Projekt

SYSTEMFÖRNYELSE

Metodram för förnyelse av informationssystem

Lars-Åke Johansson
Stig Berild
Roland Dahl
Mats R Gustafsson
Sari Hakkarainen
William W Song

Sammandrag

Denna rapport beskriver resultaten från projektet Effektiv IT Fas III, Systemförnyelse. Bakgrunden till projektet är att många organisationer har fått ett allt större informationssystemarv som man inser att man måste förnya, men man vet inte hur man skall genomföra sådant förnyelsearbete.

Projektet har producerat en metodram, med vilken man kan definiera en strategi för att migrera ett systemarv för en viss organisation. Mångfalden avseende hur olika systemarv ser ut och målbilderna avseende vart man måste gå i sin verksamhetsutveckling medför att det inte finns EN strategi för förnyelse av systemarv utan många.

Metodramen anvisar olika delmetoder, verktyg och tekniker som kan vara användbara i olika typer av strategier för systemförnyelse. Delmetoderna är hämtade från olika internationella och nationella projekt. Metodramen har provats i ett par konkreta projekt under projekttiden. Resultaten av dessa prov presenteras också i rapporten.

Innehållsförteckning

Rapportens uppbyggnad och läsanvisningar

Rapportens uppbyggnad

Denna rapport består av fyra delar:

I	**Inledning**
II	**Metodram med metodhänvisning och verktygsdel**
III	**Samverkande informationssystem och migrering mot objektorienterad teknik**
IV	**Fallbeskrivningar**

Läsanvisningar

Inledningsavsnittet presenterar syftet med rapporten och vad som ligger bakom uppkomsten av metodramen. Dessutom finns här en beskrivning av vad metodramen representerar.

Avsnittet "Metodram med metodhänvisning och verktygsdel" beskriver själva metodramen och dess olika delar samt hur man kan skapa olika strategier för systemförnyelse.

Utredning och beskrivning av verksamhet och informationssystem är väsentliga instrument i samband med förnyelse av informationssystem. Det gäller speciellt i avseendet hur de förnyade systemdelarna skall se ut och hur förnyelsearbetet skall bedrivas.

I verktygsavsnittet görs en strukturering av olika typer av verktyg. Detta kan vara intressant för att hitta rätt typ av verktyg för den uppgift i systemförnyelsearbetet som skall lösas. Många verktyg löser endast vissa speciella typer av uppgifter och man får då kanske välja flera verktyg som kan kopplas ihop för att lösa en uppsättning av olika typer av uppgifter.

Resterande avsnitt går in på ett antal aspekter som kan vara aktuella vid arbete med att hitta en strategi för systemförnyelse.

Många organisationer har en strävan mot att hitta mer decentraliserade styrformer med ansvar och befogenheter förlagda till lokala enheter. Avsnittet om samverkande system beskriver mer ingående vad detta innebär och hur informationssystem kan se ut som bygger på denna styrningsfilosofi. Särskilt om man tänker sig att migrera äldre system i denna riktning är det viktigt att veta vad som verkligen kännetecknar denna typ av system.

Ibland vill man migrera mot objektorienterade plattformar vari ingår objektorienterade databaser. Det råder en debatt om vad som verkligen skall kallas objektorienterade databaser och vilka egenskaper dessa egentligen skall ha. I avsnittet om objektorienterade databaser försöker vi därför systematisera olika typer av egenskaper som dessa databaser kan ha och när de kan vara intressanta.

CORBA är en standard för att kommunicera objekt i en distribuerad miljö. Denna ansats har visat sig vara klart intressant för att migrera gamla system mot nya modernare. Det sker genom att gamla och nya systemdelar kan kommunicera i en samverkande miljö. Denna teknik är speciellt lämplig då man skall migrera mot en objektorienterad miljö.

CORBA kan också vara intressant då man vill kommunicera mellan relativt autonoma systemdelar enligt vad som ovan refererats till som samverkande system. Meddelande-samverkan kan realiseras med hjälp av kommunicerande objekt.

Jämförelse och integration av modeller blir aktuellt i flera situationer när det gäller förnyelse av informationssystem. Dels kan det gälla att jämföra modeller avseende gamla systemdelar och modeller som beskriver nya systemdelar, vilka tas fram i en separat utvecklingsprocess. Dels kan det vara fråga om modeller som arbetas fram i olika delar av verksamheten. Dessa kan beröra samma begrepp eller objekt och man måste komma underfund med hur man ser på de gemensamma objekten. Sådana analyser kan ge vid handen att verksamhetsstrukturer måste vidareutvecklas och ge underlag till avgränsning avseende hur uppgifter skall fördelas mellan delprocesser. Denna problematik behandlas
i avsnittet om verktygsval i samband med systemförnyelse.

I rapporten presenteras vidare några prov i pågående skarpa projekt. Syftet har varit att prova den metodram som tagits fram och vissa deltekniker som diskuteras i rapporten. Konkreta prov ger mycket av erfarenheter och en del råd ingående i metodramen har tagits fram till följd av proven i fallprojekten. De båda fallen är av ganska olika karaktär och speglar ytterligare att angreppssätten för migrering måste formas på ett sätt som passar det enskilda fallet. Det rör sig dels om en sjukvårdsverksamhet, där samverkan måste förbättras mellan olika delar i verksamheten genom s k vårdprocesser, dels en bankverksamhet som måste ta fram nya tjänster för kunderna samtidigt som man måste hitta en rationellare plattform för att sköta och förändra sina system. I båda fallen finns ett besvärande systemarv som måste hanteras.

I Inledning

Systemförnyelse måste ske på ett strukturerat sätt. Systemförnyelse måste utgå ifrån ett verksamhetsmässigt perspektiv. Om man vet hur man vill utveckla verksamheten, har man också en grund för på vilka sätt och i vilka avseenden man kan förnya informationssystemen. Detta är fundamentalt för att förnya informationssystem.

Det saknas i hög grad kunskap om hur man kan förnya informationssystem på ett systematiskt sätt. Det rör sig ofta om svårigheter att sätta samman olika typer av kunskap dels om att hantera gammal systemuppbyggnad, dels om nya systemplattformar och moderna angreppsätt och sätt att hantera system. Arvssystemen använder en viss typ av teknologi. Man strävar mot system som bygger på en annan teknologi. Det saknas kunskap om hur man kan få olika typer av teknologi att samverka.

I denna rapport föreslås en metodram som kan användas för att formulera strategier för successiv migrering. Tillkomsten av denna metodram bygger på antagandet att det är mycket svårt att hitta ensartade, allmängiltiga tillvägagångssätt för att förnya informationssystem.

Förnyelseprocesser måste istället innehålla olika komponenter för att svara mot krav som verksamheten ställer. Dessa krav kan vara av mycket skiftande karaktär. Dessutom måste man utgå ifrån hur arvssystemen ser ut. Detta kan ge upphov till helt olika typer av förnyelse av informationssystem. Med andra ord kan informationssystem förnyas på en mängd olika sätt. Detta beror främst av två olika orsaker:

1) Informationssystem som behöver förnyas kan vara beskaffade på en hel mängd olika sätt. Ibland kan informationssystemen vara helt nedslitna genom att de är mycket svåra att förändra. Rättningar och ändringar är genomförda som "patchar" med ostrukturerade hopp till nya delar. Både gammal och förändrad programkod är dåligt beskrivna.
   I andra fall kan informationssystemen vara välstrukturerade och välbeskrivna, men de behöver förändras så att de kan få nya egenskaper och göra nya saker.

2) Syftet för en förnyelse kan vara av många olika slag. Det kan finnas en mängd olika verksamhetsmässiga skäl till att man genomför en förnyelse. Detta medför att man behöver olika typer av förnyelse för att kunna gå olika verksamhetsmässiga skäl till mötes. Ett skäl kan t ex vara att en tilläggstjänst skall kunna levereras tillsammans med de produkter man idag levererar. Om man inte klarar detta äventyras företagets konkurrensmöjligheter på marknaden. Konkurrenterna har redan börjat agera med liknande tjänster. I andra fall måste man strukturera om programkoden för att kunna genomföra förändringar på ett betryggande sätt. När man utför ändringar idag medför detta oacceptabelt stora driftsstörningar.

Dessa olika skäl medför att man måste formulera medvetna strategier för förnyelse så att de olika åtgärder som vidtas är de rätta för att uppnå en viss effekt och för att rätt typ av kunskap skall kunna aktualiseras. Den totala massan av kunskap som kan bli aktuell för att genomföra olika typer av förnyelse är stor.

1. Översikt över metodramen

En första översikt över den utvecklade metodramen följer nedan. Den innehåller följande delar:

- Verksamhetsanalys i samband med systemförnyelse
- Systemgranskning
- Strategival
- Verktygsval
- Genomförande
- Successiv reviewing

1.1 Verksamhetsanalys i samband med systemförnyelse

Verksamhetsanalysen sörjer för att förnyelsearbetet blir förankrat i vad som är motiverat av de krav som finns i verksamheten.

En mängd saker kan vara aktuella i verksamheten som kan kräva olika inriktning på förnyelsearbetet. Verksamhetsdelen är av fundamental betydelse vid allt förnyelsearbete. Initiala funderingar på förnyelse av informationssystem kan visa sig behöva vara av helt annan karaktär efter en mer systematisk genomgång av vad som är för handen i verksamheten och av vad som skall styra förnyelsearbetet.

Detta är egentligen inte unikt för förnyelse av informationssystem utan gäller vid allt förändringsarbete av informationssystem. Det som är väsentligt är dock att olika aspekter av verksamheten systematiskt behandlas så att rätt krav ställs på förnyelsearbetet och på vad som skall uppnås. Det innebär att såväl aspekter som har med krav från enskilda användare som olika mål för verksamheten som helhet bearbetas. Lämpligen använder man här någon form av verksamhetsanalytisk metodik som kan fasa in i kommande steg i förnyelseprocessen, nämligen strategivalssteget.

1.2 Systemgranskning

I systemgranskningssteget gör man gör klart för sig hur de existerande systemen ser ut. Vilken status, vilka brister och vilka förtjänster de har. Vad värdet är av lagrad information. Hur den är lagrad. Hur kod och behandlingsdelar ser ut.

Denna granskning görs speciellt i ljuset av krav som är särskilt viktiga ur verksamhets-synpunkt, vilket skall ha kommit fram under verksamhetsanalysen. Systemdelar som är särskilt viktiga för att t ex realisera nya affärsmöjligheter granskas speciellt. Hur ser de existerande plattformarna ut? Kommer de att stödjas framgent av leverantören? Hur kan de vidareutvecklas? Vilken kompetens har vi för att sköta de existerande systemen? Hur kan standarden höjas? Detta är frågor som är särskilt viktiga att svara på inför strategivalssteget. Man måste ha så bra underlag som möjligt för att välja en god strategi för det vidare förnyelsearbetet.

När det gäller det faktiska läget avseende de existerande systemen är det viktigt att skapa en så tillförlitlig och korrekt bild som möjligt. Det är viktigt att beslutsgrupper och management varken får en för positiv eller en för negativ bild av hur informations-systemen verkligen ser ut. Det är viktigt att beskriva vilka effekter som kommer att inträffa baserat på hur informationssystemen kommer att behöva hanteras mot bakgrund av hur de ser ut.

1.3 Strategival

Strategivalssteget är ett av de viktigaste stegen i en förnyelseprocess. Där avgörs vilken inriktning man skall ta i förnyelsearbetet. En väsentlig grund för hur avgränsningen skall göras är verksamhetsdelens resultat. Är detta arbete inte utförligt gjort kan inte goda strategival göras.

Att välja strategi kan innebära att man kommer att besluta om att också bygga nytt. Det kan gälla helt nya system eller att delar byggs nya och ansluts till gamla delar, eventuellt i en successiv strategi. Till grund för dessa avgöranden ligger bedömningar om effekter och om vilka uppoffringar som måste göras vid olika alternativa strategier. Vad innebär det att förnya på ett visst sätt? Vad kostar det? Vad får vi ut på kort och på lång sikt? Hur ställer det sig att bygga nytt och att förnya systemdelar på olika sätt? Skall vi välja nya plattformar i samband med förnyelsearbetet? Vad får vi ut av detta? Vilka är de direkt mätbara effekterna? Vad är bedömningseffekter i form av t ex ökad flexibilitet på basis av nya eventuella affärsmöjligheter?

1.4 Verktygsval

Den strategi man väljer styr i hög grad vilka åtgärder som skall ingå i en systemförnyelseprocess. Beroende på de olika åtgärder som man vill skall ingå i processen finns det en hel mängd olika verktyg på marknaden att tillgå för att stödja olika typer av åtgärder.

