bakgrund

De datornätverk som finns i dagens företag utnyttjar flera hundra applikationer och system av olika ålder och från olika leverantörer. För femton år sedan var det tillräckligt för ett företag att komma åt information som var lagrade på var och en av dessa applikationer för att klara ett viktigt affärsinitiativ. Men eftersom dagens företag ofta har enorma IT tillgångar har detta resulterat i ett behov att integrera alla dessa applikationer för att kunna klara av sina affärer både internt och externt [1].

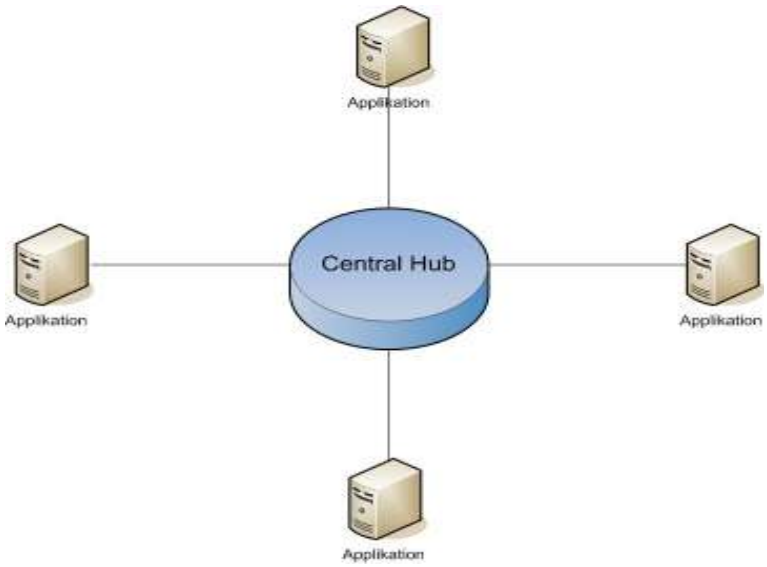
Historiskt sett har det gjorts ett antal försök att integrera flera applikationer med mer eller mindre lyckat utfall. Från ett tidigt skede byggdes integrationen på ett sätt som kan beskrivas som punkt–till-punkt integration. För varje koppling mellan två applikationer skrevs kod för att hantera transport, meddelande format osv. Att lösa problemet på detta vis visade sig fungera alldeles utmärkt ner det bara handlade om två applikationer som skulle integreras. Stora problem uppkom när ytterligare applikationer skulle integreras. Som ett exempel skulle kunna ges ett företag med fem applikationer, där alla applikationer behöver kopplas samman med varandra. Antalet sätt att koppla samman dessa fem applikationer skulle då bli n*(n-1)/2 = 10. Se figur 1.



Figur 1 Punkt-till-punkt integration

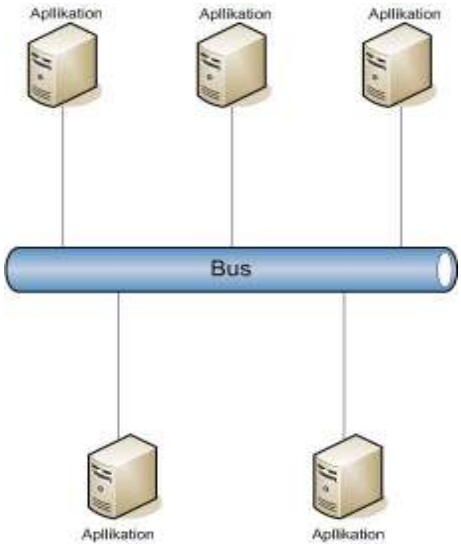
Eftersom var och en av dessa kopplingar mellan två applikationer har sin egen kod för transport, meddelande och format hantering blir applikationerna väldigt hårt bundna till varandra. Detta medför att ett system av hårt bundna applikationer blir väldig bräckligt. Om t.ex. applikation A är beroende av att kunna hämta information från applikation B innan den svarar på en förfrågan från applikation C och applikation B för tillfället inte är tillgänglig kan systemet krascha eller hänga sig allt beroende på att applikationerna i systemet är för hårt bundna till varandra. Ett annat stort problem med punkt-till-punkt integration är att allt eftersom systemet växer och fler och fler applikationer läggs till och andra tas bort är att det blir väldigt fort ohanterbart eftersom ny kod måste skrivas för varje koppling en ny applikation ska ha med en annan applikation i systemet. Så att lägga till en ny applikation eller byta ut en redan existerande applikation i ett stort system blir en väldigt komplex och tidskrävande uppgift, allt beroende på att applikationerna i en punkt-till-punkt integration är allt för hårt bundna till varandra.

En lösning som implementerades för att lösa problemen som punkt-till-punkt integrationen medförde var den så kallade ”hub and spoke” arkitekturen även refererad till som ”message broker”. Denna arkitektur bygger på en centraliserad server (hubben) som till vilken alla noder (spokers) är kopplade. Alla kommunikation mellan noder sker via servern istället för en direkt kommunikation mellan noderna. Detta medför att kommunikationslogiken mellan två parter inte behöver hårdkodas i respektive part. Detta minskar avsevärt antalet kopplingar mellan noderna i systemet (från n^2 till n) och medför en abstraktion mellan parterna som i sin tur gör att ändring, bortagande och införande av nya noder i arkitekturen förenklas avsevärt pga. att kommunikationslogiken inte finns i respektive nod utan i den centralt belägna servern [4]. Se figur 2.



Figur 2 "Hub and Spoke" arkitektur

Framtidens integrationslösningar (Enterprise Service Bus, ESB se kap 4) kommer att byggas på bus topologi där alla noder kopplas på den gemensamma bussen (se figur 3). Hubbar kan förenas för att forma vad som logiskt sett är en enda entitet och som tillhandahåller en enda punkt för kontroll men som egentligen är en mängd fysiskt distribuerade komponenter. Detta är vad som ofta refereras till som en buss. Med denna arkitektur blir systemet inte lika sårbart som en "hub and spoke" arkitektur eftersom det inte har en central server och därmed ingen "central point of failure". Bus topologin har funnits i många år och är inget nytt koncept men den har fått nytt liv genom införandet av ESB.



Figur 3 Bus topologi

1.2 Syfte och mål

Syftet med vår rapport är att få en djupare förståelse för tjänsteorienterad integration (SOA) samt ESB, där vi ska ge vår syn på det diffusa begreppet ESB. Detta ska ske genom att vi kommer läsa böcker, artiklar och whitepapers. Vi kommer även att besöka folk som jobbar med ESB och hoppas på att de kan hjälpa oss att öka vår förståelse för begreppet.

1.3 Arbetsgång

Rapporten har utförts genom teoretiska studier av ESB, SOA, Web Services och XML. Under rapporten har vi haft kontinuerlig kontakt med vår handledare, som gett oss feedback på det vi gjort. Vi har även haft kontinuerlig kontakt med Per Björkegren på Sogeti som hjälpt oss när vi behövt någon att bolla våra idéer och funderingar med. För att få oss en uppfattning hur folk som jobbar med ESB ser på begreppet har vi förutom enkäten varit ute hos företag och pratat med personer på IBM, Microsoft, Oracle, KnowIT, Zystems By Semcon och Volvo IT.

1.4 Disposition

I kapitel 2 beskrivs tjänsteorienterad integration, där beskrivs SOA, Web Services och XML. Vi kommer gå igenom hur de olika delarna relaterar till varandra samt för och nackdelar med tjänsteorienterad integration. I kapitel 3 presenterar vi en referensmodell som är framtagen för att styra utvecklingen och implementationen av en tjänsteorienterad integration. Modellen ingår i utvecklingskoncept DIA som är en ansats för kontrollerad integrationsutveckling. I kapitel 4 behandlar vi begreppet ESB, dess definition för och nackdelar samt varför man ska införa en ESB. I kapitel 5 sammanställer vi svaren vi fått på den enkätundersökning som vi gjort. Vi har skickat ut enkäter till leverantörer, kunder och konsulter som stor erfarenhet av ESB. I kapitel 6 ger vi våra reflektioner på detta arbete och i kapitel 7 presenterar vi vår slutsats.

1.5 Metod

Vi har gjort en enkätundersökning som riktar sig till leverantörer av ESB-lösningar, kunder som köpt en ESB-lösning och konsulter som jobbar med ESB-lösningar. Svaren ska ge oss en inblick i hur de ser på begreppet ESB. Vi ska även genom teoretiska studier fördjupa våra kunskaper inom tjänsteorienterad integration och ESB. Sedan ska vi förklara tjänsteorienterad integration på ett övergripande sätt samt ge vår syn på ESB.

2 Allmänt om tjänstorienterad integration

I detta kapitel presenteras tjänstorienterad integration som är ett sätt att bygga distribuerade system. Systemet är uppbyggt av tjänster som var för sig utför en specifik uppgift. Kapitlet behandlar egenskaper för tjänsteorienterad integration samt ger en förklaring av några vanliga men dock så viktiga begrepp inom området.

2.1 SOA

I detta delkapitel behandlas begrepp och koncept inom service oriented architecture, ”SOA”. Kapitlet börjar med en introduktion till SOA och fortsätter med begrepp som gjort SOA möjligt att implementera på ett effektivt sätt.

2.1.1 Definition av SOA

SOA (Service Oriented Architecture) kan definieras enligt följande:

“SOA is an integration architecture approach that is based on the concept of a service. The business and infrastructure functions that are required to build distributed systems are provided as services that collectively, or individually, deliver application functionality to either user applications or other services. SOA specifies that within any given architecture, there should be a consistent mechanism by which services communicate. That mechanism should be loosely coupled and should support the use of explicit interfaces.”[5].

2.1.2 Huvudkomponenter i SOA

Det finns ett antal termer som spelar en viktig roll i SOA. Några av dessa termer är service, meddelande och dynamisk upptäckt, och dessa tre kommer det nu att gås närmare in på.

2.1.2.1 Service

En service (tjänst) är en komponent som är kapabel att utföra en speciell uppgift [6]. En tjänst presenteras utåt i form av sitt gränssnitt som är definierat på ett sätt att det är oberoende av den hårdvara, operativsystem och det programspråk som har använts för att implementera tjänsten. Gränssnittet är en beskrivning av ett kontrakt som klargör hur interaktionen mellan den som tillhandahåller tjänsten och den som använder sig av tjänsten ska utföras. Det som beskrivs i kontraktet är bl.a. sådant som hur tjänsten ska anropas, vad som returneras (om något alls returneras) och format på meddelanden som accepteras som förfrågning och svar. Detta möjliggör för tjänster byggda på olika

system att kunna kommunicera med varandra på ett enhetligt och universellt sätt. En tjänsts gränssnitt är uttryckt i affärs koncept och termer istället för i teknologiska termer [7].

I kommunikation mellan tjänster och tjänster eller mellan tjänster och icke tjänster utgörs bl.a. av följande enheter [7]:

- **Service Consumer:** Är en enhet i SOA som använder sig av tjänster för att tillgodose sig själv med information som kan erhållas från en eller flera tjänster. Service consumern kan vara en tjänst, applikation eller annan typ av mjukvaromodul.

- **Service Provider:** Är en enhet som nås via en adress i ett nätverk. Service providern accepterar och exekverar förfrågningar från service consumern. Den tillhandahåller kontraktet och den implementationen som utgör tjänsten. Service providern kan vara en applikation eller annan typ av mjukvaromodul.

- **Service Registry:** Är en adressförteckning som kan nås via en adress i ett nätverk och som innehåller tillgängliga tjänster. Den accepterar och lagrar kontrakt från service providers och tillhandahåller dessa kontrakt till intresserade service consumers.

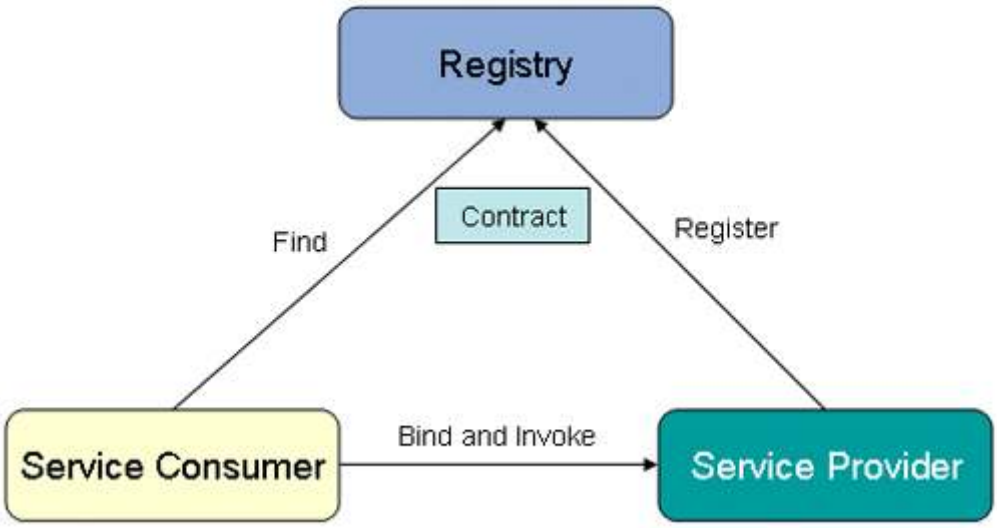
När ett system är uppbyggt av tjänster blir det lätt att underhålla då man inte behöver bry sig om hur tjänsten är implementerad. Det spelar ingen roll om implementationen eller hårdvaran bakom tjänsten ändras så länge som tjänsten inte ändrar sitt kontrakt. Användandet av tjänster möjliggör även egenskapen att skapa tjänster som består av två eller flera tjänster. På så vis blir systemet väldigt flexibelt, lätthanterligt och kostnadseffektivt eftersom man har möjligheten att återanvända tjänster.

2.1.2.2 Meddelande

Service providers och service consumers kommunicerar m.h.a meddelande. Det kontrakt som en tjänst visar utåt via sitt gränssnitt är ju som tidigare nämnts plattforms och programspråks-oberoende. Detta medför att tekniken att definiera meddelanden också måste vara plattforms och programspråks-oberoende. Den teknik som då används för att definiera meddelanden är XML (se 2.1.6) [8].

2.1.2.3 Dynamisk upptäckt

Med dynamisk upptäckt menas att en service consumer kan lokalisera en service provider endast när den behöver det. Detta realiseras med ett service registry. Som exempel kan antas en klient som vill ha uppgift om namnet på den person som äger den bil med registreringsnumret ABC 123. Klienten frågar tjänsteregistret efter lämplig tjänst som kan tillhandahålla information om bilägare. Tjänsteregistret returnerar den information som behövs för att kontakta tjänsten och klienten kontaktar tjänsten och begär den information om bilägare den behöver och får då denna information tillbaka av tjänsten om kontraktet mellan klienten och tjänsten har uppfyllts.



Figur 4 *Interaktion mellan service registry, service consumer och service provider.*

Enligt Hashimi[8] medför användandet av ett service registry följande fördelar:

- Skalabilitet av tjänster. Med andra ord menas att man kan lägga till tjänster inkrementellt.

- Service consumers och providers är löst kopplade. Genom att tjänster registrerar sin existens i en registertjänst möjliggörs löst kopplade komponenter i ett IT-system. Klienter behöver alltså inte statiskt veta var en specifik tjänst är belägen utan detta

görs dynamiskt vid förfrågan till registertjänsten. Detta innebär att klienter kan välja mellan olika tjänster utan att binda sig till en specifik tjänst på implementationsnivå.

- Enkelt att uppdatera och ändra i den bakomliggande implementationen. Detta är möjligt pga. att själva gränssnittet för en tjänst inte ändras bara för att den underliggande implementationen ändras. Det påverkar heller inte en service consumer pga. att det inte finns någon ”hård kodad” koppling mellan provider och consumer som kan påverkas av en ändring i implementationen hos service providern.

- Service consumern får en möjlighet att leta efter tillgängliga tjänster.

2.1.3 De karakteristiska dragen hos SOA

Vad som karakteriserar ett IT-system byggt på SOA kommer vi att närmare gå in på i följande kapitel. Några av de viktigaste karakteristiska dragen tar vi upp här och bör has i åtanke vid design och implementation av tjänster för att uppnå de fördelar SOA medför.[9]

2.1.3.1 Löst kopplade tjänster.

Koppling i detta avseende beskriver det beroende som existerar mellan mjukvaromoduler. Mjukvarosystem med löst kopplade moduler som har flexibla väldefinierade gränssnitt är lätta att konfigurera medan hårt kopplade mjukvarosystem är svåra att konfigurera beroende på att man inte vet vilka krav som individuella modulerna inom mjukvarosystemet har. I SOA vill man uppnå en lös koppling mellan service provider och service consumer. Service consumern ska inte ha någon vetskap om implementationen bakom service providern. Lös koppling uppnås genom att använda ett service registry.

2.1.3.2 Väl definierade gränssnitt

En tjänst ska exponera utåt ett väldefinierat gränssnitt i form av ett kontrakt som consumers kan använda för att förstå hur man ska kontakta tjänsten, vad tjänsten kan uträtta och vilken typ av meddelande format tjänsten arbetar med och vilken typ av format som returneras. Ett väldefinierat gränssnitt avslöjar inget om den bakom

liggande implementationen av tjänsten och möjliggör då att ändring i implementation hos tjänsten kan göras utan att detta påverkar de tjänster eller icke-tjänster som utnyttjar tjänsten.

2.1.3.3 Baserat på öppna standarder

Tjänster ska designas och implementeras baserat på öppna standarder. Att basera SOA på öppna standarder medför fördelar så som möjligheten att minimisera riskerna att låsa sig till standarder och produkter från en enda tillverkare. Det ger även möjligheter att kunna skapa tjänster som kan hantera ett stort antal olika typer av förfrågningar. Web Services (se kap 2.1.5) är ett exempel på en teknologi som är öppen standard och som används för att implementera tjänster i en SOA.

2.1.4 Fördelar och nackdelar med SOA

Fördelarna med att implementera SOA är många och vi kommer här att gå in på några av dem.

2.1.4.1 Effektivitet

SOA tillhandhåller ett lager av abstraktion som möjliggör för företag och organisationer att återanvända sina existerande IT-tillgångar genom att exponera dem utåt som tjänster och på det viset tillhandahålla nödvändiga affärsfunktioner både intern och externt. Med hjälp av SOA kan företag och organisationer återanvända sina existerande applikationer på ett effektivt sätt. Istället för att skapa helt nya applikationer kan man skapa dem genom att kombinera tjänster som redan är exponerade av existerande applikationer [9]. Detta medför att ett företag snabbt kan anpassa sig till nya krav genom att ändra i sitt IT-system med hjälp av redan existerande IT-resurser och tillgångar.

2.1.4.2 Enklare att integrera och att hantera komplexitet.

När man pratar om integration i samband med SOA rör detta hur tjänsten är specificerad (gränssnitten) och inte hur tjänsten är implementerad. Detta skapar ett oberoende mellan tjänstens funktionalitet ur användarens perspektiv, och tjänstens implementation och infrastruktur ur ett tekniskt perspektiv vilket leder till minimal

inverkan när ändringar sker på implementation och infrastrukturnivå. Genom att ”klä in” existerande IT-resurser och tillgångar belägna på olikartade system i specifikationer i form av gränssnitt gömmer man komplexiteten och möjliggör för enklare integration.

2.1.4.3 Val av IT-lösning

I och med att SOA är plattformsoberoende behöver företagen inte längre binda sig vid en specifik utvecklingsplattform. Detta leder till att utvecklarna kan utnyttja de fördelar som varje utvecklingsspråk och/eller plattform har. Eftersom att tjänster kommunicerar via sitt gränssnitt kan implementationen göras på valfritt sätt, vilket leder till att varje tjänst kan göras så effektiv som möjligt. Detta kallas ofta för ”best-of-breed”, eftersom det alternativ som passar företagets behov bäst kan väljas.

Två starka aspekter till varför ett företag skulle tjäna på att implementera sitt IT-system baserat på tjänste-orienterad-arkitektur (service oriented architecture – SOA) är möjligheten att på ett effektivt sätt ändra i sitt existerande IT-system och att kunna minska kostnader i sin affärsverksamhet. I ett allt hårdare marknadsklimat med ständigt ökad konkurrens och växande krav från partners och kunder måste man snabbt kunna anpassa sig till nya villkor och krav som råder på marknaden.

Möjligheten att på ett effektivt sätt implementera SOA har på senare tid realiserats på grund av det relativt nya konceptet Web Services som in sin tur bygger på användandet av SOAP/XML meddelanden. Dessa begrepp kommer att behandlas i kommande stycken.

2.1.5 Web Services

Web services är tjänster som bygger på standardiserade plattformsoberoende protokoll som har funnits länge och som anses säkra. Med införandet av web services har företagen enats om följande standarder för att få applikationerna att kunna kommunicera med varandra[3]:

- XML: se 2.1.6
- HTTP: Ett kommunikationsprotokoll för överföring av HTML och XML- data över Internet.

- UDDI: Universal Discovery, Description, and Integration register är ett XML baserat register som kan ses som en telefonkatalog för Web Services. I detta register kan sökning göras efter Web Services och deras associerande URL och WSDL sidor.
- WSDL: Web Service Definition Language är ett XML baserat språk för att beskriva Web Services och hur man får tillgång till dem.
- SOAP: Simple Object Access Protocol är ett XML baserat protokoll som möjliggör utbyte av information mellan applikationer över HTTP. SOAP är plattforms- och programspråksoberoende.

2.1.5.1 Vad är en web service?

En web service är en adresserbar mjukvaruresurs som utför en specifik uppgift. I och med att den har en adress kan applikationer eller andra web services utnyttja dess funktionalitet. För att kunna anropa en web service behöver man få reda på adressen till den. Adressen finns i en UDDI där alla web services finns registrerade, genom en förfrågan till UDDI´n kan man få adressen till en web service som matchar förfrågningen. Anledningen till att man använder en UDDI är att man vill ha löst kopplade komponenter. Eftersom man får adressen till en web service via en förfrågan till UDDI´n kan det vara olika adresser som returneras vid olika körningar. Detta är en styrka med web services då länkningen sker dynamiskt, i dagsläget utnyttjas inte detta fullt ut av utvecklare pga. att de inte litar på web services som är utvecklade av andra.

En annan styrka med web service är att den är plattforms- och programspråksoberoende, detta möjliggörs genom att web services använder sig av XML. Varje web service har ett eget XML-schema (se figur 5) som ska mappas (se figur 6) mot ett XML-schema från den anropande applikationen. Nu sker en kontroll att web servicen får all den information den behöver. Efter att ha mottagit all information så utför tjänsten sin uppgift för att sedan returnera ett svar till applikationen. Om mappningen inte stämmer ges ingen tillgång till tjänsten.

Då web services utför en specifik uppgift kan man kombinera flera stycken om man vill utföra en större uppgift. Detta leder till att man kan skapa nya applikationer utan

att behöva skriva någon ny kod. I och med att man inte behöver skriva någon kod för att använda web services kan detta göras av personer som inte är tekniska experter pga. att dom inte behöver bry sig om hur de är implementerade. På så vis hålls kostnaderna nere för att hantera systemet.

Web services är den mest förekommande typen av tjänst och det är den som har bidragit till att SOA har blivit så populärt som det är idag. Anledningen är att web services gör det möjligt att utnyttja applikationer oavsett om dom befinner sig i ett lokalt nätverk eller på webben.

2.1.6 XML

XML (Extensible Markup Language) är ett dokumentstrukturdefinitionsspråk, som är skapat för att skapa språk utifrån de kriterier som definieras av XML. Med andra ord definierar XML en syntax som man ska följa när man skapar sitt eget språk [2]. Nedan visas ett enkelt XML dokument.

```
<address>
 <name>
   <title>Mrs.</title>
     <first-name>
      Johanna
      </first-name>
     <last-name>
      Larsson
      </last-name>
 </name>
   <street>
    1401 Main Street
   </street>
   <city state = "NC">Springfield</city>
    <postal-code>
     34829
    </postal-code>
</address>
```

Figur 5 *Ett enkelt XML dokument.*

En ”tag” är texten mellan den vänstra vinkelparentesen (<) och den högra vinkelparentesen (>). Det finns startparenteser så som <name> och slutparenteser så som </name>.

- Ett element ”starttaggen”, ”sluttaggen”, och allting där i mellan. I vårt exempel ovan innehåller elementet <name> tre barnelement: <title>, <first-name> och <last-name>.
- Ett attribut är ett namngivet värde innanför starttaggen hos ett element. I exemplet ovan är state ett attribut till <city>.

Med exemplet ovan i färskt minne kan man dra slutsatsen att XML taggar är självbeskrivande på det viset att det beskriver vad de olika delarna i ett dokument är. Till skillnad från HTML taggar som beskriver hur delar i ett dokument ska placeras ut på en webbsida eller ett annat slags dokument och som är skapat för integration mellan människa och dator är XML taggar skapade för integration mellan två datorer.

På grund av att XML kan struktureras på ett sådant vis att varje viktig del av information kan identifieras, är det möjligt att skriva kod, en så kallad parser, som kan processa XML dokument utan mänsklig ingripande. Men eftersom man kan skapa sina egna unika element och strukturera ett XML dokument på så många olika vis måste det finnas ett sätt att definiera sin egen typ av XML dokument med avseende på struktur och namn på element. Lösningen på detta är XML scheman. Ett XML schema definierar följande:

- De olika typer av element som kan förekomma i dokumentet.
- De olika typer av attribut som kan förekomma i dokumentet.
- Vilka element som är barnelement.
- Ordningen på barnelement.
- Antalet barnelement som kan finnas för ett element.
- Om ett element kan vara tomt eller innehålla text.
- Datatyper för element och attribut i dokumentet.

När ett dokument sänds från en sändare till en mottagare är det viktigt att båda parterna är överens om dokumentets struktur. Med ett XML schema kan sändaren beskriva den data

som finns i dokumentet på ett sätt som mottagaren kan förstå. Som en förklaring kan ges ett exempel gällande datum. Ett datum som 05-12-2007 tolkas i vissa länder som den 5 december 2007 och i andra som den 12 maj 2007. Ett XML element med den fördefinierade XML datatypen date skulle kunna se ut så här:

<date type="date">2007-12-05</date>

Detta garanterar att en ömsesidig förståelse existerar mellan parterna då datatypen date kräver formatet ”YYYY-MM-DD”.

