IT-stöd för säljande servicetekniker

Integration genom tjänster

Innehållsförteckning

[Inledning 4](#_Toc347080160)

[Bakgrund 4](#_Toc347080161)

[Problemet 5](#_Toc347080162)

[Syfte 5](#_Toc347080163)

[Målgrupp 5](#_Toc347080164)

[Avgränsning 5](#_Toc347080165)

[Metod 5](#_Toc347080166)

[Problembeskrivning 6](#_Toc347080167)

[Arkitekturansats 6](#_Toc347080168)

[Säljprocessen 6](#_Toc347080169)

[Begreppsmodell 7](#_Toc347080170)

[Systemförteckning 10](#_Toc347080171)

[Problembeskrivning 11](#_Toc347080172)

[Om enheters identitet 11](#_Toc347080173)

[Krav på IT-stödet 11](#_Toc347080174)

[Problemområden kondenserade 12](#_Toc347080175)

[Arkitekturansats 13](#_Toc347080176)

[övergripande arkitektur och Användningsfall 13](#_Toc347080177)

[Logisk vy 17](#_Toc347080178)

[Utvecklingsvy 19](#_Toc347080179)

[ERP-fasad 20](#_Toc347080180)

[Säljprocess 21](#_Toc347080181)

[Fysisk vy 23](#_Toc347080182)

[Slutsatser 23](#_Toc347080183)

[Diskussion 23](#_Toc347080184)

[Referenser 25](#_Toc347080185)

[Bilagor 26](#_Toc347080186)

[Begreppslista 26](#_Toc347080187)

# Inledning

## Bakgrund

Det har sedan en tid blivit vanligare att företag och organisationer gör satsningar för att röra sig bort från komponentbaserade lösningar för att fokusera mer på att ett tjänstebaserat synsätt på sin IT-flora. De tidigare komponentbaserade lösningarna har ofta haft hög detaljupplösning för att tillgodose nichade behov medan de tjänstebaserade gränssnitten har blivit vidare och mer generaliserade för att kunna tillgodose ett större antal klienter i mer heterogena sammanhang. Då dessa tjänster ofta utgör vissa kärnprocesser i organisationen är det inte ovanligt att de bygger på ett flertal underliggande komponenter och tjänster som arbetar tillsammans för att leverera värde. Detta samarbete ställer höga krav på detaljinkapsling och tydliga avgränsningar mellan komponenterna de implementerande komponenterna.

[LARM AB] är ett internationellt bolag inom säkerhetsbranschen och erbjuder vakter och säkerhetslösningar för både företag och privatpersoner. De har gjort en stor satsning på den privata marknaden genom hemförsäljning där larminstallatörer besöker privatpersoner i deras hem och utför både själva installationen av larmutrustningen samt genomför den slutgiltiga affären.

Tidigare har säljprocessen till stor del skett manuellt genom att installatörerna fört bok över de komponenter som installerats hos den nyblivna kunden samt de tjänstetillval denna gjort vid installationstillfället. Detta underlag har sedan lämnats över till den administrativa personalen på [LARM AB] som slutfört affären formellt. I den administrativa behandlingen av säljet var det inte ovanligt att personalen upptäckte att enskilda installatörers lagersaldon skiljde sig mot bokförda volymer och att vissa installationer/sälj avvek från normen med avseende på vilka komponenter som blivit sålda/inkluderade i installationen. Företagets verksamhetsutvecklare kunde efter en genomlysning av sälj- och lagerprocesserna slå fast att enskilda installatörer ofta ”lånade” komponenter från varandras lager när deras egna lagernivåer blev låga utan att rapportera tillbaka till administrationen, men också att det ibland helt saknades komponenter i lagrena, varpå dessa aldrig ens erbjöds kunderna.

[LARM AB] kom fram till att ett tjänstebaserat IT-stöd för hela säljprocessen skulle implementeras och genom detta skulle larminstallatörerna få bättre möjligheter att sköta den egna lagerhållningen; vilket både skulle underlätta det administrativa uppföljningsarbetet och dessutom uppmuntra installatörerna att arbeta tillsammans med varandra med en gemensam lagerhållning och på så vis sälja större installationer till kunderna.

För att realisera detta IT-stöd bestämde sig [LARM AB] för att utveckla en mobilapplikation för installatörernas sälj- och installationsarbete, samt en samordnande tjänst för att automatisera och koordinera de tjänster och verksamhetssystem som implementerar de administrativa funktionerna som är förknippade med försäljningen. Det högst prioriterade kravet på lösningen var att inget sälj skulle gå förlorat och att hela säljet så långt som möjligt skulle genomföras vid ett besökstillfälle, utan kompletterande besök och installationer. Dessutom fanns höga krav på att de mobila applikationerna skulle vara feltoleranta och ha korta svarstider för att göra försäljningsarbetet så smidigt som möjligt.

Alla [LARM AB]s underliggande IT-system erbjöd integrationsmöjligheter i viss mån. Dock visade det sig att företagets ERP-system rotat sig så pass djupt i organisationen att dess tekniska egenskaper starkt influerat det sätt säljverksamheten bedrevs på. Ett faktum som påverkade integrationsmöjligheterna med övriga system och i förlängningen även möjligheterna för det nya tjänstebaserade IT-stödet att till fullo stödja det säljförfarande som organisationen önskade sig.

## Problemet

Hur bör en samordnande tjänst som realiserar IT-stödet för den komplexa säljprocessen implementeras övergripande för att säkerställa att [LARM AB] inte tappar ett sälj, hur bör denna tjänst verka för att isolera användarna från potentiella avbrott i underliggande system och tjänster; och hur bör dessa implementeras för att isoleras ifrån läckande implementationsdetaljer från andra verksamhetskomponenter?