Varje enskilt verktyg är dock ofta inriktat på att utföra en ganska liten del av de olika arbetsuppgifter som bör ingå i olika typer av förnyelseprocesser. Det innebär att det finns ett antal svårigheter när det gäller att hitta lämpliga verktyg.

Det kan vara svårt att förstå vilka verktyg som man behöver för en viss förnyelseprocess. Hur kan man koppla användande av olika verktyg till varandra? Hur kan man utnyttja ett resultat från ett verktyg som ”indata” till ett annat verktyg? Hur kan man få en hög produktivitet samtidigt som man gör rätt saker? Det finns relativt litet kunskap att tillgå för att skapa en samverkan mellan olika typer av verktyg.

En annan svårighet ligger i hur man skall kunna hitta rätt verktyg på marknaden om man vet vilken typ av verktyg man behöver. Ibland är verktygsutvecklarna själva ganska dåliga på att beskriva vad ett visst verktyg verkligen gör.

I detta arbete skulle man vilja ha några få större verktyg att luta sig mot, men verkligheten är att man i regel får hitta ett antal delverktyg som skall fogas samman i sitt användande så att man får en meningsfull helhet att fungera.

Exempel på verktyg är sådana verktyg som

- finner hur olika anrop mellan olika delar av koden ser ut,
- föreslår hur man skall hitta en bättre struktur för ett antal program,
- mäter ett antal komplexitetsfaktorer i ett antal program,
- översätter från en databasstruktur till en annan,
- översätter från ett programmeringsspråk till ett annat eller mellan olika versioner,
- undersöker standards och strategier för namngivning i program,
- hjälper en programmerare ett få en överblick över program som han/hon tidigare inte haft kontakt med.

Man förstår lätt av ovanstående grova listning att det inte är lätt att dels göra klart för sig vilka verktyg man egentligen behöver och dels kunna hitta och ange exempel på konkreta verktyg som kan vara användbara i ett visst fall.

En viktig uppdelning av olika verktyg är verktyg för reverse engineering och för reengineering. Verktyg för reverse engineering är inriktade på att gå in och granska hur system och systemdelar egentligen ser ut. Reengineeringsverktyg är inriktade på att på ett eller annat sätt förändra system eller systemdelar i olika avseenden. För att man skall få ett gott reengineeringsarbete utfört krävs dock att man har fått en god överblick över hur ett system ser ut.

Ett centralt verktyg i många verktygsmiljöer är repository-verktyget. Detta är i regel underskattat. Om systemdelar har undersökts och förbättrats i olika avseenden skall beskrivningar av systemdelarna lagras i ett dictionary eller repository.

Många leverantörer av både reverse engineeringsverktyg och reengineeringsverktyg lagrar olika delresultat från verktygsanvändningen i ett repository, som är mer eller mindre specifikt till sin lagringsstruktur.

Det torde vara av värde för de flesta företag som bedriver systemförnyelse att dessa verktyg är så öppna som möjligt så att olika delresultat som kommer från olika verktyg kan lagras i samma utbyggda repository-verktyg.

I dessa avseenden kan verktygstillverkarna vara olika benägna att ta hänsyn till olika standardförslag som har kommit fram. Om man har valt verktyg från flera olika leverantörer så gör detta problem i allmänhet sig påmint. Försök till standardisering inom detta område är t ex CDIF (Case Data Interchange Format) [1].

1.5 Genomförande

I genomförandesteget skall den strategi som har definierats i strategivalssteget och som där har motiverats på olika sätt till följd av bl a verksamhetsanalysarbetet, genomföras och implementeras.

Det innebär att projektaktiviteter och olika delprojekt måste specificeras och organiseras för att genomföra vad som ligger i strategin.

Det är viktigt att man kan svara på, för alla delar och alla aktiviteter som bedrivs, varför de genomförs. Detta måste kunna besvaras ur verksamhetssynpunkt av olika ledningspersoner som leder projekt.

Det innebär att man måste kunna svara på frågan varför vi skall förbättra dokumentationsstandarden på ett visst program eller varför vi skall beskriva verksamhetsregler baserat på en begreppsmodell för ett annat program. På vad sätt kommer dessa åtgärder in i helheten?

Vidare kommer reverse engineeringsarbete att avlösas av reengineeringsarbete. Man måste veta har långt man skall bedriva reverse engineeringsarbetet innan man börjar med reengineeringsarbete. När måste man gå framåt efter att ha studerat det man har? När skall man gå framåt och förändra? Det gäller både verksamhet och informationssystem.

Det är viktigt att man har tillgång till rätt kompetens för att bedriva och genomföra förnyelsearbete. Denna kompetens måste planeras innan man kommer till genomförandedelen.

Många gånger är det svårigheter med inställning till förnyelse och kunskaper om förnyelse hos personer i verksamheten, som är ett av de stora problemen för att komma någon vart med förnyelsearbetet.

En del personer som under många år har sysslat med äldre system kan ha blivit ovilliga till förnyelse eller upplever förnyelse som ett hot. Man vill därför inte delta i den kunskapsutveckling som är nödvändig för att man skall kunna migrera informationssystemstrukturer till moderna arkitekturer.

Personer som enbart behärskar äldre arkitekturer och realiseringsmiljöer måste ges chans och tid att tillgodogöra sig ny kunskap. Det är ofta fråga om att bygga upp nya referensramar. Olika förutsättningar i tänkandet ändras.

Det kan t ex innebära mycket arbete att lära känna en ny miljö som är objektorienterad. Ett helt nytt tänkande måste in. Det gäller dock att kunna relatera till vad man kan och har tillämpat hitintills. Vad är nytt i tänkandet?

Mycket av ny kunskap måste in för att bedriva förnyelse mot nya moderna plattformar. Det innebär i själva verket att kompetens som har med forward engineering måste nyttjas. Man skall därför akta sig för att enbart använda kunskap som finns inom ramen för äldre underhållsarbete. Detta är inte tillräckligt.

Om man t ex skall bedriva remodellering (beskrivning) av vad som görs i ett visst system eller en viss systemdel, måste man först ha fram en god, innehållsrik begreppsmodell och dels kunna skapa exakthet och fullständighet i denna modell. Detta är inte gjort i en handvändning. Det kräver kunskap om modellering och om nya metoder (som t ex kan inbegripa regelmodellering).

Komponenter i denna moderna modell skall sedan kunna kopplas till gamla program och programdelar som senare skall bytas ut. Man har på så sätt lagt en grund avseende vad som verkligen görs i ett viss system, på ett verksamhetsbaserat plan.

Genomförande kan vidare innebära av att man dels nyutvecklar vissa delar och dels samtidigt skapar gränsytor så att nya och gamla delar kan samverka. De äldre delarna kan därefter bytas ut successivt.

1.6 Successiv reviewing

En förnyelseprocess skall drivas enligt en vald strategi. Strategins olika delar skall vara motiverade av olika effekter som strategin skall resultera i.

En systemförnyelseprocess kan ta flera år i anspråk. En mängd olika delaktiviteter kan ingå.

Men olika förutsättningar och antaganden kan ändras under arbetets gång. Detta måste beaktas. Vi måste undersöka att de olika åtgärder som ingår i strategin fortfarande är lika adekvata med tanke på ny kunskap som har uppnåtts. Finns det anledning att ändra på strategin och på vad som planeras ingå i den? Har det hänt saker i verksamheten?

Gäller nya omgivningsfaktorer och affärsmål som föranleder oss att anpassa den strategi som vi har valt för arbetets bedrivande?

Hela strategin kan behöva omprövas eller så är det endast mindre detaljer som behöver justeras. T ex kan en viss verksamhetsregel ha blivit mindre viktig, eftersom marknaden har förändrats och helt nya tjänster måste sättas in för att möta konkurrensen.

I andra fall har kanske en helt ny plattform vuxit fram och blivit ett alternativ som man vill gå in för. Den fanns inte tillgänglig under den beslutsprocess som genomgicks för att hitta en strategi för reengineering av ett antal system.

En ny plattform kan t ex innebära att sättet att bedriva reverse engineering samt framförallt sättet att genomföra reengineeringsarbetet helt har förändrats.

Förnyelse av informationssystem innebär att ett aktivt förnyelsearbete genomförs. Förändringar i och utanför verksamheten gör att förändringsarbete i form av förnyelse av informationssystem inte kan genomföras på ett mekaniskt sätt.

Olika bedömningar som görs av vad som kommer att hända i verksamheten är förknippade med osäkerhet vid ett visst tillfälle. Om kunskapen ökar och genom att man skapar bättre bedömningar om vad som kommer att hända ger detta anledning att se över vilken strategi för förändringsarbete som man vill tillämpa.

2. Begrepp

I denna rapport används begreppet förnyelse av informationssystem. Om än inte en direkt översättning av engelskans reengineering, utnyttjas det med denna eller liknande innebörd på många ställen i rapporten. Detta är dock inte fullständigt konsekvent genomfört utan vi har tillåtit oss en viss frihet i användningen.

Med systemförnyelse avses en aktiv förändring av informationssystem i syfte att förbättra dem i olika avseenden. Det är fråga om en aktiv förändring och omstrukturering.
Detta till skillnad mot reverse engineering av informationssystem som inriktar sig på att ta reda på vilken status som systemen egentligen har.

3. Olika aspekter av förnyelse

Förnyelse av informationssystem aktualiserar i hög grad två aspekter:

1) Att lägga till nya egenskaper till informationssystem.

   Med nya egenskaper avses sådana egenskaper som den användande miljön upplever dem. Dessa egenskaper måste ofta infogas för att t ex nya affärsmässiga mål skall kunna klaras i verksamheten. Systemen måste kunna representera kunskap om nya saker, ha ny typ av funktionalitet för att kunna söka efter information, etc.

2) Att systemen skall nyttja ny typ av teknologi.

   Informationsteknologin går hela tiden framåt. Nya programmerings- och realiseringsspråk blir tillgängliga. Nya typer av databashanterare introduceras.

Kunskap blir möjlig att representera med hjälp av nya media som ingår i informationssystemen. Nya gränssnitt blir möjliga att utnyttja, etc.

Ny teknologi kan innebära att informationssystem blir lättare att rätta, underhålla och vidareutveckla. Detta är anledningar till att man ibland strävar efter att förnyelse också skall medföra att nyare teknologi nyttjas i en förnyelseprocess.

4. Hur undvika nya arv

När man sysslar med förnyelsearbete vill man förbättra äldre system i olika avseenden. En aspekt som är viktig att beakta, när man gör bedömningar om vilken strategi som är lämplig för förnyelse, är att undvika att man skapar ett nytt arv när man förbättrar informationssystem i något avseende.

Ett klassiskt exempel på detta är när man strävar efter att skapa ny funktionalitet hos ett system så att man t ex kan realisera nya affärsmässiga möjligheter. Man bygger skyndsamt på och ändrar befintlig kod så att de nya funktionerna kommer till, men tidspressen gör att man inte är tillräckligt omsorgsfull med hur man gör detta. Olika hopp görs i koden. Strukturen blir dålig i programmen. Knapphändig kommentering görs av systemen. Inga beskrivningar görs av vad olika programdelar gör och hur man accessar data. Gamla delar som man ser är dåligt strukturerade har man inte tid att göra något åt, man bara kopplar dessa till den nya och justerade koden.

Resultatet blir att man har ett system som för tillfället kan göra vissa nya saker, men som har en kodstruktur som är ännu mer komplex är den man hade tidigare, och en ny person har ännu svårare att komma in i koden och göra justeringar.

Ett annat exempel kan vara att man lyfter över all funktionalitet i ett CASE-verktyg, som inte på något sätt är öppet eller bygger på accepterade standards. Meningen är att man inte skall gå in i den av verktyget genererade koden, utan man kan bara arbeta med systemen genom de beskrivningsspråk som verktyget bygger på och som är av leverantörsspecifik art.

I båda dessa fall löper man stor risk att bygga upp ett nytt arv som kan vara svårt att arbeta med och vidareutveckla på längre sikt. I det första fallet är koden fortsatt dålig och kanske ännu svårare att underhålla än den var innan, i det senare fallet kanske leverantören upphör med sin verksamhet.

Vi berör alltså här frågan om att vid förnyelsearbete också lägga en grund som gör att fortsatt förnyelse- och förbättringsarbete skall kunna genomföras bättre och rationellare än idag.

Ett sätt att undvika nya arv är att skapa plattformar och systemarkitekturer som bygger på öppna system i olika avseenden. Olika delverktyg som nyttjas i arkitekturen skall bygga på beskrivna teknologier och skall kunna fås att kommunicera med varandra.