Nedan visas ett exempel på ett XML dokument med tillhörande XML schema:

```
<?xml version="1.0"?>
<note>
<to>Martin</to>
<from>Karl-Johan</from>
<heading>Reminder</heading>
<body>Don't forget meeting on Monday with Per!</body>
</note>
```

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3.com"
xmlns="http://www.xxxx.com"
elementFormDefault="qualified">
<xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

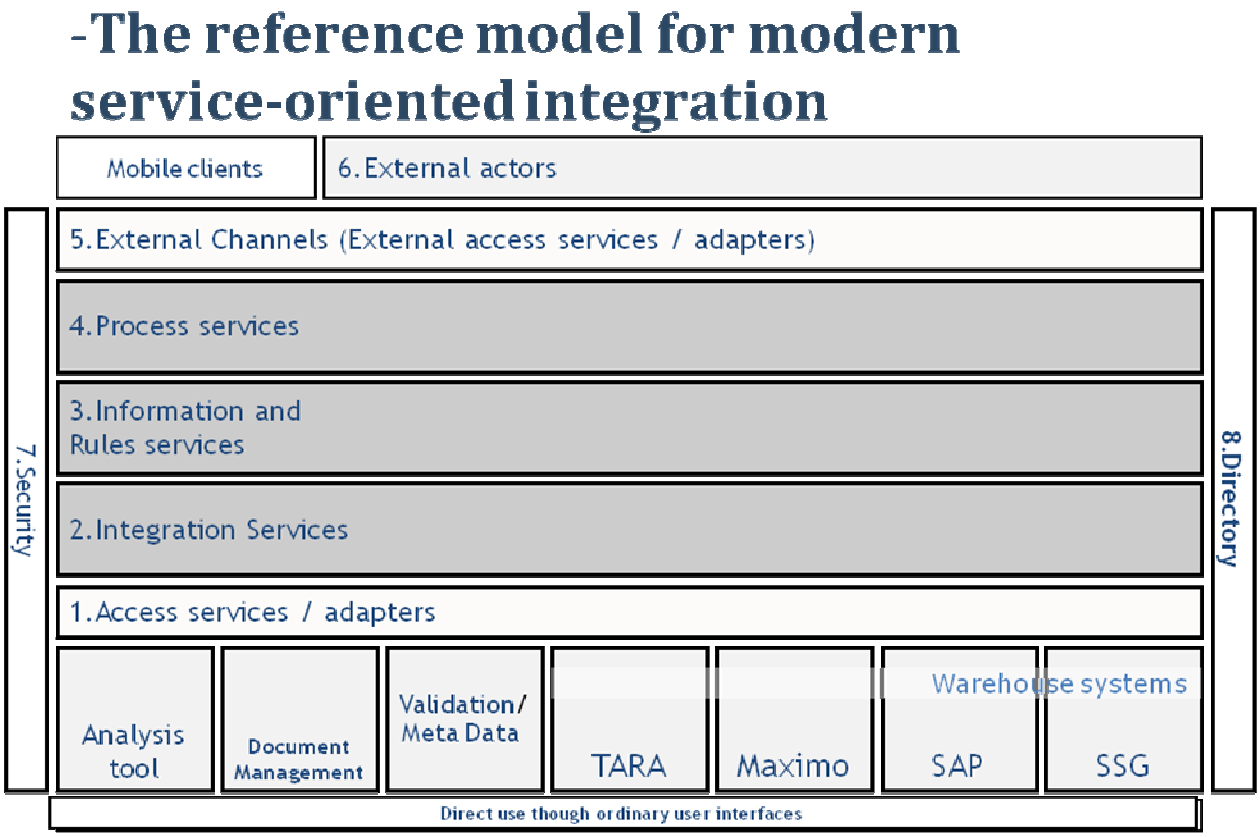
Figur 6 Ett XML dokument med tillhörande XML schema.

2.2 Sammanfattning

SOA är en arkitektur, ett sätt att tänka när man bygger ett IT-system. Med SOA vill man uppnå löst kopplade mjukvarukomponenter, vilket innebär att en ändring i en komponent inte ska medföra att ändring i andra komponenter är nödvändigt. Samtliga komponenter implementeras som tjänster med publika eller publicerade gränssnitt vilka beskriver ett kontrakt som måste uppfyllas för att kunna utnyttja tjänsten. En tjänst (service) är en komponent som är kapabel att utföra en uppgift. Tjänstens gränssnitt är publicerat för att möjliggöra återanvändning av sig själv för komponenter som kan vara fysiskt avlägsna och helt ovetande om implementationen bakom den mjukvarukomponent som utgör tjänsten. En tjänst visar utåt sitt gränssnittskontrakt vilket definierar hur tjänsten beter sig och vilken typ av meddelande (ex: SOAP/XML) tjänsten accepterar och vad som returneras, om något returneras. Ett gränssnittskontrakt ska vara plattformsoberoende vilket medför att en klient kan komma åt tjänsten oavsett geografiskt läge, operativsystem eller programspråk klienten är implementerad i. De tjänster som finns tillgängliga registreras i en så kallad registertjänst, en registertjänst innehåller information om alla tillgängliga tjänster och hur man kontaktar dom och vilken tjänst som tillhandahålls.

3 Referensmodell för modern tjänsteorienterad integration

För att kunna överblicka ett integrationssystem och få en uppfattning hur detta system ska fungera har Per Björkegren på Sogeti Karlstad tagit fram en referensmodell för tjänsteorienterad integration. All integration menar han skall designas och implementeras enligt denna modell. Referensmodellen ingår i Dynamic Integration Architecture(DIA) som är en ansats för kontrollerad integrationsutveckling.[38]



Figur 7 Per Björkegrens Referensmodell

3.1 Vad är en referensmodell?

Enligt [10] kan en referensmodell beskrivas som följande:

"A reference model is an abstract framework for understanding significant relationships among the entities of some environment, and for the development of consistent standards or specifications supporting that environment. A reference model is based on a small number of unifying concepts and may be used as a basis for education and explaining standards to a non-specialist. A reference model is not

directly tied to any standards, technologies or other concrete implementation details, but it does seek to provide a common semantics that can be used unambiguously across and between different implementations."

3.2 Varför en referensmodell?

Det har visats sig i modern integration och SOA att utveckling inom projekt behöver styras för att säkerställa fördelarna med service orientering. I många fall behövs också en mindre teknisk beskrivning vid ett integrationsprojekt för att kunna samarbeta med viktiga aktörer inom verksamheten som inte ”pratar” och förstår den tekniska nivå som utvecklare och IT-arkitekter ligger på.[38]

Ett annat syfte med en referensmodell är att IT-arkitekter och utvecklare har en mall att utgå ifrån när de ska styra och implementera en integrationsplattform. Genom att dela upp referensmodellen i logiska delar(lager) blir det lättare att hitta produkter som klarar de krav som ställs för en specifik uppgift, samtidigt som det blir lättare att överblicka systemet för de som använder det. Det underlättar också för företag som tillverkar produkter då de vet vad systemet behöver och kan anpassa sina produkter för att passa in i referensmodellen.[10]

Referens modellen för tjänsteorienterad integration används sammanfattningsvis till:

- Bestämma omfattning och diskutera runt en högnivå lösning.
- Beskriva olika scenarior.
- Bestämma vilka huvudsakliga tjänster ett integrationsfall ska bestå av.
- Analysera och definiera hur olika produkter av mjukvara möter de krav för SOA och integration.

De flesta leverantörer av integrationsprodukter har sin eget sätt att beskriva sin SOA produkt och det är därför lämpligt att ha sin egen beskrivning i form av en referensmodell så som den beskriven här, då det blir lättare att kunna jämföra och utvärdera vilken produkt(er) som passar just ett specifikt fall.

3.3 Vad representerar de olika lagren i referensmodellen?

Vi kommer här att beskriva de olika lagrena i referensmodellen var för sig. De olika lagrena är numrerade enligt Figur 7 ovan.

Lager 1: Access services/adapters. Den kod som gör det möjligt att koppla samman applikationer med en integrations tjänst (så som en ESB, EAI, MOM) kallas för en adapter. Adaptrar sköter kommunikationen mellan integrations tjänster och applikationen där meddelanden skickas och tas i mot över ett specifikt protokol. Adaptrar är nödvändiga i de fall där applikationer som är såkallade föråldrade system (legacy systems) vilka inte själva kan exponera sig som Web Service. En adapter möjliggör för två applikationer som ”pratar” över olika protokol att kommunicera med varandra. De flesta integrations tjänster levereras med en mängd olika transport normaliserings adaptrar så som adaptrar för Web Services, FTP, HTTP/S, MSMQ, SQL, EDI, SMTP med flera.

Lager 2: Integrations services.

En integration tjänst är en fristående komponent som anropas som web service. Integrationstjänsten säkerställer ett definierat dataflöde mellan två aktörer, som kan vara en applikation eller en annan service. I integrationsvärlden implementeras en integrationstjänst som en orkestrering i en integrationsprodukt, t.ex. Microsoft Biztalk.[38]

Lager3: Information and rules services.

En information service är en komponent som använder sig av andra services för att bearbeta data och ta fram information utifrån gällande verksamhets behov.

Information services använder sig av olika integration-, rules- och directory services, för att utföra sin uppgift.

En information service är alltid inblandad när det rör sig om interaktion med externa komponenter för att bl.a. säkerställa säkerheten och äktheten.

Rules services innehåller alla verksamhetsregler, t.ex. vid order över 1 000 000kr måste kundens betalningsförmåga kollas.[38]

Lager 4: Process services.

Process services kan användas för att övervaka verksamhetsprocesser, hålla processbaserade sammansatta applikationer samman och för att övervaka och kontrollera informationsflöden.

Lager 5: External Channels.

External channels är de kanaler som systemet kan använda sig av för att kommunicera, det kan t.ex. vara HTTP, FTP eller SOAP.

Lager6: External actors.

External actors är aktörer som på något sätt vill kommunicera med systemet, det kan t.ex. vara andra system eller personer.

Lager7: Security.

Här hanteras allt som rör säkerheten, t.ex. kryptering.

Lager 8: Directory.

Directory kan ses som en katalog som innehåller information, t.ex. vilka tjänster som finns, behörighetsstyrning, roller mm.

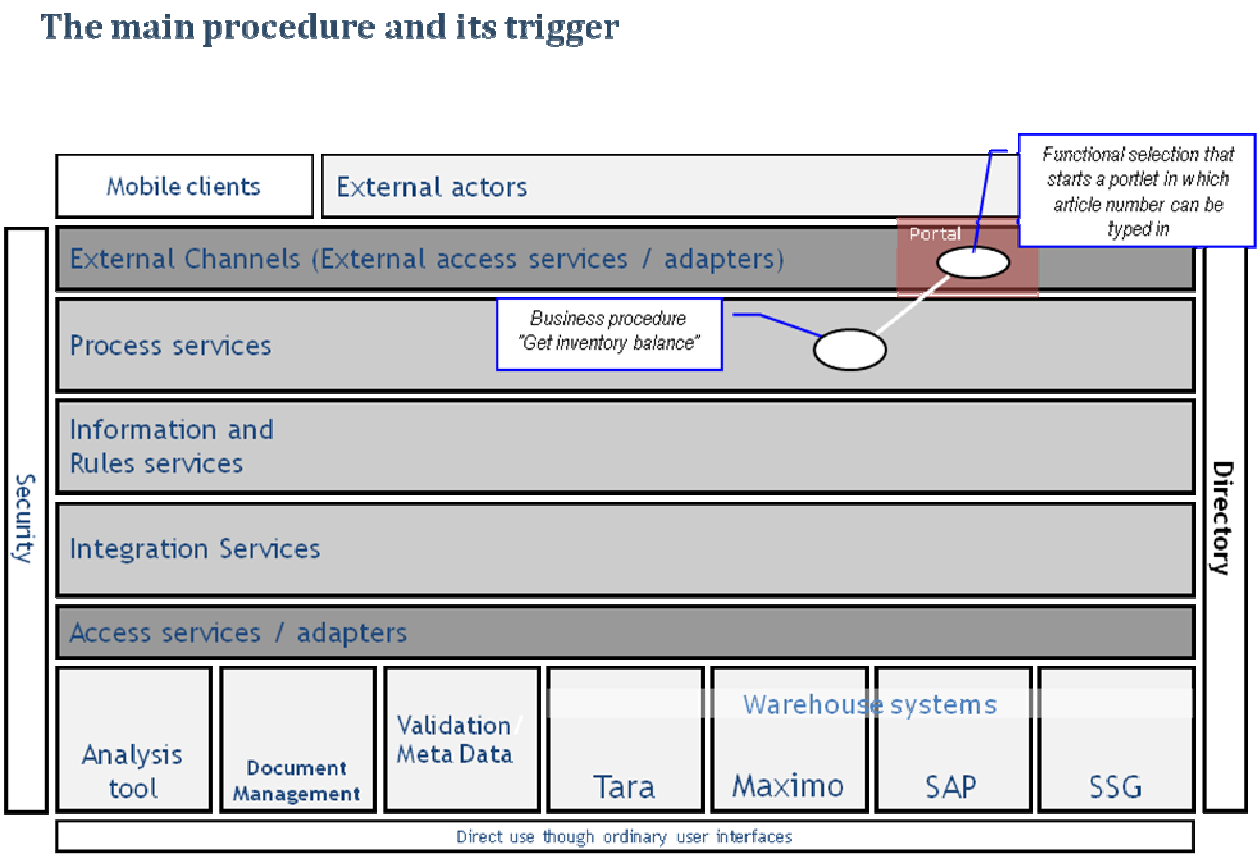
Warehouse systems är exempel på affärssystem som kan vara inblandade i en integrationslösning. t ex SAP.

Analysis tools är analysverktyg som kan användas för att övervaka och kontrollera flöden inom integrationslösningen. t ex BAM (Business Activity Monitoring).

Document management eller dokumenthantering fokuserar på att administrera elektroniska dokument under deras aktiva livscykel, med t.ex. check in/check out, versionshantering, statushantering, åtkomst- och behörighetskontroll som viktiga egenskaper.

3.4 Ett praktiskt exempel

För att kunna visa önskad information på en skräm kommer vi nedan att förklara och visa varje steg systemet måste göra för att kunna presentera den önskade informationen.

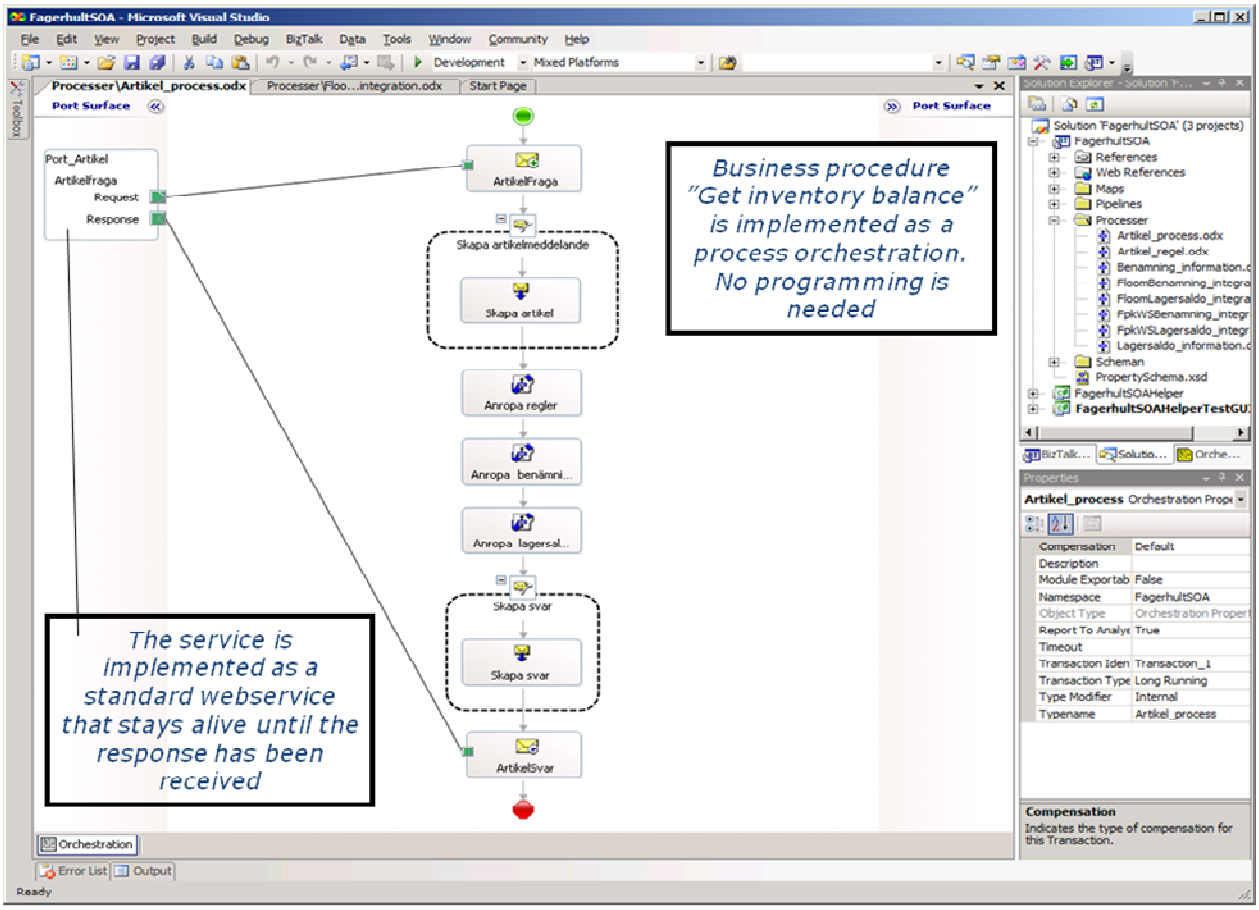


Figur 8 The main process and its trigger

I detta skede är det någon som har matat in ett artikelnr i en portal för att få reda på lagersaldot. Detta triggar igång en process som i slutändan ska kunna presentera ett svar, vilket lagersaldo angivet artikelnr har.

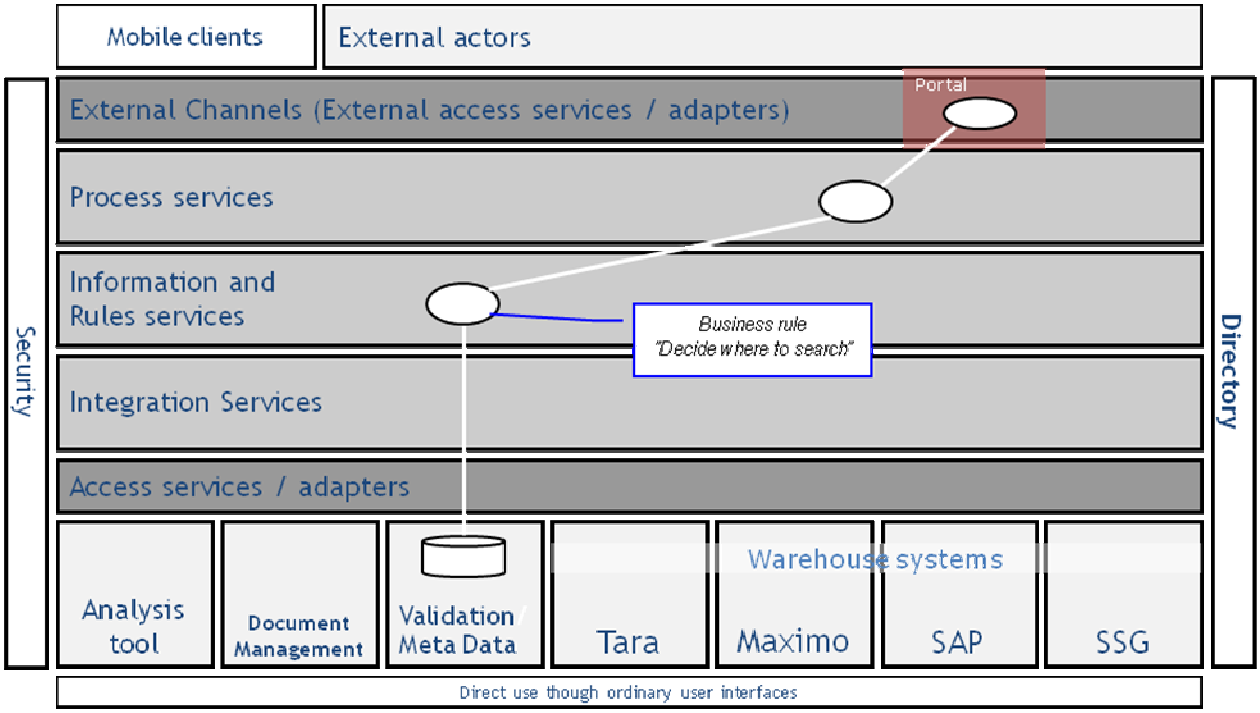
Som ni kan se på nästa bild är processen en rad händelser som sker i sekventiell följd. Händelserna kommer att utföras i tur och ordning vilket leder fram till ett svar som kommer att returneras.

Ett system består av många olika processer som i sin tur består av en specifik uppsättning händelser. Då systemet är tjänsteorienterat kan tjänster återanvändas i flera processer eller händelser.



Figur 9 The main process orchestrering

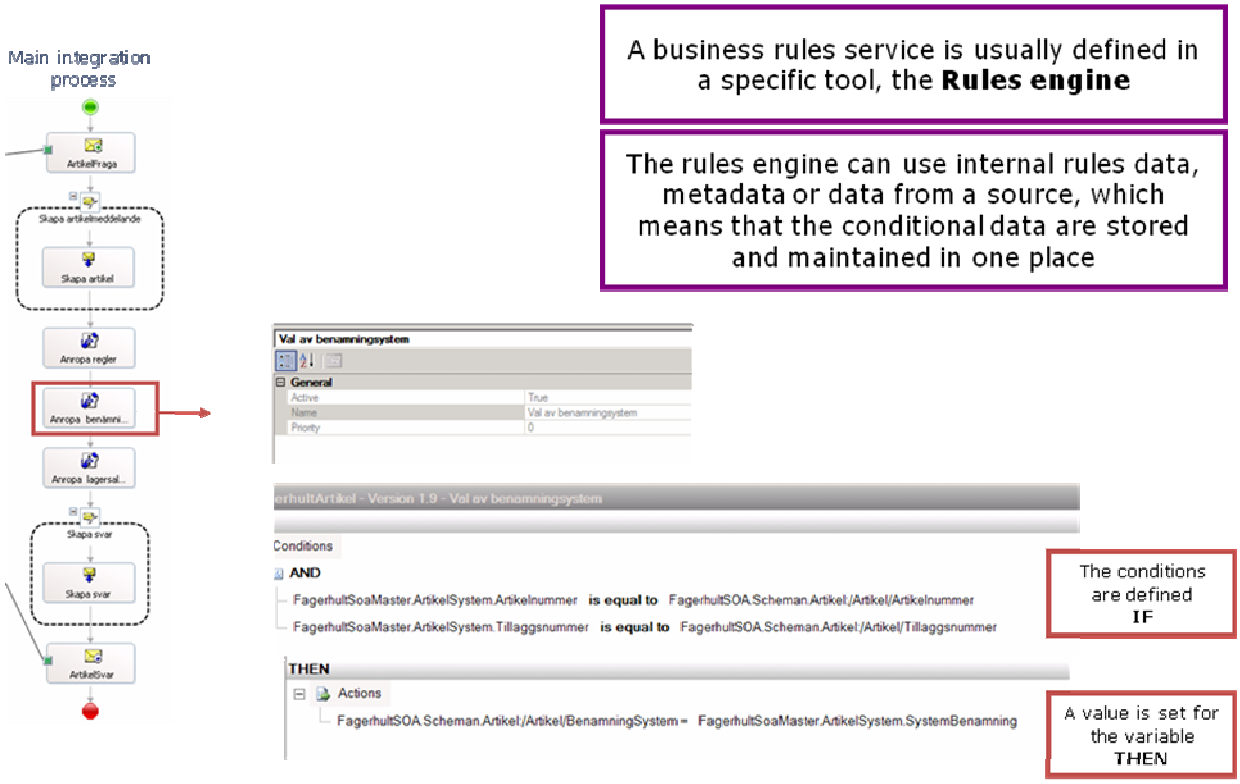
The Business rules service



Figur 10 Business rules

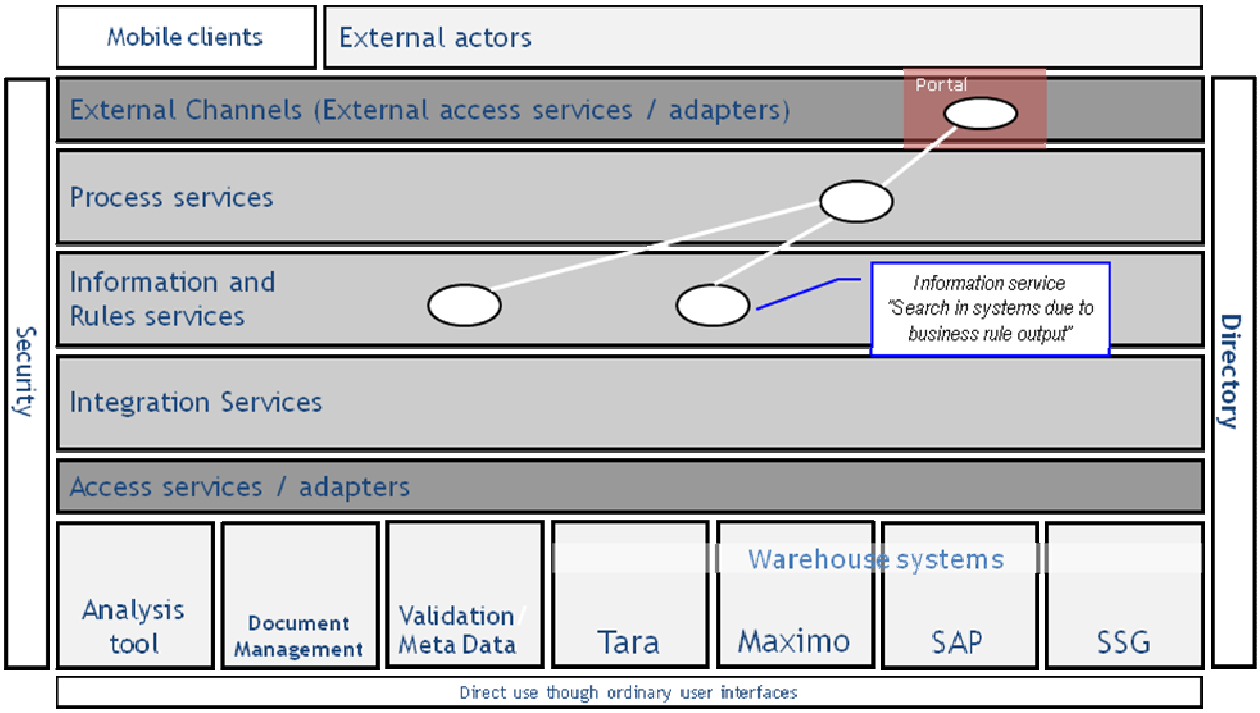
I figur 10 anropas verksamhetsreglerna för att avgöra vart man ska söka efter önskad information. Beroende på vilken typ av artikel som efterfrågas sker sökningen på olika sätt. Verksamhetsreglerna är oftast definierade i ett specifikt verktyg, ”Regel motorn”. Se figur 11.