## Syfte

Syftet med denna uppsats är att belysa vilka problem det innebär att integrera ett antal system, med varierande grad av direkt verksamheteninverkan, för att realisera ett IT-stöd för en hel verksamhetsprocess; samt presentera ett förslag på en arkitektur som överkommer dessa problem i allmänhet samt kapslar in och motverkar [LARM AB]s ERP-systems bieffekter i synnerhet.

## Målgrupp

Uppsatsen skrivs som en del av examinationen för Dataföreningens kurs ”Certifierad IT-arkitekt” omgång 45. Den tänkta målgruppen är aspirerande IT-arkitekter i allmänhet och integratörer i synnerhet.

För att tillgodogöra sig denna uppsats på ett lämpligt sätt bör läsaren ha grundläggande förståelse för UML, detta är dock inget krav då de flesta modellerna även beskrivs med text.

En viss kännedom om mönster för integration och systemutveckling i distribuerade miljöer kan vara till hjälp för att snabbt sätta sig in i resonemang. Mönstrena kommer även i förekommande fall redovisas i referenslistan för eventuell inläsning.

## Avgränsning

Uppsatsens fokus ligger på konceptuell och logisk nivå och ämnar endast beskriva lösningar på dessa nivåer. Teorier som presenteras i denna uppsats kommer baseras på vedertagna mönster och resonemang, dessa teorier kommer inte nödvändigtvis verifieras mot en teknisk implementation.

Då målet med uppsatsen är att belysa arkitektoniska detaljer kring integration, kommer kringliggande applikationer och system endast beröras i den mån det tillför något till uppsatsens resonemang. Uppsatsen kommer inte heller föreslå några förändringar i, eller byten av, system eller applikationer även om det kan verka lämpligt för att uppnå uppsatsens syfte.

## Metod

Uppsatsen skrivs på svenska men innehåller en del engelska ord och uttryck där direkta översättningar inte är möjliga eller skapar förvirring. Dessa uttryck beskrivas utförligare i en ordlista.

Jag har valt att dela upp uppsatsen i två huvudsakliga avsnitt, där det första är en mer detaljerad problembeskrivning för att formulera de krav och förutsättningar som ligger till grund för den tänkta arkitekturen och knyta an dessa mot uppsatsens problemformulering. I den andra delen presenteras ett arktitekturförslag som en potentiell lösning anpassad efter problembeskrivningen.

### Problembeskrivning

Problembeskrivningen utgörs dels av de funktionella och icke-funktionellla krav som [LARM AB] ställt på IT-stöden men även till stor del av en genomlysning av organisationens befintliga IT-stöd som är av intresse för uppsatsen. Detta avsnitt kommer alltså utöver de uttalade kraven även fokusera på att lyfta fram de egenskaper i den befintliga miljön som är arkitekturdrivande

### Arkitekturansats

Uppsatsen utgår ifrån Kuchtens 4+1 modell för att beskriva arkitekturen för den lösning som föreslås. Modellen bygger på dessa fem vyer

* **Scenarier** – i denna vy beskrivs interaktionen mellan användare/system, mellan processer och komponenter för att belysa de arkitekturdrivande användningsfallen.
* **Logisk vy** – I denna vy ligger fokus på att beskriva systemets funktioner ur slutanvändarens perspektiv
* **Utvecklingsvy** – beskrivningar anpassade för utvecklare och programmerare. Fokus ligger på beskriva strukturer som klasser och komponenter
* En **processvy** där fokus ligger på systemets beteende under exekvering. Exempelvis för att beskriva parallellitet och hur trådar och processer kommunicerar med varandra
* **Fysisk vy** – beskriver hur systemet installeras/distribueras på olika fysiska noder och miljöer. Beskriver också hur olika delar i systemet är sammanlänkade fysiskt

Varje vy kan sedan brytas ned med finare upplösning i nivåerna *övergripande, logisk* och *fysisk.*

Eftersom det inte är uppsatsens syfte att vara en komplett arkitekturbeskrivning så kommer endast de vyer och de nivåer som anses tillföra något att användas. I vyerna kommer arkitekturen beskrivas på en avslappnad form av UML så långt detta är möjligt och relevant. Jag har valt att använda aktivitetsdiagram för att modellera flöden eftersom jag personligen tycker de är bättre än exempelvis sekvensdiagram på att förmedla information oberoende av teknik och implementationsdetaljer.

# Säljprocessen

[LARM AB] bedriver traditionellt försäljarbete för att bygga upp en bas av potentiella kunder. En stor del av företagets kunder har tillkommit genom säljarbete på säkerhetsmässor, reklam i media men även genom att kunden själv tagit kontakt med bolaget. Efter att en kontakt med en potentiell kund upprättats, lagras dennes kontaktuppgifter i [LARM AB]s CRM-system och ett nytt säljteam tar vid för att försöka boka in möten med kunden och en av företagets installationstekniker för att diskutera en skräddarsydd säkerhetslösning. Mötet äger rum i kundens hem (eller fastighet/lokaler i de då kunden inte är en privatperson) och målsättningen är att teknikern ska installera larmsystemet och slutföra försäljningen på plats.

Det faktiska säljet som [LARM AB] avser att implementera IT-stöd för inleds med att teknikern åker ut till kunden för att genomföra installation och konfiguration av säkerhetslösningen. Teknikerns bil fungerar som lager för det material som behövs för att utföra installationen och i de fall vissa komponenter saknas kan teknikern komplettera sitt lager genom ett lageruttag från en kollegas bil.