Exempel på sådana delverktyg är databashanterare och gränssnittsbyggare. Specifikationer som skapas skall beskrivas i kända beskrivningsspråk. Hur olika delverktyg kommunicerar med varandra samt vad som görs och vilka objekt eller begrepp som hanteras skall beskrivas.

Sådana beskrivningar kan göras i ett beskrivningsverktyg (repository-verktyg) som bygger på kända tekniker och som kan kommunicera via beskrivbara kommunikations-gränssnitt.

Man bör vara vaksam på verktygsmiljöer som bygger på helt egna principer. Dessa miljöer kan bli kortlivade. Nya teknologier kan dyka upp som blir mer slagkraftiga.

Nya teknologier kommer alltid att dyka upp. Man kommer alltid att befinna sig i en migreringssituation, men det viktiga är att man både kan beskriva nya och gamla arkitekturer samt hur man kan kommunicera mellan dessa miljöer. Då kan man också skapa sig realistiska möjligheter till en successiv migrering som man har möjlighet att överblicka och klara av.

Detta innebär inte att man inte skall satsa på sammanhållna miljöer, men man måste kunna beskriva sätt där den nya miljön kommunicerar med komponenter som organisationen har till äldre befintliga systemdelar.

5. Förnyelse av människor i verksamheten

Förnyelse av informationssystem är ingen mekanisk aktivitet. Denna process involverar i hög grad människor. Många hävdar att en av de mest svåra frågorna i samband med migrering är förmågan att "förnya" människorna som ingår i verksamheten.

Om inte människorna i verksamheten vill skapa förnyelse och förändring, blir det ingen förändring. Även om beslutsfattare skapat ett beslut om att man skall genomföra förnyelse, kommer ingen sådan förnyelse att ske om inte människorna i verksamheten delar denna syn och detta behov.

Det finns en mängd orsaker till att det kan saknas vilja att delta och se till att förnyelse genomförs:

- Man förstår inte motiven till en förändring. Om man inte inser behovet av eller motiven för en förändring ser man ingen orsak till att delta i förändringen.

- Man vill inte delta i en förnyelse. Av olika skäl är man inte motiverad att delta i en förnyelse- och förändringsprocess. Olika förhållanden i verksamheten medverkar till att olika medarbetare inte ser sig motiverade att delta i en förändringsprocess som en förnyelseprocess ofta är. Det kräver ansträngning.

- Man vill inte lära något nytt. Man tycker att man kan vissa arbetsuppgifter väl och man inser inte, eller vill inte inse, att man måste lära nya saker och hitta nya sätt att sköta uppgifter i verksamheten.

- Man anser inte att man kan lära något nytt och de enda uppgifter man kan arbeta med är de som man just nu arbetar med. Man tror inte att man har förmåga att lära nya eller andra sätt att sköta arbetsuppgifterna.

I alla dessa aspekter är teknologin trots allt inblandad på något sätt. Teknologiutveck-lingen ändrar på olika förutsättningar. Man kan genomföra uppgifter på ett nytt sätt eftersom ny teknologi har blivit tillgänglig. Teknologiutvecklingen medför också att

nya plattformar kan skapas så att hela systemförändringsarbetet kan genomföras på ett annorlunda sätt.

Det är alltså inte bara människor som använder informationsteknologi i sina verksamheter som ofta har en tvekan att lära nytt och på detta sätt delta i en förnyelseprocess, det är också i hög grad personer som arbetar med informationsteknologin i sig som har problem att lära nya typer av plattformar och nya sätt att skapa och förändra system. Beteendet och attityderna kommer tillbaka.

Det är lätt att komma i en försvarsposition och tycka att det vi har minsann klarar en hel del – det behöver inte förnyas. Man rationaliserar ett visst sätt att arbeta.

Människors förmåga och möjligheter att lära nya sätt att arbeta och att använda ny typ av teknologi är ofta underskattade. Det finns för många miljöer där man räknar med att en viss generations människor måste till för att man skall få en reell förnyelse till stånd.

Om man ger människor möjlighet att lära nya referensramar och, framför allt, kunna relatera till gamla referensramar, finns det troligen större möjligheter för att personer som arbetar på ett sätt som man vill förnya inser att det finns goda möjligheter till att lära nytt.

Människor har alltså ett arv i sitt sätt att tänka och arbeta och man måste också låta människor få en chans att migrera. Denna process tar tid, den måste också styras.

Om man skall få dessa processer att fungera på ett bra sätt måste man ha medvetna och målinriktade förändringsprocesser. Olika strategier måste vara klara och formulerade. Olika tveksamheter i svar och inriktning kommer bara att medverka till att olika argument som skapar dålig motivation kan florera.

De mänskliga migreringsprocesserna måste bli mer medvetna och aktiva.

6. Referenser del I

[1] CDIF: CASE Data Interchange Format – Overview, EIA/IS-106, Interim Standard, 1994.

II Metodramen

Detta avsnitt innehåller en mer detaljerad redogörelse för de olika metodramsstegen. För varje steg diskuteras också vilka olika metoder och delverktyg som kan vara aktuella.

1. Verksamhetsanalys i samband med systemförnyelse

I verksamhetsanalyssteget skapar man utgångspunkterna för förnyelsearbetet. När man går in och börjar arbeta med verksamhetsanalysen finns egentligen bara en hypotes om att det är en förnyelseprocess som eventuellt skall bedrivas för att uppnå olika effekter.

I verksamhetsanalyssteget tar vi reda på om dessa antaganden är riktiga och vilka åtgärder som behövs för att ta tillvara olika möjliga effekter.

Det gör man genom att intressera sig för vilka mål verksamheten har och om det finns anledning att bedriva någon form av förändringsarbete för att hitta nya målvärden eller för att se till att de effekter infinner sig som man har räknat med.

De olika analysområden som bör ingå i en verksamhetsanalys skulle kunna vara följande:

- Analys av vilka nya verksamhetsresultat (produkter) som måste skapas
  – Vad är det specifikt som förorsakar tankar på förnyelse?
- Hur verksamheten skall styras
  – Autonom styrning, central styrning?
- Önskad struktur på verksamheten
  – Hur ser verksamheten ut?
- Önskad struktur och egenskaper på det förnyade systemet
  – Specifikt på vilket sätt måste systemet bli bättre?
- Hur affärsutvecklingen skall bedrivas
  – Informationsförsörjning?
- Slutsatser och analys av "business intelligence".

För att kunna bedriva en effektiv verksamhetsanalysprocess behöver man goda beskrivningstekniker. Detta inte minst för att effektivisera kommunikationen mellan människor. Det gäller både beskrivning av existerande verksamheter och informationssystem, men också nya verksamheter och förnyade informationssystem.

Exempel på några typer av beskrivnings- eller modelleringstekniker som kan vara användbara är följande:

- Begreppsmodell, som är oberoende av lagringsform
- Regelmodell
- Funktionsmodell
- Processmodell

- Rutinmodell
- Övergångsmodeller för lagring
- Övergångsmodeller för operationers koppling mot program.

Verksamhetsanalysen skapar en bas för vilket förnyelsearbete som skall bedrivas. Kanske skall man inte förnya informationssystem alls. Om man kommer fram till att det finns effekter att hämta om man förnyar informationssystem, så inriktar man sig här på vilken typ av förnyelse man kommer att behöva.

Det finns flera metoder som har en riktning som denna. Några exempel är F3-metodiken och MBI-metodiken .

1.1 F3-metodiken

F3, ett EU-projekt inom Esprit-programmet, hade till uppgift att ta fram en modern, effektiv metod för att fånga in, beskriva och validera krav på informationssystem [1-4, 6-7].

Namnet F3 (eller Fcube), som står för ”From Fuzzy to Formal”, sammanfattar den ofta intrikata naturen i kravanalysuppgiften där det gäller att gå från det "luddiga" – de ofta vaga initiala behoven och önskemålen – till den mer formella definitionen av kraven.

Projektet pågick under 30 månader och avslutades på nyåret 1994. Förutom SISU deltog 10 andra parter och sammanlagt 6 europeiska länder var representerade i projektet. Koordinator för projektet var SEMA Group, Frankrike.

Sveriges (SISUs) huvudsakliga uppgift i projektet var att ta fram en metod och ett datorstöd för verksamhetsanalys ("Enterprise Modeling").

F3s verksamhetsmodell (Enterprise Model) används för att beskriva och kommunicera uppfattningar om en verksamhet. Man skapar en modell av verksamheten med fokus på den eller de aspekter som är speciellt intressanta. Några av de viktigaste perspektiven är de statiska, dynamiska och intentionella perspektiven, det vill säga verksamhetens begrepp, aktiviteter och mål.

I F3s verksamhetsmodell ingår därför fem olika delmodeller, var och en inriktad på att beskriva verksamheten ur ett speciellt perspektiv: målmodell, begreppsmodell, aktivitetsmodell, aktörsmodell och informationssystemkravmodell.

Målmodellen upprättas i syfte att klarlägga frågeställningar som: Varför finns verksamheten, vad är det vi vill uppnå? Vilka är de övergripande/strategiska målen? Vilken målstruktur har vi, vilka delmål kan vi se? Vilka är verksamhetsmålen, förändringsmålen? Vilka problem finns i verksamheten, vad är deras orsaker? Vilka möjligheter kan vi se för verksamheten och vilka är förutsättningarna för dessa?

I målmodellen visas två viktiga typer av samband: influens- och motiveringssamband. Influensrelationerna uttrycker påverkansrelationer mellan olika komponenter i målmodellen. Den situation som beskrivs av den ena komponenten påverkas av den situation som beskrivs av den andra. Influensen kan vara positiv eller negativ, graden kan vara mer eller mindre stark. T ex kan ett problem förhindra uppfyllandet av ett existerande mål. I detta fall har det aktuella problemet en starkt negativ influens på målet.

Ett motiveringssamband uttrycker existensmotivering mellan komponenter i målmodellen. En komponents existens motiveras av en annan komponents existens. Sambandet kan naturligtvis vara mer eller mindre starkt.

Begreppsmodellen fokuserar på vad det är som hanteras i verksamheten. Vilka ”saker” och ”fenomen” är av intresse? Vilka är verksamhetens ”objekt”? Vad är de olika begreppens definitioner, vad innebär de? Vilken gemenskap har vi om idéer, språk och terminologi i verksamheten? Menar vi samma sak, har vi olika uppfattningar?

Aktivitetsmodellen beskriver hur verksamheten går till – hur man vill att den skall gå till.
Vilka arbetsuppgifter finns? Vad produceras? Åt vem? Den beskriver kommunikation och samspel i form av olika material- och informationssamband mellan aktiviteter, informationsbehov som finns i olika affärsförlopp.

Aktörsmodellen beskriver olika typer av aktörer i verksamheten och deras inbördes samband. Det kan vara organisatoriska enheter, den organisatoriska strukturen, det kan vara roller, och rollers fördelning på organisatoriska enheter. Mänskliga individer ”spelar” eller utför olika roller, etc.

Informationssystemkravmodellen används för att formulera och analysera mål, problem och krav för det informationssystem som skall utformas. Krav på informationssystemet kan förfinas till funktionella och icke-funktionella krav.

I IS-kravmodellen beskrivs:

– IS-mål, som uttrycker intentioner för infosystemet och/eller dess delsystem/-komponenter. De kan uttryckas som mätbara eller icke-mätbara egenskaper, mål, visioner eller riktlinjer. Exempel: ”Målet för IS är att förbättra produktiviteten i remitteringsaktiviteten med 30%”

– IS-problem, som uttrycker oönskade tillstånd för verksamheten eller dess omgivning eller problematiska fakta om den nuvarande situationen allt m a p det infosystem som skall utvecklas. Exempel: ”Ett problem är att systemets framtida användare har mycket begränsad tid att delta i utbildning om hur systemet handhas och används”

– IS-krav, som uttrycker ett krav på en speciell egenskap som det tänkta infosystemet skall uppvisa. Exempel: ”Svarstiden för frågor från processen AUM-003 avseende kundsaldon måste vara kortare än 2 sek i 95% av alla fall förutsatt en belastning av 500 transaktioner/sek.”

Funktionella krav uttrycker ett definitivt krav avseende en funktionell egenskap hos informationssystemet eller något av dess delsystem. Icke-funktionella krav uttrycker alla slags krav, begränsningsregler och restriktioner, som inte är funktionella, och som avser infosystemet som skall utvecklas eller utvecklingsprocessen. Icke-funktionella krav refererar ofta till hur funktionella krav realiseras i systemet. Infångandet av icke-funktionella krav kan därför ofta underlättas genom analys av funktionella komponenter.

I informationssystemkravmodellen används samma typer av relationer som i målmodellen: influens- och motiveringssamband. Mål motiverar andra mål och krav, problem influerar mål, etc.