The Business rules service



Figur 11 Rules engine

The Information service

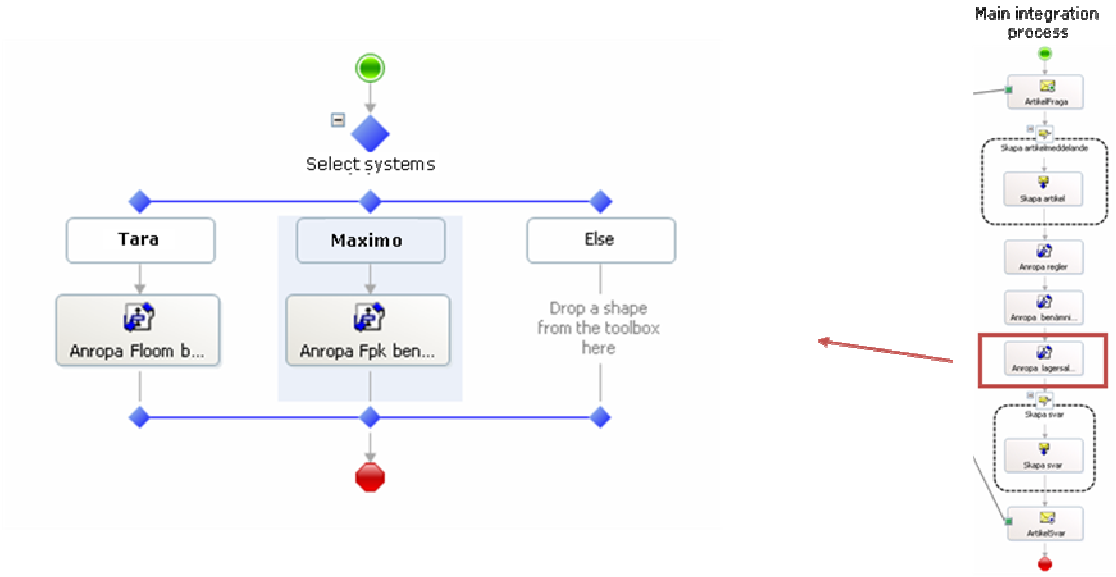


Figur 12 Information services

Information services kommer att anropa integration services men vilka tjänster som anropas beror på svaret från rules services. Verksamhetsreglerna kan begränsa urvalet tjänster att anropa.

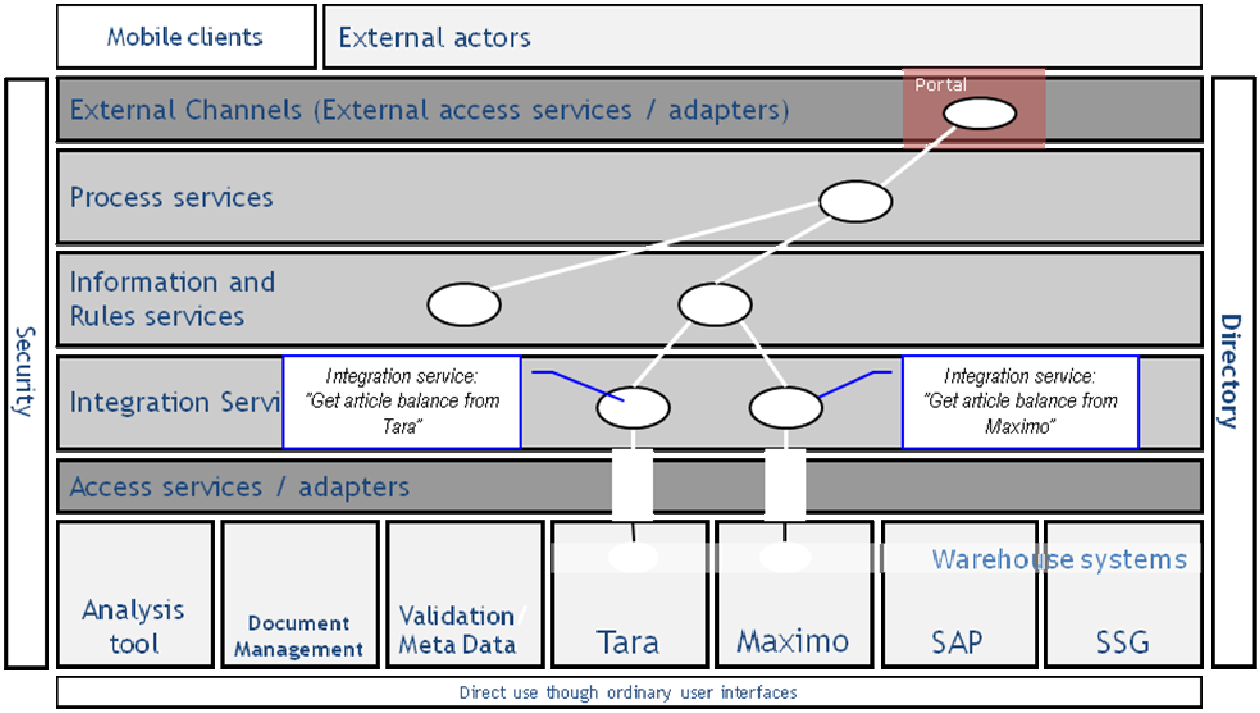
The Information service

In an integration solution platform the interfaces are implemented as schemas, which physically is and XML file (XSD-format)



Figur 13 Information services in main orchestrering

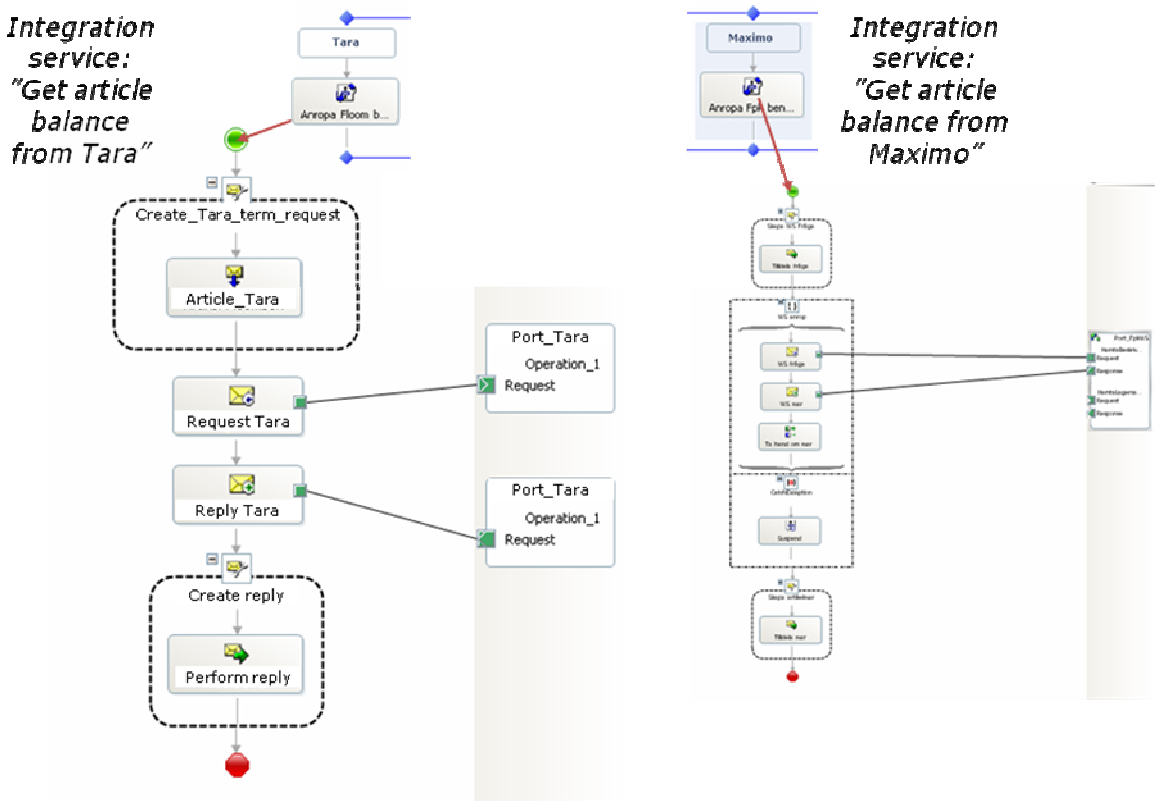
The Integration services



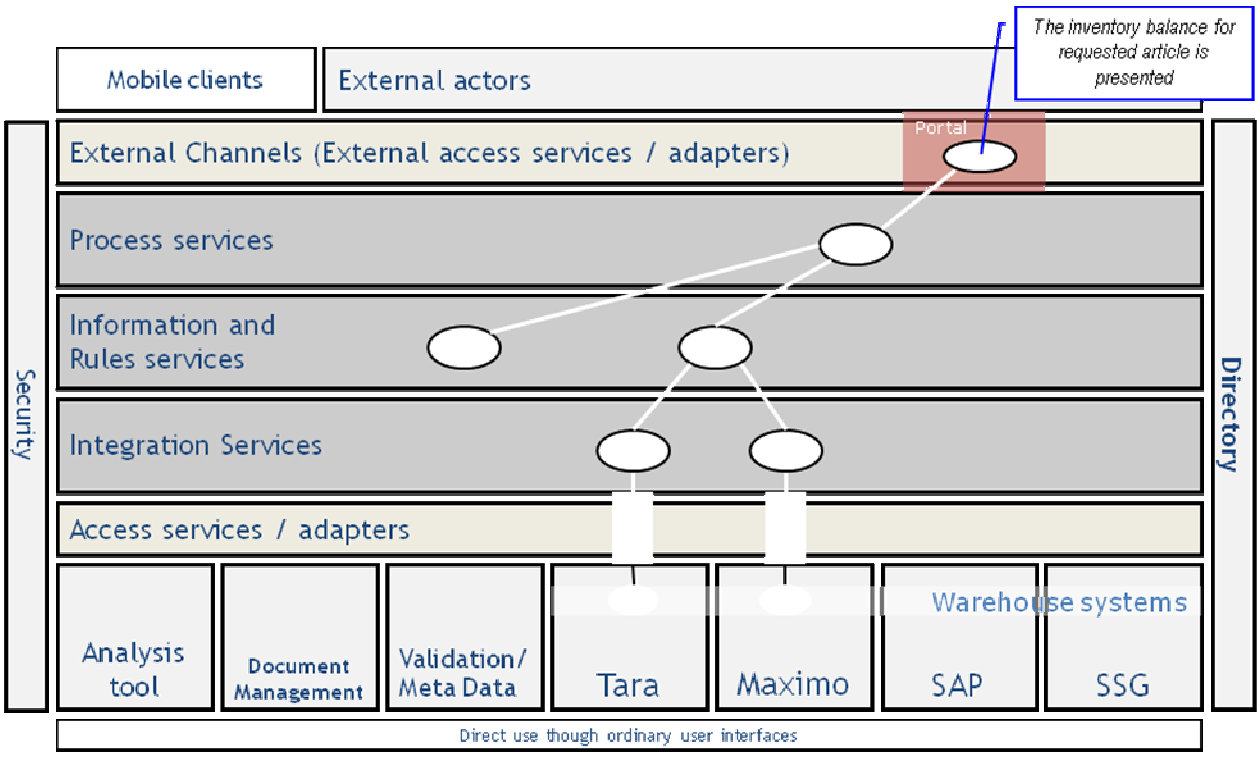
Figur 14 Integration services

Integration services kommer här att hämta den önskade informationen från de olika affärssystemen via adaptrar. Adaptrarna gör att man kan kommunicera med de olika affärssystemen. Det krävs en adapter för varje affärssystem då de kommunicerar på olika sätt.

The Integration services

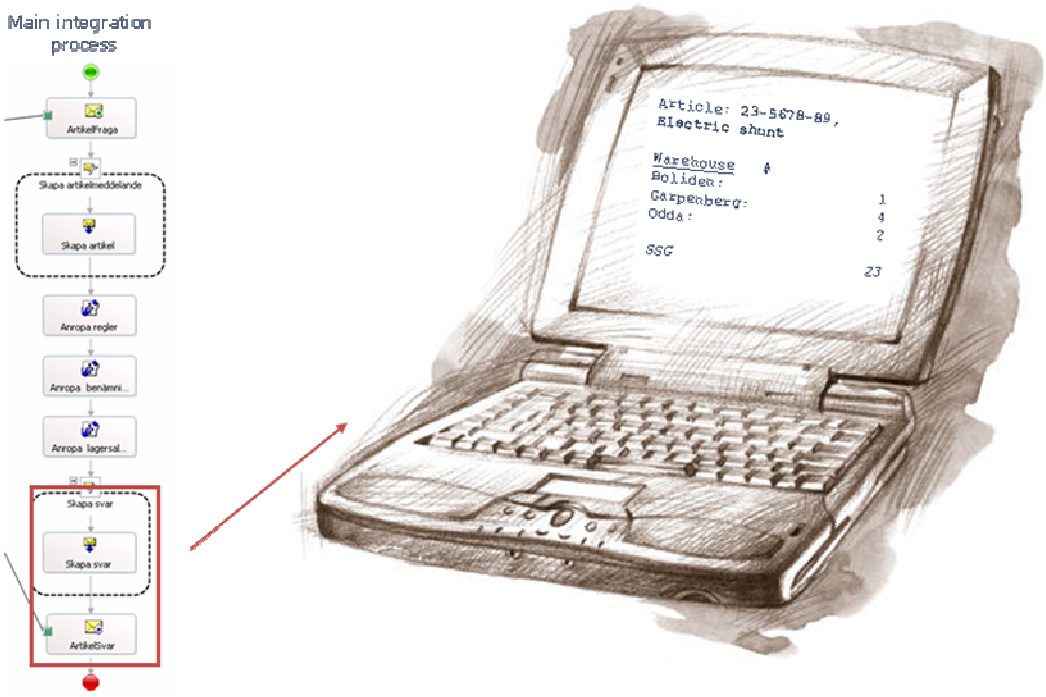


Figur 15 Integration services Get article balance



Figur 16 Reply saldo

Svaret som man får på sin förfrågan skickas stegvis tillbaka den som ställt förfrågan.



Figur 17 Show saldo

Svaret man får från de olika affärssystemen bearbetas och presenteras på ett lämpligt sätt. Ovan ser ni ett exempel på hur det skulle kunna se ut. Nu har förfrågan gått igenom hela referensmodellen.

3.5 Sammanfattning

I modern tjänsteorienterad integration behövs en referensmodell för att styra upp arbetet och få en överblick över systemet. Den referensmodell vi använder är framtagen av Per Björkegren på Sogeti och ingår i DIA som är en ansats för kontrollerad integrationsutveckling. En referensmodell är ett abstrakt ramverk för att förstå betydande relationer mellan entiteter. Modellen består av ett antal lager där varje lager har en specifik uppgift. De olika lagren i Per´s modell är:

- Access services/Adapters
- Integration services
- Information and Rules services

- Process services
- External channels
- External actors
- Security
- Directory

Ett syfte med referensmodellen är att få personer med olika tekniska bakgrunder att kunna prata och förstå varandra. Detta blir möjligt då det blir lätt att följa ett scenario i referensmodellen för att se vad som händer. Ett annat syfte är att utvecklare och IT-arkitekter har en mall att följa när de ska designa och implementera ett system.

Referens modellen för tjänsteorienterad integration används sammanfattningsvis till:

- Bestämma omfattning och diskutera runt en högnivå lösning.
- Beskriva olika scenarior.
- Bestämma vilka huvudsakliga tjänster ett integrationsfall ska bestå av.
- Analysera och definiera hur olika produkter av mjukvara möter de krav för SOA och integration.

4 Begreppet ESB

Begreppet ESB är ett relativt nytt begrepp som idag används flitigt ibland företagen inom integration. Det finns idag ingen vedertagen definition av vad ESB är. Vissa menar att det är en produkt[17][19] medan andra påstår att det enbart är ett sätt att implementera en tjänsteorienterad arkitektur[14]. Bland dem som anser att ESB är en produkt finns stora oklarheter vilka funktionaliteter som ska ingå i produkten för att den ska vara en ESB. Om man läser på olika tillverkares och analytikers hemsidor om vad ESB är, så kommer man att få lika många olika förklaringar som antalet sidor man tittar på. Vi ska i detta kapitel ge vår syn på begreppet ESB.

4.1 Definitioner av ESB

ESB refereras ofta som en grundstomme för SOA implementationer. ESB enligt Gartners definition är en ny arkitektur som utnyttjar web service teknologin, message oriented middleware teknologi, intelligent routing och transformation.[39] Genom ESB:n flödar mjukvaro tjänster och applikations komponenter [18]. Det finns ingen definition inom branschen som alla aktörer är överens om. Här följer några definitioner från aktörer inom branschen.

IBM:

An Enterprise Service Bus (ESB) is a flexible connectivity infrastructure for integrating applications and services. An ESB can power your service-oriented architecture (SOA) by reducing the number, size, and complexity of interfaces between those applications and services.
An ESB performs the following functions:

- *Route messages between services*
- *Convert transport protocols between requester and service*
- *Transform message formats between requester and service*
- *Handle business events from disparate sources*

An ESB should allow your organization to focus on your core business needs rather than the IT infrastructure required for connecting the programs together. An ESB should allow you to add new services or make changes to existing services with little or no impact to the use of existing services.[14]

Forrester:

Infrastructure software that makes reusable business services widely available to users, applications, business processes, and other services.[12]

Webmethod:

The term Enterprise Services Bus (ESB) is one that is being used increasingly with respect to Service-Oriented Architecture (SOA), yet there is still much confusion and debate as to what an ESB really is. The easiest way to understand ESB is to draw an analogy with message-oriented middleware (MOM). MOM provides features such as queuing, guaranteed delivery, routing, and publish/subscribe in support of Event-Driven Architecture. Similarly, an ESB provides capabilities that help to support the implement of Service-Oriented Architecture. In essence, an ESB can be viewed as the middleware for enabling SOA. However, whereas MOM solutions are typically proprietary, ESBs are largely based on the implementation of Web services standards such as SOAP, WSDL, UDDI, and the growing suite of WS- standards. While ESBs have been touted as an alternative to business integration solutions, webMethods' view is that ESBs address a much narrow set of requirements. webMethods offers ESB functionality as part of our broader business integration solutions to provide greater value for customers requiring synergy across different levels of their integration infrastructure. We understand that an ESB on its own is not a complete integration solution and to address business-level needs, companies require additional higher-level/value functionality, such as business process management, business activity monitoring, support for trading partner integration, and e-business standards support.[se bilaga A.3]*

Oracle:

It provides a much-needed intermediary layer that facilitates data delivery, service access, service reuse, and service management of an enterprise SOA implementation. ESB also supports intelligently-directed communication and mediates relationships among loosely coupled and decoupled business components. [13]

Microsoft:

En Enterprise Service Bus kan ses som en samling av arkitektuella mönster baserade på traditionell enterprise application integration(EAI), MOM, Web services, .NET and

Java interoperabilitet, värdsystem integration och interoperabilitet med tjänsteregister och tillgångsförråd. Applikation av dessa mönster tillhandahåller en infrastruktur som möjliggör flexibel och säker återanvändning av tjänster och möjligheten till snabb orkestrering av tjänster till ny end-to-end affärs process.[40]

Genom att läsa, enligt oss, dessa oklara och motsägande definitioner får man inte mycket inblick och förståelse för vad en ESB egentligen är. Vi ska i kommande stycken av detta kapitel ge vår syn på begreppet ESB.

4.1.1 Vår definition av begreppet ESB

Då det finns så många olika definitioner av begreppet finner vi det meningslöst att lägga till ytterligare en diffus definition.

4.1.2 Vad är ESB?

ESB är ett väldigt diffust begrepp som har olika betydelse för olika personer, det gör att det är svårt att säga exakt vad en ESB är. De flesta personer som vi pratat med och de artiklar och whitepapers som vi läst är dock eniga om att ESB är en integrationslösning som bygger på öppna standarder. Där slutar likheterna i deras beskrivningar av ESB. Det som gör att det skiljer så mycket på beskrivningarna är till största del funktionaliteten.

Det som några av de vi frågat som t.ex. IBM och Oracle är överens om och som även vi till fullo håller med om är att en ESB är en middleware-komponent som stödjer en implementation av SOA inom en koncern eller företag. De egenskaper som gör en ESB så lämplig vid implementation av SOA är dessa:

- Avskiljer den vy som den som använder tjänsten har från egentliga implementationen av tjänsten.
- Avskiljer tekniska aspekter vid tjänsteintegration.
- Integrerar och sköter tjänster inom företaget.

Genom att avskilja den syn på en tjänst som den som använder tjänsten har från den egentliga implementationen ökar kraftigt flexibiliteten hos arkitekturen. Det gör det möjligt att byta ut en tjänst utan att de som använder tjänsten kommer att påverkas eller märka det och utan att någon ändring behöver göra i arkitekturen för att en tjänst har bytts ut.

Denna avskiljning mellan de som använder tjänster och de som tillhandahåller tjänster görs bäst genom en mellanhand. Denna mellanhand publicerar tjänster till de som använder dem. De som vill använda en tjänst binder sig mot denna mellanhand för att komma åt en tjänst och detta sker då utan någon direkt koppling mellan den som vill komma åt tjänsten och de som tillhandahåller tjänsten. Mellanhanden mappar förfrågningen till den egentliga implementationen av tjänsten.

Det finns flera andra aspekter som denna mellanhand måste kunna hantera för att kunna möjligöra en SOA:

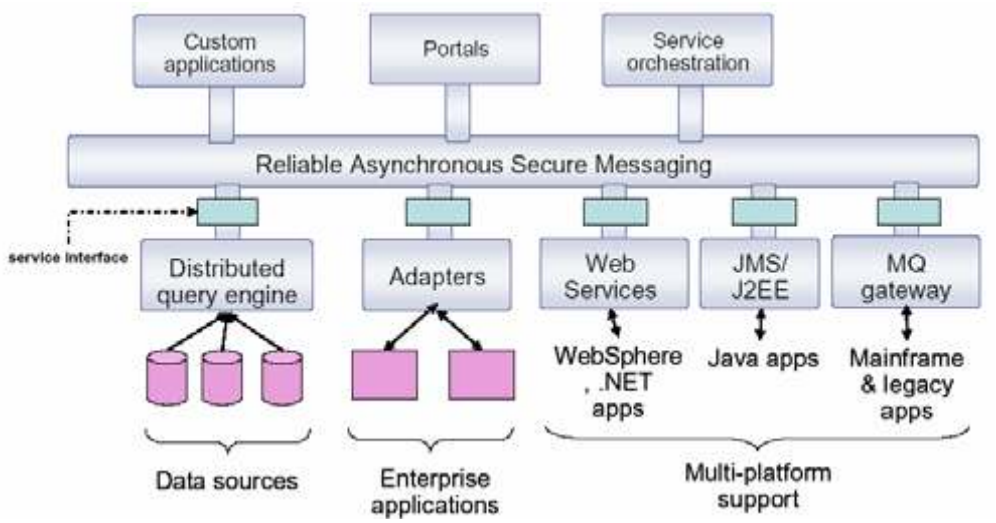
- Mappa tjänsteförfrågningar från ett protokol till ett annat.
- Transformera data format.
- Stödja ett antal olika säkerhetsmodeller mellan den som vill använda tjänsten och den som tillhandahåller tjänsten.
- Samla ihop tjänsteförfrågningar.
- Tillhandahålla meddelande egenskaper så som publish/subscribe och asynkron request/response.

Det stöd som denna mellanhand tillför är den roll en ESB spelar i en SOA.

Då funktionaliteten som en ESB ska klara varierar väldigt mycket beroende på vem man frågar kommer vi i kapitel 4.3 lista de funktionaliteter som alla vi pratat med är överens om måste ingå. Vi ser på ESB som en integrationslösning och anser att det inte finns någon begränsning hur mycket funktionalitet som kan kopplas på lösningen. Den grundfunktionalitet som en ESB måste ha enligt oss är: routing, transformering,

konvertering och connectivity. Vi anser att lösningen bör anpassas efter behov och önskemål därför ändras funktionaliteten hos en ESB från fall till fall.

Den bild som brukar förekomma i skrifter och diskuterande texter gällande ESB ser ut enligt följande:



Figur 18 ESB illustration

Bussen fungerar som en sammanbindande komponent som tjänster, applikationer, portaler och datakällor kan koppla upp sig mot. På bussen exponeras allt som tjänster vars adress återfinns i en databas i bussen. Bussen tillhandahåller funktionalitet som styr meddelandehanteringen i systemet. Den är hjärtat i systemet och alla meddelanden går genom bussen som i sin tur förmedlar, transformerar och konverterar dem.

Det pågår en diskussion om ESB är ett mönster eller produkt.[14][19] Vi har svårt att placera ESB som antingen mönster eller produkt utan tycker att det är båda delarna.

4.2 Funktionalitet och egenskaper

- **Routing**

  När meddelanden ska skickas bestäms vart de ska skickas beroende av den typ av adressering som gäller. De vanligaste typerna är Web service-adressering och innehållsbaserad (content-based). Web service-adressering fungerar så att när ett tjänsteanrop når bussen läser bussen i meddelandet till vilken webbadress det ska till och skickar förfrågan vidare till service providern som är kopplad till den adressen. Detta görs utan att den anropande tjänsten vet vart mottagaren finns. Det är detta som kallas location transparancy.[30]

  Content-based routing använder sig av regler och affärslogik som appliceras på innehållet i meddelandena för att avgöra till vilken service provider meddelandet ska skickas.

  För att systemen ska bli flexibla ska routingen klara av att sköta denna hantering både dynamiskt och statiskt.

- **Transformering**

  Transformering är en väldigt viktig uppgift för ESB´n då tjänsternas tjänstegränssnitt ska matchas mot det inkommande meddelandet. Utformningen på det anropande schemat måste överensstämma med mottagarens för att få tillgång till tjänsten. Det kan även vara så att sändare och mottagare inte använder samma dataformat då krävs en transformering för att de ska kunna kommunicera med varandra. Transformeringen kan t.ex. ske mellan XML scheman eller till XML från datakällor som inte är XML och tvärtom.

- **Konvertering**

  För att möjliggöra att applikationer som inte använder samma protokoll ändå ska kunna kommunicera med varandra, krävs att ESB´n kan ta emot ett meddelande skickat med ett protokoll, bearbeta det och skicka iväg det med ett annat protokoll till mottagaren. T.ex. en service provider erbjuder en service baserad på protokollet HTTP medans en service consumer endast är kapabel att kommunicera via ett annat

protokoll, MQ. För att dom ska kunna kommunicera med varandra krävs att ESB´n kan översätta från HTTP till MQ.

Detta är väldigt vanligt i miljöer där gamla resurser har gjorts om till tjänster, men nyare applikationer kan inte använda dessa tjänster pga. att de använder olika transport protokoll. Så istället för att ha en adapter på tjänsterna är det bättre att låta ESB´n hantera detta.[32]

- **Adaptrar (Connectivity)**

  ESB´n ska ha adaptrar så att man kan koppla på gamla system som inte är uppbyggda av tjänster. Många av dagens system är fortfarande inte uppbyggda av tjänster så adaptrarna är och kommer att vara en viktig del av ESB´n. Med hjälp av adaptrarna exponeras även de gamla systemen som tjänster. Det måste också finnas adaptrar för att klara av att kommunicera med olika system via t.ex. FTP, http, SMTP, MQSeries, MSMQ, EDI mfl.