Väl på plats installerar teknikern en central larmenhet och till denna kopplas ett antal periferienheter (rörelsedetektorer, kameror, röksensorer) som tillsammans utgör själva larminstallationen. Den centrala enheten förses även ett så kallat PROM-nummer för att kunna identifieras unikt, och ett GSM-chip för att den ska kunna kommunicera med [LARM AB]s centrala larmövervakningssystem.

När enheten är installerad rapporterar teknikern in kund- och enhetuppgifter till [LARM AB] och larmutrustningen sätts sedan i ett testläge, där installation och konfiguration verifieras mot det centrala övervakningssystemet.

Efter att testet genomförts sammanställer teknikern materialåtgång och eventuella lageruttag i en arbetsorder som registreras hos [LARM AB]. Därefter erbjuds kunden teckna diverse tilläggstjänster via abonnemang och får slutligen signera ett kontrakt. Teknikern registrerar slutligen kontraktet tillsammans med en kundorder hos [LARM AB] och säljet ansers i och med detta vara genomfört.

## Begreppsmodell

Ur processbeskrivningen ovan kan ett antal begrepp och informationsflöden utläsas, *figur1* ordnar in dessa begrepp i en begreppsmodell för att beskriva hur de hänger samman och hur de relaterar till varandra. Slutligen ges en övergripande beskrivning över de system som är involverade i processen och vilka begrepp som berörs av dem.



Figur

**Test**Avser det test som verifierar larminstallationen och utgör kvittens för en Arbetsorders uppfyllnad.

**Enhet**En Enhet är den centrala enheten i en larminstallation som [LARM AB] förhåller sig till med avseende på övervakning. Enheten identifieras dels genom det inprogrammerade PROM-numret, men också genom ett enhetsnummer. Enhetsnumret kan vara aktivt eller inaktivt, till exempel om kundens avtalsperiod gått ut. Enheten associeras också med ett GSM-nummer för kommunikation med larmcentralen.

**Kund**Kunden är den primära kontakten i händelse av larm och även den person eller organisation som anses vara betalningsansvarig för både abonnemang och den GSM-trafik som säkerhetslösningen genererar. Kunden identifieras genom ett referensnummer eller ett kundnummer

**Kontrakt**Ett Kontrakt reglerar de avtal och abonnemang kunden tecknat med [LARM AB].

**Kundorder**Kundordern specificerar de tilläggstjänster en kund har valt för en viss larminstallation och utgör underlag till den faktura kunden sänds för teknikerns arbete under installationstillfället.

**Arbetsorder**Arbetsordern specificerar de olika periferienheter utöver Enhet som ingår i en installation. Denna utgör en del av slutfaktureringen till kunden efter installationstillfället samt är underlag för teknikerns lön.

**Lageruttag**Avser det uttag en tekniker gör direkt från [LARM AB]s eller kollegas lager.

**Periferienhet**Den extrautrustning som utgör en del av hela säkerhetslösningen

Dessa begrepp har vissa inbördes beroenden som illustreras i *figur*

**

Figur Beroenden

Från figuren kan vi utläsa att

* En **Enhet** är beroende av en Kund
* Ett **Test** är beroende av en Enhet och en Kund
* **Kontraktet** är beroende av ett Test, en Enhet, en Kundorder och en Kund
* **Kundorder** är beroende av en Kund
* Ett **Lageruttag** är beroende av en Arbetsorder
* **Arbetsordern** är beroende av en Enhet och ett Test

### Systemförteckning

Följande är en förteckning över de system som ingår i processen samt en beskrivning av de begrepp från modellen som systemen berör

**SAP**SAP utgör [LARM AB]s ERP-lösning och hanterar information om samtliga begrepp i modellen. SAP är det centrala systemet som det dagliga administrativa arbetet utgår ifrån i organisationen.

**Canonix**Larmövervakningssystemet är utvecklat internt och heter Canonix. Detta operativa system håller ordning på larmenheter och kontaktpersoner (typiskt Kund) i händelse av larm. Canonix är också det system som används för att testa och verifiera larmkonfigurationer. Systemet är en vital del i företagets leveranser och har därför nära noll neretid.

**Superoffice**[LARM AB]s CRM-lösning levereras genom Superoffice och det är genom detta system som tekniker erhåller information om potentiella kunder. Superoffice berörs inte direkt av tjänstelagret som ska utvecklas för säljprocessen, då det bara har en initial roll tills teknikern påbörjat arbetet ute hos kund. Däremot hänvisar referensnumret i Kundobjektet till motsvarande kundinformation i Superoffice.

**Handyman**Handyman är den mobilapplikation som används av tekniker för sälj- och installationsarbetet ute hos kunden. Det är denna applikation som kommer kommunicera med det tjänstelager som behandlas i denna uppsats. Handyman hanterar av förklarliga skäl information om samtliga begrepp i domänen.

# Problembeskrivning

I detta avsnitt redovisas de krav som organisationen ställt på IT-stödet samt de tekniska förutsättningar som har inflytande på arkitekturens förmåga att tillgodose dessa krav. Då uppsatsens fokus ligger på tjänstelagret mellan klienten och [LARM AB]s linjesystem kommer endast de krav med bäring på tjänstelagret att redovisas. Slutligen ställs kraven mot de tekniska förutsättningarna för att identifiera några problemområden som arkitekturen bör adressera i synnerhet.