"Inter-modellära" samband
De olika delmodellerna anlägger och avgränsar olika specifika perspektiv på verksamheten. Men vi ser omedelbart att det också finns behov av att kunna beskriva ytterligare samband som går mellan komponenter som återfinns i olika delmodeller.

Ansvar och befogenhet t ex är viktiga begrepp som måste kunna beskrivas i en verksamhetsmodell. Ansvar innehas av en aktör, avser ofta en aktivitet och mäts mot ett mål. Befogenhet avser aktörers rätt och möjlighet att disponera olika typer av resurser (begrepp). Aktiviteter utförs av aktörer, målvärden sätts mot egenskaper, som fenomen och förhållanden som hanteras i verksamheten, innehar, o s v.

För att upprätthålla spårbarhet finns vidare ett antal viktiga inter-modellära samband mellan komponenter i IS-kravmodellen och komponenter i andra delmodeller. Så t ex bör ett IS-mål alltid vara motiverat av antingen ett mål (i målmodellen) eller en aktivitet (i aktivitetsmodellen).

Kraven kan länkas direkt till och förfinas ytterligare i den s k "Target"- eller NFR-modellen och indirekt till IS-modellen. NFR-modellen (Non-Functional Requirements) och IS- (informationssystem) modellen är två andra modeller som ingår i F3-ansatsen, men som ligger utanför verksamhetsmodellen.

I NFR-modellen representeras relationer och beroenden som finns mellan icke-funktionella krav så att inkonsistenser och konflikter mellan dessa kan lösas upp. Andra viktiga syften med NFR-modellen är att möjliggöra spårbarhet av ursprung och motiv på verksamhetsnivån för icke-funktionella krav. Det är också att möjliggöra referens av utvärderingsresultat (mätningar) mot önskade resultat. Detta gör tillsammans att implikationer av ej uppnådda eller icke uppnåbara krav kan följas från utvärderingsnivån till verksamhetsnivån.

Icke-funktionella krav uttrycks i kvantitativa termer så att förutsägelser och spårning kan göras. Ett ”atomiskt” icke-funktionellt krav t ex refererar alltid till en viss typ av fakta och uttrycks som en trippel: "<target, property, measurement>" d v s på en komponent i målsystemet, en egenskap som denna komponent skall uppvisa och det mått och måttenhet som skall gälla för egenskapen ifråga.

F3-metoden i ett systemförnyelseperspektiv
F3-metoden fokuserar alltså på analys, verifiering och validering av krav och ger bl a stöd för kommunikationsprocessen genom projektgemensamma verksamhetsmodeller.

Om en kravspecifikation upprättas enligt metodens intentioner blir kravspårbarhet möjlig genom direkt koppling från design via systemkrav till verksamhetsprocesser och verksamhetsmål.

Som uttryckts ovan är det i verksamhetsdelen som utgångspunkterna för ett förnyelse-arbete skapas. Målen, begreppen, aktiviteterna/uppgifterna, aktörerna i verksamheten och deras samband analyseras och beskrivs och ger tillsammans motiven till de krav på egenskaper som det förnyade systemet skall uppvisa.

Om, omvänt, det existerande systemet uppvisar dessa och dessa egenskaper, vilka blir då implikationerna för önskade effekter i verksamheten? Vilka mål som verksamheten har och vilka möjligheter som verksamheten ser kan därmed komma att påverkas negativt?

Det är uppenbart att krav på förnyelse av informationssystem uppstår när de gamla systemen inte längre upplevs kunna tillgodose de behov man har i verksamheten – det finns en "mis-match" mellan systemets egenskaper och verksamhetens behov av systemstöd. Ibland kan det vara systemens funktionalitet som inte lever upp till vad man förväntar sig, men ofta är de gamla systemen väl beprövade och "det de gör, det gör de bra".

Inte så sällan berör systemens otillräcklighet olika typer av icke-funktionella egenskaper. Producerad information uppvisar för dålig aktualitet, de säkerhetsmässiga aspekterna är inte tillräckligt tillgodosedda, användbarheten är nedsatt genom alltför ålderdomligt människa/maskin-gränssnitt. Deras arkitekturella och strukturella egenskaper gör dem mindre lämpade och möjliga att bygga ut och modifiera, deras slutenhet gör dem icke-kommunikativa, etc.

Det torde vara uppenbart att en extensiv och noggrann verksamhetsbeskrivning och analys, där också systemens funktionella och icke-funktionella egenskaper penetreras relaterat till detta, bör förstärka möjligheterna att hamna rätt, att angripa rätt saker i ett förnyelsearbete.

1.2 MBI-metodiken

MBI-metodiken (Mål, Beslut, Information) är en metodik som används för att bedriva verksamhetsanalys [5]. Det innebär att man utröner vilka mål en verksamhet skall ha, vilka uppgifter som behöver genomföras i verksamheten, hur den skall styras och informationsförsörjas. Genom att genomföra en sådan analys har man en god grund för att avgöra vilken inriktning som en förnyelseinsats skall ha.

Som tidigare påpekats kan en förnyelseinsats genomföras på en mängd olika sätt och en förnyelseinsats kan få helt fel inriktning.

Man kanske kommer fram till att informationssystemen överhuvudtaget inte skall förnyas, utan man måste bygga nya informationssystemdelar för att få önskad effekt. Ett underlag för en nyttobedömning avseende olika förändringsalternativ är ett resultat som kommer ut från verksamhetsanalysen.

Metodiken inriktar sig på hur hela verksamheten skall bedrivas. Som ett resultat av detta kan man bättre avgöra vilken informationsförsörjning som verksamheten egentligen behöver. Denna analys kan sedan leda till vad som behöver förnyas eller förändras på annat sätt.

Metodiken fokuserar på ett antal aspekter av verksamheten som man bedömer är viktiga för att få den inriktad på rätt sätt.

En av dessa aspekter är de uppgifter som skall definieras och avgränsas i verksamheten. Detta styrs i stor omfattning av vad som skall utföras i verksamheten. I detta arbete

engageras i hög grad ledning och beslutsfattare på olika nivåer. Man vill uppnå en situation där olika aktörer är medvetna och eniga om vilka uppgifterna är.

Uppgifter samlas till enheter som får en relativt autonom styrning vad avser de egna besluten och hur man använder de egna resurserna för enheten.

Uppgifternas definition baserar sig i sin tur på vilka mål verksamheten skall ha. Målen definieras av ledningen, men mål som finns på lägre nivåer måste stå i samklang med överordnade mål av olika typ. Analyser av hur mål uppfattas på olika nivåer blir därför aktuell.

När uppgifterna är definierade i olika enheter kan man definiera de olika beslut som måste kunna tas och information som krävs för att dessa beslut skall kunna fattas. Besluten kan beskrivas så väl att informationsbehoven blir möjliga att definiera på en noggrann nivå.

I metodarbetet ingår också att definiera vilka effekter i form av måltermer som man räknar med att kunna uppnå i verksamheten, bl a till följd av olika förändringsåtgärder.

Som en del i detta arbete ingår att uppskatta vad olika förändringsalternativ i form av nybyggnation och andra åtgärder kan ge i form av effekter. Detta blir en viktig del av nytto-/uppoffringsanalysen och vilka typer av förnyelseåtgärder som kan bli aktuella.

På denna punkt ser vi alltså hur en verksamhetsanalytisk metod kan kopplas samman med uppbyggnaden av ett förnyelsearbete.

Metodiken har följande skeden och steg:

Skede 1
- Avgränsning av verksamhetsområde.
- Beskrivning av verksamheten. Funktionsstruktur.
- Analys av problem och möjligheter.
- Beslut om prioriterade funktioner.

Skede 2
- Påverkbara förhållanden. Bedömning av intäktspotential.
- Samordning av operativa mål – beslutsstruktur.
- Beslutsanalys.

Skede 3
- Beskrivning av informationsbehov. Slutlig avgränsning av informationssystem.
- Värdering av systemalternativ.

2. Systemgranskning

För att bygga upp ett förnyelsearbete som blir målinriktat och ger effekter som man kan uppfatta som meningsfulla, behöver man främst uppfylla två saker:

1) Man vet vilken verksamhet man skall förnya, vilka målen är och vad vill man åstadkomma i verksamheten.

2) Man vet hur de nuvarande systemen ser ut och hur de borde se ut för att realisera olika mål i verksamheten, ta tillvara olika affärsmöjligheter som man ser i verksamheten, o s v.

Den första punkten attackeras i verksamhetsdelen av metodramen, den andra i denna del, systemgranskningsdelen.

Det är viktigt att man systematiskt tar reda på hur de existerande systemen verkligen ser ut. Diskussioner i olika läger och olika tyckande får inte styra beslutsfattandet på ett ad hoc-artat sätt.

Ett förnyelsearbete kan orsaka stora kostnader och kommer att förbruka resurser under flera år. När man beslutar om en strategi för att förbättra ett systemarv är det därför väsentligt att det är korrekt bedömt, så att man inte råkar ut för överraskningar på denna punkt när förnyelsearbetet sätter igång.

Det finns ett antal frågor som bör besvaras under systemgranskningsarbetet.

1) Vad upplevs som problem med systemet idag?

   Hur stämmer detta med måldelen för verksamheten?
   Främst gäller detta de yttre egenskaperna för informationssystemet. Vad saknas i systemet idag – både ifråga om funktionalitet och representationsformer?

2) Har systemet kända logiska fel?

   Vilka fel vet man om i det system som skall förnyas? Har det uppenbarats fler fel på senaste tiden, fel som är besvärande för verksamheten och som man har varit osäker på orsaken till?

   Hur har felen uppstått? Har systemens komplexitet varit en bidragande orsak till dessa fel eller vilka faktorer har spelat in?

3) Hur väl är systemet beskrivet och dokumenterat på olika nivåer?
   Går systemet att förstå för en person som ej har arbetat med det på senare tid?

   Det är av stort värde om ett informationssystem är väl beskrivet och dokumenterat. Det är annars mycket svårt för personer som inte har varit med om att bygga eller underhålla ett visst informationssystem att förstå vad olika delar av systemet gör.

   Beskrivning behövs på olika nivåer. Vissa beskrivningar är överordnade, man beskriver vad ett system kan göra utan att beröra tekniska lösningar. Andra nivåer har att göra med t ex vilka databashanterarprodukter som har valts för att lagra och komma åt data.

   Minst tre typer av informationssystembeskrivningar bör finnas:
   – logiska beskrivningar av vad systemet gör och vilken information det kan representera
   – delar som beskriver presentationsformer, människa/maskin-gränssnitt och interaktion
   – beskrivningar av vad som lagras och hur man lagrar det.

   Vilken typ av beskrivningar är det speciellt som fattas?
   Vilka effekter skulle man kunna nå om systemet beskrevs bättre?

4) Är systemet komplext?

   – storlek genom delar och storlek i varje del
   – onödig komplexitet genom dålig struktur

   I denna fråga ligger en hypotes om hur informationssystem strukturmässigt bör se ut. Om man har ett system som är väl strukturerat med olika delar som anropas på ett överblickbart och tydligt sätt så har man ett system som är lättare att underhålla, vidareutveckla och förändra.

   Om man å andra sidan har ett system som är dåligt strukturerat genom att det har delar som är dåligt definierade, det har påbyggda delar där det är oklart vad varje del utför och hur de olika delarna anropas, har man ett system som är svårt att underhålla och vidareutveckla.

   I detta ligger också följande. Om man har ett system som är dåligt strukturerat så upplevs det som komplext genom att man har svårt att förstå vad som händer i programmet. Om man å andra sidan har ett system som har en god struktur kan det ofta utföra komplexa saker, men man upplever fortfarande inte systemet som svårt att förstå eller svårt att förändra.

   Det finns sätt att mäta systems komplexitet enligt mer entydigt definierade metoder. Om man har sådana mättekniker kan man göra mer meningsfulla jämförelser av komplexitet och uppfatta olika mätresultat mot en gemensam referensram.

5) Hur lämpligt är systemet för framtida underhåll och förändring?

   Om man står inför ett avgörande av huruvida man skall förändra ett system eller inte gör man dels en bedömning av om man kan göra omedelbara förändringar så att man kan ta hem vissa effekter på kort sikt. Dels måste man också göra bedömningar av huruvida man kan fortsätta att underhålla och vidareförädla informationssystemet på längre sikt. Man kommer då bl a in på bedömningar av hur väl dokumenterat och hur väl strukturerat systemet är. Detta har berörts under särskilda punkter ovan.

   En annan mycket viktig fråga är huruvida man vill arbeta med andra plattformar och andra referensramar (kunskapsramar) framgent. Om skillnaden är stor mellan dagens system och de framtida plattformar som planeras, kan ett större steg behöva tas och förnyelsen blir mer omfattande. Detta även om nuvarande system är väl strukturerat. Å andra sidan kan man anse att ett välstrukturerat system blir lättare att förändra till ett förnyat system på en annan plattform än det eljest hade varit. Detta eftersom reverse engineeringsarbetet blir lättare.