- **Stöd för olika MEP´s(Message Exchange Pattern) t.ex.**

  Request/response

  När en tjänst eller en applikation vill komma åt en tjänst skickar dom ett meddelande (request) till ESB´n som behandlar meddelandet och skickar det vidare till lämplig tjänst. Den mottagande tjänsten utför sitt jobb och skickar ett meddelande (response) tillbaka till ESB´n som kollar upp vart meddelandet ska och skickar iväg meddelandet till den tjänst/applikation som skickade requestmeddelandet. Tjänsterna har ingen vetskap om varandra utan de ”ser” bara ESB´n. När request/response används på bussen sker detta asynkront, dvs att en request kan skickas iväg men tjänsten låser sig inte i ett läge där den väntar på svar utan den fortsätter. När svaret kommer tillbaka bearbetas svaret och tjänsten/applikationen fortsätter utföra sin uppgift.[33] Request/response kan användas både synkront och asynkront.

Publish/subscribe

Den tjänst/applikation som vill publicera (publish) ett meddelande skickar det till ESB´n som undersöker vilka som har anmält sitt intresse för att prenumerera (subscribe) på denna händelse. Därefter skickas meddelandet ut till de som vill ha det. På detta sätt kommunicerar tjänster/applikationer med övriga utan att veta om vilka dom är eller vart dom finns. En annan händelse skulle kunna vara att en server går ner, då triggas en händelse och de som vill ha den informationen kommer att få det via ett meddelande. Publish/subscribe skickar meddelanden asynkront vilket leder till ökad skalbarhet och mer dynamisk nätverks topologi.[34]

- **Transaktions hantering**

  När bussen blir anropad påbörjas ett flow som ska utföra ett antal uppgifter i följd, antalet kan vara 1-∞. Uppgifterna utförs en och en i en sekvens. För att det inte ska kunna bli inkonsistens i systemet finns en funktion som kallas transaktions rollback. Funktionens uppgift är att ställa tillbaka systemet i det skick som det var innan den senaste flowen påbörjades, dvs rulla tillbaka de uppgifter som blivit utförda. Detta ska ske om det av någon anledning inte gick att genomföra alla uppgifter i det aktuella flowet.

- **Plattform och språkoberoende**

  En av de stora fördelarna med ESB är att applikationer på olika plattformar kan kommunicera med varandra. De kan även vara skrivna i olika språk då bussen transformerar meddelandena det interna formatet XML och sedan till det format som mottagaren använder. T.ex. kan en applikation utvecklad i Java kommunicera med en applikation utvecklad i .NET.

- **Baserad på öppna standarder**

  För att bussen ska kunna användas av alla, oberoende av språk eller plattform ska den baseras på öppna standarder så som XML, HTML, SOAP, WSDL, UDDI och WS*. Detta gör att alla kan kommunicera med bussen på ett standardiserat sätt.

- **Säkerhet**

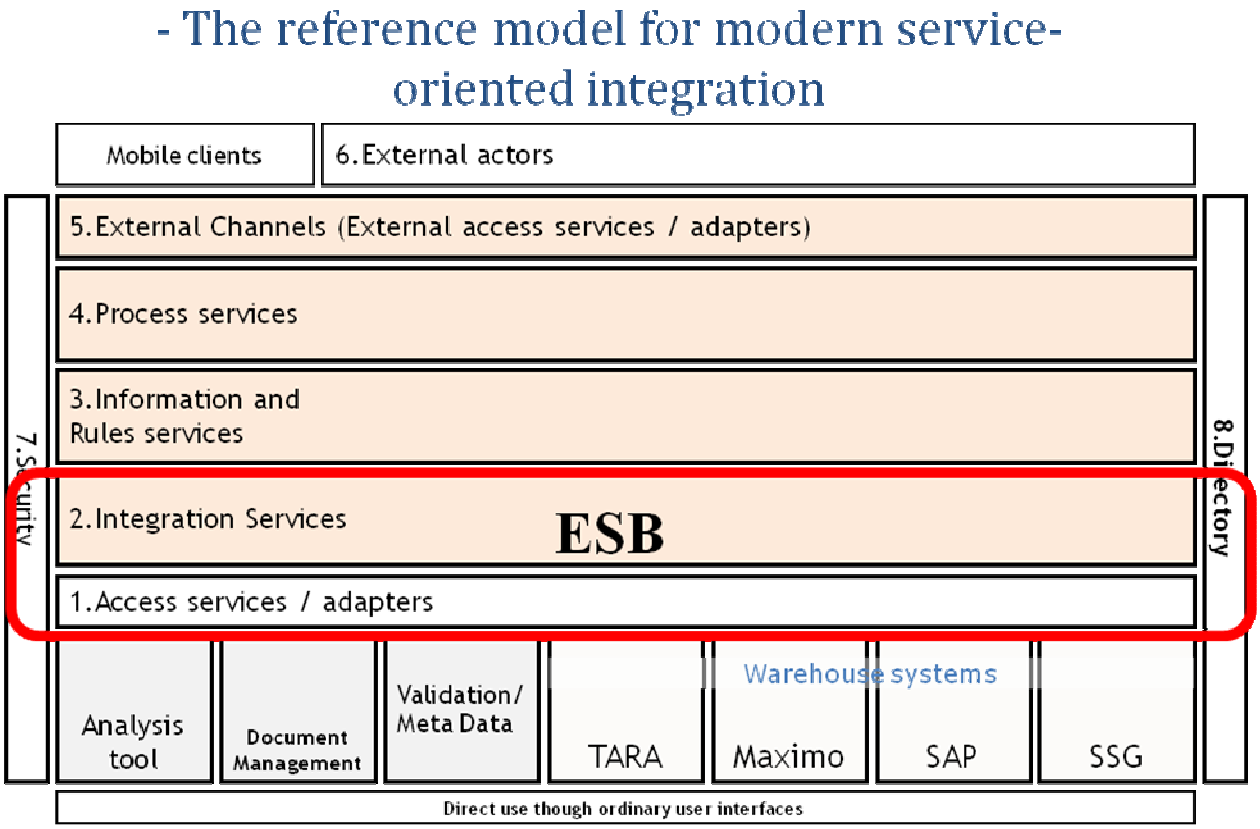
  Ska ha ett ramverk för att säkerställa säkra överföringar av meddelanden mellan applikationer.

- **Queuing**

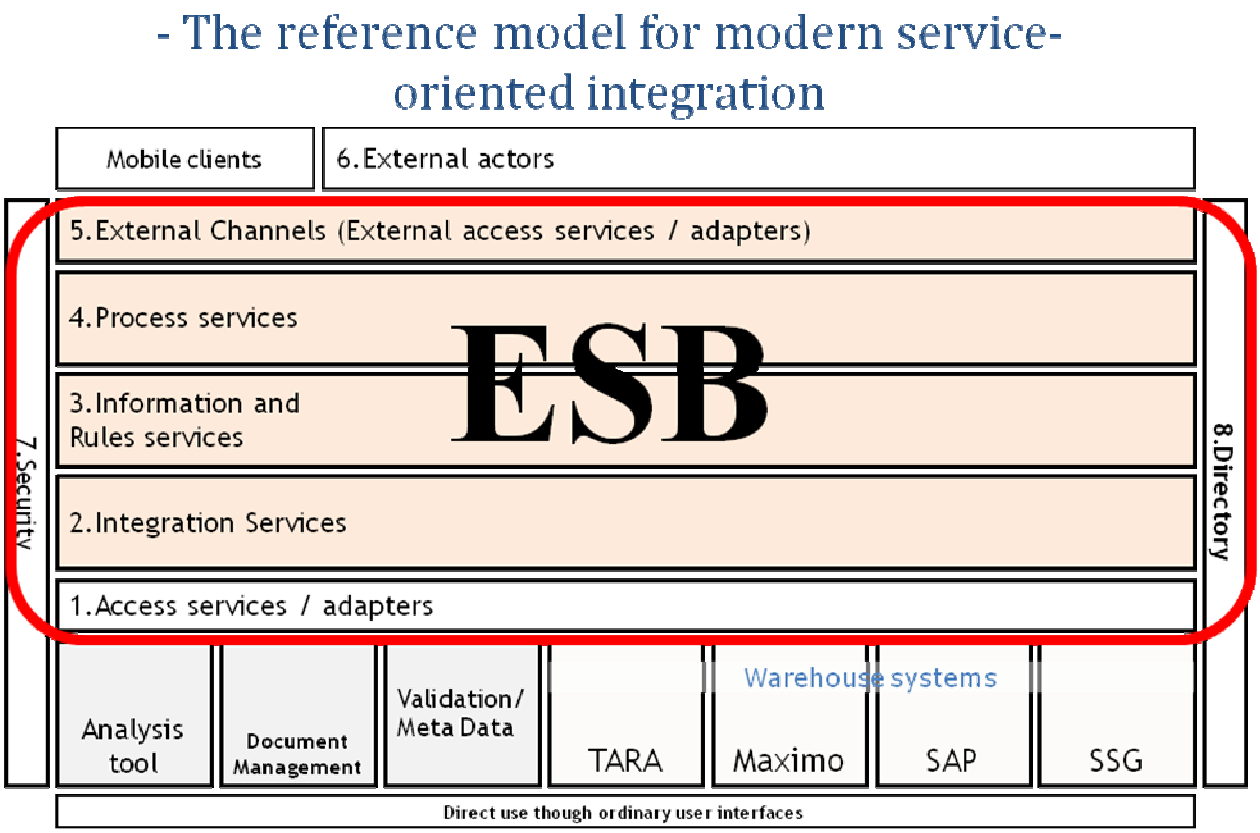
  ESB´n måste klara av att hantera att anropade tjänster för tillfället inte är tillgängliga. Då måste anropet läggas i en kö för att sedan skickas när tjänsten blir tillgänglig.

4.3 ESB i referensmodellen

Då en ESB möjliggör en tjänsteorienterad integrationslösning då den stödjer konceptet SOA fullt ut kan man koppla ihop den med referensmodellen som presenterades i kapitel 3. Då funktionaliteten hos en ESB kan variera väldigt mycket skiljer det sig också väldigt mycket vilka delar av referensmodellen som kan anses vara ESB funktionalitet. Med referensmodellens hjälp kan vi visa vilken roll en ESB har i en tjänsteorienterad integrationslösning.



Figur 19 Referensmodellen med minimal ESB



Figur 20 Referensmodellen med utökad ESB

För att kunna implementera en SOA måste både applikationer och infrastruktur stödja SOA principer. För en applikation att kunna medverka i en SOA krävs att man skapar tjänste- gränssnitt för redan existerande och nya funktioner vilket kan göras direkt eller genom adapters. För att en infrastruktur skall vara anpassad för SOA måste förmågan att kunna dirigera (routa) och transportera tjänsteförfrågningar till korrekt tjänstetillhandahållare finnas på plats. Det är här en ESB kommer in i bilden när man pratar om SOA då den tillhandahåller denna förmåga till infrastrukturen[37].

Det kanske viktigaste och mest grundläggande en ESB tillför till en infrastruktur är att den möjliggör löst kopplade komponenter och möjligheten för olikartade system och applikationer på olika plattformar byggda i olika språk att kommunicera med varandra oberoende av var tjänsterna är belägna[37].

4.4 Varför ESB?

Om man vill bygga sitt system kring SOA-konceptet med löst kopplade komponenter och web services då är ESB en väldigt viktig pusselbit för att göra detta möjligt.

I företag som växer eller minskar underlättar det förändringen om man har infört en ESB med tjänster. Dels blir det lättare att integrera nya system som t.ex. uppköpta bolag har i drift och dels kan det bli lättare att sälja av eller outsourca delar av företaget då dess funktioner är löst kopplade via tjänster. Det kan även medföra minskade kostnader i och med att man kan återanvända tjänsterna[11][14][16]. För att underlätta återanvändning bör tjänsterna läggas in i ett register så att de är lätta att hitta.

Införandet av en ESB kan ha stor påverkan på ett företags system, då införandet även bör ha en väl genomtänkt referensmodell som hela systemet ska vara uppbyggt kring. I och med att man har en referensmodell att utgå ifrån, kan alla som jobbar med systemet se hur systemet är uppbyggt och få en överblick över det. De får även lättare att förstå vad som krävs för att uppnå de ställda kraven enligt referensmodellen. Detta leder till att det blir betydligt lättare att integrera affärssystem, dels de från olika plattformar, en ESB är plattformsoberoende, men även affärssystem inom samma plattform. Eftersom man inte är beroende av en specifik plattform kan man välja ut de egenskaper/funktioner som passar företaget bäst oavsett plattform, detta kallas best-of-breed[15].

Ett företag som implementerar en ESB kommer att få ett flexibelt system då det är väldigt lätt anpassa systemet efter önskemål.

Kostnaderna kan även minskas genom att IT-personalen som har att göra med ESB'n och dess tjänster att göra, bara behöver klara av att hantera vissa produkter och protokoll samt de tjänster som de äger[21]. Övriga tjänster känner de bara till via dess interface och de behöver inte veta hur eller vart de är implementerade, ansvaret för en tjänst och dess implementering ligger hos den som äger tjänsten.

4.5 När ska man implementera en ESB?

En ESB-lösning kan alltid implementeras, men det har störst värde om man vet att man har applikationer eller tjänster med olika meddelandeformat/transportprotokoll[bilaga A.2]. Om man har många tjänster så att det blir svårt att hålla reda på dem är det lämpligt att implementera en ESB så att man på ett enkelt och bra sätt underlättar återanvändning. Ett företag som behöver en flexibel integrationslösning som man snabbt kan anpassa efter nya krav eller affärsmöjligheter kommer att ha stor nytta av att implementera en ESB-lösning.

Investeringskostnaden för en ESB-lösning kostar från ca 400000kr, för att man ska kunna motivera investeringen krävs att man har ett behov eller vet att man kommer få det längre fram. Man kan utveckla sina system enligt SOA konceptet utan att ha en ESB-lösning.[29] Men man får ingen affärsnytta med ESB´n utan SOA, självklart går det men det lönar sig inte.[31] Märker man sen att man har ett behov av en ESB-lösning kan man införa den när behovet uppstår. Då kan man vänta med investeringskostnaden till dess att ett behov uppstår.[31]

4.6 Fördelar med en ESB

Dagens affärsmarknad kräver att man snabbt kan anpassa sina system för att inte tappa kunder. En ESB möjliggör enkel och snabb integration av applikationer och processer, vilket medför att man kan agera snabbt och anpassa systemet direkt när behovet uppstår[11].
En ESB medför både verksamhets- och IT fördelar.

4.6.1 Verkamhetsfördelar

- Förbättra verksamhetens flexibilitet
  En ESB gör företaget snabbare, mer tillgängligt och mer anpassningsbart. Till skillnad från många traditionella infrastrukturer som är svår ändrade kan förändringar göras

enkelt och smärtfritt i en ESB pga. dess design. Denna flexibilitet gör att det blir enkelt att lägga till eller ta bort affärspartners direkt när behovet uppstår. Detta medför även att man kan publicera produkter eller tjänster snabbare.[11][14][21][26][28]

- Intelligentare verksamhet

  Igenom att använda sig av en ESB kan man få en inblick i verksamheten som man tidigare inte hade. Detta möjliggörs av att man i en ESB övervakar all aktivitet som går på bussen. Man kan samla in information som senare skrivs ut som rapporter eller diagram för att se hur verksamheten går. Om man vill kan man få ut informationen i realtid.[21][28][27]

- Minskade kostnader

  ESB minskar företagets kostnader då dess effektivitet och flexibilitet bidrar till att mer kan utföras med mindre personal. Utveckling av mjukvara går mycket fortare pga. att man kan utnyttja redan färdiga tjänster vilket i längden leder till att kostnaden för projekten kan hållas nere.[11][14][21][25][26][28]

4.6.2 IT fördelar

- Minskade kostnader

  En ESB bidrar på flera sätt att minska kostnaderna. Dess arkitektur är uppbyggt på ett sådant sätt att man inte behöver periodiska uppgraderingar vilket håller investeringskostnaderna nere. Nyutveckling eller uppdatering av befintliga tjänster bygger på att man återanvänder så mycket man kan av redan färdigutvecklade tjänster, vilket leder till att man snabbt och enkelt kan leverera nya tjänster. I och med att en ESB bygger på öppna standarder blir det lättare att hitta kompetent personal och man behöver inte lägga ner lika mycket pengar på utbildning av personalen. Öppna standarder leder också till att underhållet underlättas.[21][22][24][25][28]

- Ökad flexibilitet

  Eftersom tjänsterna är löst kopplade är det inga problem att göra ändringar i systemet för att anpassa sig och få systemet att fungera enligt nya krav. Detta kan göras medans systemet är i drift pga. att tjänsterna är löst kopplade. Vilket leder till att ändringar kan göras snabbt eftersom man inte behöver vänta på att systemet ska tas ur drift för att kunna uppdateras.[11][14][22][23][28]

- Best of breed

  ESB är plattforms- och programspråksoberoende vilket leder till att man kan välja den plattform eller det språk som passar bäst för en specifik applikation.[11][15][28]

- Ökad produktivitet och minskad utvecklingstid

  Genom att använda sig av färdiga tjänster kan man snabba på utvecklingstiden och öka produktiviteten vilket leder till att man snabbare kan få färdiga produkter. [20][22][28]

- Ökar systemets pålitlighet

  Införandet av en ESB ökar driftsäkerheten som systemet då det inte finns någon single-point-of-failure. Den fysiskt decentraliserade bussen ökar skalbarheten medans den logiskt centraliserade ökar tillgängligheten. I och med att man kan göra konfigurationsändringar utan att behöva ta systemet ur drift ökar dess ”upptime” och därmed även pålitligheten.[21][22][23][24][28]

4.7 Nackdelar med en ESB

En nackdel som kan bli följden av att begreppet ESB är så ”hett” är att företag köper en ESB utan att egentligen ha behovet. Många kan falla för säljsnacket att ESB är en revolutionerande lösning på alla integrationsproblem, vilket inte är helt sant. Detta kan leda till dyra investeringskostnader som kunde ha blivit lösta igenom en billigare och bättre anpassad integrationslösning. ESB är ingen universell lösning som löser alla problem.[31]

Då ESB är relativt nytt på marknaden har den inte testats under någon längre tid så man vet inte om den är driftsäker. Den baseras även på standarder som är under utveckling vilket kan leda till olika versioner av standarderna. Detta kan leda till att ESB´s från olika leverantörer så småningom kan använda sig av olika versioner dvs. inte blir kompatibla med varandra. En annan sak som talar emot ESB är att det inte finns någon klar definition om vad det är eller vad det ska klara av. Varje leverantör av en produkt har sin uppfattning om vad det är, samt att IT-arkitekter som jobbar med integration har en annan syn på det.[35] Vem har rätt? Vem ska man tro på? Detta leder till att det skapas missförstånd när man diskuterar ESB.

4.8 Sammanfattning

Det finns inte en enda definition av ESB som är exakt stämmer överens med en annan och det blir då väldigt svårt att beskriva exakt vad en ESB är. Några vi frågat menar att en ESB är en integrationslösning som bygger på öppna standarder och andra som säger att en ESB är en middware-komponent som stödjer en implementation av SOA. Dessa två definitioner är dock inte helt olika då SOA bygger på öppna standarder men inte lika heller på grund av att SOA är så mycket mer än bara öppna standarder. De funktioner vi anser bör finnas med i en ESB för att kunna lösa integrationsproblem på ett bra sätt anser vi och många av de vi frågat är routing, transformering, konvertering och connectivity. Vi har också konstaterat att en integrationslösning bör anpassas efter behov och önskemål därför ändras funktionaliteten hos en ESB från fall till fall.

5 Reflektioner

Arbetet med denna rapport har varit väldigt givande och vi har lärt oss mycket under tiden. Det har varit svårt att hitta information om ESB då det inte finns så många artiklar och böcker i ämnet. Vi har blivit tvungna att läsa whitepapers som kommer från leverantörer och de är med stor sannolikhet vinklade för att passa deras produkt. Med tanke på att vi inte hade någon erfarenhet av integration innan denna rapport var det svårt att greppa ämnet till en början då det var olika definitioner och förklaringar vart man än läste. Detta gjorde att våra frågor som ligger till grund för enkätundersökningen inte riktigt blev så bra som vi hade tänkt. Nu när vi fått fördjupad kunskap inom ämnet skulle vi ha formulerat frågorna på ett annorlunda sätt. Vår metod har sina svagheter i att tolkningen av svaren kan bli missvisande pga. antalet svar.

När vi besökte företag och pratade med dem angående integration och ESB borde vi ha gjort en riktig intervju som vi kunde ha använt oss av i rapporten.

6 Undersökning

För att ta reda på hur företag inom de olika segmenten leverantör, konsult och kund ser på begreppet ESB har vi skickat ut en enkät som de har fått svara på. Intresset för ämnet ute bland företagen har varit jättestort och vi har fått många svar som gett oss bättre insikt om vad de lägger in i begreppet ESB. Vissa företag har vi även besökt och pratat med dem angående deras syn på begreppet. Nedan kommer vi sammanställa svaren från enkätundersökningen. Vill ni läsa företagens fullständiga svar kan ni läsa dem bland bilagorna.

6.1 ESB-leverantörer

För att få djupare förståelse när det gäller ESB har vi ställt några frågor till företag som ligger i spetsen vad gäller integrationsutveckling.

1. Hur definierar ni begreppet ESB?

Microsoft

An Enterprise Service Bus can be thought of as a collection of architectural patterns based on traditional enterprise application integration (EAI), message-oriented middleware, Web services, .NET and Java interoperability, host system integration, and interoperability with service registries and asset repositories. Application of these patterns provides an infrastructure that enables the flexible and secure reuse of services and the ability to rapidly orchestrate services into new end-to-end business processes.

IBM

An Enterprise Service Bus (ESB) is a flexible connectivity infrastructure for integrating applications and services. An ESB can power your service-oriented architecture (SOA) by reducing the number, size, and complexity of interfaces between those applications and services.
An ESB performs the following functions:

- Route messages between services
- Convert transport protocols between requester and service
- Transform message formats between requester and service
- Handle business events from disparate sources

An ESB should allow your organization to focus on your core business needs rather than the IT infrastructure required for connecting the programs together. An ESB should allow you to add new services or make changes to existing services with little or no impact to the use of existing services.

Oracle

An ESB provides a much-needed intermediary layer that facilitates data delivery, service access, service reuse, and service management of an enterprise SOA implementation. ESB also supports intelligently-directed communication and mediates relationships among loosely coupled and decoupled business components.

WebMethods

The term Enterprise Services Bus (ESB) is one that is being used increasingly with respect to Service-Oriented Architecture (SOA), yet there is still much confusion and debate as to what an ESB really is. The easiest way to understand ESB is to draw an analogy with message-oriented middleware (MOM). MOM provides features such as queuing, guaranteed delivery, routing, and publish/subscribe in support of Event-Driven Architecture. Similarly, an ESB provides capabilities that help to support the implement of Service-Oriented Architecture. In essence, an ESB can be viewed as the middleware for enabling SOA. However, whereas MOM solutions are typically proprietary, ESBs are largely based on the implementation of Web services standards such as SOAP, WSDL, UDDI, and the growing suite of WS-* standards. While ESBs have been touted as an alternative to business integration solutions, webMethods' view is that ESBs address a much narrow set of requirements. WebMethods offers ESB functionality as part of our broader business integration solutions to provide greater value for customers requiring synergy across different levels of their integration infrastructure. We understand that an ESB on its own is not a complete integration solution and to address business-level needs, companies require additional higher-level/value functionality, such as business process management, business activity monitoring, support for trading partner integration, and e-business standards support.

2. Vilka funktionaliteter anser ni måste ingå i en modern tjänsteorienterad integrationslösning för att få kallas ESB?

   Endast ett företag svarade uttryckligen att tre funktionaliteter måste ingå i en ESB, dessa tre är: routing, transformering och virtualisering av endpunkter. Övriga var mer försiktiga och nämnde funktionaliteter som bör vara med i en ESB, dessa är:

   - Tjänste register
   - SOA Governance management
   - Säkerhet, transaktionshantering, loggning
   - Protokollkonvertering
   - Transformering
   - Routing
   - Quality of service tex belastnings balansering
   - Övervakning
   - Publish/subscribe
   - Ramverk för adaptrar
   - Vara byggd för hög skalbarhet och tillgänglighet
   - Ha en utvecklingsmiljö som är integrerad med övriga delar av en SOA-arkitektur, för att underlätta återanvändbarhet i alla led.
   - Centraliserad felhantering
   - Orkestrering
   - Rules engine

3. Har ert företag en ESB lösning?

   Alla företagen har en ESB lösning men Microsoft är det enda företaget som inte har en ESB produkt i sitt sortiment. De komponerar ihop en ESB lösning utifrån kundens önskemål med sina andra produkter.

4. Har ni planerat att släppa en ESB lösning?

   Då alla har en ESB lösning blir inte denna fråga aktuell.

5. Vilka funktionaliteter har er ESB lösning?

   Microsoft

Eftersom vi inte har någon ESB produkt anpassar vi lösningen efter kundens behov och önskemål. Det är upp till kunden vilka funktionaliteter som ska ingå.

IBM

Se bilaga A.2. Väldigt förenklat dock:

Basfunktionalitet:

a. Pub/sub
b. Message Transformering
c. Protokoll konvertering
d. Event Management

Tillägg:

Adapter ramverk och adaptrar gärna med stöd för kanoniska format.

Oracle

Oracle ESB is a standards based infrastructure component of the Oracle SOA Suite delivering loosely coupled data and enterprise application integration that is (i) fast, (ii) secure, (iii) reliable and (iiii) highly available. Oracle ESB features a multi-protocol message bus with centralized monitoring management of distributed services where all services are exposed as standard web services using WSDL. Oracle ESB contains message flows utilizing adapters, transformations and routing rules to distribute data throughout and beyond the enterprise.

WebMethods

Se PPT(se A.3)

6. Hur är bussen uppbyggd rent tekniskt? Vilka komponenter utgörs bussen av?

Microsoft

Som ni kan se på bilden(se bilaga A.4) utgörs bussen av WCF (Windows Communication Foundation) tillsammans med Biztalk och/eller SQL Service

Broker. Bussen är uppbyggt på .NET ramverket ock kan tex köras på en Windows server(se A.4).

IBM

Se bilaga – Då vår syn på ESB inte begränsas av våra egna komponenter är frågan inte helt enkel att svara på. I tillägg skulle jag även vilja framhålla att vi har flera komponenter delvis med överlappande funktionalitet varför jag vill påstå att en ESB från IBM kan vara allt ifrån en Message Broker eller en WebSphere ESB till en fullfjädrad lösning med samtliga nämnda produkter inkluderade – frågan är vilka krav man har att upfylla.