## Om enheters identitet

Definitionen av det PROM-nummer som programmeras in i en Enhet ägs av [LARM AB], som själva utvecklat algoritmen för att generera det unika identifikationsnumret. Trots detta har Enhetsnummer blivit en de facto standard för identifiering av en larminstallation och det är detta nummer organisationen refererar till i kontrakt, kund- och arbetsorder och fakturor. Detta beror på att verksamhetens dagliga arbete sker i ERP-systemet, och det är detta system som äger denna definition.

Kopplingen mellan PROM-nummer och enhetsnummer upprätthålls i SAP och detta, tillsammans med status om huruvida enhetsnumret är aktivt eller inte, synkroniseras mot Canonix genom nattliga batchkörningar. På det viset kan Canonix koppla ett specifikt PROM-nummer mot ett motsvarande enhetsnummer samt inaktivera övervakning av enheter med inaktiva enhetsnummer.

SAP allokerar ett antal enhetsnummer i en pool och när en ny Enhet registreras reserveras ett enhetsnummer från denna pool. Om en kunds kontrakt av någon anledning avslutas, raderas dennes enhet från systemet och enhetsnumret som den associerats med återgår till SAPs interna pool igen. Det är dock inte ovanligt att enheten har inaktiverats innan kontraktet löper ut och när enhetsnumret återförs till poolen följer även aktiveringstatusen med. Detta medför att det finns en risk att nya enheter blir tilldelade ett inaktivt enhetsnummer vilket riskerar att inaktivera övervakningen av enheten om denna status hinner bli synkad med Canonix.

För att komma till bukt med detta har [LARM AB] ordnat så att SAP returnerar ett felmeddelande om en ny enhet lagras med ett inaktivt enhetsnummer. Enheten kommer kunna lagras, men inga andra entiteter (exempelvis Kund- och Arbetsorder) kommer kunna kopplas mot den så länge dess enhetsnummer är inaktivt. I och med att det är SAP som tilldelar enhetsnumret kommer detta fel inte gå att upptäcka innan det sker, någon måste försöka lagra enhetsinformationen innan felet uppdagas. SAP tillhandahåller inte heller något API för att aktivera enhetsnumret externt, utan personal måste istället manuellt aktivera enhetsnumret och lagra enheten igen.

## Krav på IT-stödet

I huvudsak ska tjänstelagret tillgodose klientens behov av följande funktioner:

* Kunna rapportera in larmkonfiguration i form av kund- och enhetsuppgifter för registrering i företagets ERP-lösning samt det centrala larmövervakningssystemet
* Kunna rapportera in säljunderlag i form avkontraktsinformation och materialåtgång till företagets ERP-system.

Problematiken kring återanvändning av enhetsnummer behöver också hanteras eftersom det har bäring på hela säljprocessen. I det fall en inrapporterad enhet blir tilldelad ett inaktivt enhetsnummer måste en administratör notifieras så att felet kan åtgärdas omgående.

Följande egenskapskrav har ställts på IT-stödet

* Återanvändbarhet  
  Funktionerna för att registrera kontrakt och materialåtgång måste gå att återanvända eftersom en del eftersälj och tekniskt underhåll av larmutrustningen kan komma att bli aktuellt efter att larmsystemet installerats.
* Möjlighet att utföra hela säljet med så få server-anrop som möjligt

Då mobilklienten är beroende av 3G-anslutning är det önskvärt att hålla nede datakommunikationen av både stabilitets- och kostnadsskäl

* Minimal klientpåverkan vid avbrott i linjesystem  
  Under normala kontorstider ska det vara möjligt att rapportera in säljet trots avbrott i ERP-system.

Funktioner för att initiera test och verifikation av larminstallationen ligger utanför tjänstelagret och exponeras av berört linjesystem direkt.

## Problemområden kondenserade

Genom problemanalysen kan vi härleda följande konkreta problem

* **Tjänsterna kräver tillstånd** för att koordinera mellan de tre operationerna
* **Generaliserade tjänster** krävs för att möjliggöra återanvändning. Tjänsterna för inrapportering av kontrakt/kundorder och arbetsorder/lageruttag måste hållas rena från processlogik relaterad till säljprocessen. Koordination måste läggas ovanpå dessa tjänster
* **Garanterat ”tjänsteavbrott”** i samband med återanvändning av enhetsnummer, något som Handyman måste skyddas ifrån.

Det är dessa utmaningar utöver de uttalade kraven som uppsatsen avser behandla i det kommande arkitekturavsnittet.

# Arkitekturansats

För att angripa problemställningen och beskriva en arkitektur som tillgodoser de krav [LARM AB] ställer på tjänstelagret har jag valt en ”utifrån och in” approach (top-down) där jag kommer utgå ifrån en samling användningsfall som i princip mappar 1:1 med de funktionella kraven. Användningsfallen kommer sedan brytas ned i mer detalj i efterkommande avsnitt, där arkitekturen även kommer stämmas av mot de egenskapskrav och problemställningar som beskrivits tidigare.

## övergripande arkitektur och Användningsfall

*Figur XX* illusterar hur tjänstelagret stödjer [LARM AB]s interna och externa Handyman-användare genom de två tjänsterna Säljprocess och ERP-fasad. Tanken är att användarna bara ska behöva förhålla sig till dessa tjänster utan att behöva känna till vilka system som ligger bakom.



Figur - Tjänstelager, konceptuell

Utifrån de funktionella kraven har två övergripande användningsfall identifierats:

* Registrera Larmkonfiguration
* Registrera Säljunderlag

Kravet på återanvändbarhet kräver dock att användningsfallet Registrera Säljunderlag bryts ned ytterligare för att exponera funktionaliteten för att registrera Kundorder och Arbetsorder. Figur 4 illustrerar alla identifierade användningsfall samt de beroenden som finns mellan dem.