6) Hur lämpligt är systemet ifråga om

   – vad det skall producera?

   Är de nya tjänster som systemet skall producera väsentligt annorlunda än de som systemet producerar idag? Vad fordras för att de nya eller förändrade resultat som systemet skall producera skall kunna realiseras? Vilka ombyggnader krävs ?

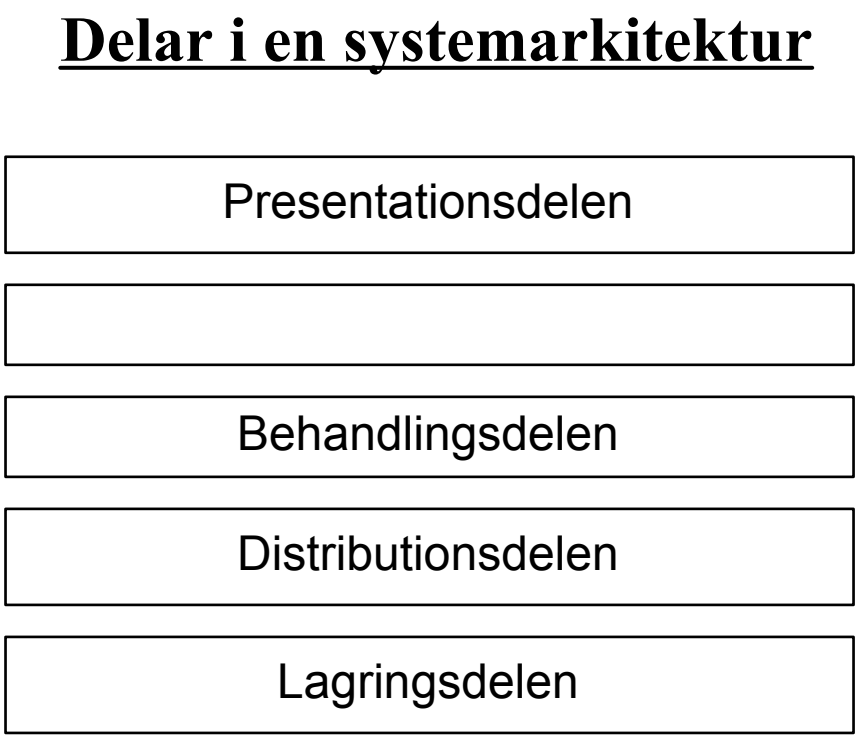
   – styrningskonformitet?

Om verksamheten skall styras annorlunda, hur väl passar den existerande system-strukturen med de nya styrformerna? Antag t ex att en verksamhet nu skall styras med hjälp av avgränsade, i hög grad självstyrande delar. Detta från dagens mer centrala styrformer, där alla beslut skulle upp centralt för beslut. Om man decentralt skall kunna fatta beslut och styra mot mål och följa upp mål självständigt måste också informationssystemen vara anpassade till detta. Systemen måste vara avgränsade till den nya verksamheten. Det är svårt att hantera kompromisser här.

– struktur?

En annan aspekt som är viktig för ett informationssystem är vilken struktur det har. En strukturaspekt är den som behandlats ovan, nämligen hur systemet är uppdelat i förhållande till den verksamhetsstruktur som man strävar mot. En annan aspekt är den ”konfigurationsstruktur” det har. Med detta avses hur systemet är strukturerat med avseende på olika grundfunktionalitet som alla system mer eller mindre har.

Låt oss betrakta följande bild för att kunna diskutera arkitektur för ett system:



Figur 1

Äldre informationssystem har inte denna arkitektur. Modernare arkitekturer har ofta mer eller mindre släktskap och kan diskuteras efter denna indelning. Modellen kan användas för att diskutera äldre arkitekturer såväl som modernare – till vilka man vill migrera informationssystem.

Nedan presenteras de olika delarna närmare:

- *Presentationsdelen*. Den del som presenterar data för användaren. Det kan ske via ett fönstergränssnitt, på papper eller via multimedia.

- Den andra delen är *säkerhetsdelen*. I denna del omhändertas olika funktioner som har med säkerhet att göra. I nyare system höjs kraven kring säkerhet. I äldre system kan säkerhet ha hanterats mer rudimentärt. Olika externa säkerhetspaket kan anropas från denna del. Sådana paket kan köpas från externa leverantörer. Olika ambitioner för säkerhet kan realiseras inom ramen för denna modul. Det kan vara lämpligt att realisera säkerhetsfunktioner på ett samlat sätt så att man inte ”skvätter ut” olika säkerhetsfunktioner i olika delar av koden när man vill ändra i dessa funktioner.
- *Behandlingsdelen*. I denna del beskrivs de olika beräkningar och operationer som görs med data. Data kan ofta baseras på viktiga verksamhetsregler som finns i verksamheten och som är viktiga att ha klart för sig ifrån verksamhetens sida sett.

- Den fjärde delen är *distributionsdelen*. Här ombesörjs kommunikation med andra system. Speciellt med starkt decentraliserade och nedbrutna systemstrukturer är denna del väsentlig. Man måste välja ut ett sätt att kommunicera med omgivande system. Ett exempel kan vara kommunikation via meddelanden eller kommunikation med access till gemensamma kommunikationsobjekt. I såväl äldre system som nyare system kan kommunikation realiseras genom access mot gemensamma databaser.

- *Lagringsdelen*. Ombesörjer lagring av data. Det kan ske via olika typer av fil- och databashanterare, o s v. Det viktiga är att lagring sker på ett logiskt sätt sedan olika händelser inträffat för olika objekt i verksamheten, etc. Lagringsdelarna kan också omfatta delar som ombesörjer lagring av data på ett dataeffektivitetsmässigt bättre sätt, men som kan te sig en aning ologiskt från ett verksamhetsmässigt betraktelsesätt. Sådana anpassningar bör man dock hålla i speciella hanterings- eller anpassningsdelar så att man inte sammanblandar dessa med de mer logiska lagringsoperationerna på data. Detta beroende på att om man sedermera vill byta databashanterare så skall man inte behöva ändra i alltför många koddelar.

Detta är struktureringsaspekter som kan bli aktuella i en förnyelse och vid bedömning av hur det existerande systemet ser ut härvidlag.

Det kan innebära att man vid en förnyelse vill förbättra strukturen i dessa avseenden. Om man bygger till eller anpassar systemet med addering av ny funktionalitet kan man behöva förbättra systemdelar så att man strävar mot en strukturering av ovanstående typ. Det innebär att man utnyttjar en målarkitektur.

– gränssnitt?

Gränssnittet är något som användarna finner alltmer viktigt. Användarna ser i sin omgivning moderna gränssnitt av fönstertyp eller som omfattar multimedia. Man kommer in på tankar om att denna typ av gränssnitt är intressant för den egna arbetssituationen. Det blir alltmer ett allmänt intresse av att äldre system får mer moderna gränssnitt.

I systemgranskningssammanhang kan det bli aktuellt att försöka bedöma om det är motiverat att ha gränssnitt av mer modern typ. Vilka effekter kan man få ut av det? Olika funktioner som t ex "cut and paste" kan vara intressanta och innebära en effektivisering i arbetet genom att förnyad inmatning undviks.

Granskningsaspekter beträffande gränssnittet aktualiserar ytterligare frågeställningar som har med själva arbetssituationen att göra.

Det är inte bara att bygga om äldre system av transaktionsorienterad typ med avseende på gränssnittet. Ett modernt gränssnitt innefattar ofta en föreställning om att man skall kunna mata in och hantera händelser allteftersom de uppstår i verksamheten. Ett transaktionsorienterat system klarar ofta inte detta. Det förutsätter istället att transaktionerna kommer i en viss ordning så att konsistens i systemet kan upprätthållas. En modernisering avseende gränssnittet kräver i detta fall en större insats.

Det finns numera etablerade tekniker att skapa moderna gränssnitt till äldre system där detta och det förnyade systemet har någorlunda samma tidsorientering. Vissa grundpaket finns att anskaffa för att underlätta arbetet. Skall systemet omarbetas från ett batch-system till ett mer interaktivt system, krävs dock mer strukturella omarbetningar.

7) Förändringsplattform för framtiden?

En annan aspekt av de existerande systemen som är viktig att granska innan förnyelsearbetet påbörjas är den plattform som systemen nyttjar. Dels är det viktigt att ta reda på exakt vad denna plattform består av, vilka kompilatorer o s v, som används. Dels är det viktigt att göra klart vilka plattformar man vill gå emot under förnyelsearbetet och de målplattformar som förnyelsearbetet skall resultera i. Valet av plattform styrs naturligtvis av den arkitektur man vill ha på systemet i framtiden. Arkitekturkraven styr vilka plattformar som blir möjliga att använda.

Med plattform menar vi här:
- Operativsystem
- Kompilatorer
- Datalagringssystem som nyttjas och principer de bygger på, t ex relationsansatsen
- Kommunikationsteknologi
- Interaktionsformer för användarna
- Vertikal arkitektur för systemet.

Med arkitekturmässiga aspekter menar vi här företrädesvis de horisontella arkitekturaspekterna. Hur är systemet uppdelat i förhållande till de verksamhetsmässiga funktionerna och processerna? Har varje funktionell del sitt eget delsystem är eller är det klustrat på olika sätt så att ett större system stödjer många delprocesser eller funktioner?

När det gäller de plattformsmässiga aspekterna måste man göra klart för sig hur man vill att systemet skall se ut i framtiden. När man vet detta och när man vet hur systemet ser ut idag, kan man börja planera hur det skall förändras för att den nya arkitekturen skall uppnås.

Man måste göra klart för sig vad man vill uppnå med den nya typen av plattform och vilka förväntningar man har. I vissa fall kan man ha förväntningar som inte alltid går att uppfylla. Effekterna infinner sig inte. Det kan finnas många skäl till detta. Ett är kunskapsskälet. Man kanske helt enkelt inte kunde lära sig att utnyttja den nya plattformen, kunskap saknades i organisationen. Man kunde inte använda den nya plattformen på rätt sätt. För den objektorienterade plattformen, exempelvis, finns det

många utsagor om att dessa och dessa effekter kommer att infinna sig om man bara börjar använda denna plattform. Detta kan också vara föreställningar som inte kommer att infria sig. Det finns många olika myter om olika typer av plattformar.

Bedömningar om vilka effekter som är realistiska att förvänta att de kommer att infinna sig, på kort och på lång sikt, måste granskas på basis av adekvat kunskap.

Detta har ofta med de referensramar som en viss organisation har att göra. Vilken kunskap har man idag? Vad fordras för att en viss grupp medarbetare skall lära om till en annan kunskap och referensram? Om man inte uppnår denna kunskap kommer man att felanvända eller inte att förstå de nya arkitekturer och plattformsdelar som kan vara aktuella i en viss migreringsprocess.

3. Strategival

Metodramens strategivalsdel behandlar det steg där de viktiga besluten om på vilket sätt man skall förnya sitt informationssystemstöd skall göras.

Resultaten från steg 1 och 2, d v s aspekter från analys av verksamheten och vad man vill med denna samt analyser av hur informationssystemen ser ut idag, finns att utgå ifrån. Vilken status har systemen för att möta de nya kraven? Hur skall de förändras?

Beslut kring strategi för förnyelse kan innebära allt ifrån att inte göra någonting alls till att man skapar en nyutvecklingsinsats.

Grundtanken är att det är svårt att hitta ensartade angreppssätt för att förnya informationssystem. De åtgärder man måste sätta in beror i hög grad av vilka målen är och hur de existerande systemen ser ut.

För ett diskutera hur man kan formulera strategier och vilka åtgärder som kan bli aktuella kommer vi i det följande att diskutera ett annat typfall, eller scenarier, som kan illustrera hur dessa situationer styr valet och pekar på behovet av olika typer av åtgärder för att förnya informationssystemarvet.

3.1 Scenarier

De olika scenarierna har rubriksatts enligt:

1) Underhållskostnader
2) Delning av verksamheter
3) Autonoma delverksamheter
4) Nya verksamhetstjänster
5) Regionalisering
6) Extern påverkan, som lagar etc
7) Nya krav på egna processer
8) Arbetssätt och rutiner ändras
9) Ny utgångspunkt för förändring
10) Bas för återanvändning
11) Datorleverantören tar nya strategiska beslut
12) Krav på uppfattbarhet och människa/maskingränssnitt

13) Nya och gamla krav måste samexistera

Under varje rubrik diskuteras vad som kännetecknar de olika situationerna. Scenarierna har sin bas i olika målbilder, mer eller mindre från verksamheten. Syftet är också att peka på hur strategiformuleringarna bör kopplas till verksamhetsperspektivet.