Beträffande teknik kan man framhålla följande grova indelning:

i. J2EE – WebSphere ESB med stöd för XML-based messaging, JMS, SOAP and JCA
ii. C++ - WebSphere Message Broker med stöd för any messaging (Inklusive XML, Flatfile, binary, COBOL Copytext, comma separated), JMS, SOAP.

HW – Data Power (XML based message transformation)

Oracle

Generellt kan sägas att Oracle ESB är en J2EE-applikation, som internt kommunicerar via JMS-standarden. För mer detaljerad beskrivning se sidorna 4-6 i dokumentet *ORACLE-SOA_SUITE-ESB.DOC*.

WebMethods

The webMethods Fabric suite is built in Java, and the components for the bus is:

- webMethods Broker
- webMethods Infravio X-Registry
- webMethods Infravio X-Broker
- webMethods ESP

7. Är er lösning hårt bunden till specifika produkter så som en speciell typ av server från ett specifikt företag?

Microsofts lösning är den enda som är hårt bunden till specifika produkter.

8. Hur möjliggörs en fysisk decentraliserad och logiskt centraliserad infrastruktur i er ESB lösning?

   Microsoft

   Se bilaga A.4

   IBM

   Vad avser man med distribuerad ?
   IBM's ESB är central i bemärkelsen att man inte har exvis "destination look-up", "transformation" i service –end points , utan gör detta centralt "i bussen". Det finns dock inget krav på att ESB:n skall ligga i en maskin, utan den kan "utbredas" både geografiskt och logiskt.

   Oracle

   Denna uppdelning uppnås m h a att Oracle ESBs möjlighet att driftsättas (deployas) på flera applikationsservrar, som jobbar mot en gemensam metadatabas. Deployment kan göras symmetriskt (alla tjänster lika fördelade på alla applikationsservrar) eller asymmetriskt (olika tjänster på olika appl.servrar). För mer info – se dokumentet *aesb-advanced-architecture-presentation.pdf*

   WebMethods

   WebMethods has a long history in EAI and B2B integration, as a vendor providing integration it is cruicial to incorporate these two requirements into the platform, avoiding the centralized server which becomes the bottleneck or single point of failure. All components of the webMethods Fabric suite supports load-balanced or clustered setups. All administration and setup of these distributed components can be handled by the MyWebMethods portal, where portlets for configuring and setup are provided.

9. Hur kan fysiskt decentraliserade tjänster så som: mappning, orkestrering osv. verka tillsammans som en logisk enhet?

   Microsoft

Dessa tjänster är inte fysiskt decentraliserade utan finns samlade i en Biztalk server. Se bilaga A.4.

IBM

Orkestrering lägger IBM inte i ESB:n -lagret utan i Process lagret. Om ni med orkestrering avser meddelandeflödes ”orkestrering”, så kan detta utvecklingsarbete och UNIT test utföras (skapa/modelera flödena) ”de-centraliserat”. Meddelande flödena driftsätts dock alltid centralt på ESB server (ESB servers kan självfallet distribueras ut rent fysiskt, men ses logiskt som centrala entiteter). Samma sak gäller mappning. Utvecklingsarbete och UNIT test kan ske lokalt, men driftsättning görs centralt. Detta är en aspekt som kan vara relevant vid uppfyllande av SoX’s ”separation of concerns” – applikationsutvecklare skall inte veta / behöva bry sig om vem som konsumerar producerar meddelanden och ej vilka format som funnits på vägen.

Oracle

Samverkan med orkestreringsverktyget Oracle BPEL Process Manager sker m h a JMS, liksom all kommunikation mellan de olika komponenterna i Oracle SOASuite, som är SOA-delen av Oracle Application Server.
När det gäller det som vi kan kalla för “bastjänster”, dit vi kan räkna mappning, så separeras inte dessa fysiskt från övriga delar av ESBn. Istället är Oracle ESBn byggd för att kunna skalas upp, t ex via klustring över flera applikationsservrar (symmetriskt eller asymmetriskt).
Nackdelar om man skulle anropa t ex fysiskt decentraliserade mappningstjänster är bl a

- den overhead som anrop till externa tjänster ofrånkomligen kostar, och som går stick i stäv med tanken om ESBn som meddelandeförmedlare med maximal genomströmningshastighet.
- kravet på att de decentraliserade tjänster som anropas har samma tillgänglighetsgrad som ESBn själv. Eftersom man inte kan garantera att så

är fallet krävs extra, och onödiga, felhantering om t ex mappningstjänsten inte är tillgänglig när den anropas från ESBn.

WebMethods

With the webMethods ESP, any service is available for anyone (with the righ priviligies) regardless of where they are residing and how they are called (WS or Messaging), services becomes reusable between the environments, they can publish the service publically through the registry using a mediation or not.

10. Vilka fördelar har er ESB lösning jämfört med andra existerande ESB lösningar? Med andra ord varför ska man välja er lösning?

Microsoft

Eftersom vår lösning kan anpassas helt efter kundens behov säljer vi inte på kunden mer än vad den behöver. Köper man en färdig produkt kanske man får betala för funktionalitet som man inte har nytta av. Våra produkter är utvecklade att fungera tillsammans så att man ska kunna använda dem tillsammans medan det finns konkurrenter som köper färdiga produkter som anpassas för att passa deras andra produkter. Vi anser inte detta vara en optimal lösning då man inte utnyttjar resurserna på ett effektivt sätt.

IBM

IBM rankas som en av 5 ledare inom ESB av Forrester's rapport "The Forrester Wave™: Enterprise Service Bus, Q2 2006", en rapport som inte analyserat IBM's komletta erbjudande utan endast WebSphere ESB, som har en delmängd av funktionaliteten i IBM's samlade erbjudande.

IBM's ESB bygger på teknologi som utvecklats under lång tid. Många av produkterna är väl beprövade och har utomordentligt (världsklass) bra prestanda. Nyare komponenter är måhända tidigare på mognadsskalan och kan förväntas nå högre prestanda och stabilitet, men håller redan hög kvalitet och fyller det gap som tidigare funnits inom ex.vis J2EE erbjudandet.

IBM's lösning kan skala både horisontellt och vertikalt såväl ur prestanda synvinkel som ur administrativ synvinkel. IBM stöder öppna protokoll och öppna standarder . IBM är drivande i flera standardiseringsorgan och når därför tidigt hög mognad i produktstöd för standarder under utveckling.

IBM's ESB i kombination med" ITCAM (IBM Tivoli Composite Application Monitoring) for SOA" medger utomordentligt bra stöd för Systems managemet av miljön. Såväl governance av tjänster på bussen (OBS inte nödvändigtvis begränsat till WebServices) som SLA uppfyllnad och val av QoS kan ske i runtime genom inbyggt stöd för anrop till WebSphere Service Registry and Repository.

Oracle

Nedanstående fördelslista från dokumentet *ORACLE-SOA_SUITE-ESB.DOC*.
Vill också tillägga att Oracle SOASuite (där Oracle ESB ingår) vunnit ett antal upphandlingar på den svenska marknaden det senaste året, både inom privat och offentlig sektor. I dessa upphandlingar har flertalet av de stora konkurrenternas produkter deltagit. Slutsatsen man kan dra från dessa upphandlingar är att funktionellt är det mycket tuff konkurrens mellan olika leverantörers produkter. De flesta erbjuder mer eller mindre samma funktioner. Därför är det andra saker som avgör, där Oracles välintegrerad produktsvit Oracle SOASuite visat sig vara ett vinnande koncept.
Oracles lösningar för skalbarhet och tillgänglighet är också faktorer som attraherat kunder.

WebMethods

Many… the advantages of a complete integrated stack of products is big. Governance and Management of SOA is one big advantage, the adapters, the lists becomes long…

11. Har ni möjlighet att ge oss kundreferenser där er ESB lösning har implementerats och en kontaktperson som vi kan kontakta för eventuella frågor?

    Företagen lämnar inte ut kundreferenser.

12. Vilka fördelar medför en ESB lösning?

    Nedan följer en lista med fördelar som företagen tagit upp i sina svar:

    - Reliable delivery
    - Location transparency
    - Pub/sub
    - Event driven management
    - Robust technology
    - Scalable technology
    - Multiple protocol support
    - Multiple message format support including XML,COBOL Copytext, CSF etc.
    - Flexibility
    - Decrease costs
    - Business intelligence
    - Service Virtualization
    - Dynamic Discovery and Invocation of Services
    - System Heterogeneity
    - Abstraction of Business Logic from Technical Implementation Details

13. Ser ni några nackdelar med en ESB lösning?

    Endast en av de utfrågade ser en nackdel med ESB, de anser att de öppna standarderna kan vara en nackdel pga. att de ständigt utvecklas och det kan leda till förvirring.

14. När ska man använda en ESB lösning?

    Man kan alltid använda sig av en ESB lösning, frågan är bara om införandet tillför verksamheten något.

15. Hur ser ni på framtiden för ESB?
    Alla är överens om att ESB kommer att finnas kvar i integrationslösningar en lång tid framöver dvs. framtiden är ljus.

16. Här skulle vi vilja att ni ”mappade” in er integrationslösning och dess komponenter i vår referensmodell för modern tjänsteorienterad integration.

    De flesta av leverantörerna har valt att hänvisa oss till deras egna referensmodeller(se bifogade svar) utom Oracle som mappade in sina komponenter i vår referensmodell. Sammanställningen blir svår att göra då det skiljer en del på de olika referensmodellerna. Modellerna påminner om varandra men vi har inte den fördjupade kunskapen inom ämnet eller om komponenterna för att kunna göra en rättvis mappning mellan företagens komponenter och vår referensmodell. Därför hänvisar vi er till bilagorna där ni finner företagens svar.

6.2 ESB-kunder

För att få reda på hur kunder som köpt och driftsatt en ESB ser på begreppet lät vi några företag svara på vår enkät. Vi ville även få svar på om ESB uppfyller de högt uppsatta förväntningarna om att en ESB löser alla integrationsproblem.

1. Hur skulle ni definiera begreppet ESB?

   - Försäkringskassan

     En standardiserad integrationsplattform som kombinerar meddelandehantering, webservices, datatransformering och intelligent routing för att koppla ihop och koordinera interaktioner mellan ett stort antal applikationer i en organisation med dess affärspartners.

   - Nordea

     Fleksibel integrasjons hub

   - SKF

Den del av vår centrala integrationsmiljö där vi håller gemensamma meddelanden/tjänster tillgängliga för producent- och konsumentapplikationer.

- Volvo

A service bus aid Service consumers in accessing services

- *Even when the service consumer and provider do not share the communication method*

Services can be exposed and further improved at the service bus

- *For example, multiple low lever application services can be composed into a high level business service*

Som ni ser är definitionerna mer likt varandra när vi frågar kunderna. Den definition som skiljer sig från de övriga är Volvo's som är mer fokuserad på tjänster. Till stor del handlar definitionerna om meddelandehantering, vilken är en av de huvudsakliga uppgifterna för en ESB-produkt.

2. Är det en tjänsteorienterad integrationslösning ni har infört/implementerat?

   Samtliga tillfrågade har valt att implementera en tjänsteorienterad integrationslösning för att ha möjligheten att använda sig av tjänster som är löst kopplade, vilket ökar återanvändningsmöjligheterna och gör dem lätta att kombinera för att skapa mer komplexa tjänster.

3. Vilken funktionalitet efterfrågade ni när ni valde er ESB-lösning?

   Det som efterfrågades var en integrationslösning som skulle klara av att på ett bra sätt integrera applikationer och tjänster som låg på olika plattformar. Dels ville man ha en lösning som klarar av webservices, transformering, routing, konvertering samt att lösningen skulle klara av att koppla sig samman med affärssystem på ett enkelt och smidigt sätt antingen via webservices eller via adaptrar. En annan viktig del i valet av lösning var lösningens flexibilitet, snabbhet i utveckling samt dess prestanda.

4. Varför ville ni köpa en integrationslösning, vad var problemet?

   Företagen har haft lite olika anledningar till att skaffa en integrationslösning. Några av företagen har ett problem som har uppstått medans andra rustar för framtiden och ligger steget före. Ett företaget har en egenutvecklad integrationslösning idag men vill övergå till en mer standardiserad lösning i samband med att deras egenutvecklade lösning skulle vidareutvecklas. Så istället för att vidareutveckla sin egen komplexa lösning har man valt att köpa en

ESB-lösning som är enkel att underhålla samt att det blir lättare att snabbt lära upp ny personal då lösningen enbart bygger på öppna standarder. Ett annat företag har valt en ESB-lösning på grund av ökade flexibilitetskrav i ett stort transformeringsprojekt.

För att snabb kunna integrera uppköpta bolag och nya affärspartners har ett företag valt att införa en ESB-lösning för att möjliggöra SOA/webservice kommunikation och processrelaterad affärsintegration. Anledningen till att de valt en SOA arkitektur var att de ville koppla loss sina applikationer från peer-to-peer lösningar.

5. Blev alla problem lösta efter införandet av den nya integrationslösningen?

   De flesta företagen har inte fått samtliga problem lösta direkt i och med införandet av en ESB-lösning. Ett företag har uppgett att de har fått alla sina problem lösta, de har dock inte satt hela sitt system i drift via ESB-lösningen.

6. Vilken verksamhetsnytta har ni fått av att införa en ESB-lösning?

   Den verksamhetsnytta som införandet har medfört för företagen är bla återanvändning av tjänster, intern och extern hantering av XML och komplett transformation av affärssystem.

7. Varför valde ni att implementera en ESB-lösning?

   En anledning till att implementera en ESB-lösning är att man vill kunna integrera på ett smartare och enklare sätt samt att man vill undvika egenutvecklade”spagetti”-lösningar. En annan anledning är att man vill införa SOA.

8. Har lösningen uppfyllt era förväntningar?

   Det verkar som de flesta upplever att deras förväntningar till stor del har blivit uppfyllda även om de haft lite problem.

9. Har ni haft några problem i samband med implementation/driften?

   Alla företagen har haft någon form av problem i samband med implementation/drift. Problemen som de har haft är att det tagit för lång tid för bussen att hantera meddelanden, kompetenskraven har varit väldigt höga samt

att det varit problem med java runtime pga att det var svårt att veta hur olika parametrar skulle dimensioneras.

10. Har införandet av en integrationslösning medfört verksamhetsnytta som ni inte hade räknat med?

    Ingen av företagen har upplevt att införandet medfört någon ytterligare verksamhetsnytta.

11. Är implementationen lokal eller global?

    Implementationerna har varit både lokala och globala.

12. Är er ESB-lösning avsedd för B2B och internt bruk eller endast för internt bruk?

    På denna fråga finns båda varianterna representerade.

13. Från vilket företag kommer er lösning?

    De företag som finns representerade här är:
    SeeBeyond(Sun)
    BEA
    IBM

14. Varför valde ni det företagets lösning?

    De faktorer som bidragit till valet av ESB-lösning har till stor del varit pris, prestanda samt funktionalitet.

15. Rekommenderar ni andra att implementera en ESB-lösning?

    Samtliga rekommenderar andra företag att implementera en ESB-lösning.

16. Med facit i hand skulle ni ha löst ert problem på ett annat sätt?

    Inget av företagen skulle med facit i hand valt en annan tekniskt lösning för att lösa sina problem. Ett av företagen svarade att de skulle ha löst det organisatoriskt på ett annat sätt.

17. Hade ert företag någon integrationslösning innan införandet av er ESB-lösning

Samtliga företag hade även innan införandet av ESB-lösningen någon form av integrationslösning. Några företag hade utvecklat sina egna integrationslösningar medans andra har valt att köpa färdiga lösningar. Ett företag hade tidigare flera lokala integrationslösningar men efter införandet av en ESB-lösning har de nu en global.

6.3 ESB-konsulter

Intresset för ämnet har varit stort från många konsultfirmor men det är bara ett företag som tagit sig tid med oss, Zystems. Nedan listas därför deras svar på frågorna.

1. Hur definierar ni begreppet ESB?

   An Enterprise Service Bus is a flexible infrastructure that handles the **mediation of messages** between the applications being integrated in a **consistent manner**.

2. Vilka funktionaliteter anser ni måste ingå i en modern tjänsteorienterad integrationslösning för att få kallas ESB?

   - Routing
   - Conversion (protocol)
   - Transformation (data)
   - Connectivity (transport)
   - Logging/monitoring/management
   - Service Registry

3. Vilka funktionaliteter i en ESB efterfrågas mest av era kunder?

Allt ovan

4. Vilka företag på marknaden anser ni levererar en ESB-lösning idag?

- Zystems By Semcon
- Flera företag levererar produkter med vars hjälp man kan bygga en ESB men produkt räcker inte, man behöver också koncept och metod för att realisera en ESB
- Flera av våra konkurrenter har individer med stor integrationskompetens men inget gemensamt integrationskoncept

5. Finns det företag på marknaden som säger sig leverera en ESB lösning som ni inte anser vara en ESB? Vilka?

Se ovan

6. Vilka/Vilken ESB lösning(ar) har ert företag valt att driftsätta hos kunder?

Baseline implementerat på IBM WebSphere integrationsprodukter (främst WebSphere MQ och Message Broker, men även ICS, WPS, WPG, WESB)

7. Varför väljer ni just dessa/denna ESB lösning?

Baseline är unikt i sitt slag. IBM WebSphere är den produkt (enligt vår åsikt) som har störts förmåga att hantera olika plattformar. Den är dessutom mycket skalbar och robust.

8. Vilken ESB lösning har ni på ert företag bäst erfarenhet av?

Se ovan

9. Hur många ESB lösningar har ni driftsatt?

Ett 40-tal

10. Är det stora skillnader i pris mellan olika ESB lösningar?

Ja

11. Hur ligger de olika ESB lösningarna i pris i förhållande till varandra? Ange gärna prisintervall.

Vi har ingen uppfattning om andra leverantörer än IBM. Inom IBM är licenskostnaden för de olika produkterna avhängig antal processorer mm, medan Baseline betingar ett fast pris. Vårt tjänsteerbjudande Baseline Start-kit (fn 400.000 kr) ger kunden alla verktyg och kunskaper som behövs för att komma igång med en integrationplattform (ESB) på 2-3 veckor.

12. Vilka fördelar medför en ESB lösning?

- Enhetlighet
- Lös koppling
- Kontroll
- Flexibilitet
- Spårbaret

13. Ser ni några nackdelar med en ESB lösning?

Man måste ha en viss volym avseende informationsutbytet eftersom en ESB har en relativt hög initialkostnad. Den ställer också nya krav på kompetens, organisation och ägarskapsfrågor.

14. När ska man använda en ESB lösning?

Vid stora krav på informationsutbyte mellan relativt många system i en heterogen miljö

15. Hur ser ni på framtiden för ESB?

Ljus. Konceptet har funnits länge under andra namn (t.ex. Hub and Spoke) och kommer säkert att få nya namn och features i framtiden. Enligt vår åsikt är en robust ESB en förutsättning för att SOA skall fungera i praktiken, i alla fall så länge inte alla system i hela världen är WebService enablade.

7 Slutsats

Begreppet ESB definieras på olika vis beroende på vem man frågar och det visar vår undersökning klart och tydligt. Vi har valt att inte bidra med ännu en definition av ESB pga. att detta bara skulle röra till det ännu mer. Vi har istället valt att försöka att titta på de delar i de definitionerna från de vi frågat och andra definitioner vi har granskat för att bryta ner dem till delar som de nästan alla är överens om. Utifrån detta har vi kommit fram till att några menar att en ESB är en integrationslösning som är baserad på öppna standarder och att några menar att en ESB är en middleware-komponent för att möjliggöra SOA. Vad det gäller begreppet ESB finns vad det verkar inga rätt eller fel eftersom ingen vedertagen definition som kan anses vara ömsesidigt accepterad inom branschen. När det gäller vår syn på ESB menar vi att en integrationsprodukt med funktionalitet som minst innefattar transformering, routing, konnektivitet, och konvertering och som bygger på öppna standarder, möjliggör en implementation av en tjänsteorienterad integrationslösning (SOA) kan kallas för en ESB.

Vi anser att tjänsteorienterad integration kommer vara det sätt som framtida integrationslösningar kommer att byggas på vilket även de företag vi har varit i kontakt med också tror. Vi tror dock att begreppet ESB kommer att försvinna inom några år pga. att det råder så stor oklarhet om vad det egentligen är. Integrationslösningar kommer dock att byggas på det sätt som de görs idag med en central knytpunkt (buss eller hub) som hanterar kommunikationen (transformering, routing, konnektivitet, konvertering) mellan tjänster och som kommer att baseras på öppna standarder och möjligöra en tjänsteorienterad arkitektur.

Referenser

[1] *Woodgate Scott, BizTalk Server 2004 UNLEASHED, Sams Publishing 2005, ISBN 0-672-32598-5.*

[2] *Hunter David, Beginning XML (3rd edition), Johan Wiley & Sons Incorporated 2005, ISBN 0-7645-7077-3.*

[3] *Stateful Web services using WSE (Inspec)*

[4] *http://dotnet.sys-con.com/read/121831.htm (. Net Journal, artikel).*

[5] IBM Redbooks. Patterns: SOA with an Enterprise Service Bus in WebSphere Application Server V6. Durham, NC, USA: IBM, 2005. p20. http://site.ebrary.com/lib/kaubib/Doc?id=10112536 & ppg=40.

[6] SprottDavid,UnderstandingService-OrientedArchitecture. http://www.msarchitecturejournal.com/pdf/Understanding_Service-Oriented_Architecture.pdf

[7] Stojanovic, Zoran (Editor). Service-Oriented Software System Engineering: Challenges and Practices.Hershey, PA, USA: Idea Group Publishing, 2005. http://site.ebrary.com/lib/kaubib/Doc?id=10075731&ppg=23

[8] Hashimi, S. (2003), Service-Oriented Architecture Explained. http://www.ondotnet.com/pub/a/dotnet/2003/08/18/soa_explained.html

[9] http://eprints.ecs.soton.ac.uk/11209/01/ecstr_iam05_004.pdf

[10] http://www.oasis-open.org/committees/soa-rm/faq.php

[11] http://www-306.ibm.com/software/info1/websphere/index.jsp?tab=landings/esbbenefits

[12] BEA_ForresterReport_wp.pdf

[13] http://www.oracle.com/technology/tech/soa/mastering-soa-series/part2.html

[14] http://www-306.ibm.com/software/info1/websphere/index.jsp?tab=integration/esb

[15] best-of-breed-esbs.pdf

[16] http://www.looselycoupled.com/opinion/2005/warner-esb-infr0114.html

[17] http://www.capeclear.com/download/whitepapers/ESB_Whitepaper.pdf

[18] http://www.mindtree.com/white_paper/EAI-integration-approaches.pdf

[19] http://www.polarlake.com/files/esb.pdf

[20] http://www.bijonline.com/index.cfm?section=article&aid=268

[21] http://www.neudesic.com/media/Neudesic%20-%20Neuron%20ESB%20White%20Paper%20(P1).pdf

[22] http://www.fiorano.com/whitepapers/fiorano_esb.pdf

[23] http://www.bea.com/content/news_events/white_papers/BEA_AL_Service_Bus_wp.pdf

[24] http://www.ebizq.net/views/download_raw?metadata_id=7382&what=white_paper

[25] http://www.polarlake.com/files/PolarLake_19Feb07_US.pdf

[26] http://www.vitria.com/Business_Accelerator/esb.php

[27] http://www.mindtree.com/white_paper/EAI-integration-approaches.pdf

[28] http://www.nascio.org/awards/nominations/2006Kentucky7.pdf

[29] http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci1226497,00.html
[30] esb_for_soa.pdf
[31] http://www-128.ibm.com/developerworks/webservices/library/ws-soa-esbarch/?ca=drs-tp3307
[32] http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1228852,00.html
[33] http://en.wikipedia.org/wiki/Request-response
[34] http://en.wikipedia.org/wiki/Publish/subscribe
[35] ESB Oriented Integration Landscape - A Business Analysis.pdf
[36] http://www-306.ibm.com/software/info1/websphere/index.jsp?tab=landings/esbbenefits
[37] IBM Redbooks: Patterns: Implementing an SOA Using an Enterprise Service Bus. ibm.com/redbooks
[38] Per Björkegren Sogeti Karlstad www.sogeti.se
[39] http://technology.amis.nl/blog/wp-content/images/AMISQuery14febIntroESB.ppt
[40] http://www.microsoft.com/biztalk/solutions/soa/esb.mspx

Bilagor

A Frågeankät Tillverkare

Tjänsteorienterad integration, ESB

Martin Bood och Karl-Johan Fisk

Examensjobb

Karlstads Universitet

1. Hur definierar ni begreppet ESB?
2. Vilka funktionaliteter anser ni måste ingå i en modern tjänsteorienterad integrationslösning för att få kallas ESB?

3. Har ert företag en ESB lösning?

   Om svaret på fråga 3 är nej, svara istället på fråga 4 och hoppa över fråga 5-8.

4. Har ni planerat att släppa en ESB lösning?

5. Vilka funktionaliteter har er ESB lösning?
6. Hur är bussen uppbyggd rent tekniskt? Vilka komponenter utgörs bussen av?
7. Är er lösning hårt bunden till specifika produkter så som en speciell typ av server från ett specifikt företag?
8. Hur möjliggörs en fysisk decentraliserad och logiskt centraliserad infrastruktur i er ESB lösning?
9. Hur kan fysiskt decentraliserade tjänster så som: mappning, orkestrering osv. verka tillsammans som en logisk enhet?
10. Vilka fördelar har er ESB lösning jämfört med andra existerande ESB lösningar? Med andra ord varför ska man välja er lösning?
11. Har ni möjlighet att ge oss kundreferenser där er ESB lösning har implementerats och en kontaktperson som vi kan kontakta för eventuella frågor?

12. Vilka fördelar medför en ESB lösning?
13. Ser ni några nackdelar med en ESB lösning?
14. När ska man använda en ESB lösning?
15. Hur ser ni på framtiden för ESB?

16. Här skulle vi vilja att ni ”mappade” in er integrationslösning och dess komponenter i vår referensmodell för modern tjänsteorienterad integration. Referensmodellen bifogas separat.

A.1 Oracle

Oracle

1. Hur definierar ni begreppet ESB?

An ESB provides a much-needed intermediary layer that facilitates data delivery, service access, service reuse, and service management of an enterprise SOA implementation. ESB also supports intelligently-directed communication and mediates relationships among loosely coupled and decoupled business components.

2. Vilka är de funktionaliteter anser ni måste ingå i en modern tjänsteorienterad integrationslösning för att få kallas ESB?

1. Virtualize Endpoints: *From resources to services*

2. Transform: *Convert data to target formats*

3. Route: *Reliably transport and route data over a variety of protocols*

De tre ovanstående är grundfunktionerna som måste finnas. Sen bör en ESB även

- vara byggd för skalbarhet och hög tillgänglighet
- ha ett övervakningsgränssnitt som gör att trafiken i ESBn kan övervakas på ett effektivt sätt.
- ha en utvecklingsmiljö som är integrerad med övriga delar av en SOA-arkitektur, för att underlätta återanvändbarhet i alla led.