Figur – Övergripande användningsfall

Användningsfallen har delats in logiskt i de två paketen Säljprocess och ERP-fasad, som motsvarar de tjänster som kommer utgöra tjänstelagret. Användningsfallen illustreras nedan och tjänsterna kommer beskrivas i mer detalj i kommande avsnitt

#### Aktivitet: Registrera Arbetsorder

*Figur XX*  illustrerar flödet för att Registrera en Arbetsorder. Flödet startar genom att en Arbetsorder läses in och valideras. Vid valideringsfel returneras en felkod och flödet avslutas, om inga fel påträffas lagras Arbetsordern och Lageruttag och flödet avslutas.



Figur - Användningsfall- Registrera Arbetsorder

Användningsfallet Registrera Kundorder har utelämnats avsiktligt då det i princip är identiskt med flödet ovan, bortsett från den information som användningsfallet hanterar.

#### Aktivitet: Registrera Larmkonfiguration

Användningsfallet inleds med att inkommande data valideras, vid valideringsfel returneras en felkod och flödet avslutas, annars fortsätter flödet med att PROM-nummer och kontaktuppgifter registreras i Canonix. Därefter lagras Kund och Enhet i SAP och flödet avslutas. Hela flödet illustreras i *Figur 6* nedan.



Figur – Användningsfall – Registrera Larmkonfiguration

#### Aktivitet: Lagra säljunderlag

*Figur 8* illustrerar användningsfallet Lagra säljunderlag, som inleds med att inkommande data valideras och eventuellt avslutas med felkod om något valideringsfel inträffar. Om data är giltigt säkerställs att aktuellt enhetsnummer är aktivt och sedan registreras Arbetsorder och Kundorder innan flödet avslutas



Figur – Användningsfall – Lagra Säljunderlag

## Logisk vy

*Figur ZZ* illustrerar den logiska strukturen för tjänstelagret i form av ett komponentdiagram. Jag har valt att dela in förekommande komponentdiagramm i tre lager:

* **Publikt gränssnitt –** i detta lager specificeras de publika gränssnitten som konsumenter kommunicerar över.
* **Middleware-**lagret grupperar implementationsdetaljer och infrastruktur för att leverera de tjänster som exponeras i det publika gränssnittet.
* **Backend** motsvarar de bakomliggande linjesystem som tjänstelagret kommunicerar med genom middleware. Detta lager finns med i modellerna som ren information och är inte signifikant i övrigt för arkitekturen som illustreras.

****

Figur - Tjänstelager, logisk struktur

Tjänsten Säljprocess exponerar all funktionalitet som implementerar säljprocessen och kommunicerar direkt med Canonix. Tjänsten har också tillgång till SAPs API:er. ERP-fasad utgör de generella och återanvändbara tjänsterna och kommunicerar via samma API:er mot SAP. Visst arbete delegeras även till ERP-fasad genom Säljprocess.

För att göra det möjligt för tjänstekonsumenter att rapportera in data till ERP oberoende av backend:ets driftstatus har båda tjänsterna tillgång till köhantering. Detta för att det ska vara möjligt för tjänsterna att kunna pausa operationer och lagra data när SAP har driftstopp, för att sedan kunna återuppta arbetet när SAP är uppe igen. För att isolera klienter från dessa potentiella driftstopp krävs asynkron kommunikation mellan tjänst och backend. *Figur xx* illustrerar hur kommunikationen bör ske; synkront mellan klient och tjänst, och asynkront via en kö och middleware mellan tjänst och backend.



Figur - Synkron klient, asynkron backend

## Utvecklingsvy

I det här avsnittet beskrivs hur tjänsterna bör implementeras för att möta de krav som ställts på IT-stödet. De egenskaper som är gemensamma för de båda tjänsterna kommer beskrivas på en generell nivå medan mer specifkika implementationsdetaljer kommer beskrivas separat för de båda tjänsterna.

**Asynkrona meddelanden och feltolerans**Den huvudsakliga utmaningen är att erbjuda klienter möjligheten att registrera Kontrakt och Arbetsorder även då SAP har driftstörningar, men detsamma gäller även för användare av Säljprocessen. För att skydda klienter från dessa driftstörningar krävs asynkron hantering av inkommande data vilket i sig innebär att potentiella fel måste upptäckas så tidigt som möjligt i anropet, medan klienten fortfarande finns tillgänglig för att ta emot eventuella felmeddelanden. Så snart tjänsten delegerat arbetet vidare och returnerat svar till klienten måste systemet garantera att all data når sin slutdestination, sk Guaranteed Delivery eller Reliable Messaging.

Den asynkrona kommunikationen bör alltså implementeras genom att en kö introduceras mellan tjänsten och den aktör som blir delegerad arbetet. Genom att utföra strikt datavalidering innan varje överlämning kan risken för fel längre ner i anropskedjan minskas. Eventuella valideringsfel rapporteras omedelbart tillbaka till anropande part och terminerar anropet. Detta ger konsumenten en chans att återhämta sig alternativt försöka på nytt med korrigerat data. Endast validerade data får passera ut ur tjänsten och placeras på kö för nästa aktör.

Detta förfarande gör också att de tjänster som klienter kommunicerar med i princip bara implementerar validering och vidaresändning ut till en kö, en så kallad Messaging Gateway eller Service Gateway. Den fakstiska logiken implementeras istället i en separat komponent som matas med data genom kön. Denna komponent läser data från kön och försöker skriva till relevant entitet i SAP, därefter tas data bort från kön. Vid driftstopp i SAP kommer operationen naturligtvis misslyckas, datat ligger då i kön och adaptern försöker sedan på nytt vid ett senare tillfälle.

