# Siebel Project Review

# Slutrapport för Rikspolisstyrelsen

Referensnummer

Version: 2013-04-29 Slutrapport

Författare:
Simon Mills
Gustaf Söderlund
John Gareth Roberts
Peter Pohl

# Innehållsförteckning

# Versionshistorik

| | | |
|---|---|---|
| Preliminärrapport | 2013-04-28 | IBM_SiebelProjectReview_preliminär_rapport |
| Slutrapport | 2013-04-29 | IBM_SiebelProjectReview_slutrapport_versionshantering_sv |

# Sammanfattning

## *Bakgrund*

2010/11, fattade Rikspolisstyrelsen (RPS)beslutet att implementera Siebel, dels så man kunde spendera mer pengar på nya projekt och mindre pengar på förvaltning, men också för att möjliggöra en flytt till en standardplattform som möjliggör funktionalitet att implementeras snabbare. RPS implementerar Siebel som sitt huvudsakliga ärendehanteringssystem för både ’kriminal-fall’ och civila fall. Siebel kommer ersätta en kombination av pappersbaserade lösningar samt terminalsystem och Java-system.

## *Mål med genomlysningen*

RPS har engagerat IBM för att genomlysa Siebel-programmet vad gäller följande fokusområden:

1. Möter programmet sina definierade mål?
2. Är programmets processer och strukturer etablerade och kommunicerade?
3. Är programmets ansvarsfördelning, gränser och beroenden förstådda?
4. Är resursmodellen adekvat vad gäller antal resurser och kompetens?
5. Är en lämplig arkitektur på plats och skalenligt anpassad?

## *Sammanfattning av observationer*

Kostnaden för Siebel-programmet är högre än förväntat, villket betyder att kostnadsbesparingar kommer ta längre tid att realisera än förväntat. En av huvudmålen med att implementera Siebel var att sänka utvecklingskostnader och förvaltningskostnader. Men, implementationen hos RPS har blivit anpassad till hög grad, vilket innebär att löpande kostnader kommer bli högre än om Siebel inte hade blivit anpassad. Detta behöver dock inte betyda att man inte kommer reducera kostnader, men att det kommer ta längre tid än planerat att uppnå dessa.

Styrningen av programmet är komplext, men kvalitetssäkringsprocesser – testning, ändringshantering, konfigurationsstandarder – har inte tillämpats hart. Resultatet blir en förvirrad rapporteringsstruktur, ett program som inte reagerar snabbt och sämre leveranskvalitet än tänkt vid projektets uppstart.

Kunskap om Siebel är väldigt begränsad på RPS. Det betyder att RPS är beroende av externa leverantörer för att implementera projekten. Det verkar också ifrågasatt om lämpligheten i att använda Siebel för att implementera den komplexa ärendehantering som RPS har.

## *Huvudsakliga rekommendationer*

Den här rapporten innehåller många förslag och rekommendationer. Om RPS bara kunde genomföra tre av dessa, borde de vara:

1. Återgå till ett standardiserat Siebel-system genom att ta bort kostsamma anpassningar
2. Strukturera om projekten som ett gemensamt Siebel-program, med en Siebel Design Authority-funktion med rådighet över lösningens kvalitetssäkring.
3. Förbättre Siebel-kompetensen hos RPS och kommunicera fördelarna med PÄR/PUST till nyckelpersoner och användare

Detaljer om hur dessa kan implementeras är preciserade i rapporten.

# Introduktion

## *Omfattning av IBMs genomlysning*

RPS har bett IBM att genomföra en analys af Siebelprogrammet. RPS implementerar Siebel Public Sector som deras huvudsakliga ärendehanteringssystem för 'kriminal-fall' och civila fall. Det här körs som två projekt: PÄR och PUST.

Analysen täcker 5 områden:

1. Möter programmet sina definierade mål? När RPS initierade Siebel-programmet dokumenterades 13 mål som behövde uppnås. Den första sektionen beskriver dessa objekt och utvärderar hur nära Siebelprogrammet är att realisera dessa.
2. Är programmets processer och strukturer etablerade och kommunicerade? Den här sektionen av rapporten jämför de processer och den styrmodell som används inom Siebelprogrammet och jämför den med de modeller vi typiskt skulle förväntat oss i hanteringen av ett program av den här typen med den här storleken.
3. Är programmets ansvarsfördelning, gränser och beroenden förstådda? Som ett resultat av det förra området, avgör den här sektionen hur väl programmens team arbetar tillsammans och hur programmet relaterar till andra RPS-team.
4. Är resursmodellen adekvat vad gäller antal resurser och kompetens? Ett program av den här typen är beroende av folk med specifika kompetenser och erfarenheter för att bli framgångsrikt. Framförallt är det nödvändigt att teamen har en god erfarenhet av implementation av Siebelprojekt.
5. Är en lämplig arkitektur på plats och skalenligt anpassad? Siebel är implementerad i en flerskalig arkitektur (databas, webb-server, applikationsserver) och opererar i komplexa miljöer med andra lager som nätverk, säkerhetsinfrastruktur och gränssnitt mot andra system. Den här sektionen ser på hur RPS implementation står sig jämfört med 'best practices' inom Siebel-implementation

## *Tillvägagångssätt*

IBM genomförde intervjuer med nyckelpersoner på RPS och hos de externa leverantörerna under December 2012 samt deltog i demonstrationer av PÄR och PUST under januari 2013 för att komplettera informationsinsamlingsfasen. Intervjuerna, tillsammans med ett antal document (som redovisats I appendix), innehöll frågor som möjliggjorde att ovan områden kunde analyseras. Slutligen, baserat på vår erfarenhet med andra Siebel-projekt, har vi dokumenterat våra observationer och gjort rekommendationer som RPS kan följa för att förbättra programmet. Även om en genomlysning normalt resulterar i både negativa och positiva observationer har IBM blivit ombedda att fokusera på förbättringsområden, vilket innebär att endast ett fåtal av de positiva observationerna presenteras här.

# 1. Möter programmet sina definierade mål?

## *1.1 Observationer*

Före RPS fattade beslutet att implementera Siebel som en standardplattform, genomfördes en förstudie (enl. Appendix Förstudierapport implementation av PUST i Siebel v1 17). Följande områden identifierades i denna förstudie som mål med Siebelprogrammet;

- Fullgod funktion kan erbjudas med “Standardplattformen Siebel”
- Avveckling av RAR och DUR kan ske I tid
- Lägre kostnader för utveckling och förvaltning
- Hög grad av återanvändning
- Sänkta ledtider för utveckling
- Utökade användarfunktioner
- Införandet hos andra myndigheter blir lättare
- Multi-kanal-stöd (surfplattor, smartphones och självbetjäning)
- Off-line-lösning
- Framtida säkerhetslösning som standard (enligt NATO-standard)
- Rik informationsmodell inkluderad
- Verktyg för gallring av data
- Stöd för dokumenthantering

I ekonomiska termer, beräknas SiebelPUST att kosta över 67 MSEK, jämfört med en implementationskostnad på 195 MSEK för en egenutvecklad applikation (JavaPUST inclusive planerade ytterligare faser). Men, det verkar råda oenighet om huruvida dessa uppskattade siffror (från förstudien) är korrekta och vi har fått ta del av olika siffror beroende på vem vi frågat. När det gäller kostnaden för att implementera Siebel, uppskattade en extern leverantör kostnaden för de två första faserna till 17MSEK (MAPS och Pust 2 fas 1). Utfallet förväntas nu landa på nästan dubbla den uppskattningen enligt en av de vi intervjuat.

Det andra målet med att ersätta RAR/DUR-systemen i tid är extra viktigt för PoA. PoA (och RPS) har fått till ett undantag för att fortsätta köra RAR och DUR-systemen till slutet av 2014, även om systemen inte är fullt godkända enligt lag. Regeringen har nämnt att inga ursäkter kommer godtas vad gäller att nå det målet i tid. Det är därför en stark drivkraft för RPS att undvika ett misslyckande gentemot uppdragsgivaren.

Vi har också tagit del av ett dokument gällande den generella IT-strategin, som skapades för perioden 2010-2015. När dokumentet skrevs (2010), spenderades 93% av systemkostnaden på förvaltning och 75 på modernisering och utvecklande av ny funktionalitet. IT-strategin fokuserar också på användandet av

breda standardplattformar och att erbjuda tjänster gentemot resten av organisationen genom hög grad av återanvändning. I tillägg till detta nämner IT-strategin att målet är att använda IT för att driva processen mot en modernisering av polisen.

I intervjuer har några nämnt att implementationen av Siebel I första hand är en organisatorisk förändring och ett nytt arbetssätt. En av de intervjuade såg till och med plattforms-implementationen som en möjlighet att tvinga fram organisatorisk förändring. Många av de intervjuade nämnde att, för att Siebel skall bli en framgång utifrån ett verksamhetsperspektiv, måste slutanvändarna använda och acceptera systemen och systemet måste ge ett bättre stöd för poliserna (ex ökad effektivitet) än de gamla systemen. Flera av de intervjuade har nämnt en oro att slutanvändarna kommer undvika att använda Siebel eftersom systemet uppfattas som användarovänligt och att en implementation av Siebel kan leda till lägre effektivitet, lägre uppklarandenivåer och ökad stress. Det finns också oro att poliserna kommer återgå till de gamla systemen (RAR och DUR). En av de intervjuade nämnde att eftersom Siebel inte kommer erbjudas för mobila klienter (eller i ’off-line’-läge) i det här skedet, kommer poliserna inte använda det, eftersom de tycker om att använda sina laptops.

Flera har uttryckt sin oro över avvikelse från en standardiserad Siebel-implementation. RPS har ett mål, som nämns i ovan förstudie, att använda standardfunktioner till 61% av implementationen och att använda 6% av enklare konfigurationer vid migrationen av JavaPUST till Siebel (MAPS). Flera av de intervjuade har nämnt att stadardiserad Siebel har använts i betydligt lägre omfattning, framförallt i SiebelPUST (se i sektionen Arkitektur).

’Ett-till-ett’-migrationen från JavaPUST till SiebelPUST innebär att samma funktionalitet som redan finns där på plats kommer implementeras i ett nytt system, men med ett års försening.

## *1.2 Analys*

För tillfället är RPS på väg att nå några av de mål som nämnts I förstudien (ovan), såsom att uppnå samma funktionalitet (som redan implementerats i JavaPUST) och reducera mängden utveckling jämfört med ett egenutvecklat system. Några av de nämnda målen kommer inte nås i den utsträckning som man förväntat sig, såsom att reducera förvaltningskostnader. Några mål har vi inte sett något bevis för att man är på väg att nå, såsom ‘off-line’-funktionalitet och att använda Siebel på smarta (mobile) klienter.

Det är viktigt att ha en korrekt och gemensam förståelse för siffrorna bakom besparingen i utvecklingskostnader, så jämförelser blir relevanta. Det här är speciellt viktigt i det här programmet eftersom sänkta utvecklingskostnader är en av de huvudsakliga målen med att implementera Siebel.