3. Har ert företag en ESB lösning?

Ja. Den heter Oracle Enterprise Service Bus, och är en av komponenterna i Oracles middleware, som heter Oracle Application Server men som också kalls Fusion Middleware. En förpackning av Oracle Application Server kallas för Oracle SOASuite, och innehåller de komponenter som behövs för att bygga SOA-baserade lösningar, däribland ESBn.

5. Vilka funktionaliteter har er ESB lösning?

Oracle ESB is a standards based infrastructure component of the Oracle SOA Suite delivering loosely coupled data and enterprise application integration that is (i) fast, (ii)

secure, (iii) reliable and (iiii) highly available. Oracle ESB features a multi-protocol message bus with centralized monitoring management of distributed services where all services are exposed as standard web services using WSDL. Oracle ESB contains message flows utilizing adapters, transformations and routing rules to distribute data throughout and beyond the enterprise.

6. Hur är bussen uppbyggd rent teknisk? Vilka komponenter utgör bussen?

Generellt kan sägas att Oracle ESB är en J2EE-applikation, som internt kommunicerar via JMS-standarden. För mer detaljerad beskrivning se sidorna 4-6 i dokumentet *ORACLE-SOA_SUITE-ESB.DOC*, samt de pdf-presentationer jag sänt Er via epost.

7. Är er lösning hårt bunden till specifika produkter så som en speciell typ av server från ett specifikt företag?

Nej. Oracle ESB ingår i Oracle Application Server, där ett stort antal komponenter (inkl ESBn) kan köras i applikationsservrar från andra leverantörer, och på många olika typer av hårdvara och operativsystem, samt mot olika leverantörers databaser.

Oracle kallar detta för att produkten är "Hot Pluggable".

8. Hur möjliggörs en fysisk decentraliserad och logiskt centraliserad infrastruktur i er ESB lösning?

Denna uppdelning uppnås m h a att Oracle ESBs möjlighet att driftsättas (deployas) på flera applikationsservrar, som jobbar mot en gemensam metadatabas. Deployment kan göras symmetriskt (alla tjänster lika fördelade på alla applikationsservrar) eller asymmetriskt (olika tjänster på olika appl.servrar). För mer info – se dokumentet *aesb-advanced-architecture-presentation.pdf*

9. Hur kan fysiskt decentraliserade tjänster så som: mappning, orkestrering osv. verka tillsammans som en logisk enhet?

Samverkan med orkestreringsverktyget Oracle BPEL Process Manager sker m h a JMS, liksom all kommunikation mellan de olika komponenterna i Oracle SOASuite, som är SOA-delen av Oracle Application Server.

När det gäller det som vi kan kalla för "bastjänster", dit vi kan räkna mappning, så separeras inte dessa fysiskt från övriga delar av ESBn. Istället är Oracle ESBn byggd för att kunna skalas upp, t ex via klustring över flera applikationsservrar (symmetriskt eller asymmetriskt).

Nackdelar om man skulle anropa t ex fysiskt decentraliserade mappningstjänster är bl a

- den overhead som anrop till externa tjänster ofrånkomligen kostar, och som går stick i stäv med tanken om ESBn som meddelandeförmedlare med maximal genomströmningshastighet.
- kravet på att de decentraliserade tjänster som anropas har samma tillgänglighetsgrad som ESBn själv. Eftersom man inte kan garantera att så är fallet krävs extra, och onödiga, felhantering om t ex mappningstjänsten inte är tillgänglig när den anropas från ESBn.

10. Vilka fördelar har er ESB lösning jämfört med andra existerande ESB lösningar? Med andra ord varför ska man välja er lösning?

Nedanstående fördelslista från dokumentet *ORACLE-SOA_SUITE-ESB.DOC*.

Vill också tillägga att Oracle SOASuite (där Oracle ESB ingår) vunnit ett antal upphandlingar på den svenska marknaden det senaste året, både inom privat och offentlig sektor. I dessa upphandlingar har flertalet av de stora konkurrenternas produkter deltagit. Slutsatsen man kan dra från dessa upphandlingar är att funktionellt är det mycket tuff konkurrens mellan olika leverantörers produkter. De flesta erbjuder mer eller mindre samma funktioner. Därför är det andra saker som avgör, där Oracles välintegrerad produktsvit Oracle SOASuite visat sig vara ett vinnande koncept.

Oracles lösningar för skalbarhet och tillgänglighet är också faktorer som attraherat kunder.

WHY ORACLE?

Oracle SOA Suite is the only comprehensive and integrated SOA suite in the industry. While other vendors claim to have similar platforms, Oracle SOA Suite provides several unique differentiators over other products.

Realize Greater Organizational Flexibility

Oracle SOA Suite can help you achieve greater organizational flexibility better than any other solution in the market. Oracle ESB facilitates this with the following:

- *Standards Based Integration* – Oracle ESB utilizes standards that reduce barriers to integrating systems enabling better decision-making, and new products to be rolled out quicker and easier than before.
- *Business Resiliency* – Oracle ESB simplifies change by enabling configuration-based changes to service artifacts such as schemas, transformations, routing rules and content based routing filter expressions without requiring costly redeployment.
- *Business Visibility* – Oracle ESB increases business visibility into enterprise systems by exposing registering all adapter and routing services as standard Web Services via WSDL into UDDI service registries.

Eliminate Middleware Complexity

Oracle SOA Suite can reduce your costs and middleware complexity better than any solution available from any other vendor. It is the industry's only SOA Suite technically engineered to be a single product. Oracle SOA Suite differs from other market solutions in four key areas:

- *Single Development Framework* – Oracle SOA Suite is the only SOA suite that provides a single integrated design time environment to develop enterprise applications, to compose Web services, to create enterprise portals, and to orchestrate business processes. You learn one tool to target the entire platform.
- *Single Deployment Architecture* – Oracle SOA Suite is the only SOA suite that provides a common architecture for scalability, availability, workload distribution, resource management, security, and metadata management. You spend less time integrating your middleware infrastructure.
- *Single Management Architecture* – Oracle SOA Suite is the only SOA suite that has a common identity management and systems management architecture. You monitor and manage users and systems centrally, lowering cost and improving security.
- *Single Metadata Management System* – Oracle SOA Suite is the only SOA suite that leverages a common metadata management system across all components, speeding up application development and leading to more maintainable applications.
- *Easy to Adopt* – All of the SOA Suite components are built upon and support industry standards, to ensure that they can be incrementally adopted and easily integrated into an organization's existing information technology infrastructure.

11. Har ni möjlighet att ge oss kundreferenser där er ESB lösning har implementerats och en kontaktperson som vi kan kontakta för eventuella frågor?

Nej tyvärr. Oracles policy är att inte lämna ut uppgifter om kontaktpersoner hos kunder förutom som en del av införsäljningen av produkten hos nya kunder, och då endast efter överenskommelse med kontaktpersonen i fråga.

Officiella kundreferenser finns det däremot. Se t ex en kort beskrivning av Deutsche Post i dokumentet *ORACLE-SOA_SUITE-ESB.DOC*. Ett annat företag som är en stor användare av Oracles ESB är jobbförmedlingssajten Monster.com.

12. Vad anser ni vara de främsta fördelarna med en ESB lösning?

- **Reliable Messaging:** Reliable transport of data continues to be a basic need for any integration solution. While the principle of SOA calls for standards-based, platform-independent messaging protocols, this principle does not inherently allow for reliable delivery of data. Standards (such as WS-RM) are emerging to support this capability, but they are not yet mature or widely adopted.
- **Service Virtualization:** SOA implies a basic architectural paradigm in which any service consumer can access a service provider from any platform. This, in turn, implies that the appropriate protocol and syntactic mediation is in place to insulate consumers and providers.

- **Dynamic Discovery and Invocation of Services:** In order to optimize reuse of services, service consumers require an intermediary function to understand the characteristics of the service request, and facilitate the connection with the provider. In an ideal SOA, this relationship would be brokered at run-time.
- **Policy Management:** Access by known and unknown service consumers results in the need for an abstracted policy management model that is capable of enforcing authentication, authorization, and encryption in addition to more complex business-level policies independent of the service provider implementation.
- **System Heterogeneity:** Today's new applications are tomorrow's legacy, as one can observe in common applications as well as the software used to connect them. This proliferation of new technology is inevitable, and system landscapes must be architected to support such change.
- **Abstraction of Business Logic from Technical Implementation Details:** One goal of SOA is to provide a layered approach to developing systems that insulates changes in technology from changes in business process, and vice versa. In effect, this "separation of concerns" must be designed into the architecture from the start.

13. Ser ni några nackdelar med en ESB lösning?

Inga nackdelar med ESBn som mönster och/eller produkt. Däremot finns en nackdel, eller ska vi kalla det fälla i införandet och användandet av ESBn. Nämligen att både leverantörer av ESB-lösningar, och kunder som implementerar sådana, tenderar att försöka lösa problem med ESBn som den inte är det mest optimala verktyget att lösa.

Att t ex införa logik eller process-orkestrering enbart m h a en ESB tenderar att utmynna i väldigt mycket specialanpassningar, som ofta är properitära, egenbyggda lösningar, vilka över tiden blir mycket dyra och svåra att underhålla.

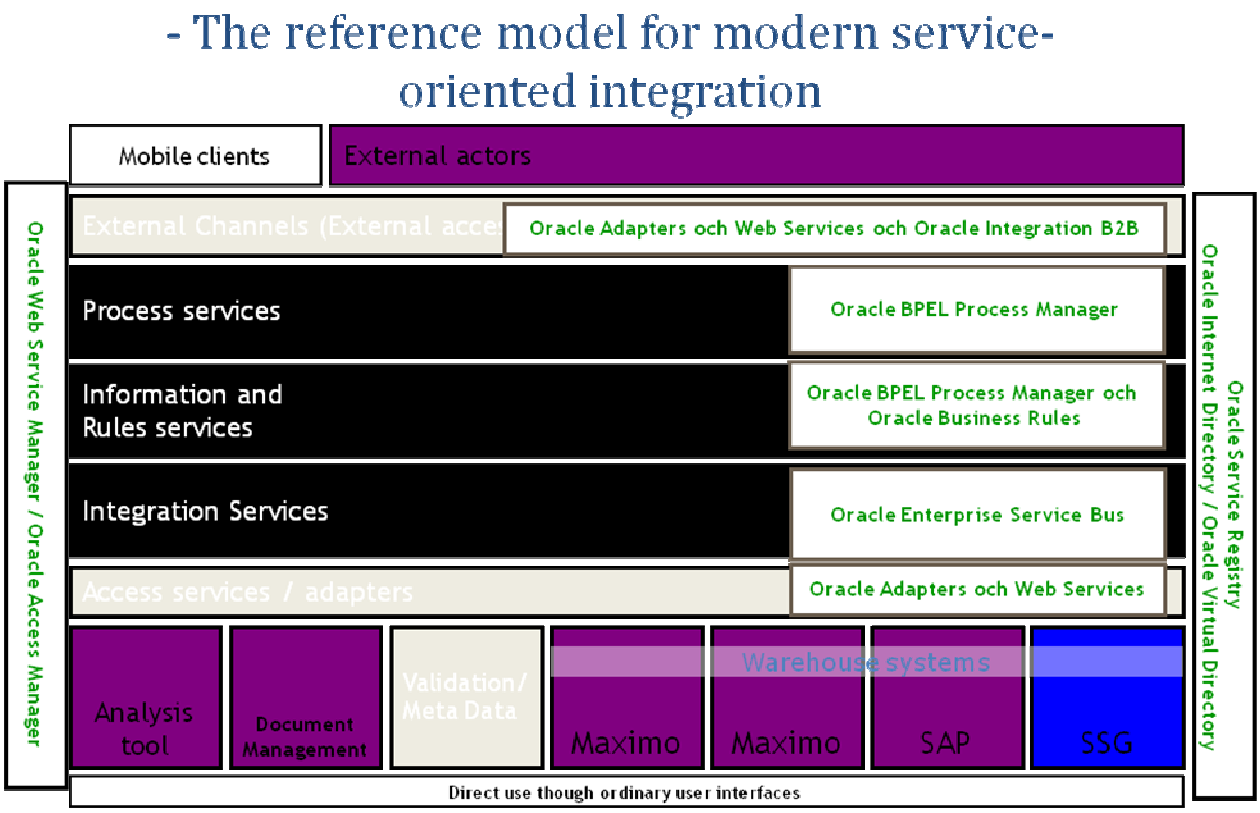
14. När ska man använda en ESB lösning?

För modern systemintegration, och inte minst för sk *Service Virtualization*. För en beskrivning av vad detta är – se dokumentet *Wiring Through an Enterprise Service Bus.doc*.

15. Hur ser ni på framtiden för ESB?

ESBn har sin självklara roll i framtidens SOA-baserade IT-arkitektur, och är en av flera "enablers", för att bygga fullskaliga SOA-lösningar.
Arkitekturen kommer att vidareutvecklas, i takt med att SOA förädlas. T ex kommer nästa version av Oracle Enterprise Service Bus att ha stöd för den nya specifikationen Service Component Architecture (SCA). För mer info ang SCA se dokumentet *SCA_White_Paper1_09.pdf*.

16.


- The reference model for modern service-oriented integration
Mobile clients
External actors
Oracle Web Service Manager / Oracle Access Manager
External Channels (External acce
Oracle Adapters och Web Services och Oracle Integration B2B
Process services
Oracle BPEL Process Manager
Information and Rules services
Oracle BPEL Process Manager och Oracle Business Rules
Integration Services
Oracle Enterprise Service Bus
Access services / adapters
Oracle Adapters och Web Services
Oracle Service Registry
Oracle Internet Directory / Oracle Virtual Directory
Analysis tool
Document Management
Validation / Meta Data
Warehouse systems
Maximo
Maximo
SAP
SSG
Direct use though ordinary user interfaces

A.2 IBM

IBM

1. Hur definierar ni begreppet ESB?

Här har jag saxat från www.ibm.com.

An Enterprise Service Bus (ESB) is a flexible connectivity infrastructure for integrating applications and services. An ESB can power your service-oriented architecture (SOA) by reducing the number, size, and complexity of interfaces between those applications and services.
An ESB performs the following functions:

- Route messages between services
- Convert transport protocols between requester and service
- Transform message formats between requester and service
- Handle business events from disparate sources

An ESB should allow your organization to focus on your core business needs rather than the IT infrastructure required for connecting the programs together. An ESB should allow you to add new services or make changes to existing services with little or no impact to the use of existing services.

2. Vilka funktionaliteter anser ni måste ingå i en modern tjänsteorienterad integrationslösning för att få kallas ESB?

Jag känner inte till att vi har någon e.g. krav på vad en ESB måste innehålla för att få kallas ESB, tvärtom så ser vi att det finns ”domän”-specifika service-bussar som uppfyller domänspecifika krav, men kanske missar generiska krav såsom öppenhet, flexibilitet mm, som måste kunna ingå i en ESB. I många praktiskt förekommande fall har företag och organisationer någon form av befintlig infrastruktur som inte enkelt kan ersättas momentant. En **ESB** bör därför klara av att integrera med domänspecifika service bussar, här är snarast en fråga om språkbruk om detta skall anses ingå i ESB:n eller vara domäner utanför ESB. Det är också önskvärt att en ESB medger löskoppling (säker leverans, event hantering), routing (ev. content based), publish subscribe, protokollkonvertering,meddelande transformation, såväl asynkrona- som synkrona-scenarier. Det är en fördel om ESB:n har ramverk för adaptrar mot exempelvis affärssystem eller teknlogier, gärna med stöd för transformering mellan generiska affärsrelevanta objekt och applikationsspecifika affärs objekt. Stöd för XML, XSLT, XPATH och WSDL är självfallet också klara värde bidragande egenskaper.

3. Har ert företag en ESB lösning?

Ja – Men i egentlig mening är IBM’s syn att en ESB är en arkitektonisk stil, som konceptuellt kan bestå av flera Service Bussar (bla sådana som löser existerande eller domänspecifika integrationsbehov). Initialt ville därför IBM inte se ESB som en produkt

(vilket fortfarande är en förhärskande sanning), utan just som en arkitektoniks stil. Dock fanns genom konkurrenters annonsering av ESB produkter ett tryck från kunder att IBM skulle leverera en ”ESB produkt”, varför IBM så småningom släppte en komponent som heter WebSphere ESB, detta är emellertid inte den enda komponent i IBM’s kompletta ESB offering.

IBM’s kompletta ESB lösning är (man måste inte ha alla ingående komponenter, utan kan välja delar av den utifrån de behov man står inför):

a. WebSPhere MQ (en kö-mekanism som medger säker leverans, synkron eller asynkron meddelande hantering och är på så vis en hörnsten i löst koppade arkitekturer).
b. WebSphere Application Server (IBM’s J2EE flaggskepet) har JMS server implementaion i form av SI-Bus. Denna Java server komponent är förutom JMS kompliant även MQ-kompliant, dsv en messsaging node i SI-Bus uppträder som en Q-Manager i MQ om man kopplar samman dem. På samma sätt uppträder en WebSphere MQ’s Q-manager som en Messaging Node i WebSphere Application Server’s SI-Bus. Med J2EE miljön kommer JCA (Java Connector Architecture) kring vilken många nya adaptrar byggs (IBM betraktar inte adaptrar som kärnkomponent i ESB:n, men jag nämner dem ändå).
c. WebSphere ESB – en java baserad broker med routing, transformation, pub-sub. Då den byggts på WebSphere Application Server använder den JMS och SI-Bus för meddelandebaserad leverans. Vidare stöds sjävfallet SOAP över http för WebService. Denna broker transformerar och routar företrädesvis XML meddelanden (i Java miljö), t.ex. WS, men tranformering till andra format går självfallet också. Flödes och meddelande modellering och transformering sker med WebSphere Integration Developer (eclipse baserat), med vilken stöd för service component architecture (SCA) och business objects (BO eller Service Data Object SDO) kommer, men det är inte riktigt på agendan för ESB. [Motivet för att överhuvudtaget nämna SDO och SCA är att detta är ramverket kring vilket Adaptrar modellereas i WebSphere ESB.] Denna ESB komponent kan enkelt samverka med övriga ESB komponenter via MQ, JMS och http.
d. Websphere Message Broker – Detta är en högprestanda broker med stöd för såväl JMS, MQ och SOAP över http som bärare. Här stöds alla tänkbara format och transformeringar. Flödes- , meddelande- modellering , och transformering sker med WebSphere Message Broker Toolkit (eclipse baserat). WebSphere Message Broker har ett kraftfullt programmeringsspråk som heter ESQL och stöder dessutom Java-kod. Till denna komponent kan knytas ett adapter ramverk som inte är JCA baserat utan bygger på en J2SE plattform, detta är ett historiskt arv och kommer sannolikt på sikt migreras till JCA adaptrar i någon form.

e. IBM har även hårdvara som gör XML transformering ”at the speed of the wire” och säkerhetshantering med certificate mm. Dessa produkter marknadsförs under namnet DataPower. Syfte med denna hårdvara är att lasta av WebSphere Message Broker och WebSphere ESB vid extremt höga prestanda krav.

Ytterligare information kring IBM och ESB kan ni inhämta på: http://www.ibm.com/software/integration/wsesb/v6/faqs.html#do

Angränsande till ESB har IBM som ni ser i vår referensakrkitektur funktionella områden såsom , Interactions Services (presentations lager, portal , any device komponenter), Process Services, Information Services, Partner Services (B2B/EDI), Business Applications Services, Access Services (här ligger bland annat ramverk för adaptrar mm).

I en angränsande domän som vi kallar för SOA Governance Lifecycle adresserar IBM :

- Discovery & re-use , publicering av tjänster
- QoS, Monitorering och hur väl tjänsten uppfyller SLA, om de används mm. Kriterier som är viktiga ur driftsperspektiv.

Analogin till UDDI är bara delvis korrekt då IBM’s approach adresserar väsentligt bredare (bla annat fler transport bindings än http)och framförallt även täcker in monitorering och driftsaspekten. Intuitivt kanske detta inte förefaller relevant för ESB:n, men samtidigt kan ESB både provider och consumer av tjänster och har därför relevans såväl i develpoment time som i runtime (för dynamiskt uppslag av tjänster).

Om svaret på fråga 3 är nej, svara istället på fråga 4 och hoppa över fråga 5-8.

4. Har ni planerat att släppa en ESB lösning?

5. Vilka funktionaliteter har er ESB lösning?

Se ovan och länken ovan. Väldigt förenklat dock:

Basfunktionalitet:

a. Pub/sub
b. Message Transformering
c. Protokoll konvertering
d. Event Management

Tillägg:

Adapter ramverk och adaptrar gärna med stöd för kanoniska format.

6. Hur är bussen uppbyggd rent tekniskt? Vilka komponenter utgörs bussen av?

   Se ovan – Då vår syn på ESB inte begränsas av våra egna komponenter är frågan inte helt enkel att svara på. I tillägg skulle jag även vilja framhålla att vi har flera komponenter delvis med överlappande funktionalitet varför jag vill påstå att en ESB från IBM kan vara allt ifrån en Message Broker eller en WebSphere ESB till en fullfjädrad lösning med samtliga nämnda produkter inkluderade – frågan är vilka krav man har att uppfylla.

   Beträffande teknik kan man framhålla följande grova indelning:

   i. J2EE – WebSphere ESB med stöd för XML-based messaging, JMS, SOAP and JCA
   ii. C++ - WebSphere Message Broker med stöd för any messaging (Inklusive XML, Flatfile, binary, COBOL Copytext, comma separated), JMS, SOAP.
   iii. HW – Data Power (XML based message transformation)

7. Är er lösning hårt bunden till specifika produkter så som en speciell typ av server från ett specifikt företag?

   Alla existerande servrar stöds ej, men bra nära. Linux (de mest namnkunniga distributionerna), Windows, System z, diverse UNIX.

   Jag är inte säker på om detta fullständigt svarar på er fråga?

8. Hur möjliggörs en fysisk decentraliserad och logiskt centraliserad infrastruktur i er ESB lösning?

   Vad avser man med distribuerad ?

   IBM's ESB är central i bemärkelsen att man inte har exvis "destination look-up", "transformation" i service –end points , utan gör detta centralt "i bussen". Det finns dock inget krav på att ESB:n skall ligga i en maskin, utan den kan "utbredas" både geografiskt och logiskt.

9. Hur kan fysiskt decentraliserade tjänster så som: mappning, orkestrering osv. verka tillsammans som en logisk enhet?

   Orkestrering lägger IBM inte i ESB:n -lagret utan i Process lagret. Om ni med orkestrering avser meddelandeflödes "orkestrering", så kan detta utvecklingsarbete och UNIT test utföras (skapa/modelera flödena) "de-centraliserat". Meddelande flödena driftsätts dock alltid centralt på ESB server (ESB servers kan självfallet distribueras ut rent fysiskt, men ses logiskt som

centrala entiteter). Samma sak gäller mappning. Utvecklingsarbete och UNIT test kan ske lokalt, men driftsättning görs centralt. Detta är en aspekt som kan vara relevant vid uppfyllande av SoX's ”separation of concerns” – applikationsutvecklare skall inte veta / behöva bry sig om vem som konsumerar producerar meddelanden och ej vilka format som funnits på vägen.

10. Vilka fördelar har er ESB lösning jämfört med andra existerande ESB lösningar? Med andra ord varför ska man välja er lösning?

    IBM rankas som en av 5 ledare inom ESB av Forrester's rapport ”The Forrester Wave™: Enterprise Service Bus, Q2 2006”, en rapport som inte analyserat IBM's komletta erbjudande utan endast WebSphere ESB, som har en delmängd av funktionaliteten i IBM's samlade erbjudande.

    IBM's ESB bygger på teknologi som utvecklats under lång tid. Många av produkterna är väl beprövade och har utomordentligt (världsklass) bra prestanda.

    Nyare komponenter är måhända tidigare på mognadsskalan och kan förväntas nå högre prestanda och stabilitet, men håller redan hög kvalitet och fyller det gap som tidigare funnits inom ex.vis J2EE erbjudandet.

    IBM's lösning kan skala både horisontellt och vertikalt såväl ur prestanda synvinkel som ur administrativ synvinkel. IBM stöder öppna protokoll och öppna standarder . IBM är drivande i flera standardiseringsorgan och når därför tidigt hög mognad i produktstöd för standarder under utveckling.

    IBM's ESB i kombination med” ITCAM (IBM Tivoli Composite Application Monitoring) for SOA” medger utomordentligt bra stöd för Systems managemet av miljön. Såväl governance av tjänster på bussen (OBS inte nödvändigtvis begränsat till WebServices) som SLA uppfyllnad och val av QoS kan ske i runtime genom inbyggt stöd för anrop till WebSphere Service Registry and Repository.

11. Har ni möjlighet att ge oss kundreferenser där er ESB lösning har implementerats och en kontaktperson som vi kan kontakta för eventuella frågor?

    Jag kan tyvärr inte besvara denna fråga, utan att först konsultera de kunder som berörs. Jag skulle nog vilja ha mer kött på benen avseende ert examensarbete innan jag gräver i detta.

12. Vilka fördelar medför en ESB lösning?

Svårt att lista alla – en hel del finner ni ovan:

- Reliable delivery (via MQ / SI-Bus)
- Location transparency
- Pub/sub
- Event driven management
- Strong programming language (ESQL and support for Java in both WMB and WESB)
- Robust technoogy
- Scalable technology
- World class performance
- World class navigation in complex data structures (WebSphere TX – from Mercator acquired by IBM)
- Multiple protocol support
- Multiple message format support including XML,COBOL Copytext, CSF etc.
- Excellent integration with WebSphere Service Registry and Repository
- Excellent Monitoring from Tivoli

13. Ser ni några nackdelar med en ESB lösning?

Egentligen inte, men ibland har det ett värde att gömma den (nödvändiga) tekniska komplexiteten bakom ett produktnamn. IBM har flera. Ibland uppfattas detta som komplext, det är då viktigt att förstå att valet av komponenter i en ESB bör göras utifrån de krav som ställs, samt tillägg som skalbarhet, flexibilitet, ”övervakningsbarhet”, mm. Gör man detta tycker jag vår ESB’s upplevda komplexitet enkelt kan vändas till en styrka.

14. När ska man använda en ESB lösning?

En ESB kan eg alltid implementeras, men det har störst värde om man vet att man har applikationer eller tjänster med olika meddelande format, transport protokoll och där man i förväg vet affärskrav kommer att påverka vilka applikationer eller tjänster som kommunicerar med vilka över tiden. När behov av flexibilitet och ”speed of change” är viktigt får man ut mest värde ur en ESB.