Resonemanget ovan har ett tydligt fokus på data vilket gör den typiska databäraren i kommunikationen i tjänstelagret är ett dokumentbaserat meddelande.

**Köhantering**För att inte försätta tjänster i en evig loop till följd av misslyckade överföringar måste en köhanteringskomponent kunna skilja på ogiltiga meddelanden, så kallade poison messages, och återförsök vid eventuella driftstopp. De senare måste tillåtas, medan de ogilitiga/icke-hanterbara meddelandena måste läggas ut på en alternativ datakälla, tillsammans med eventuella felmeddelanden, för uppföljning av personal hos [LARM AB].

**Kommunikationsprotokoll**Det föreslagna kommunikationsprotokollet för tjänsterna i tjäsntelagret är Web Services/SOAP. Detta främst för att protokollet tillåter starkt typade interface, vilket underlättar valideringen som beskrivits i föregående punkt. SOAP är dessutom en mogen teknik som de flesta plattformar kan hantera. För att underlätta vidare integration och öka användningsytan kommer tjänsterna endast implementera det mest elementära från SOAP-standarden WS-I Basic Profile version 1.1. Senare SOAP standarder erbjuder en del ytterligare funktionalitet som kan vara relevanta för lösningen, exempelvis säkerhet och pålitlig meddelandehantering (reliable messaging). Att använda den äldre och mer rudimentära standarden är en godtagbar kompromiss då arkitekturen kommer hantera många av funktionerna som standarden saknar explicit.

Ett alternativ till SOAP kan vara POX över http, vilket hade gjort tjänstelagret mer lättkonsumerat. Eftersom tjänsterna är starkt beroende av datavalidering är det dock inte särskilt attraktivt då POX inte erbjuder något inbyggt stöd för starkt typade kontrakt.

**Säkerhet**Användarautentisering sker lokalt vid access mot linjesystem (Canonix och SAP) och hanteras av dessa direkt och i de respektive komponenter som exponerar systemen för tjänsterna. Tjänsterna i sig bör anropas säkert över HTTPS.

**Transaktioner**Canonix exponerar inga möjligheter för transaktionella skrivningar i vanlig mening, där det ges möjlighet för commit och/eller rollback även från klienter. Istället ses skrivningar som atomära operationer som utförs direkt vid anrop utifrån. Anropen kan antingen lyckas eller misslyckas och det ges ingen möjlighet för klienter att ångra eller avbryta en operation, klienterna måste istället förlita sig på kompenserande operationer. Den enda operation som tjänstelagret behöver är att lagra larmkonfiguration och den enda anledningen till att denna operation misslyckas är om Canonix är offline – vilket invaliderar hela säljprocessen då den är beroende av att larmutrustningen testas mot systemet.

SAP exponerar däremot transaktioner om så önskas och jag föreslår att dessa kapslas in i ett SAP-API som fungerar som adapter mot systemet. Funktionerna mot SAP exponeras då istället som atomära operationer som i fallet med Canonix. Tänkbara fel i operationer mot SAP beror antingen på att systemet är offline eller på grund av ogiltigt indata. Genom att validera inkommande data kan vi nästan utesluta ogilitigt indata, felet beror då istället på ändrade strukturer och format i SAP - i detta fall kommer vi förlita oss på hanteringen av poison messages.

### ERP-fasad

**Leverantörsagnostiska interface**För att skapa generella tjänster som går att konsumera utan djupare detaljkunskap om SAP bör tjänsterna inte läcka SAP-specifika implementationsdetaljer genom sina interface. För att dölja dessa implementationsdetaljer föreslås att meddelandekontrakten baseras på kanoniska datamodeller (Canonical Datamodels/Canonical schema) vars strukturer illustreras i *figur zz* nedan.



Figur ERP-fasad, meddelandestruktur

De kanoniska datamodellerna är de meddelandetyper som exponeras i tjänstens gränssnitt ut mot konsumenter, och för att matcha detta mot SAPs interna datastrukturer måste mappning ske innan data skickas vidare mot SAP. *Figur xx* illustrerar de komponenter som tillsammans utgör tjänsten ERP-fasad



Figur - ERP-fasad, komponenter

Det publika gränssnittet utgörs här av en enkel Service Gateway som är exponerad som en SOAP Web Service. Servicen tar emot och validerar datat innan det sänds ut på en kö avsedd för säljunderlag.

Vidare arbete utförs av en Affärslogik-tjänst som regelbundet, eller genom trigger, läser ut meddelanden ur kön med hjälp av en köhanteringskomponent. För att översätta meddelandet till SAP:s format används en mapping-komponent avsedd för det specifika meddelandet/entiteten.

De meddelanden som inte går att översätta eller av någon annan anledning inte går att lagra i SAP tas om hand av köhanteringskomponenten och läggs ut i en databas avsedd för poison messages.

### Säljprocess

**Denormaliserade datakontrakt**Eftersom det är önskvärt att begränsa trafiken mellan klient och server exponerar tjänsten endast två operationer där indataformatet består av denormaliserade data som utgör flera domänobjekt/entiteter. De diskreta entiteterna bryts sedan ut från meddelandet för vidare processering. *Figur xx* visar en konceptuell modell över hur meddelandena som används i gränssnittet är uppbyggda.