Generellt observerade vi många IT-relaterade mål med Siebel-programmet i förstudien ovan (återanvändning, snabbare införande och en standardplattform), men vi ser brister i verksamhetsrelaterade mål i förstudien (om dessa observerats i andra förstudier bör dessa tydligt kommuniceras även i denna). Större investeringar inom en plattform för kärnverksamheten är normalt rättfärdigad genom positiva verksamhetseffekter. Exempel på detta borde vara en högre grad av effektivitet på fältet (ex fler fall hanterade per polis), högre uppklarande-grad, snabbare handläggning av informationsbegäran etc. Vad gäller verksamhetsmål nämner denna förstudie endast att Siebel kommer ge en jämförbar användarupplevelse, jämförbar prestanda etc.

Lägre utvecklings- och förvaltningskostnader är oftast en av fördelarna med en standard-plattform. Det här förutsätter dock att mängden 'standard-innehåll' upprätthålls på en hög nivå. Men, RPS har redan avvikit från standard-plattformen (se komplexitetsberäkningen i Arkitektur). RPS har också avvikit från den gemensamma plattformen (Siebel Core) till två (och potentiellt tre) cores. Om man jämför med den existerande lösningen (egenutveckla hela ärendehanteringslösningen i Java), borde implementationen av en standardplattform leda till lägre utvecklings- och förvaltningskostnader, även om RPS flyttar till två eller tre cores. Det borde heller inte vara så svårt att sänka förvaltningskostnader från en nivå på 93% (av systemkostnader). Det är viktigt att ha både en korrekt och gemensam förståelse för siffrorna bakom kostnadsbesparingarna inom utvecklingskostnader, så man jämför äpplen med äpplen. Det här är extra viktigt eftersom sänkta utvecklingskostnader är en av de huvudsakliga målen med införandet av Siebel.

För att nå målet i att undvika ny utveckling genom återanvändande av standard-komponenter, krävs det till att börja med att standardkomponenter och plattformsfunktioner används. I SiebelPÄR verkar man ha uppnått detta än så länge, men i SiebelPUST verkar detta inte vara fallet (se Arkitektur). Men, om man jämför med JavaPUST, bör återanvändningen ökat även om man använt en låg grad av standardkomponenter och funktioner.

Målet med att minska ledtider för utveckling och samtidigt upprätthålla hög kvalitet kan nås, men bara om RPS implementerar standardiserad Siebel-funktionalitet och enklare konfigurationer istället för att använda sig av –scripting- och anpassningar, något som verkar vara fallet just nu.

Det verkar som att målet med att använda Siebel på andra klienter, som smartphones och laptops, liksom 'off-line'-funktionalitet, inte kommer inkluderas i nästa release av SiebelPUST. Vi har inte fått någon information om när detta kommer inkluderas i Siebelplattformen.

Om uppskattingen av implementationskostnader är felaktiga (som i exemplet med uppskattningen av kostnader för de två första PUST-faserna, behöver business caset förändras och 'ROI':n kommer försenas. Det här behöver man ta i beaktande. Men, om införandet av Siebel på 'fältet' resulterar i en ökad effektivitet (även med en lågt procenttal) måste detta också tas i beaktande och inkluderas i 'business caset'.

## *1.3 Rekommendationer*

Gå igenom förstudien ovan (samt eventuella andra förstudier) och säkerställ att verksamhetsvärdena med att införa Siebel som en standardplattform är betonade, gärna tillsammans med verksamheten. En liten effektivitetsökning (eller uppklarandegrad) för poliserna/handläggarna kommer resultera I ett kraftigt förbättrat business case. Gå även igenom uppskattningen av implementationskostnaden, om nödvändigt, för att säkerställa att den nuvarande situationen/det nuvarande utfallet reflekteras.

Eftersom Siebel-projekten har så högt fokus (profil), kunde det vara fördelaktigt att följa utvecklingen vad gäller uppfyllnad av målen, när dessa är enade med verksamheten. Förutom att konfirmera att man är på rätt spår, är avsikten att skapa en förståelse inom hela organisationen för varför man gör det här och att projekten framskrider.

Säkerställ att Siebels standardpaket används i högre grad. Genom anpassning minskar fördelarna med en standardplattform, vad gäller att sänka ledtid och minska förvaltnings- och utvecklingskostnader. RPS behöver, om möjligt, återgå till en core. RPS behöver också sätta upp en gedigen kvalitetssäkringsprocess för att upptäcka och om möjligt undvika all framtida avvikelse från standardiserad Siebel. Det här ämnet beskrivs i mer detalj i sektionen Arkitektur.

RPS behöver förstå kostnaden involverad med att ha flera cores och flera instanser i produktion och effekten det har på förvaltning och återanvändning. Baserat på den informationen måste RPS fatta beslut on hur man skall gå vidare (eg är det nödvändigt att ha flera cores och till vilken kostnad?). Det här är ytterligare detaljerat under sektionen Arkitektur.

Den ökade funktionaliteten som nämns I förstudien, såsom möjligheten att använda Siebel på ex Smartphone eller ’off-line’, kommer bara vara värdefull om användarna kommer använda SiebelPUST och SiebelPÄR, men också om användarna får använda de beskrivna funktionerna.

# 2. Är programmets struktur och processer etablerade och kommunicerade?

## *2.1 Observationer*

### Kravfångst

Generellt inom RPS verkar inte kravfångstprocessen vara speciellt strict. Många av de intervjuade nämnde att de ursprungliga kraven är ’fluffiga’ (inte specifika, svåra att mäta och implementera) och att nya krav ständigt dyker upp och förändras kontinuerligt. Det är inte klart för oss huruvida det här beror på brist på tydlighet i processen eller att exekveringen av denna process i dessa projekt. Projekten önskar tydligare krav från verksamheten, men också tydligare säkerhets/CSL-krav. Säkerhetskrav i synnerhet verkar vara ett allvarligt problem ör projekten. Säkerhet ser en utmaning i att utvecklarna inte förstår säkerhetskraven, medan projekten ser en sen aceptans av kravuppfyllnad från Säkerhet som ett stort problem. En av de intervjuade nämnde att ”CSL gör bara acceptans av kraven i testfasen – som kan leda till rena designförändringar.” Som ett resultat av det här gapet inom kravdefinitionen och testningen mellan Säkerhet och projekten, ser projekten en risk i förseningar av GoLive och Säkerhet ser att loggnings-kostnaden går upp, eftersom utvecklarna ”utvecklar för mycket”. En av de intervjuade beskriver problemet såhär: ”Säkerhet har veto för allting som går ’live’, men de berättare inte för oss hur de vill ha det. De säger bara ’logga allting’.”

Det finns också en oro för att kraven inte hanteras i tid. En av de intervjuade nämnde: ”I mars (2012) dokumenterade vi några krav som inte lästes förrän oktober/november.”

Användarkrav och acceptanstester för användarna är potentiellt den största utmaningen vad gäller krav, speciellt vad gäller SiebelPUST. Även om projektet nämner att de haft viss kontakt med verksamheten (PoA) och gruppen som fungerar som referensgrupp för verksamheten – Metod och verksamhetsgruppen (MVG) – verkar det uppstått missar i kommunikationen. Från ett PoA-perspektiv har användarna involverats endast i liten omfattning och hade förväntat sig från SiebelPUST-projektet att involvera MVG i betydligt högre utsträckning. Det här resulterar i svårigheter med att få acceptans från slutanvändarna för SiebelPUST. Nya krav dyker upp i acceptans-testerna (där sex poliser vad involverade) och det här leder till förseningar i projektet (samt en risk att poliserna inte kommer använda systemet om inte kraven är implementerade). Från ett SiebelPÄR-perspektiv verkar det här vara ett betydligt mindre problem, där representanter från verksamheten har blivit involverade tidigare, något som lett till en ganska positiv uppfattning av systemet.

### Ändringshantering

Det finns olika syner på hur ändringshanteringen fungerar. En av de intervjuade nämnde att “under förra veckan fick vi ett ändringsförslag på måndagen, två på tisdagen, en på onsdagen och två på torsdagen. Vi har inte sett någon ändringshantering på plats. I PÄR-projektet, etablerades en ändringshanterings-funktion (Change Control Board - CCB) i augusti och hålls veckovis. Output från HPs Quality Centre genererar input till CCB. I Siebel Core-programmet skapades ett CCB från början. Med SiebelPUST verkar situationen varit annorlunda,

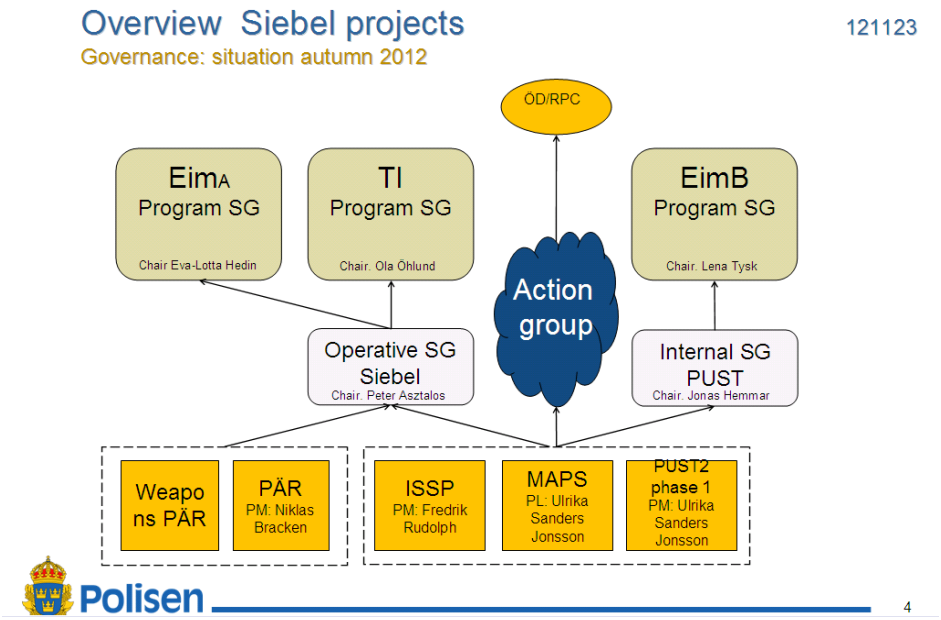
eftersom inget CCB etablerades förrän oktober 2012. En av de intervjuade nämnde: ”PoA lägger in massor av ändringsförslag (40-60 hittills) och Accenture hanterar inte ändringarna.” Många av de intervjuade delar synen att PoA gör för många ändringsförslag. Många av de vi intervjuade har också varit med om s k ’scope creep’ (ungefär att omfattningen på projektet gradvis ökar, ofta okontrollerat) inom PUST-projektet. Actiongruppens (senare PUST styrgrupp) första fokus var att ta kontroll över ändringsförslagen och minska omfattningen.

## Kvalitetssäkring

Det verkar inte finnas en särskilt tydlig koppling mellan Siebel-programmets mål och de respektive projekten (även om vissa av dessa mål kommer uppnås). Vi har inte observerat någon separat funktion som arbetar med att säkerställa att de huvudsakliga målen uppnås (såsom att sänka utvecklings-/förvaltningskostnaden, öka återanvändning av kod eller införa ’off-line’-funktionalitet), även om testning görs i projekten (såsom systemtester, integrationstester, acceptanstester etc). Men i balansen mellan tid och kvalitet, verkar tid ha fått ett starkare fokus. Det här verkar vara särskilt tydligt i SiebelPUST som har genomgått hög (potentiellt orealistisk) tidspress. SiebelPUST-projektet har tvingats genomföra omfattande prioriteringar för att hinna möta deadline. En av de intervjuade sa: ”Vi rundar hörnen när det gäller att involvera slutanvändare i processen och att samla in krav från verksamhetsfolk, såväl som juridisk kompetens och användbarhetskunskap”. Resultatet enligt flera av de intervjuade kan bli att Siebel inte möter de uppsatta verksamhetskraven.