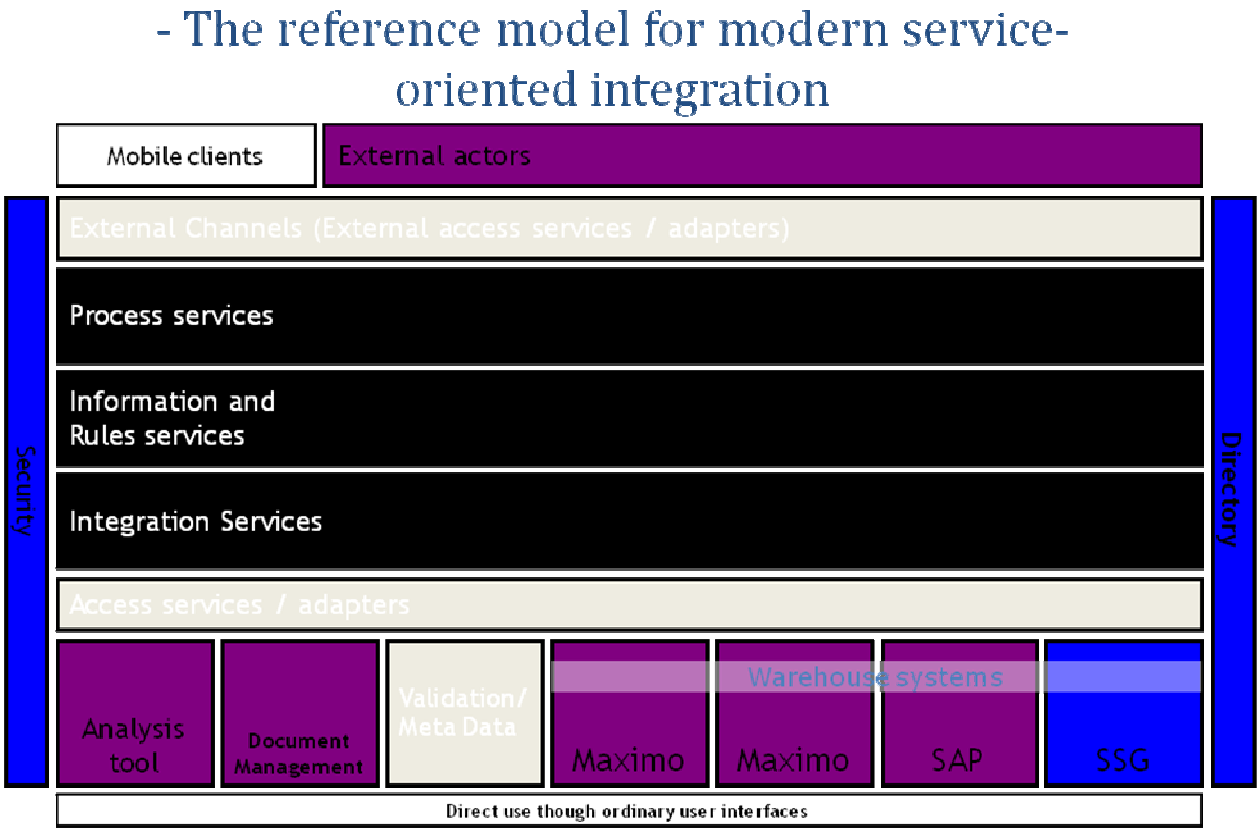
15. Hur ser ni på framtiden för ESB?

ESB är en förutsättning för en Service Oriented Architecture och har som sådan en mycket ljus framtid. Det kommer säkert att ske en del föränding, eller åtminstone debatt, kring vad man betraktar som kärnfunktionalitet i ESB och

vad som är tillägg. Detta är av mindre betydelse då en ESB av sin natur bör koppla utökad funktionalitet löst. Kandidater som jag kan tänka mig är framförallt Service Directoryt som idag ligger som en Governance funktion, men som lika gärna kan betraktas som en runtime komponent.

16. Här skulle vi vilja att ni ”mappade” in er integrationslösning och dess komponenter i vår referensmodell för modern tjänsteorienterad integration.

    Se Referensmodellen



A.3 WebMethods

WebMethods

1. Hur definierar ni begreppet ESB?

Svar: The term Enterprise Services Bus (ESB) is one that is being used increasingly with respect to Service-Oriented Architecture (SOA), yet there is still much confusion and debate as to what an ESB really is. The easiest way to understand ESB is to draw an analogy with message-oriented middleware (MOM). MOM provides features such as queuing, guaranteed delivery, routing, and publish/subscribe in support of Event-Driven Architecture. Similarly, an ESB provides capabilities that help to support the implement of Service-Oriented Architecture. In essence, an ESB can be viewed as the middleware for enabling SOA. However, whereas MOM solutions are typically proprietary, ESBs are largely based on the implementation of Web services standards such as SOAP, WSDL, UDDI, and the growing suite of WS-* standards. While ESBs have been touted as an alternative to business integration solutions, webMethods' view is that ESBs address a much narrow set of requirements. webMethods offers ESB functionality as part of our broader business integration solutions to provide greater value for customers requiring synergy across different levels of their integration infrastructure. We understand that an ESB on its own is not a complete integration solution and to address business-level needs, companies require additional higher-level/value functionality, such as business process management, business activity monitoring, support for trading partner integration, and e-business standards support.

2. Vilka är de funktionaliteter anser ni måste ingå i en modern tjänsteorienterad integrationslösning för att få kallas ESB?

Svar: Since the term ESB is a”changing” term, most of our customers has different requirements/understanding of what a ESB means to them. But generically the list below give a overview requirement list, some of our Customers incorporates BPM, BAM and other terms into the concept and then the functionality list increase.

- Service discovery and routing, to allow services within an SOA-based system to locate and invoke one another
- SOA Governance and management, capabilities to handle the life-cycle of services.
- Mediation of synchronous and asynchronous calls.
- Security, transaction management, logging and other capabilities required to build robust business applications
- Facilities such as data transformation and mapping that operate on the information flows between services
- Quality-of-service capabilities such as fail-over and load-balancing to facilitate service-level management
- Single Interface for management and monitoring of the SOA infrastructure reagardless of synchronous/asynchronous calls.

3. Har ert företag en ESB lösning?

Svar: Ja

5. Vilka funktionaliteter har er ESB lösning?

Svar: See PPT.

6. Hur är bussen uppbyggd rent teknisk? Vilka komponenter utgör bussen?

Svar: the webMethods Fabric suite is built in Java, and the components for the bus is:

- webMethods Broker
- webMethods Infravio X-Registry
- webMethods Infravio X-Broker
- webMethods ESP

7. Är er lösning hårt bunden till specifika produkter så som en speciell typ av server från ett specifikt företag?

Svar: Nej

8. Hur möjliggörs en fysisk decentraliserad och logiskt centraliserad infrastruktur i er ESB lösning?

Svar: webMethods has a long history in EAI and B2B integration, as a vendor providing integration it is cruicial to incorporate these two requirements into the platform, avoiding the centralized server which becomes the bottleneck or single point of failure. All components of the webMethods Fabric suite supports load-balanced or clustered setups. All administration and setup of these distributed components can be handled by the MyWebMethods portal, where portlets for configuring and setup are provided.

9. Hur kan fysiskt decentraliserade tjänster så som: mappning, orkestrering osv. verka tillsammans som en logisk enhet?

Svar: with the webMethods ESP, any service is available for anyone (with the righ priviligies) regardless of where they are residing and how they are called (WS or Messaging), services becomes reusable between the environments, they can publish the service publically through the registry using a mediation or not.

10. Vilka fördelar har er ESB lösning jämfört med andra existerande ESB lösningar? Med andra ord varför ska man välja er lösning?

Svar: Many… the advantages of a complete integrated stack of products is big. Governance and Management of SOA is one big advantage, the adapters, the lists becomes long…

11. Har ni möjlighet att ge oss kundreferenser där er ESB lösning har implementerats och en kontaktperson som vi kan kontakta för eventuella frågor?

Svar: we have over 1500 customers world-wide, we although with respect to ur customers do not provide customer-references without asking them, so if you are specific details you like to discuss with a client we can provide references.

12. Vad anser ni vara de främsta fördelarna med en ESB lösning?

Svar: A standard way to integrate a multitude of operative systems, applications and technologies.

13. Ser ni några nackdelar med en ESB lösning?

Svar: Regardless of what the customers pick, there are always advantages and disadvantages, but there are disadvantages, like the standards, they are always evolving and creates a confusion.

14. När ska man använda en ESB lösning?

Svar: Always I would say, it is a better way then doing “point to point” solutions or using some proprietary technology.

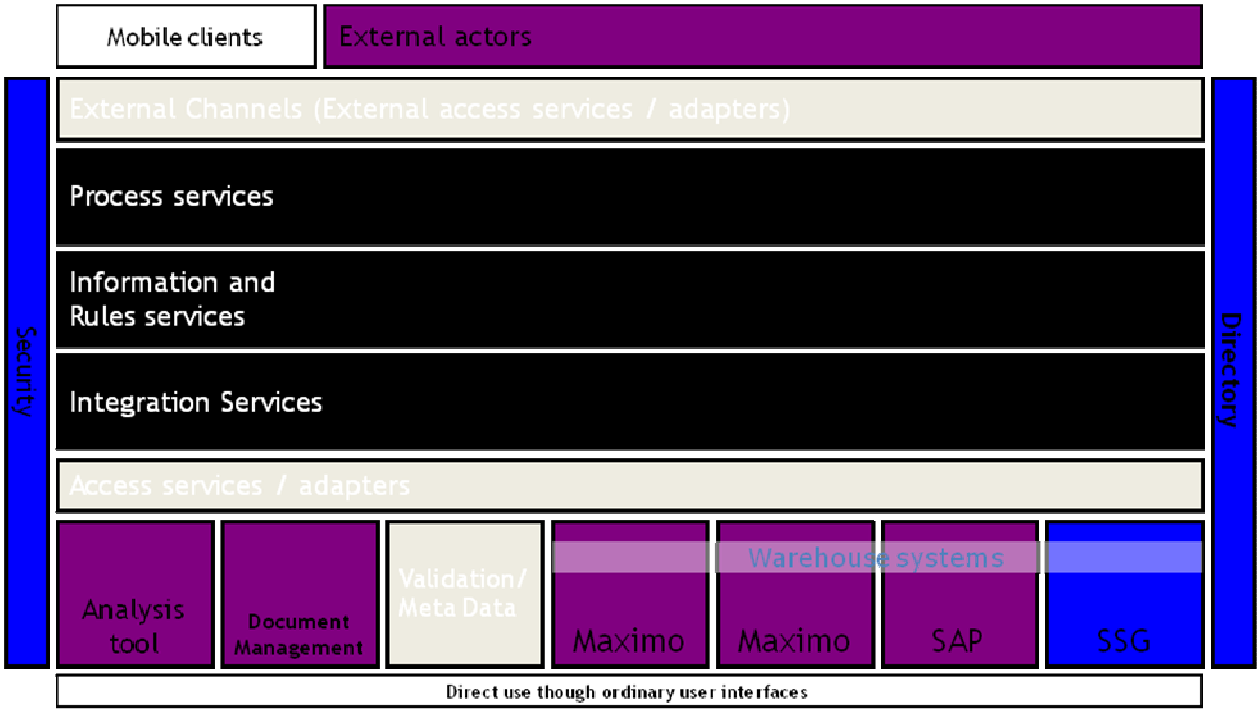
15. Hur ser ni på framtiden för ESB?

Svar: The concept of ESB is nothing new, looking back the ESB concept where something that was talked about in the Corba (OMG) initiative. I believe the ESB will evolve and incorporate a lot more into the concept. If ESB as a term will survive no one knows, but it is the best intitiative in the Integration space today.

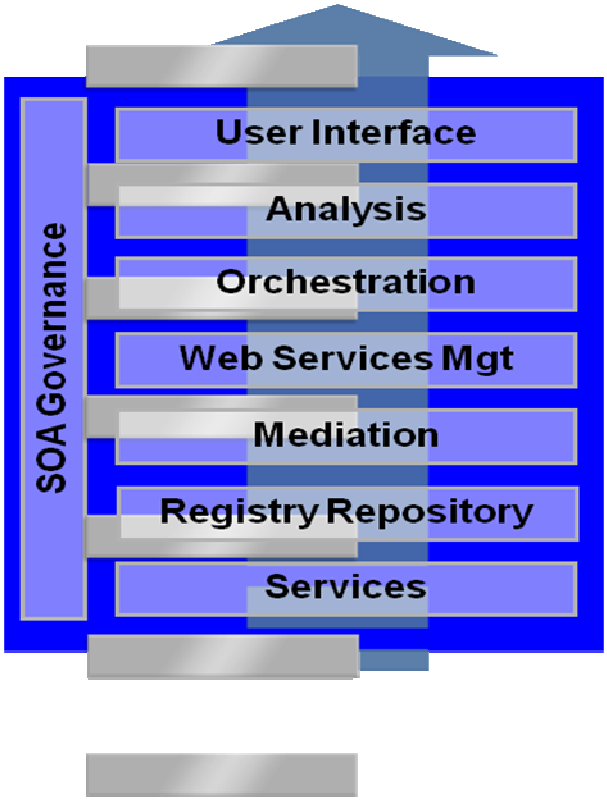
16. Här skulle vi vilja att ni ”mappade” in era produkter i vår referensmodell för modern tjänsteorienterad integration. Referensmodellen bifogas separat.

Svar: see included ppt.

- The reference model for modern service-oriented integration



Components of an SOA Infrastructure



User and System/Process Interaction

Process Analysis & Trending

Process Assembly

Service Monitoring

Connect Services

Service Registration & Reuse

Create Web Services

Services: To have an SOA, you obviously first need services. There are 3 places to get services: 1) you can buy them. The newest versions of applications such as Oracle and SAP are exposing their functionality as web services, and other applications you purchase may do so as well. webMethods is not an application vendor, so we don't play here. 2) Second, you can build them. Whether you leverage .NET or J2EE, you can develop new applications in a service-oriented way and expose new functionality as services. webMethods does not play here. 3) For most organizations, probably 90% of the business logic that runs the organizations exists on pre-existing systems that are not service enabled. Do leverage these capabilities within the SOA, you need a service enablement platform. This is where webMethods is focused, and we have done this for many years.

Registry Repository: Now that I have services, the thing that separates organizations who are successful at any scale with SOA vs. those who are simply dabbling is having a registry repository that lists what services are available, who built them, and contains additional information needed to understand how to use, support, and change the services that have been built. This is the critical component to deliver the benefit of reuse, and is the foundation for governing the SOA which we will discuss later.

Mediation: Once you have services built, and they are in a registry so they can be reused, next you need technology to allow them to communicate with each other at a technical level. This is mediation, and is important so that you don't build point-to-point SOA implementations that are brittle and hard to change.

Web Services Management: The next capability is Web Services Management. As you deploy a service oriented architecture, the fact that the services are distributed becomes a support challenge. Imagine a scenario where an application is calling 7 different services, and then the application becomes "slow". Which service is causing the slow down? What if other applications are using the same services at the same time? Web Services Management allows you to troubleshoot and monitor SOA implementations to ensure the same quality of service you would expect from a traditional client-server implementation.

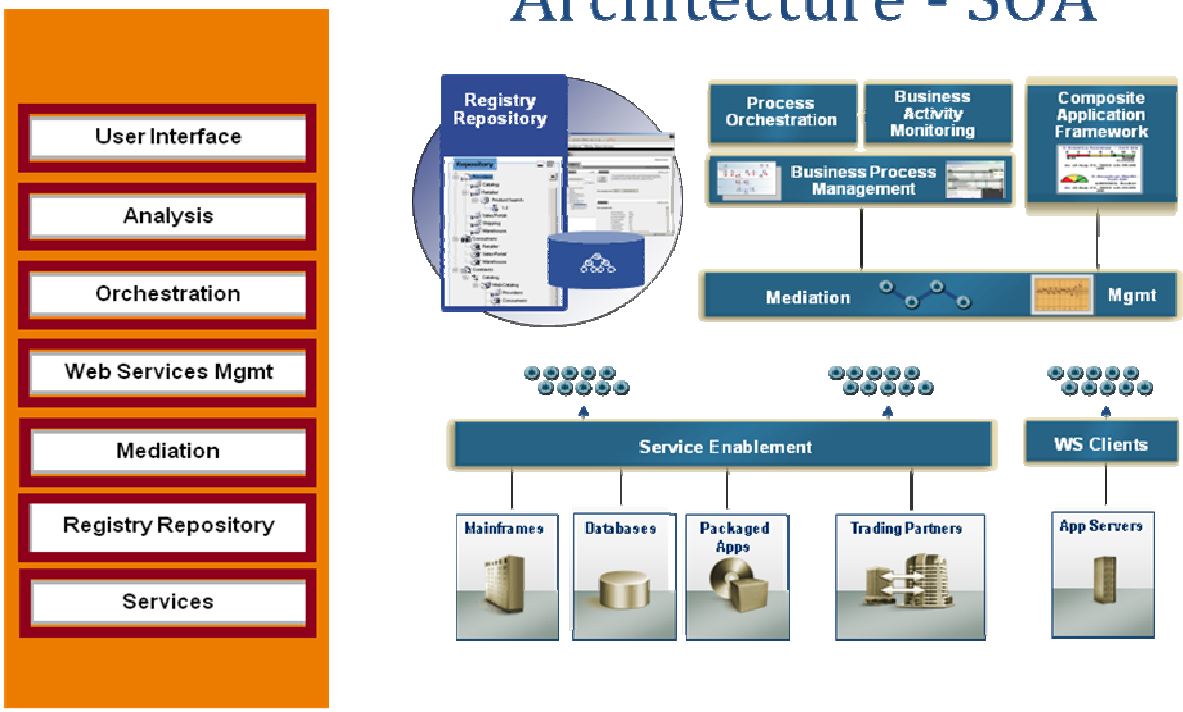
Orchestration: Once you have a set of managed web services, you can orchestrate those services into an automated process with graphical tools, lowering the cost of development and drastically improving the time to market. This is a critical capability to employ to realize the benefit of SOA.

Analysis: One of the great things about SOA is that because applications are actually made up of discrete, atomic, distributed services, it is easy to monitor between service calls and demonstrate key functionality.

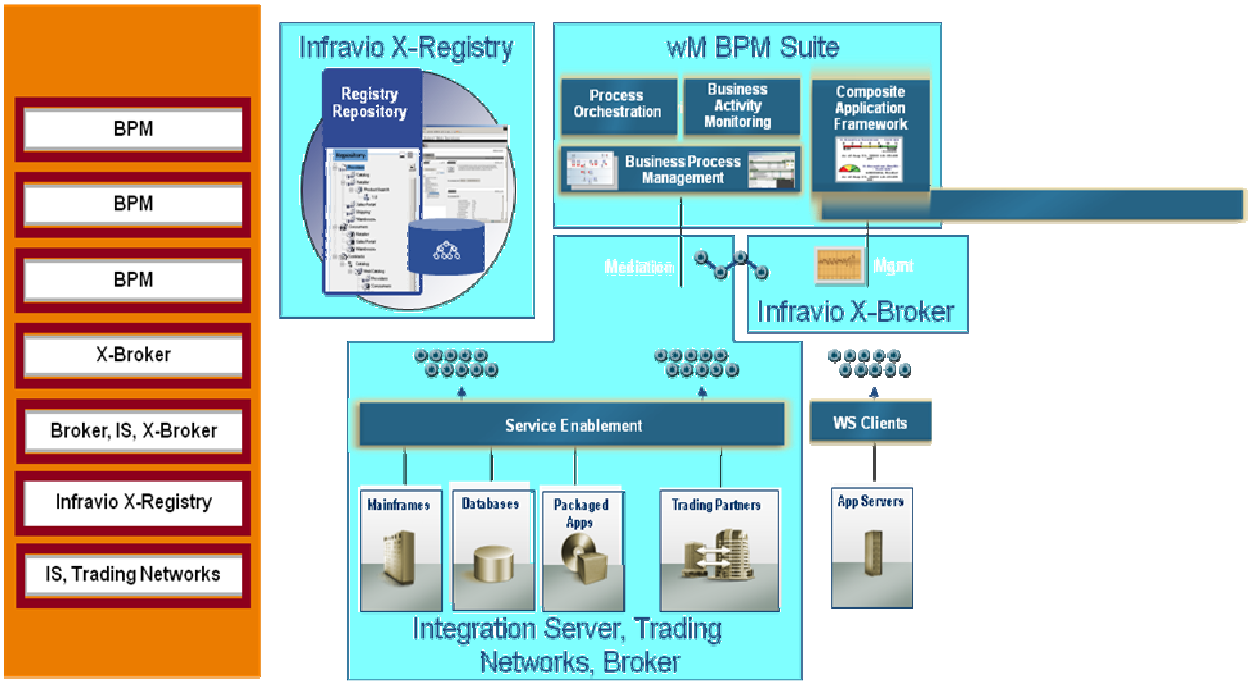
User Interface: Finally, in order to deliver SOA applications to the end user, you need a composite application environment that is optimized for web services.

SOA Governance: Encompassing all of this technology is SOA Governance. SOA Governance is the set of policies and standards that you set up within your organization to make sure that all stakeholders – developers, architects, support personnel, etc are able to interact with the SOA correctly. In the old days (ie prior to the last couple of years), most SOA Governance was done with paper. Today, technology exists to allow you to implement these polices and standards throughout the SOA, ensuring that you can scale SOA throughout and even beyond the enterprise safely and effectively.

webMethods Fabric Architecture - SOA



webMethods 7.0 Products in an SOA Architecture



A.4 Microsoft

Microsoft

1. Hur definierar ni begreppet ESB?

   An Enterprise Service Bus can be thought of as a collection of architectural patterns based on traditional enterprise application integration (EAI), message-oriented middleware, Web services, .NET and Java interoperability, host system integration, and interoperability with service registries and asset repositories. Application of these patterns provides an infrastructure that enables the flexible and secure reuse of services and the ability to rapidly orchestrate services into new end-to-end business processes.

2. Vilka funktionaliteter anser ni måste ingå i en modern tjänsteorienterad integrationslösning för att få kallas ESB?

   Vi anser inte att en ESB måste innehålla specifika funktionaliteter för att få kallas ESB, men den borde klara av eller ha stöd för följande:

   - Message routing
   - Message validation
   - Message transformation
   - Centralized exception management
   - Extensible adapter framework
   - Service orchestration
   - Business Rule Engine
   - Business Activity Monitoring

3. Har ert företag en ESB lösning?

   Ja, vi använder oss av våra produkter och skapar en ESB men vi har ingen egen ESB produkt.

   Om svaret på fråga 3 är nej, svara istället på fråga 4 och hoppa över fråga 5-11.

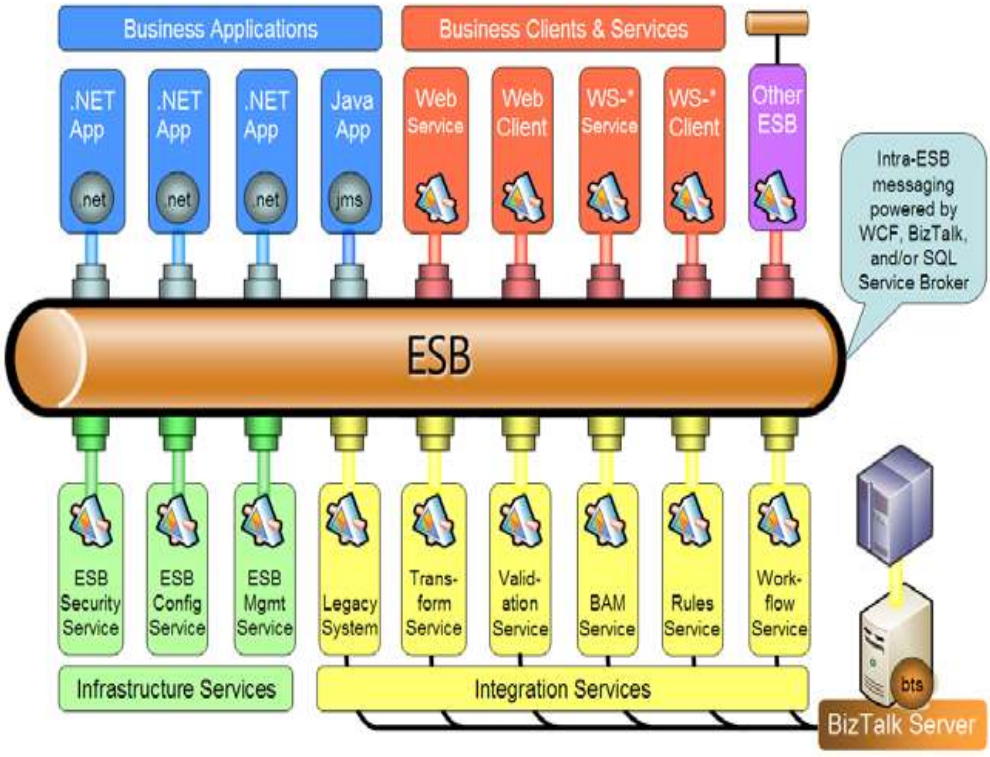
4. Har ni planerat att släppa en ESB lösning?

5. Vilka funktionaliteter har er ESB lösning?

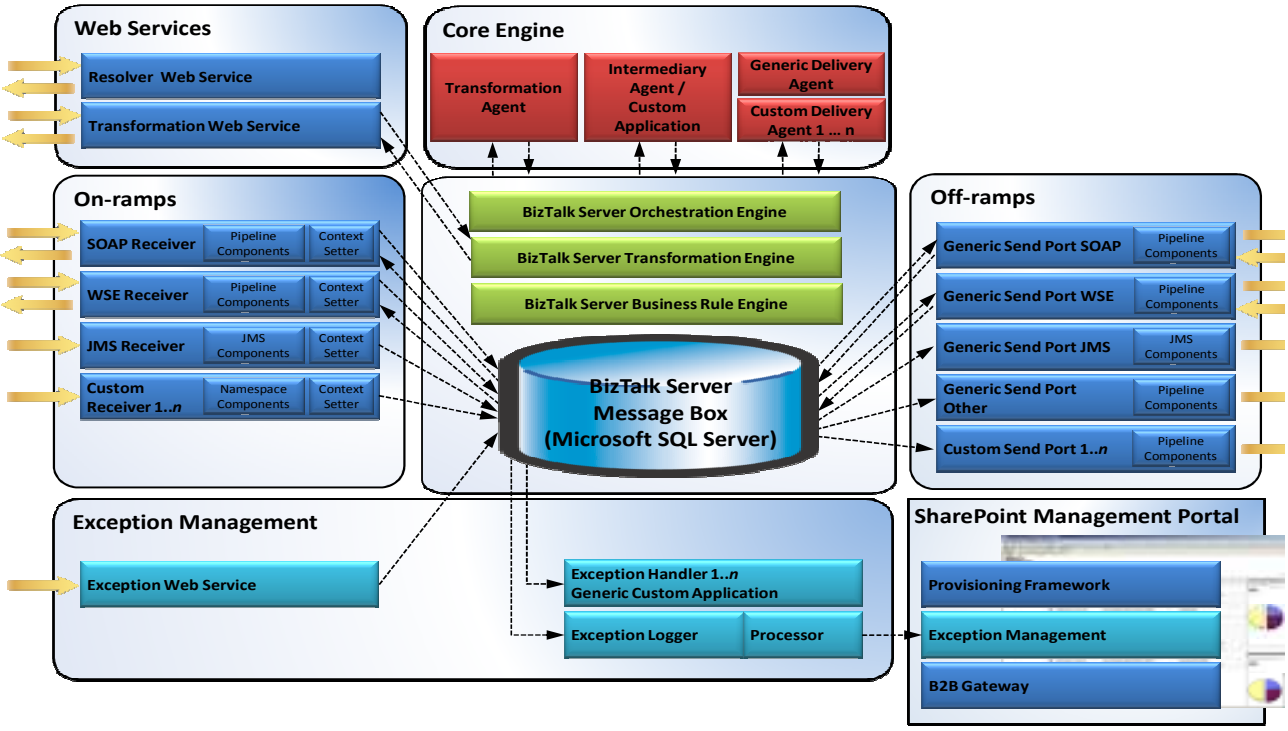
   Eftersom vi inte har någon ESB produkt anpassar vi lösningen efter kundens behov och önskemål. Det är upp till kunden vilka funktionaliteter som ska ingå.