****

Larmkonfiguration och Säljunderlag utgör huvudkontrakten/meddelandena för operationerna *Registrera larmkonfiguration* och *Lagra säljdunderlag*. Larmkonfiguration slås isär till entiteterna Enhet och Kund innan dessa ska lagras i SAP. Detsamma gäller meddelandet Säljunderlag, som utgörs av de kanoniska datamodellerna som utgör gränssnittet i ERP-fasad. Det är i synnerhet denna operation som tjänar på denormaliserade kontrakt eftersom varje entitet kräver ett separat tjänsteanrop från Säljprocess till ERP-fasad men endast ett anrop från klienten.

**Tillståndshantering**Testningen av larminstallationen kräver att kund-/enhetsdata och arbetsorder/kontrakt skickas vid två olika tidpunkter. Eftersom både kontrakt och arbetsorder är beroende av att enhetsdata är registerat i SAP behöver tjänsten hantera det scenario där den inrapporterade enheten blivit tilldelat ett inaktivt enhetsnummer. I det här scenariot är det lockande att låta den tidigare nämnda kö- och poison message-hanteringen även ta hand om fel relaterade till inaktiva enhetsnummer. Ett av kraven på lösningen är dock att personal på [LARM AB] måste meddelas när detta problem uppstår så att denne kan åtgärda problemet omgående, vilket inte är nödvändigt eller önskvärt vid övriga fel.

Lösningen som föreslås innebär att operationen Registrera Larmkonfiguration lagrar enhetsnummer och status som specificerar om det är aktivt/inaktivt i en databas så att det finns tillgängliget i ett efterföljande anrop till Lagra säljunderlag. För att korrelera de två anropen mot varandra används kundens referensnummer, som också bör lagras tillsammans med enhetsnumret. När tjänsten tar emot säljunderlag stäms referensnumret av mot tillståndsdatabasen, om motsvarande enhetsnummer är inaktivt skickas en notifiering till [LARM AB]s personal och meddelandet läggs tillbaka i kön igen. Tillståndsdatabasen bevakas sedan för statusförändringar på enhetsnumret, exempelvis genom polling. Vid aktivt enhetsnummer skickas meddelandet sedan vidare till ERP-fasad. En illustration över flödet implementerad med polling finns i *Figur xx* nedan.



Figur - Aktivitet Säkerställ aktivt enhetsnummer

Statusförändringen som personalen utför i SAP måste på något vis propageras vidare till tillståndsdabasen för att processen automatiskt ska kunna återupptas. Eftersom SAP:s API:er inte gör det möjligt att programmatiskt kontrollera denna status måste vi förlita oss på personalen som hanterar ärendet. Lösningen som föreslås är att den notifikation som skickas ut även innehåller en hyperlänk som personalen klickar på för att bekräfta att statusen är uppdaterad. Länken uppdaterar aktuellt enhetsnummer i tillståndsdatabasen och processen kan återupptas igen.

De komponenter som utgör tjänsten Säljprocess illustreras i *figur dd* nedan



Figur - Säljprocess, komponenter

Stora delar av Säljprocess-tjänsten liknar ERP-fasad, med några skillnader främst i att affärslogiken även är beroende av en Notifikations- och en Tillståndshanteringskomponent. Dessutom integreras Säljprocessen även med Canonix.

En viktig detalj som är värd att belysa är att kommunikationen med ERP-fasad inte går via dess publika gränssnitt, utan istället skickas säljunderlaget direkt till ERP-fasads kö. Eftersom meddelandet redan validerats i Säljprocessens service gateway finns det ingen poäng att gå via det publika gränssnittet. Att kommunicera direkt via fasadens kö gör det dessutom enklare att distribuera båda tjänstegränssnitten oberoende av komponenterna i middleware-lagret.

## Fysisk vy

*Figur XX* nedan visar hur kan tänkas distribueras på fysiska noder. Jag har endast valt att rita ut de beroenden mellan komponenter som innebär kommunikation över server-gränser och i de fall jag tycker det är lämpligt.



Figur - Tjänstelager, deployment

Den föreslagna distributionen omfattar en webbserver som agerar front-end mot klienter, i denna körs de publika tjänstegränssnitten – Service Gateways – exponerade över HTTPS. Komponenter för affärslogik stödfunktioner placeras på en applikationsserver. Mellan dessa noder ligger ytterligare en nod som agerar lagringsserver och huserar de databaser och köer som används i tjänsterna. Kommunikationen mellan dessa servrar gör över TCP/IP.

Distributionen motiveras enligt kravet på att klienter måste skyddas mot driftstörningar i framför allt SAP. Genom att fysiskt separera tjänstelogik från linjesystemen är tjänstelagret rimligt isolerat från fel.

Enligt den föreslagna distributionen kan även tjänstelagret i applikationsservern vara offline utan att påverka klienterna. Så länge lagringsserver och webbserver är online kommer klienter kunna rapportera in data obehindrat och inget data kommer gå förlorat. När applikationsservern är online igen och alla tjänster är startade, kan arbetet återupptas tack vare att inkommande meddelanden kunnat buffras upp i köerna.

I förslaget innebär lagringsservern tyvärr en single point of failure, något som man dock kan motverka/lindra med redundanta servrar. Om lagringsservern av någon anledning skulle vara offline kommer en anropad gateway kunna rapportera denna status tillbaka till klienten. Klienten får i det fallet åtminstone en möjlighet att vidta åtgärder, som att cacha data och försöka sända igen vid ett senare tillfälle, alternativt meddela användaren om att manuell rapportering krävs. För [LARM AB] är detta en fullt godtagbar lösning då det redan finns en väl inarbetat process för manuell rapportering av sälj inom företaget idag.