## Struktur (Programhantering och styrgrupper)

Styrmodellen mellan Siebel-projekten är komplex. Flera av de intervjuade nämnde att det var svårt att förstå strukturen. Några av de intervjuade hade starka synpunkter om svårigheterna att leda projekt på RPS. I en presentation om programstrukturen som givits till oss, bestod de tre programmen av fem projekt (se bilden) och sex styrgrupper (inclusive Action-gruppen som rapporterar till Rikspolischefen). I olika intervjuer har vi hört om upp till nio styrgrupper. En av projektledarna nämnde att han/hon rapporterade till tre styrgrupper (inget gemensamt forum), att det finns överlappning mellan de olika styrgrupperna och att det finns en tydlig utmaning vad gäller mandat i styrgrupperna. De andra projektledarna har indikerat liknande utmaningar. Många av de intervjuade tycker att Action-gruppen har gjort programhantering enklare och tydligare. Action-gruppen har också det nödvändiga mandatet för att fatta de beslut som krävs enligt flera av de intervjuade.




Som nämnts tidigare är Siebel-projekten indelade I tre olika program med olika ägare. Även om vi har förstått att TI-programmet (se ovan) inte kommer innehålla några Siebel-projekt efter MAPS är avslutat (förväntad GoLive 18:e februari), kommer det fortfarande vara två program kvar med två programägare från två separata avdelningar inom RPS (RA och PoA). En av de intervjuade nämnde att det kunde vara en fördel att ha allting under ett paraply och Eva-Lotta Hedin, RAs avdelningschef, såg inga problem med att göra något gemensamt med PoA vad gäller Siebel-program.

### Riskhantering och riskmitigering

Synen på riskahentering är också splittrad. Några av de intervjuade anser att riskhanteringen fungerar väl, framförallt vad gäller processen. Andra anser att projektledarna tenderar att vara dåliga på både att ta upp risker, men också att komma med mitigeringsförslag (förslag hur man bör hantera riskerna). De flesta riskerna förväntas mitigeras inom projektet och den första frågan när en risk tas upp är normalt vad projektledaren föreslår att man gör för att lösa problemet (mitigeringsförslag). Med de överlappande styrgrupperna verkar det uppstå förvirring vad gäller vilken styrgrupp som har auktoritet att bestämma vad gäller mitigering av vilka risker. En av projektledarna nämnde att risker som tas upp väldigt sällan hanteras/mitigeras i styrgupperna, men att hanteringen/mitigeringen har fungerat bättre i Actiongruppen. Lena Tysk, vice avdelningschef på PoA, nämnde, när vi pratade om utmaningarna inom SiebelPUST-projektet, att de valde att eskalera det här till ledningsgruppen, vilket ”ledde till att Actiongruppen bildades, något som skulle varit på plats från starten. Vi skulle ha dragit i bromsen betydligt tidigare.”

## *2.2 Analys*

Om kraven är otydliga eller fångas sent I processen finns det en stor möjlighet att det kommer uppstå många ändringsförslag, något som significant driver upp utvecklingskostnaden och risk finns att implementationen försenas. Det här verkar vara ett problem framförallt vad gäller säkerhetskraven på RPS. Om

omfattande förändringar på kraven kommer in i testfasen, tvingas projektet gå tillbaka till designfasen och det leder till en stor risk för försening i projektet, eftersom Säkerhet kan begära att krav implementeras före projektet går live. Eftersom det antingen är ett problem med hur säkerhetsgruppen formulerar krav eller hur utvecklare förstår säkerhetskraven (eller både och), innebär säkerhetskraven en hög risk för projekten.

En hög mängd ändringsförslag signalerar att de initiala kraven inte fångats eller dokumenterats ordentligt eller att omfattningen är oklar. Det indikerar också att organisationen som föreslår/begär förändringar, antingen inte varit involverade i tillräckligt hög grad i kravfasen, eller saknar den nödvändiga kompetensen om hur man fångar krav. Vad som verkar ha hänt är att ändringsförslag dyker upp sent i projektet när verksamheten blivit mer involverad och när användarna har börjat se systemen, då de förstår att Siebel inte kommer fungera på samma sätt som deras gamla system fungerar. Den stora mängden ändringsförslag verkar vara en större utmaning i SiebelPUST än i SiebelPÄR. Ett utvecklingsprojekt som tillfälligt saknar en ändringshanteringsfunktion (CCB), såsom i SiebelPUST, tappar man snabbt greppet om, något som leder till s k scope creep, ökad utvecklingstid, missade deadlines och högre kostnader.

Kvalitetssäkring är nödvändigt för att säkerställa att utvecklingsprojektet möter de huvudsakliga mål som satts upp, samt de uppsatta kraven. Testning sker, men fokus verkar vara på att möta de funktionella kraven istället för att möte målen med Siebel-programmet. Exempelvis, faktumet att det har skett en stark avvikelse från standard Siebel i SiebelPUST-projektet (se Arkitektur), med flera cores och instanser, kommer leda till en högre utvecklings- och förvaltningskostnad och reducera mängden återanvändbar kod i projekten. Testning sker också inom projektteamen och utvalda nyckelanvändare, snarare än ett dedikerat test-team.

Om programstrukturen är komplex, finns det en risk att man över-rapporterar (rapporterar samma sak till flera styrgrupper) eller en risk att saker faller mellan stolarna. Det senare alternativet är ett särskilt problem på RPS. Exempelvis tas risker upp som inte mitigeras och beslut tas utan att nödvändiga personer är informerade. Ett problem med bristande mandat i styrgrupperna kan resultera i att problem inte hanteras och att projektet därmed löper risk att bli försenat. Det verkar som att många av problemen med mandat och överlappning av styrgrupper har lösts iochmed inrättandet av Actiongruppen. Men, Actiongruppen har inte ersatt någon existerande styrgrupp, utan blivit en till. Faktumet att Actiongruppen fungerar (till skillnad från andra) kan vara ett tecken på tre saker: 1) Det är extremt viktigt att styrgrupper har mandat (Actiongruppen rapporterar direkt till Rikspolischefen), 2) Det finns ett behov att koordinera insatser mellan alla Siebel-projekt och 3) Det finns ett gap mellan operativ, taktisk och strategisk nivå som verkar överbryggats i denna grupp. Även om framsteg har gjorts i Actiongruppen (exempelvis vad gäller ändringshantering och scope creep), är det här en reaktiv lösning på problemen och visar tendenser på detaljstyrning, vilket både är tidsintensivt och innebär att arbete blir dubblerat.

Risker tenderar att inte tas upp direkt när de identifierats. Det verkar som det finns svårigheter i att förstå till vilken styrgrupp vilka risker skall lyftas upp eller ens hur risker hanteras på RPS. Riskmitigeringsprocessen verkar vara otydlig på RPS. Om riskerna inte tas upp direkt, eller inte tas upp alls, kan de bli svårare att mitigera. Det här kan leda till projektförseningar eller i förlängningen ett misslyckat projekt. Samma problem kan uppstå om riskerna tas upp, men inte mitigeras.

## 2.3 Rekommendationer

Framöver måste mer tid spenderas på kravfångst och att specificera så många av kraven som möjligt för att sedan ‘frysa’ kraven. Alla nya krav hanteras därefter som ändringsförslag. Om möjligt borde krav också reflektera beroenden till andra projekt/program. Verksamheten (främst PoA) och dess användare måste bli betydligt mer involverade i kravfasen, så att så stor del som möjligt av kraven kan fångas initialt istället för att behöva hantera dem som ändringsförslag. Det här kan resultera i projekt-tröghet, eftersom kravfångst-fasen aldrig tycks bli färdig. Det här kan hanteras genom att begränsa kravfångsfasen till ett bestämt tidsspann (kanske 3-4 veckor) i början av varje fas. Om det finns utmaningar med kravfångst eftersom omfattningen är för stor, bör utvecklingen delas upp i mindre releaser.

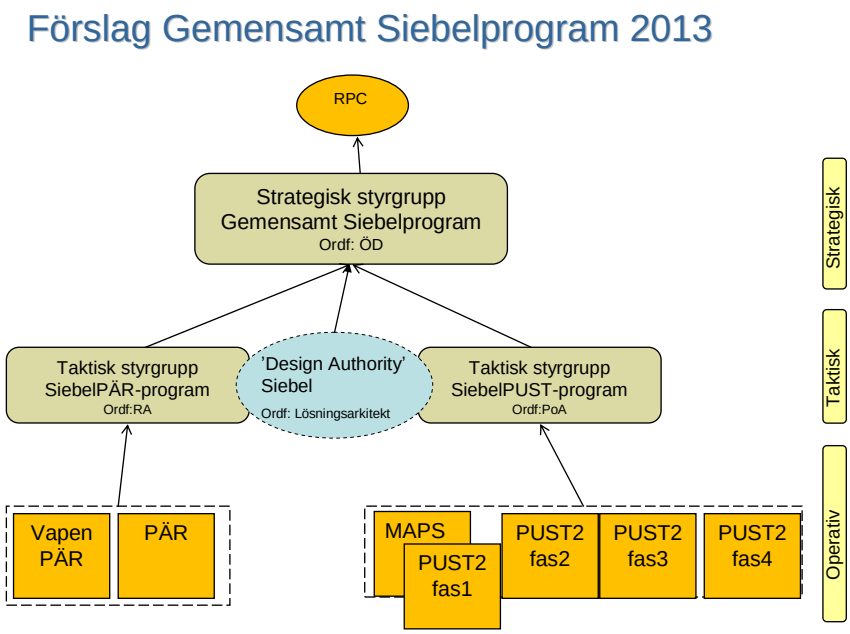
Säkerhetskraven för CSL behöver också dokumenteras och låsas i en tidig fas. Kommunikationsgapet (eller kompetensgapet) mellan Säkerhet och utvecklare behöver överbryggas. Om det verkligen är ett kompetensgap och inte ett kommunikationsgap måste Säkerhet bygga Siebel-kompetens eller Siebel-utvecklarna bygga säkerhets/CSL-kompetens, lämpligtvis bägge. Ett alternativt tillvägagångssätt vad gäller att implementera säkerhetskraven bör också övervägas, något som är beskrivet i Arkitektur-sektionen.

Alla krav som identifieras efter kraven har frysts skall hanteras som ändringsförslag vid ett ändringshanteringsmöte (CCB) från början i projektet. Vid ett CCB, värderas alla ändringsförslag och prioriteras efter huruvida de är nödvändiga för att få ett leveransgodkännande eller inte. Om verksamheten (inklusive slutanvändarna) och Säkerhet varit involverad i kravfasen och godkänt kravlistan, borde det vara få ändringsförslag att ta upp vid CCB-möten.Det borde resultera i lägre mängd ’scope creep’ och reducerad utvecklingstid. Krav bör alltid valideras gentemot programmets syfte för att säkerställa att de möter projektens mål, exempelvis sänka kostnader, sänka komplexitet, öka effektivitet etc.