3.1.1 Underhållskostnader

Ibland kan de höga underhållskostnaderna för äldre system vara skäl till att man vill förnya informationssystemen. Enligt vissa undersökningar [14] är det dock inte så ofta som underhållskostnaderna i sig är skälet till att man börjar diskutera systemförnyelse i verksamheterna, utan detta är ofta kombinerat med funderingar kring större förändringar som man misstänker kommer att ha konsekvenser också för informationssystemen.

En annan aspekt är vad man egentligen menar med underhållskostnader. Är det endast rättning av fel i system, eller är det också förändringsarbete till följd av ändrade eller utvidgade krav från verksamheten, eller är det en kombination av dessa?

Mindre förändringar av existerande funktionalitet brukar ofta räknas till underhållsarbete, medan utvidgningar av funktionalitet eller lagringsmöjlighet ses som vidareutveckling av informationssystem.

3.1.2 Delning av verksamheter

Det kan finnas andra anledningar till att man bygger upp en strategi för systemförnyelse. Ett scenario kan vara att verksamheter delas av olika skäl.

Man kan exemplifiera med vad som händer i tillverkningsindustrin. Utgångsläget har varit att man konstruerar de olika delarna som skall ingå i en komplex produkt, man producerar delarna, man sätter samman produkten och man ser till att den säljs till konsumenter.

I en framtid är det inte lika självklart att företaget konstruerar och tar fram alla komponenterna till den sammansatta produkten och/eller sätter den på marknaden.

Man kanske, till exempel, lägger vissa delar av verksamheten i separata företag. Det kan alltså vara en del i den egna verksamheten som blir ett eget företag. Syftet är att bedriva verksamheterna i delarna mer effektivt i olika avseenden. Ansvaret för att se till att de nya delverksamheterna kan klara sig själva blir mer markerat. Syftet är också att de nya verksamhetsdelarna får konkurrera med andra liknande företag om att erbjuda tjänster till huvudverksamheten. Detta ställer särskilda krav på de nya verksamheterna.

För att de nya verksamheterna skall kunna arbeta självständigt måste man också ha informationssystem som stödjer dem. I den gamla strukturen var de olika delverksamheterna mer integrerade. Det gällde också informationssystemen.

Skall man dela upp verksamheterna måste man också på något sätt dela på informationssystemen så att man kan få ett adekvat informationssystemstöd för varje verksamhetsdel. Detta är ett arbete som är omfattande. Det är heller inte helt uppenbart hur det skall gå till.

Det krävs dessutom mycket hög grad av strukturering av system för att klara detta med mindre resursinsatser. Ofta är äldre system mindre strukturerade av olika skäl – de kan ha varit det ursprungligen eller ha blivit det över tiden – och detta faktum i sig kan medföra att man behöver förbättra dem.

Det finns en mängd olika typer av åtgärder som måste genomföras för att åstadkomma en uppdelning av informationssystem. Några exempel på sådana följer nedan.

En sådan typ av åtgärd är att man måste dela på databaser. Flera delar av informationssystemet måste komma åt vissa data, vilket innebär att databaserna måste delas. Ett annat sätt att se på saken är att flera representationer av objektet läggs upp för flera verksamheter. I själva verket kommer de olika delverksamheterna att arbeta med objektet på litet olika sätt.

Detta måste kompletteras med att samband upprättas mellan de olika verksamheterna i form av informationsöverföring eller meddelandeöverföring. Detta måste till för att beroendeförhållanden skall klaras av. Det kan gälla meddelanden om att en uppgift har genomförts och att resultatinformation överförs till nästa delverksamhet för att nya uppgifter skall kunna genomföras.

Andra typer av åtgärder som kan behöva göras är uppdelningar av koden avseende funktionalitet. De nya funktionalitetsklustren hänförs till olika delverksamheter genom att man görs ansvarig för dem – både till innehåll och beträffande driftsmässiga aspekter. Generaliserade rutiner kan behöva dubbleras och kan gå in i en förvaltningsfas i respektive delverksamhet såvida man inte har avtal om samverkan kring gemensamma komponenter.

Presentations- och gränssnittsdelar kan behövas dubbleras.

I övrigt behöver bedömningar göras huruvida plattformar som kommer att användas i de olika verksamheterna kan behöva förnyas och anpassas till modern teknologi.

3.1.3 Autonoma delverksamheter

En viktig typ av förändring som man vill genomföra i vissa verksamheter är förändrad styrning. Med detta menas att man vill ha nya uppgifter definierade och/eller att uppgifterna fördelas på ett nytt sätt i verksamheten. Med detta följer en ny ansvarsstruktur som ofta är inriktad på decentralisering. Man får ett långtgående självbestämmande i sättet att genomföra uppgifter så länge som man uppfyller uppställda mål. Ansvar lämnas ut i olika delar av organisationen som ges en stor grad av autonomitet.

I arbetet med att avgränsa de autonoma delarna ingår också att skapa samband mellan dem. Sambanden är fundamentala för att helheten skall fungera och för att de olika delverksamheterna skall kunna bedrivas på ett vettigt sätt.

Ledningen för verksamheten kan på detta sätt få en ny roll. Man skall ägna sig åt att sätta mål för delverksamheterna och se till att de rätta sambanden etableras mellan delarna så att helheten kan fungera effektivt.

Denna form av autonomitet och sätt att styra delverksamheterna fordrar också att informationssystemen skall ha en struktur och en avgränsning som inte motverkar utan

stödjer det nya sättet att styra verksamheten. Informationssystemen bör delas så att de får en struktur som överensstämmer med verksamheten och verksamhetsstrukturen.

Till en sådan avgränsning hör också att informationsdelsystem skall ha samband mellan sig. Det kan t ex ske genom att meddelanden med speciellt innehåll kan/skall skickas mellan verksamhetsdelarna. Det kan ingå i ansvaret för en verksamhet att skicka meddelanden med speciellt innehåll och med vissa tidskrav. Detta är en del av genomförandet av uppgifterna för resp delverksamhet. Informationssystemen skall på detta sätt kunna skicka meddelanden enligt gällande sambandsdefinitioner mellan verksamheterna.

Detta kan innebära att informationssystemet i hög grad kan ansvaras för i de olika verksamheterna. Man kan i princip tänka sig att man köra vilka operativsystem, databashanterare samt andra plattformar som man vill i resp delverksamhet. Det viktiga är att man kan leverera de meddelanden som vars och ens ansvar kräver.

Om de befintliga informationssystemen är avgränsade på ett helt annat sätt och man vill gå över till denna styrform så måste man förändra sina gamla system.

Man kan bygga nya system med de nya önskade egenskaperna och avgränsningarna. Detta visar sig dock ofta ganska dyrt att genomföra. En mer realistisk ansats blir då att dela de gamla systemen så att man får en annan struktur. Detta är på intet sätt trivialt.

Ett tänkbart sätt att gå tillväga, om man skall dela ett system så att två autonoma delverksamheter, A och B, kan få ett vettigt informationssystemstöd så att man kan lösa sina uppgifter, kan t ex vara att man från det gamla systemet behåller de delar som A behöver och helt enkelt skrotar de delar av systemet som B behöver. De senare skapas genom att nya komponenter byggs som motsvarar B:s behov. För de delar som A kommer att använda måste man dessutom bygga om systemet så att det kan leverera de meddelanden som stämmer med de uppgifter som A har att lösa.

Det kan finnas flera olika lösningsvägar för denna situation beroende på förutsättningar och andra krav på längre sikt om plattformar o s v. Exempelvis kan man tänka sig olika typer av CORBA-lösningar.

3.1.4 Nya verksamhetstjänster

En av de viktiga orsakerna till att det uppstår behov av informationssystemförnyelse är att verksamheterna måste kunna leverera nya tjänster och produkter.

Dels ser man nya affärsmöjligheter med att gå ut med nya produkter ev med tillkopplade nya tjänster, dels blir man tvungen att leverera nya tjänster eftersom konkurrenterna har börjat göra detta – konkurrenssituationen kräver att man måste komma med liknande tjänster eller tillhandahålla andra möjligheter som gör att man kan upprätthålla konkurrenskraften.

Skall man kunna leverera nya tjänster t ex till existerande leveranser av fysiska produkter, som maskiner etc, måste informationssystemen ofta förändras, existerande delar kan behöva justeras och nya delar kan behöva läggas till.

Ett exempel kan vara att en leverantör av en produkt erbjuder en tjänst till kunden att i samband med köp också registrera produkten hos aktuell myndighet.

Uppgiften att ombesörja att detta blir gjort stöds med en nybyggd informationssystemdel som hanterar uppföljning avseende vad som har hänt med registreringen, när myndigheten har genomfört själva beställningen, ger löpande beskrivningar av vad som händer med registreringen, etc. Denna del skall kopplas till och kommunicera med den del som hanterar produkten som helhet, eftersom det finns starka relationer mellan hanterandet av registreringshändelsen och andra händelser som berör försäljning, leverans, installation, etc.

I samband med denna typ av förändringar så uppkommer också frågan om man skall modernisera plattformen. Kommer man att vilja arbeta vidare med ett förändrat system kan det migreras till en modern systemplattform.

3.1.5 Regionalisering

Verksamheter kan förändras på annat sätt i riktning mot att bli mindre centraliserade. Det kan vara fråga om en regionalisering.

En sådan struktur kan innebära, att varje regional del har alla funktioner som företaget kan erbjuda, men riktade och anpassade mot resp region. Varje del har ansvar att ta vara på de affärsmöjligheter som finns och kan komma att utvecklas inom regionen.

Det innebär i princip att alla funktioner och processer är duplicerade för varje region. Denna struktur måste motiveras med de olika fördelar som närheten till varje region kan erbjuda.

Det finns många möjligheter som informationsteknologi kan erbjuda i detta fall. Om man skall bygga informationssystemstöd för varje funktion i varje region kan det finns anledning att skapa grundlösningar som kan anpassas och användas i flera regioner.

Man behöver inte tvinga varje region att använda ett visst informationssystem, utan man kan erbjuda ett bassystem som varje region får utvärdera. Detta på basis av att man har frihet att välja det sätt att lösa sina uppgifter som man vill.

Systemen kan göras flexibla på många sätt. Man kan göra ett bassystem som kan anpassas och byggas på i olika riktningar och där arkitekturen är flexibel för utbyggnad.

Man kan också bygga ett grundsystem som har mer färdiga alternativa utformningar som kan motsvara olika mer kända typbehov. I vissa fall kan man också tänka sig parameterstyrda systemlösningar som medger att olika beteenden hos systemet kan väljas.

Olika former av migreringssituationer kan tänkas i detta fall.

Den nya styrformen skall implementeras i verksamheten. Olika aktörer skall förstå vad de nya premisserna innebär. Organisationen måste vara intresserad av att införa förändringen.

Gamla system måste kanske delas och anpassas under en övergångsperiod. De nya systemdelarna kan införas t ex del för del. Gränssnitt mellan gammalt och nytt måste beskrivas.

Beskrivning av vilka begrepp som skall hanteras måste göras och göras kända i verksamheten genom en begreppsmodell.

Olika versioner av ett grundsystem skall tas fram för varje region. Varje region kan ställa krav på ett anpassat system och skall driva en specifikationsprocess som möjliggör detta.

Beskrivningar av ett bassystem kan göras och olika anpassningar skall beskrivas lokalt, men kan även göras kända över verksamheten så att man kan tillföra varianter som redan har producerats.

3.1.6 Extern påverkan som lagar etc

Inte så sällan åläggs verksamheter krav på förändringar utifrån, krav som man har liten möjlighet att påverka och som inte är framsprungna ur verksamheten själv. Man blir tvungen att hantera dem ändå. Det gäller lagar som regering och riksdag skapar, regleringar som införs av myndigheter, tvingande krav som ställs av externa organisationer vartill man är underleverantör, etc.

Sådana lagar och regler kan orsaka att man måste göra ganska stora förändringar i ett systemarv. De kan aktualisera aspekter av verkligheten i eller utanför verksamheten som man tidigare inte alls hade kopplade till andra aspekter som är inbyggda i program.

Helt nya begrepp och regler måste beskrivas i systemet.

3.1.7 Nya krav på egna processer

Vissa verksamheter måste förbättra sitt sätt att fungera. Det kan finnas många skäl till detta. Ett skäl kan t ex vara att konkurrenterna blir allt bättre ifråga om ledtider för att ta fram en ny produkt, tidspannet från koncept till färdig produkt minskas. Om denna typ av ledtider kan hållas nere kan man uppnå både modernare produkter och billigare priser, eftersom mindre resurser har använts för att ta fram produkten.