# Slutsatser

Uppsatsens syfte var att identifiera antal problemområden inom tjänsteorienterad systemintegration och relatera dessa till de krav som ställts på IT-stödet i en säljprocess. Vidare skulle jag presentera en arkitektur som överbryggar dessa problem för att kunna tillgodose kraven i största möjliga mån.

Arkitekturförslaget bemöter problemen med väletablerade mönster och svarar väl upp mot de krav som ställts på den, men gör det genom att tillföra komplexitet. Å ena sidan erbjuder arkitekturen

* Näst intill totalt löst kopplade klienter
* Näst intill totalt isolerade klienter med avseende på driftstörningar
* Ett opakt tjänstelager som totalt kapslar in komplexiteten i de bakomliggande systemen och döljer denna för konsumenter
* Nya, generaliserade tjänstegränssnitt som går att återanvända i andra scenarior än de som legat till grund för uppsatsen

Å andra sidan kommer dessa möjligheter till kostnader i form av

* Ökad komplexitet med ett flertal abstraktionslager som gör lösningen svårare att förstå
* Ett ökat underhållsarbete och ett mer ansträngt utrullningsscenario som en följd av distribution

Även om arkitekturen föreslår distribuering av tjänsterna i Process- och ERP-lagret finns det inget tekniskt hinder att köra dessa in-process, om än endast för säljprocessen. För att göra de generella ERP-tjänsterna tillgängliga utanför säljprocessen krävs givetvis en viss distribution, men genom att inte distribuera dem i normalfallet skulle kommunikationen i Process-tjänsterna förenklas avsevärt och även underlätta utrullning och underhåll. Det skulle också ge den goda effekten att de båda tjänstelagrena tillåts evolvera oberoende av varandra (något som i och för sig skapar ett större behov av governance).

Arkitekturförslaget har legat till grund för den tekniska implementation som idag körs av [LARM AB]. Lösningen har baserats på Biztalk 2009 och Windows Communication Foundation som ryggrad och i och med detta teknikval har en del koncept som diskuterats i uppsatsen helt levererats out-of-the-box genom dessa produkter. Produkterna tillför självklart andra former av komplexitet och nya kostnader, men med en stor del av den tekniska plattformen redan utvecklad kunde [LARM AB] ta systemet i drift fort och börja göra de kostandsbesparingar man hoppats på.

# Diskussion

Det blir mer och mer tydligt att SOA handlar mycket mer om Process och mycket mindre om Teknik än vad jag som tekniker tidigare föreställt mig. Det är svårt att undgå att komplexiteten i den process som beskrivits i uppsatsen är artificiell och en direkt följd av ett verksamhetssystem som tagit för stor plats i det dagliga arbetet i organisationen. Som leverantör av skräddarsydda lösningar är det här en intressant observation eftersom detta tyder på att det fortfarande finns ett visst motstånd till att göra en genomlysning av sina egna processer, och börja förändringsarbetet där istället för att blint följa de arbetsformer som den befintliga IT:n påtvingar oss. Detta trots att SOA nuförtiden börjar ses som något moget och nästan alldagligt; och trots den aktuella organisationens egna uttalade önskemål att bli mer tjänsteorienterade.

Viss självkritik är också på sin plats. I retrospekt kan man konstatera att en organisation som saknar väl avgränsade och väldefinierade tjänsteområden och verksamhetsobjekt inte är mogen för composite services. De svårigheter implementationsteamet hade med att definiera tjänstegränssnitten och de arkitektoniska beslut som krävdes för att leverera stödet utgjorde sammantaget en ganska god grund att ifrågasätta dessa önskemål från organisationen.

Å andra sidan levererades ett IT-stöd som löste de problem man ansåg sig ha, på ett av beställaren godtagbart sätt. Så ett akademiskt rättfärdigande för arkitekturen är irrelevant, sett ur arkitektens vypunkt.

# Referenser

1. **Kruchten, Philippe.** *Architectural Blueprints - The 4+1 view model of software architecture.* u.o. : IEE Software, 1995. IEE Software 6(1995)
2. **Buschmann, Frank, Henney, Kevlin och Schmidt, Douglas C.** Facade. *Pattern-Oriented Software Architecture Vol. 5, On patterns and pattern languages.* Chichester : John Wiley, 2007.
3. **-**. Adapter. *Pattern-Oriented Software Architecture Vol. 5, On patterns and pattern languages.* Chichester : John Wiley, 2007
4. **Hohpe, Gregor, Woolf, Bobby.** Guaranteed delivery. *Enterprise Integration Patterns* [online] Hohpe, Woolf  
   http://www.eaipatterns.com/GuaranteedMessaging.html
5. **-.** Canonical Data Model. *Enterprise Integration Patterns* [online] Hohpe, Woolf  
   <http://www.eaipatterns.com/CanonicalDataModel.html>
6. **-.** Messaging Gateway. *Enterprise Integration Patterns* [online] Hohpe, Woolf  
   <http://www.eaipatterns.com/CanonicalDataModel.html>
7. **Asynchronous Queueing** [**http://soapatterns.org/patterns/asynchronous\_queuing**](http://soapatterns.org/patterns/asynchronous_queuing)
8. **Contract Denormalization** [**http://soapatterns.org/patterns/contract\_denormalization**](http://soapatterns.org/patterns/contract_denormalization)

# Bilagor

## Begreppslista

|  |  |
| --- | --- |
| Begrepp | Förklaring |
| POX | Plain Old Xml |
|  |  |
|  |  |
|  |  |
|  |  |