Om man inte redan har gjort det, rekommenderar vi att RPS sätter upp en kvalitetssäkrings-funktion som hanterar att projektet håller sig till överrenskommen omfattning och når målen. Den här funktionen kunde troligtvis fungera inom Actiongruppen, även om det här teamet även skulle behöva Siebelkompetens, något som är beskrivet i mer detalj i Resurs-sektionen. Kvalitetssäkringsfunktionenborde också utökas till att includera ett oberoende test-team, så utvecklingsteamen kan fria upp resurser för att designa och utveckla, med ett dedikerat team ansvariga för end-to-end-testning. Det här har även tagits upp i sektionen Resurser.

Vi rekommenderar en mindre complex program-struktur för att säkerställa tydlighet och reducera mängden gränser och överlappning. Program och styrgrupper skulle kunna vara separerade för SiebelPUSt och SiebelPÄR på en operativ och taktiskt nivå, men på den taktiska nivån bör man även inrätta en design/arkitektur-koordinering genom en s k ’Design Authority’-funktion. På strategisk nivå bör det vara en gemensam styrgrupp och ett gemensamt program för SiebelPUST och SiebelPÄR. Det här är för att undvika att programmen rör sig i olika riktningar. Varje styrgrupp (eller styrgruppen) skall ha det relevanta mandatet. Varje grupp skall bestå av rätt folk (representanter för relevanta avdelningar). Alla problem som inte hanteras på operativ nivå skall lyftas upp till taktisk nivå och så vidare. Anledningen till att man delar upp i strategisk, taktisk och operativ nivå är för att säkersätlla att rätt frågor diskuteras på rätt nivå och för att undvika detaljstyrning. För återkommande problem skall fokus inte bara vara på att lösa problemet, utan att finna roten till varför problemet uppstod. Det här skulle innebära en tydlighet i strukturen, begränsning av överlappningen av

styrgrupper och rapportering samt undvika att uppgifter faller mellan stolarna. Det här är ett förslag på bild över hur Siebel-programstrukturen skulle kunna se ut på RPS:




Risker bör tas upp så fort de upptäckts. Desto fortare risker rapporteras, desto enklare är de att mitigera. Projektledaren/Programledaren skall rapportera risken och komma med förslag på beslut som kan mitigera risken. Om risken identifieras som en taktisk eller strategisk risk, eskaleras den snarast möjligt upp till rätt styrgrupp (motsvarande rätt nivå).

# 3. Är programmets ansvarsfördelning, gränser och beroenden förstådda

## Generella observationer

Många av de intervjuade är överens om att roller och ansvarsfördelning är otydliga inom RPS. Det verkar finnas situationer där överlämning inte görs på rätt sätt och antingen går folk overksamma i avvaktan på att ett beslut fattas eller så bestämmer folk själva utan att koordinera / samordna med andra. Många av de tillfrågade nämner styrning i synnerhet som ett komplext område där RPS har varit mindre framgångsrik. ”Siebel-programmet handlar om en stor organisatorisk förändring i en organisation som inte är van vid organisatorisk förändring” som en av de intervjuade sa.

## Samordning och beroenden mellan programmen

Som nämnts tidigare finns det minst fem projekt och tre program i Siebel-projekten. För SiebelPUST-projekten är detta ett särskilt svårt problem eftersom det finns två programägare som representerar två separata divisioner (PoA och ITS). Dessa två ägare har mycket olika behov och olika primära fokus. ITS kommer sluta som programägare men kommer fortsatt äga Siebel Core-plattformen där både PUST och PÄR byggs.

Beroenden mellan PUST och PÄR är främst när det gäller instanser och cores. Eftersom både PUST och PÄR förväntas använda Siebel Core-plattformen måste alla förändringar av Siebel Core samt buggar som rör Siebel Core samordnas mellan projekten. En av de intervjuade nämnde "synkronisering av releaser, beroenden och tidsplaner gör det svårt att ha en kombinerad instans." När ett projekt hittar en bugg rapporteras den till andra projekt. Bortsett från detta finns det endast informell kommunikation mellan de två projekten. En av de externa leverantörerna vill också ta deras utveckling off-shore vilket skulle kunna ha en effekt på samordning. Det finns ingen lösningsarkitekt (eller end-to-end arkitekt) för hela Siebel och det verkar inte finnas någon Design Authority (se Arkitektur). PUST och PÄR-projekten verkar använda olika utvecklingsmetoder (vattenfall jämfört med agilt).

Det finns även beroenden mellan Siebel-projekten och andra IT-relaterade projekt som pågår i RPS såsom NKP (ny klientplattform), INÄS (nätverk) och MOPC (mobil plattform). Siebel är det första programmet på RPS att använda all den nya infrastrukturen. Ett exempel är att när ett Siebel-projekt berörs av ett IT-problem, såsom att MQ Broker inte är säkerhetscertifierat, blir det ett problem för Siebel-projektet.

## Förhållande, gränser och beroenden gentemot externa leverantörer

De flesta av de intervjuade verkar överens om att RPS är i händerna på externa leverantörer. Tre systemintegratörer är djupt involverade i projekten (Accenture, Deloitte och CGI) och Siebel-standardplattformen tillhandahålls av Oracle. Avtalen med systemintegratörer sker främst på ’löpande räkning’ och hanteras genom PVS. Leverantörerna förväntas generellt att göra bästa möjliga uppskattningar av kostnader för införande och rapportera uppåt i programmets ledningsstruktur. RPS syn på externa leverantörer är generellt positiva även om språket ibland kommer upp som en utmaning (många av konsulterna använder

engelska som sitt arbetsspråk). Men, många av de intervjuade har uttryckt oro över Siebel-kunskapsnivån för Accenture-konsulterna (se avsnittet om kompetens / resurser). CGI och Deloitte verkar dela utvecklingen av PÄR och tävlar i någon mening mot varandra. När det kommer till Oracle hade flera intervjupersoner förväntat sig ett starkare stöd från dem. Särskilt nämner en av de intervjuade att de ville se Oracles engagemang i genomförandet av det officiella Siebel polis-paketet (Law Enforcement). "Vi försökte få Oracle att bygga detta som en del av standardfunktioner i Siebel. Ingen reaktion från dem hittills. Jag är mycket besviken." RPS har också beslutat att minska stödet från Oracle främst på grund av höga konsultarvoden.

## Ansvar och gränser mellan avdelningarna i RPS – Beställar/Utförar-organisation

Många av de intervjuade beskriver dåligt samarbete mellan de olika delarna av RPS, särskilt i Siebelprojektet. Vissa går ännu längre och beskriver det som interna politiska strider mellan divisionerna och personliga korståg. Det verkar som att beslutet att gå vidare med Siebel har varit dåligt förankrat och kommunicerat inom organisationerna. Flera personer tycks ha känt att de var utanför diskussionerna och reagerade defensivt. RPS beställar /utförar-modell har beskrivits för oss som följer: verksamheten (PoA och RA i Siebel-projekt) fungerar som en kravställar-organisation mot IT (i det här fallet både ITS och PVS), PoA / RA har versamhetsbehov som översätts till verksamhetskrav av ITS som sedan ber PVS göra en kostnadsuppskattning på att exempelvis utveckla ett system. PVS svarar med en uppskattning. ITS ber sedan PVS att börja utvecklingen för den beräknade kostnaden med PVS som leverantör av IT (och mycket annat). ITS å andra sidan håller budget för IT-investeringar vilket innebär att om PVS går över budget måste de komma tillbaka till ITS och be om mer pengar.

Enligt flera av de intervjuade är detta inte en ovanlig situation. PVS är ansvariga för att ge uppdrag till externa leverantörer och håller avtalen med dessa leverantörer. ITS roll beskrivs som ledande, styrande och uppföljande av IT (och hantering gapen mellan vad som behövs och vad som finns där).

I Siebel-projekten är RA de som kravställer PÄR och PoA är de som kravställer PUST. Men i projektet som ansvarar för förflyttningen av JavaPUST-funktionalitet till Siebel (MAPS) fann sig ITS plötsligt sig vara en av de som kravställer (eftersom det sågs som ett rent IT-projekt som inte hade något att göra med verksamheten). "PoA ville inte äga MAPS - det drevs hårt av PoA - vilket var ett stort misstag. ITS fick ansvaret i stället. "

De flesta intervjuade är överens om att det finns ett styrningsproblem i RPS, delvis som en följd av denna komplexa uppställning/struktur. Men beroende på vem vi har intervjuat finns det olika svar om var problemet ligger. Kritik mot verksamheten (PoA i synnerhet) är att de inte tar ansvar för Siebel-migrationsprojektet, att de inte är lika engagerade som de borde vara (särskilt inte i början) och att de måste vara mer aktiva och bli bättre på att sälja Siebels fördelar till resten av verksamheten (t.ex. poliser). Kritiken mot ITS är att de har fattat ett beslut att gå med Siebel ("trots att ingen ville") utan att lyssna till resten av organisationen och att de är inblandade i detaljstyrning av projekten. Kritik mot PVS är att de alltid har en tendens att överskrida budgeten, att de inte är proaktiva, att de har gjort ett dåligt jobb att upphandla resurser för projektet och att de fortfarande inte har byggt upp egen kompetens på Siebel (varken i utveckling eller förvaltning). Som en av de intervjuade från ITS nämnt: "Vi har börjat kräva leveransansvar från en organisation som aldrig har behövt hantera detta ansvar innan. PVS ledarskap är ganska reaktivt. "

I vår sista intervju i informationsinsamlingsprocessen nämndes det att RPS är på väg att ändra i beställar/utförar-modellen genom att ge en del av IT-budgeten till verksamheten (PoA / RA) i stället för ITS. ITS å andra sidan får sätta den IT-strategiska planen och övervaka att den följs.

### Kommunikation

De flesta av de intervjuade påpekar bristen på kommunikation mellan och inom divisionerna. En av de intervjuade nämnde: "Vi har brist på kommunikation överallt - mellan PoA och RA, mellan programansvariga och mellan projekten". En annan uttryckte det så här: "Det är inte bara kommunikationsproblem inom PVS, det är också mellan ITS och PoA. Även de externa konsulterna har dålig kommunikation mellan varandra". Även om inte alla intervjuade känner lika starkt som detta sammanfattar de alla att kommunikationen, på ett eller annat sätt, måste förbättras. Misstro nämns som en av anledningarna till denna brist på kommunikation, speciellt mellan ITS och PVS. Dessutom kan det faktum att många av de intervjuade inte känner sig delaktiga i beslutet spela en roll. Arkitekter har nämnt att de inte var inblandade, säkerhetsgruppen nämnde de inte var inblandade och personer i verksamheten nämnde att de inte var inblandade.

En annan utmaning är kommunikationen mellan strategisk, taktisk och operativ nivå. Många av de intervjuade nämnde att informationen i högsta ledningen inte verkar filtrera ner till de lägre nivåerna.