För att förbättra verksamheter så måste man kunna kommunicera om dem. Det finns många olika sätt att kommunicera om verksamheter. Ett sätt som idag är populärt i många industriella verksamheter, men också inom många tjänstesektorer såsom sjukvård, försäkring m fl, är att tala om processer. Med process (eller processtyp) menar man ofta en enhet som har till uppgift att producera ett visst resultat. För detta behöver processen ha tillgång till olika typer av resurser. Processägaren har ansvaret för att processen fungerar väl, bl a med avseende på att det som skall produceras verkligen kommer fram.

Avsikten med att införa ett processynsätt i en verksamhet är ibland att bryta invanda mönster. Orsaken till att man behöver denna typ av förändring kan t ex vara att det har funnits enheter i verksamheten som har blivit "sig själva nog", och som inte har ägnat tillräckligt med ansträngningar åt att klara ett samspel med andra enheter och vara en del i en helhet.

Man har endast sett de egna målen för den lokala funktionen och man har haft svårt att relatera den egna delverksamheten till helhetsmålen för verksamheten.

Sammankopplade processer skall gripa över större delar av verksamheten så att man binder samman olika delverksamheter och i ökad utsträckning ser till vad som faktiskt kommer ut ur verksamheten gentemot kunden.

Viktiga avgöranden är t ex hur hela processkedjan förbättras så att man får en bättre produkt, eller hur man kan ta tillvara kundens synpunkter på produkterna och återföra denna kunskap till konstruktionsprocessen för förnyelse av produkter, etc.

Om man vill ha en hög medvetenhet om processernas uppbyggnad och hur verksamheterna skall förbättras i olika avseenden, så behöver informationssystemen ha en följsamhet mot processerna och inte gå stick i stäv med processtrukturen. Informationssystemen skall försörja med information så att man stödjer definierade processer och hur processerna skall vara en del i verksamheten.

För att informationssystem skall kunna migreras till en struktur som kan svara upp mot ett processtänkande, så måste de oftast struktureras om så att de stödjer hela den sammankopplade processen och inte bara delar av den. Informationssystemen skall bl a stödja aspekter som har att göra med sambanden mellan processerna så att hela processkedjan kan bli bättre.

Hur man korrigerar processens sätt att fungera är en annan viktig aspekt. Hur kan man återföra information om hur processen har gått, till de som utför arbetet i olika delar av processen? En annan aspekt är hur processägare skall kunna kontrollera och övervaka olika delar av processen för att kunna besluta om korrigeringar i processen, etc.

Hur kan man migrera ett informationssystem så att det bättre svarar mot ett processtänkande?

Ett sätt kan vara att man förändrar informationssystemet så att det kan stödja och erbjuda en flexibel uppsättning informationstjänster för olika delar av verksamheten.

Det kan innebära att man bryter upp ett gammalt system så att man kan erbjuda informationstjänster för olika delar av processen. Den nya tjänsterna skall helst motsvaras av olika komponenter som lätt kan bytas ut och utvidgas.

Olika informationstjänster och lagrad information skall svara mot de verksamhetsmässiga mål som har tilldelats verksamheten avseende aktuell process.

Arkitekturmässigt kan detta innebära att olika delar i det äldre systemet struktureras om och delas upp. Vissa delar kan läggas som komponenter i en återanvändbar tjänstestruktur i en servermiljö. Andra delar kan ligga i ett antal klientmiljöer. Lagringsdelar mm görs generella så att man kan hitta, förändra och bygga ut delar. Delarna kan byggas och kompletteras så att man kan få nya tjänster på verksamhetsnivån.

Hur den nya verksamheten och de förnyade informationssystemen skall se ut måste beskrivas. Det gäller också det sätt på vilket informationssystemen stödjer verksamheten.

3.1.8 Arbetssätt och rutiner ändras

Inom ramen för en eventuellt förnyad verksamhetsstruktur kan man behöva ändra arbetssätt och rutiner.

Arbetssätten kan förändras i sig, men de överordnade uppgifterna för verksamheten bör vara klargjorda för verksamheten innan man går in i detalj för att förbättra arbetssätten.

Med arbetssätt menas här en noggrannare beskrivning avseende hur arbetet i en verksamhet egentligen skall bedrivas. På verksamhetsstrukturnivå beskriver man ofta vilka uppgifter och mål som skall finnas för en funktion eller en process. Inom ramen för fastställda uppgifter och mål, kan man sedan arbeta vidare med och utforma hur själva arbetet skall, eller kan bedrivas.

Denna utformning kan ofta ske lokalt inom ramen för funktionen eller processen, speciellt om sambanden med andra funktioner och processer är klargjorda. Enligt olika styrfilosofier anser man ofta detta som en fördel att den aktuella funktionen själv kan utforma hur arbetet skall bedrivas. Detta skapar ett större ansvar för att uppgifter och mål uppfylls och att man också har rätt att utforma arbetssätt på ett sätt som passar de egna preferenserna så länge som de överordnade målen för uppgifterna uppfylls. En större motivation kan uppnås i förändringsarbete.

Olika ändringar i uppgifterna för en funktion innebär att själva arbetssättet och rutinerna måste ändras. Om nya mål förs in eller nya uppgifter tillkommer, medför det att arbetssätt och rutiner måste ändras.

Även inom en existerande uppgiftsstruktur kan man upptäcka att det kan vara betydligt rationellare att utföra arbetet på ett annat sätt. Det kan också innebära att större ansvar och beslutsbefogenhet kan införas i rutinerna och en högre motivation kan uppnås bland befattningshavarna. Högre kvalitet i beslut och produkter kan bli ett resultat av detta.

Ett bättre utformat informationssystemstöd kan innebära att uppgifterna kan genomföras på ett rationellare sätt med mindre resursåtgång. Det kan också innebära högre kvalitet i beslut.

För att uppnå dessa olika potentialer behöver ofta informationssystemet ändras.

Olika krav och behov kan uppkomma från de enskilda personer som genomför uppgifter och rutiner. Man kan t ex tycka att gränssnittet för ett visst informationssystem är otidsenligt och olämpligt för att utföra olika arbetsmoment. Ett bättre gränssnitt på de existerande systemen skulle kunna spara resurser och/eller skapa bättre kvalitet i arbetsresultaten.

Att bygga om ett existerande informationssystem så att det får ett nytt gränssnitt kan medföra olika typer av insatser av varierande omfattning beroende på hur det existerande informationssystemet är uppbyggt. I många informationssystem är gränssnittdelen invävd och svår att urskilja, i andra har den en mer tydlig kontur.

Människa/maskin-interaktionen i olika system kan dessutom bygga på helt olika principer, vilket präglar hela systemets uppbyggnad. Ett exempel är ”välj först operation – välj sedan på vad” kontra ”välj först på vad – välj sedan operation ”. Detta är ganska olika sätt att arbeta.

Olika tekniker finns för att skapa nya gränssnitt till system. Det är viktigt att sådana förändringar görs så att man strävar mot en medveten arkitektur så att informationssystemet blir möjligt att underhålla också i framtiden.

Wrapping-baserad teknik för att skapa nya gränssnitt kan vara en möjlighet. Se avsnitt 4.

3.1.9 Ny utgångspunkt för förändring

Verksamheter måste ofta förändras snabbt för att kunna överleva. Nya affärsmässiga krav som uppstår innebär att man inte själv kan bestämma takten i sina förändringar. Man tvingas att förändra i en takt som man inte önskar och på ett sätt som kan ställa stora krav på verksamheten, på personerna i verksamheten och på dess informationssystem.

Om det är så att man förväntar sig att det inom en snar framtid kommer att uppstå avsevärda, nya krav från konkurrenter eller från helt nya legala regelsystem o s v, kan man inse att man behöver förbereda äldre systemstrukturer på ett sådant sätt att det blir lättare att införa förändringar, eller förbereda dem på annat sätt på att nya krav kommer att bli aktuella.

Många ändringar i informationssystem införs på ett forcerat eller panikartat sätt. I ett sådant läge har man små möjligheter att skapa en god arkitektur i ett förnyat informationssystem.

Ändringarna blir arkitekturlösa. Har man tid att förbereda och definiera en ny arkitektur, som man kan använda för att aktivt styra migreringsarbete, har man bättre möjlighet att skapa en miljö i vilken man kan realisera nya krav och upprätthålla en vettig systemstruktur som man kan vidmakthålla och som svarar mot en modern arkitektur.

Att förnya informationssystem på detta sätt kräver ofta en mer ingående insats för att skapa en plattform för att kunna genomföra specifika ändringar på ett effektivt sätt. Att skapa en sådan plattform kan innebära att man skapar olika arkitekturdelar som kan förändras var för sig och som har definierade gränssnitt i förhållande till andra delar.

Det kan t ex innebära att man skapar en datalagringsdel, en gränssnittsdel och en del där olika härledningar, beräkningar och utvärderingar genomförs.

Moderna databashanterare kanske kommer att användas i vissa avseenden och äldre fillagringssystem används i andra delar under ett övergångsskede, men gränssnittet till dessa delar används genom ett modernt objektorienterat gränssnitt.

En gränssnittsnivå identifieras och vissa gränssnitt byggs om enligt nya principer. Andra delar körs med ett nytt gränssnitt, men styrningen sker på ett likartat sätt som i det gamla systemet. Man tar inte steget fullt ut förrän i ett senare steg när man känner att man klarar av det.

Man utvärderar viktiga verksamhetsregler och avgör vilka delar av de gamla beräkningsdelarna i programmen som man behöver. En ny behandlingsnivå i programmassan identifieras och vissa program skrivs om medan andra behålls tillsvidare, men de aktuella verksamhetsreglerna är valida.

Beskrivningsfrågorna är viktiga i detta fall, eftersom man lägger en grund för att kunna "slå upp" vad systemet gör och vilka begrepp som egentligen hanteras.

Lagringen beskrivs genom att lagringsorienterade modeller kopplas till mer begrepps-orienterade modeller. De lagringsorienterade modellerna ändras i takt med att lagringen ändras i en successiv migreringsprocess.

Detta kan utgöra ett exempel på hur man kan förbereda ett antal system för ett större antal förändringar som måste kunna realiseras på ett effektivt sätt och kanske under kort tid.

3.1.10 Bas för återanvändning

Det kan finnas helt andra skäl till att man vill förnya informationssystemen i en verksamhet. Det kan vara så att man vill skapa en helt ny systemstruktur som är bättre i något avseende.

Man kan t ex vilja skapa en mer underhållsvänlig systemstruktur. Detta kan realiseras på olika sätt. Man kan utföra detta med olika ambitionsnivå och med olika mål och inriktning.

Ett av syftena kan vara att man vill skapa en hög grad av flexibilitet, t ex beroende på att man väntar sig nya och omfattande krav från verksamheten och medföljande förändrade krav på informationssystemen. Man vill då troligen skapa höggradigt flexibla informationssystem, som under kontrollerade former snabbt kan ändras så att de får andra egenskaper och kan utföra andra saker.

Det finns idag speciella realiseringsmiljöer för informationssystem som är inriktade på att man kan få flexibla system. De bygger på att man skapar små avgränsade delar som är tydliga och utför preciserade uppgifter. Det som hanteras – ”objekten” – och det som man vill utföra på objekten – ”operationerna” – definieras noggrant. Detta bland annat så att objekt och operationer som är lika varandra – eller samma – bara skall definieras en gång. Olika operationer och objekt som är lika skall bygga på varandra och relateras till varandra.

Om man vill ändra på objekt och operationer skall man inte behöva göra detta på många olika ställen utan olika definitioner skall finnas på speciellt utpekade ställen. Ändringarna skall utföras på ett säkert sätt, vilket bland annat uppnås om man inte behöver titta på många olika ställen.

Dessa principer och tekniker är typiska för objektorienterade arbetssätt. Vill man bygga ut systemet, kan man bygga på saker som redan är definierade. Endast nya saker behöver definieras. Effekterna kan bli återanvändning. Resurser kan sparas och en högre grad av kvalitet genom bl a färre fel kan uppnås om man hittar rätt arbetssätt i förändringsarbetet.

Denna typ av systemuppbyggnad kräver förtrogenhet med programmeringsspråk och systemkomponenter såsom databashanterare, o s v, vilka bygger på dessa principer. Det krävs kunskapsuppbyggnad och dessutom i viss mån ”avlärning” ifråga om principer som gäller för andra programmeringsspråk o s v, för personer som inte har stiftat

bekantskap med denna typ av miljöer förut. Detta måste byggas in i strategigenomförandet.