6. Hur är bussen uppbyggd rent tekniskt? Vilka komponenter utgörs bussen av?

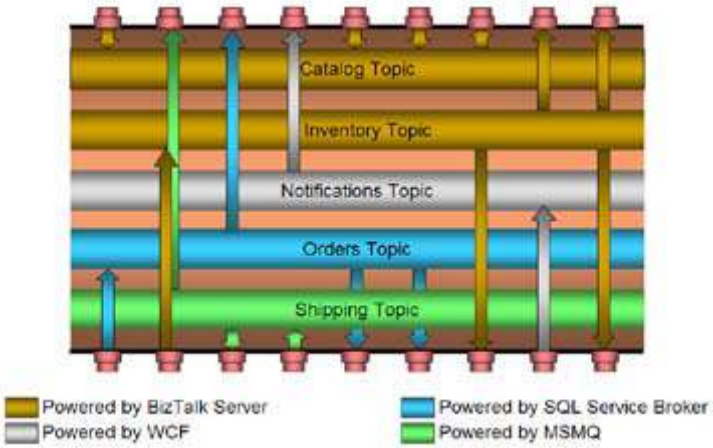
   Som ni kan se på bilden(Figur 1 och 3) utgörs bussen av WCF(Windows Communication Foundation) tillsammans med Biztalk och/eller SQL Service Broker. Bussen är uppbyggt på .NET ramverket ock kan tex köras på en Windows server(Figur 4).



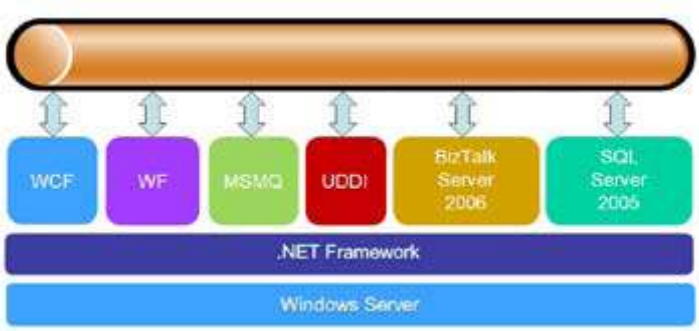
ESB infrastruktur



Microsoft ESB Guidance Architecture



Insidan av bussen



ESB´n uppbyggt på en Windows server

7. Är er lösning hårt bunden till specifika produkter så som en speciell typ av server från ett specifikt företag?

   Eftersom vår lösning är uppbyggd av våra egna produkter som arbetar tillsammans för att utgöra ESB´n kan man väl säga att den är hårt bunden.

8. Hur möjliggörs en fysisk decentraliserad och logiskt centraliserad infrastruktur i er ESB lösning?

   Se figur 1

9. Hur kan fysiskt decentraliserade tjänster så som: mappning, orkestrering osv. verka tillsammans som en logisk enhet?

   Dessa tjänster är inte fysiskt decentraliserade utan finns samlade i en Biztalk server. Se Figur 1.

10. Vilka fördelar har er ESB lösning jämfört med andra existerande ESB lösningar? Med andra ord varför ska man välja er lösning?

    Eftersom vår lösning kan anpassas helt efter kundens behov säljer vi inte på kunden mer än vad den behöver. Köper man en färdig produkt kanske man får betala för funktionalitet som man inte har nytta av. Våra produkter är utvecklade att fungera tillsammans så att man ska kunna använda dem tillsammans medans det finns

konkurrenter som köper färdiga produkter som anpassas för att passa deras andra produkter. Vi anser inte detta vara en optimal lösning då man inte utnyttjar resurserna på ett effektivt sätt.

11. Har ni möjlighet att ge oss kundreferenser där er ESB lösning har implementerats och en kontaktperson som vi kan kontakta för eventuella frågor?

Nej sådana uppgifter lämnar vi tyvärr inte ut.

12. Vilka fördelar medför en ESB lösning?

- Flexibilitet
- Skalbarhet
- Minskade kostnader
- Ökad verksamhets intelligens

13. Ser ni några nackdelar med en ESB lösning?

Nej

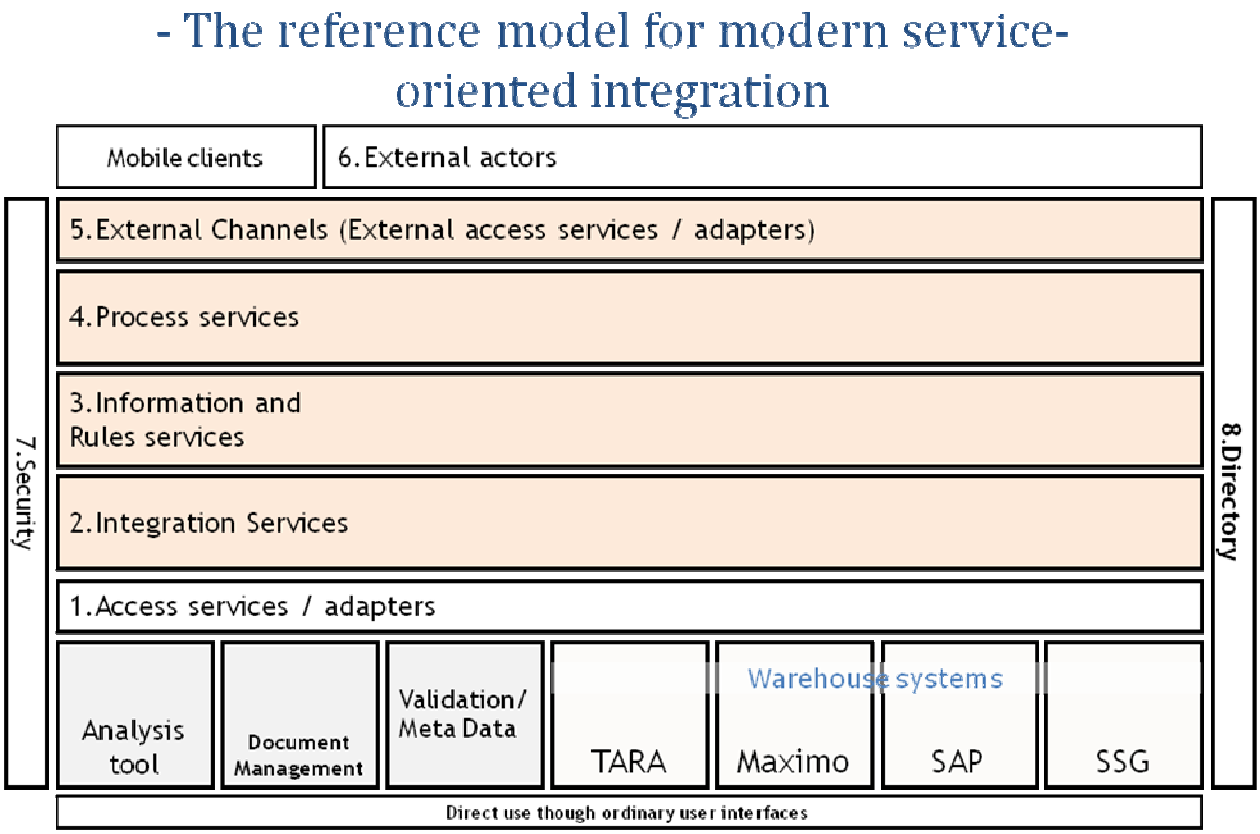
14. När ska man använda en ESB lösning?

Man kan alltid implementera en ESB men man bör noga se över sina system och se vad man vinner på att införa en ESB då initieringskostnaden är ganska stor. Vissa problem kan lösas på bättre sätt.

15. Hur ser ni på framtiden för ESB?

Framtiden för integrationlösningar ser ljus ut även om begreppet ESB kanske inte finns kvar. Då det råder delade meningar om vad en ESB är eller ska klara av kommer man antagligen att frångå detta begrepp för att undvika förvirring.

16. Här skulle vi vilja att ni ”mappade” in er integrationslösning och dess komponenter i vår referensmodell för modern tjänsteorienterad integration. Referensmodellen bifogas separat.



B Frågeankät Kunder

Tjänsteorienterad integration, ESB

Martin Bood och Karl-Johan Fisk

Examensjobb

Karlstads Universitet

1. Hur skulle ni definiera begreppet ESB?
2. Är det en tjänsteorienterad integrationslösning ni har infört/implementerat?
3. Vilken funktionalitet efterfrågade ni när ni valde er ESB-lösning?
4. Varför ville ni köpa en integrationslösning, vad var problemet?
5. Blev alla problem lösta efter införandet av den nya integrationslösningen?
6. Vilken verksamhetsnytta har ni fått av att införa en ESB-lösning?
7. Varför valde ni att implementera en ESB-lösning?
8. Har lösningen uppfyllt era förväntningar?
9. Har ni haft några problem i samband med implementation/driften?
10. Har införandet av en integrationslösning medfört verksamhetsnytta som ni inte hade räknat med?
11. Är implementationen lokal eller global?
12. Är er ESB-lösning avsedd för B2B och internt bruk eller endast för internt bruk?
13. Från vilket företag kommer er lösning?
14. Varför valde ni det företagets lösning?
15. Rekommenderar ni andra att implementera en ESB-lösning?
16. Med facit i hand skulle ni ha löst ert problem på ett annat sätt?
17. Hade ert företag någon integrationslösning innan införandet av er ESB-lösning?

B.1 Försäkringskassan

Försäkringskassan

Lite allmänt.

Vi är på väg att införa en ESB (JCAPS från Sun). Första applikationen är planerad att vara i produktion i november (2007). Vi har köpt produkterna eGate (ESB) och eInsight (BPM) som ingår i JCAPS från Sun. I köpet ingår också kö-hanteraren JMS_Grid.

Vi började söka efter en gemensam kö-hanterare för Tuxedo och WebLogic Server, det slutade med att vi köpte en ESB för att få ett mervärde utöver själva kö-hanteraren.

Vi är en site med en StorDator (Bull mainframe) i vilken vi kör ca 50 förmåner (barnbidrag, pension, tillfällig föräldrapenning, …..). Vi har påbörjat arbetet att flytta applikationer från StorDatorn till den öppna sidan (Unix Sun Solaris). Vi har applikationer i StorDatorn (Cobol), Tuxedo (C++), WebLogic Server (J2EE), lokala applikationer (.Net) och köpta standardsystem ex vis SAP. Många av applikationerna behöver utbyta information med varandra.

Vår nya applikationsarkitektur bygger på meddelandehantering (lösa kopplingar). Den är också tjänstebaserad. Vi har idag ett stort antal interna tjänster, stora som små.

Svar på era frågor.

Hur skulle ni definiera begreppet ESB?

En standardiserad integrationsplattform som kombinerar meddelandehantering, webservices, datatransformering och intelligent routing för att koppla ihop och koordinera interaktioner mellan ett stort antal applikationer i en organisation med dess affärspartners.

Är det en tjänsteorienterad integrationslösning ni har infört/implementerat?

Ja. Vi tänker alltid i form av tjänster, stora som små. Vi har det redan idag i Tuxedo och WebLogic Server. På sikt räknar vi med att kunna skapa ”större” tjänster (processer) genom att sätta ihop ett flertal mindre tjänster.

Vilken funktionalitet efterfrågade ni när ni valde er ESB-lösning?

Transformering och Webservices samt kopplingsmöjligheter; Tuxedo och WLS mot SAP och Aptic/Collect. Vi letade också efter en generell kö-hanterare.

Varför ville ni köpa en integrationslösning, vad var problemet?

Vi har idag en hel del integrationer bl a något vi kallar för ”förmånsförmedling” – förmånerna behöver prata med varandra. Idag har vi en egenutvecklad förmånsförmedlare i Tuxedo, vi tänker ersätta den med ESBn. En strategi är att vi ska använda standardprodukter där det är möjligt.

Integrationen med SAP och Aptic/Collect. Problemet var att vi inte ville göra integrationen själva utan använda en standardprodukt/verktyg. Förmånerna ligger i Tuxedo, J2EE och StorDator.

Blev alla problem lösta efter införandet av den nya integrationslösningen?

Ja. Det vi hittills har sett.

Vilken verksamhetsnytta har ni fått av att införa en ESB-lösning?

Eftersom vi inte har kommit igång ”på riktigt” så kan jag inte svara.

Varför valde ni att implementera en ESB-lösning?

Vid behövde en integrationsplattform och ville inte ha en ”spagetti”-lösning med egna integrationer i våra applikationer.

Har lösningen uppfyllt era förväntningar?

Eftersom vi inte har kommit igång ”på riktigt” så kan jag inte svara.

Men, vad säjs om ett Nja. Det har varit lite segt, allt har inte varit problemfritt. Vi har samtidigt försökt inrätta en funktion för integrationer, ett ICC (Integration Competence Center).

Har ni haft några problem i samband med implementation/driften?

Eftersom vi inte har kommit igång ”på riktigt” så kan jag inte svara.

Vi har gjort ett antal integrationer (16 st) som f n är i enhetstest. Integrationerna fungerar ok, men vi ser att vi har en hel del trimningsarbete (det tar bl a för lång tid i bussen) kvart att göra.

Har införandet av en integrationslösning medfört verksamhetsnytta som ni inte hade räknat med?

Än har vi inte upptäckt någon extra nytta.

Är implementationen lokal eller global?

Lokal.

Är er ESB-lösning avsedd för B2B och internt bruk eller endast för internt bruk?

Internt bruk till att börja med. På sikt kommer den att användas vid samverkan med andra myndigheter och medborgare (via medborgarportal).

Från vilket företag kommer er lösning?

Sun. Produkten kommer ursprungligen från Seebeyond som blev uppköpt av Sun.

Varför valde ni det företagets lösning?

Billigast. Bästa licensmodellen.

Rekommenderar ni andra att implementera en ESB-lösning?

Ja.

Med facit i hand skulle ni ha löst ert problem på ett annat sätt?

Nej.

Hade ert företag någon integrationslösning innan införandet av er ESB-lösning?

Mellan förmånerna har vi en egen lösning. Mellan Unix och StorDatorn har vi också en ”egen lösning” som vi kallar StordatorBryggor för synkron och asynkron kommunikation.

B.2 SKF

SKF

Hur skulle ni definiera begreppet ESB?

Den del av vår centrala integrationsmiljö där vi håller gemensamma meddelanden/tjänster tillgänliga för prducent- och konsumenapplikationer.

Är det en tjänsteorienterad integrationslösning ni har infört/implementerat?
Ja, ibland tjäönsteorienterad och ibland eventdriven.

Vilken funktionalitet efterfrågade ni när ni valde er ESB-lösning?

jag var inte med och valde, men med det jag vet, så låg fokus på kraven för hela integrationsmiljön såsom flexibilitet och snabbhet i utveckling + prestanda att hantera stora mängder transaktioner på kort tid.
ESB biten har nog inte kravställts särskilt mycket separat.

Varför ville ni köpa en integrationslösning, vad var problemet?
Snabbhet att integrera nya bolag och trading partners elektroniskt
Möjliggöra SOA/Webservices kommunikation framöver och processrelaterad appikationsintegration. Flexibilitet m.a.p transportprotokoll och affärsprotokoll.
Vi ville koppla lös våra applikationer från P2P lösningar.

Blev alla problem lösta efter införandet av den nya integrationslösningen?

Jajjamensan :-))

Självklart inte, men vi är bra mycket mer byxade på många plan idag än innan.
B2B protokoll flexibitiet, / B2B EDI har varit en svaghet hos vår produkt.

Vilken verksamhetsnytta har ni fått av att införa en ESB-lösning?
- t.ex. möjlighet att hantera allsjöns XML för externt och interna ändamål.
- återanvändbarhet av gemensamma meddelanden (servicar)

Varför valde ni att implementera en ESB-lösning?
Vår integrationslösning innehåller komponenter för att bära ett kanoniskt internt meddelandeformat som ligger till grund för våra servicar och den platsen utgör däremed vår ESB. Detta koncept stöds, men just begreppet ESB har vi inte sålt in särskilt hårt än.

Har lösningen uppfyllt era förväntningar?
I stort JA, men inte rörande B2B protokoll flexibitiet.
Metadata kring det kanoniska internformatet behöver kompltterande produkter för att manageras.

Har ni haft några problem i samband med implementation/driften?
Ja, JAVA runtime relaterade problem. Problem att veta hurman skall dimensionera olika JAVA parameterar.

Har införandet av en integrationslösning medfört verksamhetsnytta som ni inte hade räknat med?
Nej

Är implementationen lokal eller global?
Global

Är er ESB-lösning avsedd för B2B och internt bruk eller endast för internt bruk?
Allt

Från vilket företag kommer er lösning?
Sun

Varför valde ni det företagets lösning?
Bra prestanda/pris

Rekommenderar ni andra att implementera en ESB-lösning?
Större företag JA, mindre företag ?

Med facit i hand skulle ni ha löst ert problem på ett annat sätt?
Nej

Hade ert företag någon integrationslösning innan införandet av er ESB-lösning?
Ja, flera regionala, men ingen global

B.3 Nordea

Nordea

- Hur skulle ni definiera begreppet ESB? **Fleksibel integrasjons hub.**
- Är det en tjänsteorienterad integrationslösning ni har inf¨rt/ implementerat? **Ja**
- Vilken funktionalitet efterfrågade ni n¨r ni valde er ESB-lösning? **Effektiv, integrasjon på tvers av mange plattformer.**
- Varför ville ni köpa en integrationslösning, vad var problemet? **Stort transformeringsprosjekt med økt krav til fleksibilitet.**
- Blev alla problem lösta efter införandet av den nya integrationslösningen? **Nei, ikke helt innkjørt enda.**
- Vilken verksamhetsnytta har ni fått av att införa en ESB-lösning? **Komplett transformasjon av forretningsprosesser.**
- Varför valde ni att implementera en ESB-lösning? **For å kunne integrere på en smartere og lettere måte.**
- Har lösningen uppfyllt era förväntningar? **Vet ikke enda, midt i implementasjon.**
- Har ni haft några problem i samband med implementation/ driften? **Store krav til kompetanseheving.**
- Har införande av en integrationslösning medfört verksamhetsnytta som ni inte hadde räknat med? **Vet ikke enda.**
- Är implemtationen lokal eller global? **Global.**
- Är er ESB-lösning avsedd för B2B och internt bruk eller endast för internt bruk? **Internt bruk.**
- Från vilket företag kommer er lösning? **BEA**
- Varför valde ni det företagets lösning? **Funksjonalitet, pris og modenhet i produktet.**
- Rekommenderar ni andra att implementara en ESB-lösning? **Ja**
- Med facit i hand skulle ni ha löst ert problem på ett annat sätt? **Nei**

  Hade ert företag någon integrationslösning innan införandet av er ESB-lösning? **Ja.**

B.4 Volvo

VOLVO

1. Hur skulle ni definiera begreppet ESB?
   A service bus aid Service consumers in accessing services
   - *Even when the service consumer and provider do not share the communication method*

   Services can be exposed and further improved at the service bus
   - *For example, multiple low lever application services can be composed into a high level business service*
2. Är det en tjänsteorienterad integrationslösning ni har infört/implementerat?
   Vår implementerade lösning syftar till att hantera både tjänsteorientering (SOA) och Enterprise Application Integration (EAI)
3. Vilken funktionalitet efterfrågade ni när ni valde er ESB-lösning?
   - Transformation
   - Routing
   - Conversion
   - Connectivity
4. Varför ville ni köpa en integrationslösning, vad var problemet?
   En lång historik med point to point kopplade system, vilket är komplicerat att underhålla och förändra tillräckligt snabbt när verksamheten ställer krav.
5. Blev alla problem lösta efter införandet av den nya integrationslösningen?
   Nej, i och med att vi har implementerat IT lösningar sedan 60-talet har vi en stor ryggsäck, denna transformeringsfas har vi just påbörjat och kommer att ta många år att genomföra (om vi gör det alls för vissa system). Där vi har använt den "rätta" integrationslösningen ser vi att det ger oss de effekter vi eftersträvar.
6. Vilken verksamhetsnytta har ni fått av att införa en ESB-lösning?
   För tidigt att uttala sig om, men vi ser stora möjligheter att snabbare möta nya marknads och verksamhetskrav (nya affärsmodeller, processer, tjänster etc.)
7. Varför valde ni att implementera en ESB-lösning?
   Se 6
8. Har lösningen uppfyllt era förväntningar?
   Se 6
9. Har ni haft några problem i samband med implementation/driften?
   Ja,
   - avsaknad av gemnsamma arbetssätt och metoder
   - Kompetensbrist
   - Finansiering av ny infrastruktur
   - Vem skall äga och förvalta integrationslagret
   - Förståelse för nyttan av integration ifrån systemägare och leverantör
10. Har införandet av en integrationslösning medfört verksamhetsnytta som ni inte hade räknat med?
    Se 6
11. Är implementationen lokal eller global?
    Global per affärsområde, på sikt global för hela Volvo

12. Är er ESB-lösning avsedd för B2B och internt bruk eller endast för internt bruk?
    Avsedd för all integration både internt och externt.
13. Från vilket företag kommer er lösning?
    Flera olika, störst leverantör är IBM
14. Varför valde ni det företagets lösning?
    Öppen arkitektur som följer öppna standards, möter våra krav på skalbarhet, prestanda och robusthet.
15. Rekommenderar ni andra att implementera en ESB-lösning?
    Vi rekommenderar andra att ta ett helhetsgrepp över sin arkitektur och däri är en ESB en viktig komponent.
16. Med facit i hand skulle ni ha löst ert problem på ett annat sätt?
    Inte tekniskt men organisatoriskt.
17. Hade ert företag någon integrationslösning innan införandet av er ESB-lösning?
    Ja, Volvo har en mycket lång tradition av att integrera applikationer mha messaging teknik. Volvo utvecklade själv en produkt på 70-talet (VCom) då inget fanns att köpa på marknaden som uppfyllde våra krav.

C Frågeankät Konsultbolag

Tjänsteorienterad integration, ESB

Martin Bood och Karl-Johan Fisk

Examensjobb

Karlstads Universitet

1. Hur definierar ni begreppet ESB?
2. Vilka funktionaliteter anser ni måste ingå i en modern tjänsteorienterad integrationslösning för att få kallas ESB?
3. Vilka funktionaliteter i en ESB efterfrågas mest av era kunder?
4. Vilka företag på marknaden anser ni levererar en ESB-lösning idag?
5. Finns det företag på marknaden som säger sig leverera en ESB-lösning som ni inte anser vara en ESB? Vilka?
6. Vilka/Vilken ESB-lösning(ar) har ert företag valt att driftsätta hos kunder?
7. Varför väljer ni just dessa/denna ESB-lösning?
8. Vilken ESB-lösning har ni på ert företag bäst erfarenhet av?
9. Hur många ESB-lösningar har ni driftsatt?
10. Är det stora skillnader i pris mellan olika ESB-lösningar?
11. Hur ligger de olika ESB-lösningarna i pris i förhållande till varandra? Ange gärna prisintervall.
12. Vilka fördelar medför en ESB lösning?
13. Ser ni några nackdelar med en ESB lösning?
14. När ska man använda en ESB lösning?
15. Hur ser ni på framtiden för ESB?

C.1 Zystems By Semcon

Zystems By Semcon

1. Hur definierar ni begreppet ESB?

An Enterprise Service Bus is a flexible infrastructure that handles the **mediation of messages** between the applications being integrated in a **consistent manner**.

2. Vilka funktionaliteter anser ni måste ingå i en modern tjänsteorienterad integrationslösning för att få kallas ESB?

- Routing
- Conversion (protocol)
- Transformation (data)
- Connectivity (transport)
- Logging/monitoring/management
- Service Registry

3. Vilka funktionaliteter i en ESB efterfrågas mest av era kunder?

Allt ovan

4. Vilka företag på marknaden anser ni levererar en ESB-lösning idag?

- Zystems By Semcon
- Flera företag levererar produkter med vars hjälp man kan bygga en ESB men produkt räcker inte, man behöver också koncept och metod för att realisera en ESB
- Flera av våra konkurrenter har individer med stor integrationskompetens men inget gemensamt integrationskoncep

5. Finns det företag på marknaden som säger sig leverera en ESB lösning som ni inte anser vara en ESB? Vilka?

Se ovan

6. Vilka/Vilken ESB lösning(ar) har ert företag valt att driftsätta hos kunder?

Baseline implementerat på IBM WebSphere integrationsprodukter (främst WebSphere MQ och Message Broker, men även ICS, WPS, WPG, WESB)

7. Varför väljer ni just dessa/denna ESB lösning?

Baseline är unikt i sitt slag. IBM WebSphere är den produkt (enligt vår åsikt) som har störts förmåga att hantera olika plattformar. Den är dessutom mycket skalbar och robust.

8. Vilken ESB lösning har ni på ert företag bäst erfarenhet av?

Se ovan

9. Hur många ESB lösningar har ni driftsatt?

Ett 40-tal

10. Är det stora skillnader i pris mellan olika ESB lösningar?

Ja

11. Hur ligger de olika ESB lösningarna i pris i förhållande till varandra? Ange gärna prisintervall.

Vi har ingen uppfattning om andra leverantörer än IBM. Inom IBM är licenskostnaden för de olika produkterna avhängig antal processorer mm, medan Baseline betingar ett fast pris. Vårt tjänsteerbjudande Baseline Start-kit (fn 400.000 kr) ger kunden alla verktyg och kunskaper som behövs för att komma igång med en integrationplattform (ESB) på 2-3 veckor.

12. Vilka fördelar medför en ESB lösning?

- Enhetlighet
- Lös koppling
- Kontroll
- Flexibilitet

- Spårbaret

13. Ser ni några nackdelar med en ESB lösning?

Man måste ha en viss volym avseende informationsutbytet eftersom en ESB har en relativt hög initialkostnad. Den ställer också nya krav på kompetens, organisation och ägarskapsfrågor.

14. När ska man använda en ESB lösning?

Vid stora krav på informationsutbyte mellan relativt många system i en heterogen miljö

15. Hur ser ni på framtiden för ESB?

Ljus. Konceptet har funnits länge under andra namn (t.ex. Hub and Spoke) och kommer säkert att få nya namn och features i framtiden. Enligt vår åsikt är en robust ESB en förutsättning för att SOA skall fungera i praktiken, i alla fall så länge inte alla system i hela världen är WebService enablade.