Som ett resultat av dålig kommunikation i Siebel-projektet är kunskapsöverföringen lägre än idealisk mellan JavaPUST och SiebelPUST (vilket har lett till upprepning av misstag), arkitektur och gränssnitt skiljer mellan SiebelPUST och SiebelPÄR snarare än att maximera delade tjänster (och avvikelsen från Siebel Core blir större och större) och det finns ett dödläge vad gäller att komma överens om servicenivåer mellan PVS och ITS.

## *3.2 Analys*

Det faktum att Siebel-programmet består av minst fem projekt och tre program innebär att det finns många gränser / gränssnitt att hantera och det måste finnas en god samordning, kommunikation och samarbete mellan olika parter. Eftersom kommunikationen och samarbetet verkar vara en av de största utmaningarna inom RPS blir gränsdragningarna stora frågor. Frågan blir ännu större eftersom det finns en brist på samordning mellan programmen men den nyligen genomförda Action-gruppen är ett undantag. Resultatet är att SiebelPÄR och SiebelPUST driver sina projekt i olika riktningar vilket innebär att avvikelsen från Siebel-standard och avvikelsen från RPS egen Siebel Core ökar varje dag. Konsolidering till en standardplattform var en av de viktigaste argumenten för att genomföra Siebel (i stället för ett specialbyggd system eller flera specialiserade system) och hålla sig till en standardimplementation är ett av de bästa sätten att uppnå några av de viktigaste målen (snabb utveckling, lägre utveckling och förvaltningskostnad och hög mängd återanvänd kod). Om detta inte hanteras finns det en stor risk att de viktigaste målen endast delvis uppfylls.

Det faktum att Siebelprojekten (särskilt SiebelPUST) verkar vara beroende av andra IT-relaterade program såsom INÄS, NKP och MOPC ökar bara mängden gränser / gränssnitt och beroenden som behöver hanteras. Eftersom det verkar finnas starka beroenden mellan Siebel-projekten och dessa externa program skulle ett fel eller stopp i något av de andra programmen kunna få återverkningar på Siebel-projekten. Siebel måste plötsligt hantera dessa frågor vilket indikerar att det finns en brist på proaktiva motsvarigheter i de andra programmen. Även om Siebel-projekten tycks prioriteras visar dessa problem det motsatta.

Det är tydligt att RPS är i händerna på externa leverantörer. Siebel-projekten måste vara en framgång och RPS har inte tid att göra ett "omtag". Systemintegratörer arbetar främst på ’löpande räkning’ vilket innebär att risken fortfarande huvudsakligen sitter hos RPS. Eftersom det finns en brist på Siebel-kompetens (se avsnittet Resurser) i Sverige har flera systemintegratörer "tvingats" att använda utländska konsulter,vilket innebär att arbetsspråket är engelska, vilket resulterar i en annan nackdel för RPS. RPS har inte byggt upp Siebel-kompetens och visar inga tydliga avsikter att göra detta. Detta innebär att de har ett problem att värdera kvaliteten och nivån på konsulternas Siebel-kompetens och de förstår inte fullt ut vad de får. Relationen med Oracle är också svår. RPS anser att de blivit lovade ett polispaket med de flesta funktioner som byggs men inser nu att de kommer att behöva bygga det mesta av dessa speciella funktioner själva utan särskilt stöd från Oracle. RPS har beslutat att minska mängden konsultstöd från Oracle vilket innebär att de är ännu mer beroende av systemintegratörerna.

Program som innebär organisatorisk förändring och kulturell förändring måste förankras i organisationen för att lyckas. I korthet innebär detta att alla måste vara "ombord". Om det finns människor som tar ett personligt intresse för programmets misslyckande finns en risk att detta kommer att hindra programmets utveckling eller till och med gör att det misslyckas.

Beställar / Utförarmodellens förutsättningar måste bli tydligare. Personer måste förstå rollerna i beställande och utförande, till exempel på vilken nivå ITS och verksamheten (POA / RA) bör vara delaktiga i projekten och vem som bestämmer vad som ska rymmas inom projektet. För att beställar / utförarmodellen skall fungera måste det finnas styrmekanismer på plats. Organisationen som beställer behöver antingen få leveransåtaganden från leverantören eller ha möjlighet att använda externa leverantörer. Om det inte finns några styrmekanismer på plats kommer organisationen med utförande endast ha en liten risk (eller ingen risk) och kommer istället arbeta efter bästa förmåga med låga incitament att hålla sig inom budgeten. I så fall kommer organisationen som beställer behålla risken, men risken bör följa pengarna. Om det finns en gemensam risk kommer det också att finnas ett starkare engagemang för att lyckas från båda parter.

Att låta ITS vara programägare av MAPS är också en stor risk. Delvis på grund av förvirring om gränser mellan PoA och ITS och vad som ska hanteras där, men också det nödvändiga engagemanget från PoA i MAPS-programmet (och dess framgång). Ökat deltagande från PoA kunde ha säkerställt en tätare koppling till slutanvändarna: poliserna.

I en komplex programstruktur med flera programägare blir kommunikationen ännu viktigare mellan divisionerna och program samt inom divisionerna och program. Om beslut fattas på en strategisk nivå är det viktigt att denna information sprids nedåt i organisationen.

### *3.3 Rekommendationer*

Kommunikationen hos RPS måste förbättras. Projekt (och program) måste vara mer samordnade. Sätt upp en struktur för att säkerställa detta. På en högre nivå borde programmen ha kombinerade styrgrupper och på den översta nivån bör det finnas ett gemensamt program för alla Siebel-projekten. RPS måste se till att alla divisioner förstår att Siebel-projekten skall lyckas uppnå sina viktigaste mål. Detta hanteras delvis genom ytterligare bekräftelse av Siebel-beslutet samt med hjälp av de justerade målen och se till att risken sprids mellan avdelningarna (se rekommendationerna på beställar /utförare nedan).

Beroenden av andra program måste hanteras direkt. Om Siebel-projekten verkligen har den högsta prioriteten är det nödvändigt för andra program att säkerställa att de inte är en flaskhals för Siebel-projekten. Om det finns en risk att Siebel-projekten kommer att försenas på grund av andra program och anpassningen till dessa program inte är nödvändigt för att uppnå de viktigaste målen bör anpassningen till dessa program, om möjligt, vara utanför projektens uppdrag. Ett exempel är NKP-programmet - är det nödvändigt att släppa Siebel på den nya klientplattformen (NKP) för att Siebel ska uppfylla sina viktigaste mål, eller kan det ske i ett senare skede?

RPS tycks vara starkt beroende av externa leverantörer. Med kontrakt som bygger på ’löpande räkning’ innebär det att risken stannar hos kunden (i det här fallet RPS) medan leverantörerna ’endast’ förväntas leverera efter bästa förmåga. Exempelvis finns det i allmänhet inga viten för leverantören i samband med att inte leverera i tid och enligt kostnadsestimat. RPS bör köpa åtaganden på fast pris med relevanta viten om leverantörerna inte levererar i tid och enligt budget. Detta kommer att skapa incitament för leverantörerna att vara proaktiv och leverera i tid och inom budget. I fallet med CGI och Deloitte verkar de tävla mot varandra för ytterligare affärer. Denna sunda konkurrens innebär att RPS kommer att vara mindre i händerna på en leverantör (med två att välja från) men tanken på att leverera åtaganden för ett fast pris bör gälla även här. När det gäller engagemang från Oracle bör RPS klargöra hur Oracle avser att delta i utvecklingen av Siebels polispaket om alls. I allmänhet för att bli en mer kunnig beställare och kund ska RPS bygga Siebel-kompetens själva. På så sätt kommer de att bli mindre beroende av leverantörer och fatta mer välgrundade beslut om Siebel.

På samma sätt som RPS förväntar sig att externa leverantörer ska signera leveransåtaganden måste beställarorganisationen förvänta sig att leveransorganisationen också ska skriva på leveransåtaganden. Till exempel, om PoA i egenskap av organisation med beställer ett åtagande från PVS bör PVS agera som en utförare/leveransorganisation och bör därmed leverera i tid och inom budget. PVS bör inte få mer finansiering om de behöver mer resurser för att leverera och RPS borde fundera på att implementera någon form av vite om deadlines in följs.

En stark rekommendation är att se till att ITS inte kommer att bli ägare till något av Siebel-programmen efter MAPS har gått live i februari.

# 4. Är resursmodellen adekvat vad gäller antal resurser och kompetens?

## *Observationer*

- RPS är en av få organisationer I Sverige som använder Siebel. RPS har själva inga dedikerade Siebel-resurser i varken program- eller arkitektgrupperna. I de tidiga stadierna fanns det konsulter från Oracle på plats, men dessa är inte längre närvarande

- PÄR-projektet är bemannat genom en kombination av konsulter från kontrakt, Deloitte och CGI. SiebelPUST-projektet är bemannat av Accenture.

- RPS har en dedikerad säkerhetsgrupp som ansvarar för penetrationstester/injektionstester/malware-tester samt för att validera att säkerhetskraven blivit korrekt implementerade.

- Testning av Siebel-applikationerna utförs av projektteamen och de verksamhetsanvändare som ingår i projekten.

- Linjearkitekten tilldelad Siebel-projekten har inte fått formell träning i Siebel och kan därmed inte validera korrektheten i strategin för implementationen av verksamhetskraven.

- Många av de intervjuade påpekade att de konsulter som arbetar med Siebel-konfiguration kan ha bristande kvalitet och erfarenhet. Det fanns även oro att några av de befintliga konsulterna skulle rullas av på grund av EU-skatteregler. Detta skulle i så fall betyda att deras kunskap och erfarenhet kommer gå förlorad för projektet.

- De flesta dokument är på engelska och ibland uppstår översättningsproblem vad gäller att fånga språknyanser. Många av Siebel-konsulterna har inte engelska som modersmål och kan inte svenska vilket ökar risken för missuppfattning.

- Processen för att fånga upp verksamhetskrav verkar inte formaliserad.

## *Analys*

*Siebel och OPA resurser*

RPS är beroende av externa parter  (konsulter och systemintegratörer) vad gäller all sin Siebel- samt Oracle Policy Automation (OPA)-kunskap. Det är normalt att använda externa företag och konsulter för att implementera dessa projekt men det är viktigt att RPS har kunskap om produkterna som konfigureras så att de kan sätta en strategisk rikting och förstå hur man bäst använder Siebel och OPA.

Det är också viktigt att systemintegratörerna erbjuder kvalificerad och erfaren personal till projektet. Detta kommer att säkerställa att kraven förstås och implementeras på rätt sätt.

Kravhantering är en viktig del i implementationen av ett Siebelprojekt. Kraven ska inte bara dokumenteras utan bör även fångas upp av personer som inte bara förstå verksamhetsmiljön, men som även förstå de produkter som kommer att användas för att implementera kraven, t.ex. Siebel och OPA.

*Testning*

Komplexa Siebel-projekt kräver olika former av tester: s k unit testing, system, integration, prestanda, användaracceptans och operativ acceptans. Det är vanligt att utvecklingsgrupper gör ’unit -testning’ som en del av konfigurationsprocessen. Andra tester utförs vanligen av specialiserade testare, användare eller de som är tänkta att sköta systemet. Vid RPS verkar det som att utvecklingsgrupperna har ett bredare befogenhet än bara ’unit testning’. Detta kan resultera i att systemet inte testas så grundligt eller oberoende som krävs. Särskilt för operativa system såsom PÄR och PUST, som tillhandahåller verksamhetskritiska funktioner till polispersonal på fältet.

## *Rekommendationer*

*Siebel- och OPA-resurser*

Inrätta ett kompetenscentrum inom RPS som har Siebel och OPA färdigheter. För Siebel-kompetens bör RPS fundera över att ha åtminstone en arkitekt som är ansvarig genom hela processen (en s k ’end-to-end-arkitekt’), en huvudansvarig verksamhetsanalytiker samt en integrationsarkitekt som kan Siebel. End-to-end-arkitekten skulle ’äga’ Siebel-lösningen och se till att den implementeras med hänsyn till RPS verksamhetsmål samtidigt som man hela tiden anpassar implementationen efter bästa praxis. Arkitekten kan också fungera som en kvalitetssäkrare som är oberoende av utvecklingsgruppen och som ansvarar för att den konfigurerade tillämpningen är av hög kvalitet. Den huvudansvariga verksamhetsanalytikern kommer samarbeta med verksamhetsledningen och användarna för att säkerställa att de förstår vilka funktioner som finns tillgängliga inom Siebel och för att styra deras krav mot att använda grundläggande funktioner snarare än att anpassa produkten. Integrationsarkitekten kommer att ansvara för att äga integrationsstrategin mellan Siebel och andra system. Denna person kommer att säkerställa att integrationen återanvänder arbete som redan har gjorts vilket förhindrar dubbelarbete. Andra RPS-anställda ska också utbildas i Siebel så att de vet vilka styrkor och begränsningar som plattformen har, t.ex. säkerhet, infrastruktur, nätverk, viktiga verksamhetsanvändare.

Oracle Policy Automation är en annan viktig komponent inom Siebelprojektet. Den tillåter att verksamhetsregler implementeras med hjälp av ett naturligt språk som innebär att man blir mindre beroende av utvecklare för att implementera policies, lagar och regler. Den stora mängden anpassning av Siebel antyder att verksamhetslogik, policy och polislagstiftning har konfigurerats direkt i Siebel istället för att implementeras i OPA där de kan anropas från Siebel när det behövs. Att flytta reglerna från Siebel till OPA innebär att komplicerade regler kan skrivas av användarna istället för av utvecklarna vilket resulterar i färre misstag och kortare utvecklingslivscykler. Detta bör åtgärdas i en två-fas strategi: för det första, bedöma i vilken utsträckning policies, lagar och regler har implementerats i Siebel och vad som bör flyttas till OPA. För det andra, utbilda användarna i hur regler skrivs i OPA så att de kan bli ansvariga för implementationen av reglerna.

RPS bör också fortsätta att engagera sig i andra Oracle-kunder som använder eller funderar på att använda Siebel och OPA för ärendehantering. Kontinuerlig dialog med andra kunder kommer tillåta RPS att lära av andras projekterfarenheter.

*Testning*

RPS bör överväga att upprätta en ”testfabrik" inom Siebel-programmet som kommer att ansvara för tester genom hela kedjan (end-to-end) av Siebel-projekten. Dessa kommer att börja med verksamhetskraven för att säkerställa att de är mätbara (kan testas), arbeta med utvecklingsgrupperna för att skapa och planera ’unit testning’ och script samt validera resultaten av ’unit’-testningen och sedan hantera de större aspekterna av testning:

- Integrationstester för att validera att gränssnitten fungerar
- Systemtester för att säkerställa att verksamhetsprocesserna fungerar från början till slut. Detta omfattar samtliga applikationer och gränssnitt som berörs under en process
- Prestandatester för att säkerställa att konfigurationen skalar med antalet användare och ger acceptabla svarstider till slutanvändaren
- Icke-funktionella tester för att säkerställa att säkerhets- och juridiska aspekter täcks
- Acceptanstest tillsammans med användarna för att validera att systemet implementerar verksamhetsprocesser korrekt och att systemet är användbart
- Operativ acceptanstestning som säkerställer att det utvecklade systemet kan köras på RPS servrar och nätverk och kan återställas i händelse av fel.

”Testfabriken” kan vara på plats hos RPS eller off-shore. Varje testfas är diskret och kan hanteras av en extern grupp.

# 5. Är en lämplig arkitektur på plats skalenligt anpassad?

## *Observationer*

- RPS har komplexa säkerhetskrav som har resulterat i anpassning av Siebel. Det finns ett centralt loggningssystem (CSL) som samlar säkerhetsdata från många applikationer. Detta måste inkludera säkerhetsgranskningens kontext och även vilken data som har setts, skapats eller ändrats. Nästan varje person som intervjuats uppgav att loggningskraven var komplexa och hade därmed bidragit till projektets försening och anpassningen av Siebel.

- ’Säker kodning’ är för närvarande inte aktuellt för RPS Siebel-projekt.

- SiebelPUST implementeras som en direkt ersättare för Java-versionen av PUST.

- Oracle Repository Calculator har visat en stor mängd scripting (ca 80.000 rader) och relativt många nya objekt i kodbasen (ca. 1800).

- PÄR och PUST bygger på samma core Siebel-plattform (kallas SiebelCore). Men utvecklingen av varje applikation ligger på olika grupper. Under programmets gång har PUST-projektet utvecklat ändringar i core-plattformen som inte har återspeglats i PÄR-implementationen, något som nu resulterat i tre separata Siebel-instanser.

- Det kommer potentiellt att bli mellan 8-9,000 och upp till 20.000 användare av Siebelapplikationerna. En demo av PÄR-systemet indikerade inte några särskilda prestandaproblem, men demon utfördes på en utbildningsmiljö med endast en inloggad användare. Prestandatestning utförs med hjälp av automatiserade verktyg. När SiebelPUST-systemet demonstrerades verkade systemet hänga sig vid flera tillfällen. Möjligen kan detta bero på loggning, anrop till OPA eller timeouts orsakade av att Adobe Lifecycle inte svarar. För användare i fält kommer PUST-applikationen förlita sig på en mobil uppkoppling för att kommunicera mellan slutanvändarens enhet och Siebel-servern.

- Demon av PÄR- och PUST-systemen avsåg också användbarheten av de två applikationerna. PÄR ser väldigt mycket ut som en vanlig Siebel-implementation och innehåller ett minimalistiskt användargränssnitt. PUST-användargränssnittet har konfigurerats för att passa storleken på de skärmar som kommer att användas i fält, samt anpassats vad gäller färg och form.

- Flera av de intervjuade påpekade att de trodde att det skulle bli problem med användare som inte vill använda SiebelPUST som den för tillfället är konfigurerad. Detta på grund av det stora antalet klick och eftersom systemet är så annorlunda från vad de för närvarande använder (antingen i JavaPUST eller RAR / DUR-system).

- Integrationsgruppen anmärkte att XSDs (gränssnittsdefinitionerna) ändras ofta. Ibland två gånger om dagen och upp till 20 förändringar per vecka, vilket är högre än väntat.

## *Analys*

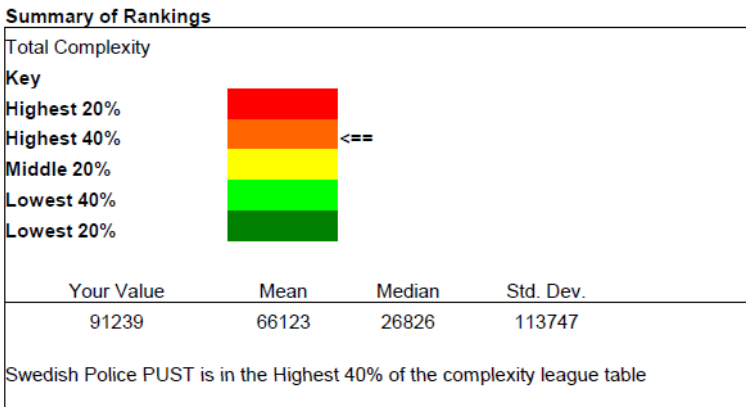
*Loggning*

Siebel Audit Trail kan fånga vilken användare som har sett, infört, uppdaterat och raderat poster i applikationen. Men den har sina begränsningar, i synnerhet kring vilka dataposter som har setts i en lista med poster, eftersom endast den post som "klickades på" loggas. Detta innebär att i en förteckning över säg 20 poster kommer endast en auditpost skrivas medan RPS kräver att alla 20 poster granskas eftersom en användare kan ha läst innehållet i en post på skärmen och skrivit ner detaljerna för användning utanför systemet. Dessutom dokumenterar Siebel Audit Trail inte sammanhanget runt en auditpost d.v.s. vilken verksamhetsfunktion som utförts.

Audit Trail är en central komponent i Siebel. Så som Siebel-implementationen är anpassad finns det en risk för att prestanda kommer att försämras när en applikation har rullas ut till många tusen samtidiga användare eftersom loggnings-motorn anropas varje gång varje användare utför någon operation inom systemet. Det innebär också ytterligare omkostnader under projektets tid samt kostnad för att bibehålla anpassningarna framöver, speciellt om loggnings-kraven förändras.

*Java vs. Siebel PUST*

SiebelPUST-projektet återimplementerar JavaPUST-systemet med funktionerna från Java-systemet och konfigurerar in dem i Siebel. Detta har resulterat i en teknisk implementation snarare än en verksamhetsimplementation där kravinsamlingen hållits till ett minimum. Det har också lett till en hel del anpassning av Siebel som kan ses från Repository Calculator som Oracle levererat(versionen från Oracle från Januari jämförde PUST med en inkorrekt Siebel-version):




***Källa: Oracle Complexity Calculator report April 2013 (PUST)***

Komplexitet av detta slag inträffar oftast som ett resultat av en kombination av faktorer: att inte ha tydliga verksamhetskrav eller att ha verksamhetskrav som inte fångats med Siebel-plattformen i åtanke vilket gör att Siebel-produkten får utföra funktioner som den inte är avsedd för, att inte ha erfarenhet av Siebel-utveckling samt att ha gränssnitt till flera externa system.

RPS bör också vara oroliga över att de redan är bland de översta 40% av de flesta komplexa implementationer som Oracle har bedömts. Detta tyder på att framtida utveckling och underhåll av Siebel-plattformen kommer att kosta mer än beräknat och riskerar därför att inte uppfylla ett av de viktigaste målen som är att sänka kostnaderna.

*Siebel-anpassning*

För att följa upp det som nämns ovan, bör man se på orsakerna till RPS plats bland de 40% mest komplexa installationerna, innehållet i kodbasen på ca 80.000 rader av script och cirka 1800 nya Siebel-objekt (verksamhetsskomponenter, verksamhetsobjekt, vyer, applets, kopplingar och länkar).

Antalet rader av script är ett problem eftersom de har tagit lång tid att utveckla och kommer att behöva underhållas och uppgraderas med varje release. Ändringar i dessa script kommer att kräva en utvecklingslivscykel och förlitar sig på att personalen har tillräcklig kunskap om Siebel. Script är dessutom inte lika högpresterande som inbyggda Siebel-funktioner och detta kommer därför att vara ett bekymmer när applikationen rullas ut till flera tusen samtidiga användare. Även om hårdvaran är tillräckligt dimensionerad för att rymma antalet användare kommer det sannolikt att vara prestandaproblem.

Onödigt hög grad av scripting är oftast ett resultat av en kombination av faktorer: verksamhets- eller tekniska krav kan inte uppfyllas med hjälp av den grundläggande kapaciteten, verksamheten är ovillig att ändra arbetssätt, utvecklarna har inte funderat på alternativ till scriptning som redan finns i Siebel t.ex. arbetsflöde och runtime-händelser.

Förutom den stora mängden av script, innehåller RPS kodbas relativt många nya Siebel-objekt som har skapats av utvecklare. Som med rader av script kommer dessa objekt behöva förbättras, underhållas och uppgraderas allteftersom projektet fortsätter att förlita sig på Siebel-kunniga utvecklare för att utföra dessa förbättringar och förändringar.

| **Short Summary of Rankings** | |
|---|---|
| New Business Components | |
| New Business Objects | |
| New Views | |
| New Applets | |
| New Joins | |
| New Links | |
| New Customer Foreign Keys | |
| New Extension Columns | |
| New SmartScript Questions | |
| New Business Component Script | |
| New Business Service Script | |
| New Application Script | |
| New SmartScript Script | |
| New SmartScript Branches | |
| New Workflow Branches | |
| New Workflow Processes | |
| New Application Browser Script | |
| New Applet Browser Script | |
| New Applet Web Script | |
| New Business Component Browser Script | |
| New Run Time Service Script | |
| New Service Browser Script | |

***Källa: Oracle Complexity Calculator report April 2013 (PUST)***

Den observerade nivån av anpassning kommer inte att hjälpa RPS att minska kostnaderna för dess implementation, åtminstone inte på kort sikt.

Det är möjligt att antalet nya objekt beror på att grundläggande föremål har klonats för att tillgodose de olika behoven hos PÄR och PUST-projekten inom SiebelCore. Även om kopiering av objekt är tillåtet i Siebel bör det inte ske alltför ofta. Klonade objekt brukar också resultera i två, tre eller ännu fler exemplar av samma förvaltningsproblem när förändringar måste tillämpas på mer än ett ställe vilket leder till mer utveckling och testning.

*Prestanda*

Det är viktigt att PÄR- och PUST-plattformarna presterar på en acceptabel nivå. Båda dessa applikationer kommer att användas av personal i frontlinjen. I fallet med PUST även av poliser på fältet som fångar information om brott. PÄR måste också fungera på rätt sätt för att säkerställa att en backlog av ärenden inte byggs upp och som kräver extra personal för att rensa.

Både PÄR och PUST arbetar i en komplex end-to-end arkitektur där en användartransaktion behöva navigera genom brandväggar, webbservrar, olika nätverk, applikationsservrar och samla in data från olika källor (t.ex. personsökning eller Adobe Lifecycle) . Det är därför inte alltid lätt att identifiera var en flaskhals orsakar prestandaproblem.

Under demon av SiebelPUST uppfattades det flera gånger som att applikationen hade hängt sig innan den bytte bild på skärmen. Detta kan vara på grund av oförutsedd belastning på testserver eller eftersom inte alla programvarukomponenter kördes på testsystemet, t.ex. Adobe Lifecycle. Detta kan oroande nog vara ett symptom på den stora mängden scripting och anpassning som orsakar systemet att sakta ner när du utför vissa åtgärder. Om scripting är en del av gränssnittet, pekar detta på potentiella problem när systemet rullas ut och det kommer att finnas många tusen samtidiga användare som anropar de olika system som är anslutna till PÄR och PUST.

Projektgrupperna använder automatiserade testverktyg för att simulera tunga laster på systemet. Detta är rätt sätt att göra det på så länge det sker som en del av en övergripande teststrategi som behandlar prestandaproblem när de hittas. Teststrategin bör även inbegripa icke-funktionella krav avseende svarstider och volymer för att säkerställa att dessa räknas in i utformningen av infrastrukturen och Siebelkonfigurationen.

*Användarvänlighet*

Den initiala implementationen av PÄR innehåller grundläggande funktioner för att söka efter en person, skapa ett ärende och söka efter ett ärende. För närvarande finns det inte en stor mängd funktionalitet. Som ett resultat, är systemet lätt att navigera och skärmarna är relativt "stilrena", d.v.s. inte för mycket data att visa. De nuvarande PÄR-användarna flyttar oftast från pappersbaserade och terminalbaserade-program, så användarnas övergång till det nya systemet förväntas inte orsaka några problem eftersom det nya systemet bör vara ett rejält steg upp från vad de använder för tillfället.

PUST implementationen är däremot en komplex verksamhetsprocess vilket återspeglas i det stora antalet Siebel-skärmar och vyer. Dessutom innehåller varje skärm många under-vyer och en del av dessa vyer innehåller många applets. Applets själva (där informationen visas) innehåller ofta en hel del datafält och har flera knappar för användaren att klicka på. Detta för att migrationen av det befintliga systemet till Siebel speglar den komplexitet PUST-processen visar idag. Skärmarna kräver ett stort antal klick för att bläddra och för att navigera runt i systemet.

Användarnas övergång till det nya systemet är ett vanligt problem i Siebel-projekt. Siebel-användargränssnittet är varken ett traditionellt Windows eller terminal-baserat system och inte heller ser det ut som ett webbaserat system även om det körs i en webbläsare. Den grundläggande Siebel-applikationen innehåller många skärmar, vyer, applets och datafält som kan förvirra en användare eftersom det inte alltid är självklart vad man ska göra eller vart man ska gå härnäst. Liksom navigationsproblem kan det finnas en uppfattning bland användare att Siebel inte ger något tillbaka till dem, speciellt när det ersätter ett system som användaren ser fungera mycket väl och som de precis lärt sig. I situationer som denna är det viktigt att komma ihåg varför en ny plattform implementeras och att Siebel ger fördelar för användarna, t.ex. minskad tid för att skriva in ett nytt fall, förmåga att klara upp ärenden snabbare, och mer tid att spendera på att lösa brott än på administration.

En sista anmärkning om användarvänlighet är att Siebel-applikationer brukar misslyckas med att följa WCAG (Web Content Accessibility Guidelines) normer som är utformade för att göra materialet mer tillgängligt för användare som kräver högre tillgänglighet/stöd. Till exempel, om RPS har synskadade användare som förlitar sig på skärmläsare eller förstoringsprogram för att utföra sina jobb kan dessa oftast inte arbeta med Siebel på grund av det sätt som Siebel-användargränssnittet återges i webbläsaren.

## *Rekommendationer*

*Loggning*

Överväg att implementera auditkraven utanför Siebel. Fånga ett absolut minimum av information för att tillfredsställa CSL och skicka sedan data till ett annat program för bearbetning innan det i sin tur skickar den sammanställda auditinformationen till CSL.

Överväg också att använda databas-audit istället för Siebel Audit Trail. Oracle Grained Audit kan utföra samma funktioner som Siebel Audit Trail men på en databas-nivå vilket innebär att loggningen skulle bli snabbare. Det innebär också att projektgruppen kommer att spendera mindre tid på icke-funktionella krav och kan ägna mer tid åt att implementera verksamhetsfunktionalitet.

En annan lösning kan vara att använda mjukvara som hanterar ’skärmavlyssning’ vilket innebär att Siebel Audit Trail kan fånga minimalt med information och arbeta diskret medan de viktiga verksamhetsprocesserna granskas visuellt. Även om detta kommer att kräva ytterligare programvarukomponenter kan detta kosta mindre än kostnaden för att upprätthålla egna auditkrav i Siebel och kostnaden för användarens tid i ett långsamt systemet på grund av de anpassade krav som har implementerats.

*Java vs. Siebel PUST*

Siebel-projekt bör inte genomföras som direkta ersättningar för befintliga system. Bästa praxis är att fånga krav, mappa dessa till Siebel, identifiera gap och hålla anpassning till ett minimum. Gap och anpassning bör kontrolleras via styrning mellan projektet och verksamheten för att säkerställa att verksamhetens mål uppfylls och att avvikelser förstås av alla parter innan kostsam utveckling startas.

Det är nog för sent att dra tillbaka alla de anpassningar som redan har utförts på SiebelPUST. Men framöver bör RPS anta en projektmetodik som fångar kraven med Siebel i åtanke, kartlägger dessa till Siebel kapacitet samt upprätthåller styrning för att minimera anpassning. Verksamheten måste delta i dessa samtal så att de är medvetna om de grundläggande funktioner som finns i Siebel så att de kan utforma sina krav på lämpligt sätt. Det bör också vara möjligt att ha samtal med verksamheten om att förändra sitt sätt att arbeta snarare än att anpassa Siebel att fungera onaturligt.

*Siebel-anpassning*

De 80.000 rader av script som finns bör analyseras för att avgöra om det finns alternativa metoder för genomförandet, t.ex. arbetsflöde, runtime-händelser och användarinställningar. För script som måste förbli bör kvaliteten på konfigurationen bedömas för att se till att de är högpresterande och effektiva. Den här analysen kommer även visa i vilken grad anpassningar varit nödvändiga och vad huvudorsakerna till anpassningarna är.

Den grundläggande orsaken till att det finns 1,800 nya objekt bör fastställas. Om detta är ett resultat av att PÄR och PUST delar core och gemensam uppsättning objekt men har olika krav så skulle det vara bättre att dela upp de två systemen helt och underhålla dem separat framöver i stället för att behålla PÄR och PUST-varianter av objekt. Alternativt bör de två systemen föras tillbaka till en gemensam core som kan upprätthållas av en grupp där de nuvarande PÄR och PUST-grupperna ansvarar för specifika krav som kommer genom versamhetsförändringar inom respektive område. Eftersom PUST-gruppen redan har utvecklat bort sin version av core från den ursprungliga versionen är det viktigt att de två teamen konsoliderar tillbaka till en enda core så snart som möjligt. Testning av dessa förändringar kommer också att behöva göras för att bekräfta att förändringarna inte har tagit bort funktionalitet som tidigare var konfigurerad och testad.

Det verkar som att implementationen av PÄR och PUST på en enda Siebel-instans är inte möjligt av juridiska skäl (t.ex. blandning kriminella och icke-kriminella ärenden / personer) men en enda Siebel-implementation skulle vara enklare och billigare i längden.

RPS bör också undvika ytterligare Siebel instanser eftersom detta kommer att öka komplexiteten och kostnaderna för hela Siebel-programmet, t.ex. internationella fall.

*Prestanda*

Prestandatestning utförs vanligtvis sist eller nära slutet i utvecklingslivscykeln. Dock bör de icke-funktionella krav som är testade utformas och utvecklas på direkten för att säkerställa att konstruktören / utvecklaren vet att biten av funktionalitet de implementerar fungerar lika bra för 8.000 användare som det gör för en enda användare. Bättre kommunikation av icke-funktionella krav genom hela livscykeln ser till att alla parter är medvetna om konsekvenserna av eventuella förändringar på användarbasis. Det bör också finnas en lista över viktiga verksamhetsprocesser och svarstider som testas som en del av ’unit test’-

livscykeln så att lågpresterande kod fångas upp tidigt i projektet snarare än i slutet vilket kan resultera i en försening i genomförandet av projektet.

End-to-end arkitekturen behöver förstås av projektets arkitekter så att flaskhalsar kan identifieras. Det är inte tillräckligt bra att konstatera att något är ett problem med "nätverket" eller "säkerhetslagret" eller "rapporteringsmotorn" eller "gränssnittet" när det i själva verket kan det vara så att någon av dessa komponenter antingen är felaktigt dimensionerade eller inte utformade för att göra vad som efterfrågas av dem. Detta gäller särskilt vid RPS där loggningskraven har placerat en stor börda på Siebel att lämna uppgifter om vad en användare har sett och vad användaren gör med uppgifterna. Som nämndes i början av detta avsnitt i audit och loggning bör man överväga en alternativ lösning för att logga om detta visar sig vara en orsak till prestandaproblem. På samma sätt förlitar sig end-to-end lösningen på många gränssnitt för att genomföra verksamhetsprocesser. Om dessa gränssnitt är beroende av script i Siebel är prestandaproblem troligt. Det är därför viktigt att validera att gränssnitten har implementerats med ett minimum av script.

*Användarvänlighet*

Det är viktigt att användarna är utbildade i grunderna i Siebel såsom navigering, söka information och skriva in data. En hel del av det initiala motståndet till användargränssnittet kan övervinnas genom att visa användarna hur man utför dessa grundläggande uppgifter särskilt genom att belysa hur kortkommandon kan göra verksamheten snabbare än att använda en mus. Det är också viktigt att relatera layouten av skärmarna till detaljerna i de processer som användarna utför så att de ser användargränssnittet som ett medel för att utföra ett arbete snarare än ett hinder.

PUST-användargränssnittet är en särskild utmaning på grund av den komplicerade karaktären i implementationsprocessen. Det system som demonstrerades kräver en hel del klick och bläddra för att ange ett nytt fall och relaterad information. Förutom att vara komplicerad och kräva detaljerad utbildning skulle det vara lätt för en användare att gå "vilse" i ansökan. Detta skulle kunna förenklas på ett par sätt:

- Använd uppgiftsbaserade (Task-based) användargränssnitt i Siebel för att bryta ner processen att mata in ett nytt fall i diskreta delmoment. Detta kommer att minska mängden information på skärmen vid alla givna tidpunkter och kommer att göra det möjligt för användaren att navigera framåt och bakåt i processen med ett mindre antal knappar än det är för tillfället. Det minskar också mängden scrollning (vilket tar tid) vilket innebär att ärenden kan tas upp snabbare. Uppgiftsbaserade (Task-based) användargränssnitt gör också att oerfarna användare kan mata in data med minimal utbildning.
- Överväg att införa en konsolliknande skärm som tillåter användaren att se förloppet av ärendet när de går igenom processen. Processen för att ange ett ärende kräver att användaren slutför en serie steg. Den konsolliknande skärmen kan visa användaren hur långt genom processen de är och de steg som fortfarande måste slutföras. Även om detta kommer att kräva att konfigurationen av en ytterligare vy kan det hjälpa med navigation och förhindra att användaren glömmer att utföra delar av processen. En sådan skärm tvingar inte användaren att utföra processen på ett bestämt sätt men det gör att användaren guidas.

- Siebel Open UI: detta skulle kunna användas för att skapa skärmar som efterliknar den befintliga processen i JavaPUST. Även om detta kan övervinna ett kortsiktigt problem med användarnas övergång för att användarna inte gillar Siebel UI införs ett nytt arv av att behöva konfigurera varje vy i Siebel så att det ser ut som det gamla programmet. Denna väg bör endast övervägas om det kan bevisas att den snabbt kan implementeras och inte resulterar i mer anpassning i kodbasen.

Området där Siebel Open UI kommer att vara tillämplig gör programmet tillgängligt för mindre duktiga användare som förlitar sig på skärmläsare och förstoringsglas. Eftersom det öppna gränssnittet kan köras i alla webbläsare och stödjer öppna standarder kan dessa hjälpmedel användas. Om man antar att det finns ett krav att stödja WCAG standarder finns det fördelar med att inrätta en testmiljö med Siebel Open UI för att se om det kan implementeras med minimal ansträngning (även om det kommer att kräva någon ytterligare konfiguration av servermiljön ).

Ett sista förslag kring användarvänlighet skulle vara att göra en användarvänlighets- och tillgänglighetsgranskning av applikationen och fastställa områden för förbättring i hur användarna kommer att använda systemet och för att garantera att den uppfyller standarder för nedsatta användare (om tillämpligt). En användarvänlighetsgranskning kommer att bidra till att öka användarnas användning av Siebel-applikationen.

# Sammanfattning av rekommendationer

Detta avsnitt av rapporten sammanfattar de rekommendationer som lämnas i denna rapport.

## *Programmets mål*

- RPS har dokumenterat fasta mål för Siebel-programmet. Det är viktigt att dessa mäts och följs (många projekt, särskilt inom den offentliga sektorn, har inga definierade mål).
- Kommunicera värdet av att hålla sig till grundutförandet av Siebel och uppmuntra verksamheten att ändra hur den fungerar.
- Tala om för verksamheten hur Siebel-programmet kan hjälpa till att förbättra hur det fungerar t.ex. öka effektiviteten och ökad upplösning.
- Kostnaden kommer inte att minska genom att anpassa Siebel och ha flera olika versioner av Siebel. Minska mängden anpassning och konsolidering till en enda kärnversion av Siebel med varianter för PÄR och PUST.
- Fokusera på användarnas övergång så att användarna är nöjda med funktionaliteten och användbarheten av Siebelapplikationerna.

## *Processer och Struktur*

- Det finns belägg för att det finns en solid styrmetod för förändring på plats vid RPS. Dock bör inte programmet förlita sig enbart på förändringskontrollen för att hantera omfattningen.
- Kravhanteringsprocessen måste formaliseras och bör slutföra så mycket som möjligt i början av varje projekts livscykel. Förlita er inte enbart på förändringskontroll (CCB).
- För krav som inte kan dokumenteras eller förutses behövs en rigorös förändringskontroll-process för att säkerställa att förändringarna är anpassade till programmets mål.
- Omstrukturera programmet för att få färre och kortare (i antal och komplexitet) rapporteringsvägar. Detta kommer att förbättra smidigheten och beslut kan fattas snabbare.
- Hantera risker proaktivt innan de blir problem.

## *Ansvar, gränser och beroenden*

- Förbättra kommunikationen genom att strukturera om programmet.
- Hantera interna programberoenden. Framför allt, gör inte Siebel-programmet till ansvarig för stora förändringar som genomförs av andra grupper, t.ex. testning av nya nätverk och stationär infrastruktur.
- Förhandla om till en åtagande/fastprismodell med leverantörerna.

- PoA och PVS bör enas över vilka åtaganden som förväntas av varandra och hantera dessa som om de hanterar en extern part, t.ex. med viten för avvikelser. Detta kan vara svårt, särskilt om en organisation inte har finansiering.

- ITS bör inte vara ägare till någon av Siebel-programmen efter MAPS har gått live i februari.

### *Resursmodell*

- Action-gruppen har snabbt löst ett antal frågor över Siebel-programmen. Denna grupp måste få fortsätta detta goda arbete.

- Upprätta Siebel-kompetens inom RPS med både Siebel- och OPA-färdigheter.

- Samordna och bemanna en formell QA process och Design Authority inom Siebel-programmen för att säkerställa att Siebel genomförs enligt bästa praxis.

- Fortsätta den befintliga dialogen med andra Oracle-kunder som implementerar Siebel Public Sector. Detta initiativ är redan etablerat och fungerar bra vid RPS.

- Även om det finns bevis för att enskilda grupper utför vissa aspekter av testning bör detta formaliseras genom en teststrategi för Siebel-programmen. Skapa en ”testfabrik” som är ansvarig för end-to-end-tester av PÄR och PUST.

### *Arkitektur*

- RPS har valt Siebel och OPA som sin ärendehanteringsplattform. Att kunna separera verksamhetslogik i Siebel från policy och lagstiftning i OPA är en av styrkorna med denna arkitektur. RPS måste se till att varje komponent används korrekt.

- Titta på alternativa lösningar för att genomföra säkerhets- och loggningskrav t.ex. Oracle Fine Grained Audit eller skärmavlyssningsprogram.

- Implementera Siebel enligt bästa praxis: fånga krav i linje med företagets mål, identifiera gap och minimera anpassningen.

- Överväg att ändra arbetssätt istället för att skräddarsy Siebel.

- Utföra en detaljerad analys av de 80.000 rader script som finns för att identifiera alternativa metoder för implementation.

- Analysera gränssnitten till PAR och PUST för att säkerställa att scriptning hålls till ett minimum och även för att bekräfta att koden återanvänds.

- Ta PÄR och PUST tillbaka till en gemensam core /kärna.

- Implementera inte ytterligare instanser av Siebel.

- Kommunicera icke-funktionella krav till utvecklarna så att koden är utvecklad med det slutliga tillståndet i åtanke.

- Titta på alternativa metoder för att genomföra PUST UI så att det krävs färre klick för att navigera i applikationen och att mindre information visas åt gången.

- Genomför en användarvänlighetsaudit för att hitta områden där användarens upplevelse kan förbättras, t.ex. genom att minska klick eller ändra skärmlayouter för att göra dem mer intuitiva.

# A Appendix – Förteckning över intervjuade personer och dokument

## A.1 Intervjuade personer

| | |
|---|---|
| Liselott Ringborg | Program Manager EIMa |
| Niklas Bracken | Project Manager EIMa |
| Johan Markborg | Architect EIMa |
| Jonas Hemmar | Program Manager EIMb/PUST2 |
| Ulrika Sanders Jonsson | Project Manager EIMb/PUST2 and TI-program |
| Olli Pekka Timperi | Architect EIMb/PUST2 and TI-program |
| Robert Lagerqvist | Architect EIMb/PUST2 |
| Tomas Centerlind | Program manager TI-program |
| Ola Öhlund | CIO |
| Per-Ola Sjöswärd | CTO |
| Predrag Mitrovic | IT strategist |
| Lena Tysk | Deputy Head Police Division |
| Eva-Lotta Hedin | Head of Justice Division |
| Peter Asztalos | Manager Application Operation PVS |
| Torbjörn Rapp | Manager Integration Center PVS |
| Jan Sahlander | Business Architect PVS |
| Erik Söderberg | Business Architect PV |
| Roger Lund | Manager Security Group PVS |
| Sverker Forsberg | Information Architect Business Protection Unit / Security |
| Pär Frid | Business Analyst Police Dept |
| Lina Bjurhager | Business Analyst Police Dept |
| Magnus Nilsson | Business Analyst Police Dept |

| | |
|---|---|
| Mats Hjelmgren | Business Analyst Police Dept |
| Karl Hjalmarsson | Application Maintenance |
| Lars Lindahl | Head of PVS Division |
| Fredrik Rudolph | Siebel Core Platform PM |

## *A.2 Dokument*

- Förstudierapport implementation av PUST I SIEBEL v1 17.pdf
- Engelsk om polisen och IT-strategi.ppt
- RPS Governance_eng.pptx
- MAPS action SG masterplan_english.ppt
- Repository Calculator – Swedish Police updated 17.4.2013.pdf