Återanvändning kan i viss mån uppnås utan objektorienterade miljöer, men det krävs medvetenhet om vad man vill åstadkomma ifråga om systemuppbyggnad. Å andra sidan kan också objektorienterade miljöer missbrukas så att effekter som är åsyftade inte uppnås.

3.1.11 Datorleverantören tar nya strategiska beslut

Det kan finnas ytterligare skäl till att man måste förnya informationssystem. Personer och organisationer i verksamhetens omgivning kan ställa krav och sätta upp restriktioner som föranleder att man ser sig nödsakad att förnya sina informationssystem.

Ett exempel på detta är leverantörer av informationsteknologi. Dessa moderniserar över tiden den utrustning man säljer i olika avseenden. Det gör man av konkurrensmässiga och andra skäl, man försöker förbättra egenskaper och prestanda hos produkterna.

Nya versioner av produkterna kommer ibland att ersätta äldre versioner, vilka leverantörerna inte längre anser sig kunna underhålla. Detta innebär att kunden kan bli tvungen att anskaffa och gå över till de nya versionerna för att kunna fortsätta att köra sina system utan problem.

T ex kan ett byte till en ny version av en mjukvara i sin tur innebära att nya maskiner måste anskaffas för att den skall kunna köras. Den nya versionen har nya egenskaper och kräver ökad maskinkraft.

Den samlade effekten kan bli att en relativt omfattande förnyelse måste genomföras och att nya investeringar måste göras. Dessa investeringar måste sättas in i ett sammanhang och ingå i en strategi så att ytterligare förnyelsesteg hänger ihop på lång sikt. Nya gränssnittspaket, nya databashanterare, o s v, måste sättas in i ett sammanhang och kunna utnyttjas i andra förnyelsesteg i verksamheten. Man bör undvika att tvinga fram förändringar utgående enbart ifrån restriktioner från leverantören.

Nya versioner av mjukvara och hårdvara kan ha starka samband med andra delar i den informationsteknologi man har. Effekterna av, i förstone tänkta partiella utbyten, kan bli stora.

Vi exemplifierar detta genom några exempel på delar i plattformar och hur delarna hänger samman med varandra.

Operativsystem. Ett tvång att byta operativsystem kan innebära att ny hårdvara behöver anskaffas. De gamla systemen som körs via operativsystemet kanske måste kompileras om. Detta kan föranleda justeringar i koden då nya förutsättningar finns inbyggda på operativsystemnivån, eller i de nya kompilatorer som hänger samman med den nya operativsystemversionen. Flera kopplade förändringsåtgärder behöver genomföras och planeras.

Hårdvara. Hårdvara åldras snabbt. Nya versioner av hårdvara kan innebära stor kapacitetsökning, men innebär ofta att stora investeringar måste göras. Byts hårdvara,

behöver ofta operativsystem bytas till en nyare version, vilket kan innebära ovan angivna konsekvenser.

Nya kompilatorer. Nya versioner av kompilatorer kan innebära nya möjligheter att realisera önskade egenskaper i ett system. Men det kan också medföra att systemen måste justeras för att de nya kompilatorerna skall acceptera existerande kod. Fel i kompilatorerna kan innebära att nya fel introduceras i systemen trots att några nya egenskaper inte uppstår hos systemen. Användarna kan plötsligt finna att deras system fått nya faciliteter och ser annorlunda ut, detta trots att man inte bett om några förändringar.

Databashanterare. Datalagringssystemen, såsom t ex databashanterare, måste förnyas om leverantörer av dessa system inte stödjer de äldre versioner man har. De nya versionerna har nya egenskaper som är svåra att ta tillvara vid ett visst tillfälle. Vissa sätt att lagra data som man använt sig av i den äldre versionen kanske inte är möjliga att använda i en ny version. Man kanske måste gå igenom hela datalagringslösningen så att de ställen där man har gjort speciallösningar uppdagas. Detta måste ändras innan man börjar använda den nya versionen.

Slutresultatet kan bli ett bättre strukturerat system, men ett omfattande arbete kan behöva genomföras.

3.1.12 Krav på uppfattbarhet och människa/maskingränssnitt

I vissa fall kan systemförnyelsearbete behöva initieras på grund av att människorna i en verksamhet kommer fram till att gränssnitten hos informationssystemen måste förnyas.

Informationssystemen kanske utför rätt saker för att verksamheten kall kunna fungera väl, men gränssnitten gör att informationssystemen är svåra att arbeta med. Olika aktörer har hypoteser om att förbättrade gränssnitt kan innebära att olika arbeten kan utföras smidigare i olika avseenden.

Detta kan ha sin grund i flera orsaker. Nya datorsystemplattformar har medföljande moderna gränssnitt. Olika aktörer tycker att dessa gränssnitt utgör en normal arbetsmiljö och bör vara aktuella för olika applikationer i den egna verksamheten. Gränssnitten börjar dessutom att uppvisa många likheter, oavsett leverantör och operativsystemtyp.

Ett antal embryon till gränssnittsstandard tar form på olika håll. Detta gör att användarna kan arbeta med olika typer av applikationer på ett likartat sätt. Man börjar känna igen sig och man kan lättare sätta sig in i nya applikationer. Användare som ännu inte fått tillgång till dessa gränssnitt blir medvetna om att de finns tillgängliga. De egna gränssnitten kan, i förhållande till detta, vara klart ålderstigna och aktörer i organisationen inser att man skulle kunna arbeta på ett bättre sätt med moderniserade gränssnitt.

Arbetsuppgifternas beskaffenhet kan innebära att man skulle vilja arbeta på ett mer parallellt sätt. Detta innebär att man behöver kunna komma åt applikationer samtidigt när man utför vissa typer av arbetsuppgifter. Detta kan vara svårt att åstadkomma med äldre gränssnitt. Moderna gränssnitt kan innebära att man snabbt hoppar mellan olika applikationer som körs igång. En applikation som man utgår ifrån kan stannas i ett viss

läge för att man behöver kontrollera några uppgifter i ett annat system. Man kan sedan återgå till den första applikationen och fortsätta sitt arbete.

Detta kan ofta åstadkommas för äldre applikationer med ett modernare gränssnitt, utan att applikationerna behöver påverkas i alltför stor utsträckning. Kravet för att göra detta är dock att det inte blir andra konsistensmässiga problem mellan applikationer, genom att de uppdaterar varandra på ett olämpligt sätt baserat på vad som händer i verksamheten. I detta fall kan hoppande mellan applikationer innebära att man måste förändra applikationerna så att de påverkar varandra på ett sätt som gör att de följer skeendet i verksamheterna.

I andra fall kan applikationer behöva förändras för att det skall vara möjligt att koppla till ett modernt fönstergränssnitt. Exempelvis kan en applikation vara uppbyggd från en filosofi som innebär ”bestäm vad som skall göras – ange sedan på vad detta skall göras”. Ett modernt fönstergränssnitt bygger ofta på en annan princip: ”bestäm på vad någonting skall göras – ange sedan vad som skall göras”.

Om en äldre applikation är av den förstnämnda typen kan det innebära en hel del arbete med att ansluta ett gränssnitt som bygger på den sistnämnda principen. Det kan dock finnas möjlighet att ansluta ett modernt gränssnitt genom att anpassa applikationen även om sättet att styra applikationen inte kommer att bli helt på det sätt som hade blivit resultatet om man hade skrivit applikationen efter den sistnämnda principen.

Att introducera moderna gränssnitt kan dock vara ett gott steg i en successiv migreringsprocess och ger användaren möjlighet att successivt lära sig att använda moderna applikationer med moderna gränssnitt.

När man skall modernisera gränssnitt i verksamheten kan det finnas anledning att genomföra denna modernisering på ett systematiskt sätt. Det innebär att man bör tänka igenom vilka typer av gränssnitt som kommer att nyttjas i verksamheten i framtiden. Alla migreringsaktiviteter bör sedan ligga i linje med dessa inriktningar.

3.1.13 Nya och gamla system måste samexistera

En annan anledning till att man ser sig nödsakad att starta en migreringsprocess kan vara att man inser att man måste skapa en successiv migreringssituation. Detta kan ha olika verksamhetsmässiga och affärsmässiga orsaker.

Förnyelsen av de olika existerande systemen i en verksamhet kommer att bli omfattande, bedömer man. Detta implicerar att det kommer att vara omöjligt att förändra alla olika informationssystem samtidigt. Det gäller då att hitta ett sätt att kunna förändra de olika systemen steg för steg. Denna process kommer att ta tid. Man måste hitta ett sätt att låta nya och gamla system samexistera under en avsevärd tid. De måste kunna samverka under hela denna tid.

Detta kan upplevas som ett medel i en migreringsprocess, men eftersom denna form av samexisterande kommer att vara under avsevärda tidsrymder kan det vara en situation som kommer att styra hur migreringsstrategin kommer att byggas upp. Vad man väljer att förnya i ett visst tidsskede kan dessutom bero av att en viss verksamhetsdel behöver ett förnyat systemstöd. Detta kan i sin tur ha att göra med att denna verksamhetsdel har fått nya uppgifter och nytt definierat ansvar.

Samverkan måste byggas upp på ett medvetet och systematiskt sätt. Det finns flera sätt att åstadkomma samverkan. De olika sätten har olika effekter och kan stämma olika bra med strukturen på verksamheten och hur man vill styra denna.

Ett sätt är att åstadkomma samverkan genom att meddelanden utväxlas mellan gamla och förnyade informationssystem. Varje meddelande byggs upp och definieras med avseende på sitt innehåll. Den nya systemdelen bör tillhöra en verksamhetsdel där det finns ett ansvar för att denna bedrivs på ett kvalitetsmässigt och uppgiftsrelaterat sätt. Den arvsmässiga systemdelen kan tillhöra en annan verksamhetsdel med annat ansvarsområde eller uppgift. Avgränsningen av vad som skall förnyas vid en viss insats bör planeras så att den nya delen som skall förnyas har en verksamhetsmässig motsvarighet.

Ett annat sätt kan vara att man utväxlar objekt mellan förnyade delar och arvsdelar i en systemstruktur. Denna strategi implicerar ofta att man förnyar genom att skapa objekt-orienterade lösningar i de delar som förnyas. Man förnyar genom att hantera objekt på ett stringent sätt med olika definierade metoder enligt verksamhetskraven. Olika objekt definieras till sina egenskaper och till sin semantiska betydelse. När man skall kommunicera med arvssystemen uppfattar de förnyade delarna att man kommunicerar med ett annat delsystem via objekt på samma sätt som med ett annat förnyat delsystem.

Man måste då skriva ett gränssnitt som verkställer alla anrop enligt det objektorienterade synsättet med metoder, etc. Dessa anrop översätts till anrop som det äldre informationssystemet förstår.

På detta sätt får man tid på sig att migrera de äldre systemdelarna steg för steg och man kan skapa en plan för varje systemdel så att det totala migreringsarbetet kan genomföras stegvis och att god kvalitet kan uppnås i varje steg.

Det tredje sättet att åstadkomma samverkan mellan gamla och nya system är via databaser. Databaserna spelar rollen att vara den "plats" som man använder för att lotsa information mellan gamla och förnyade system. Denna strategi bör dock övervägas noga i olika avseenden. Bl a måste någon känna ett ansvar för varje databas. Databaserna bör spela en roll som är klart definierad. Databaserna kan ha en roll under själva migreringsarbetet. Efter genomfört migreringsarbete bör en sådan databas avvecklas och ersättas med en som passar in i den nya verksamhets- och systemstrukturen.

Man bör tänka över hur de förnyade systemdelarna skall kommunicera med andra systemdelar. De förnyade systemdelarna bör definieras på ett systematiskt sätt. Varje förnyad del bör avgränsas med tanke på ett verksamhetsmässigt kriterium. Ett sådant kriterium bör kopplas till hur man vill att den nya verksamheten skall fungera i framtiden. Det bör bl a kopplas till uppgift och ansvar i den nya verksamheten.

4. Verktygsval

En central fråga vid förnyelse (reengineering) av informationssystem är hur man skall välja och hur man skall använda de verktyg (och de metoder de tillämpar) som finns på marknaden.

I detta avsnitt analyserar vi några av de verktyg vi har stött på, beskriver och klassificerar deras egenskaper samt lägger fram ett antal principer för när i en reengineeringsprocess de kan användas.

För att kunna göra en sådan verktygsanalys beskriver vi även en allmän arkitektur för en tänkbar reengineeringsprocess för arvssystem.

4.1 Inledning

Eftersom arvsinformationssystemen, som ofta skapats för decennier sedan, väsentligt kan försvåra möjligheterna att åstadkomma förbättringar vad avser verksamhet, informationssystem, produktunderhåll och effektivt samarbete med aktuella system och miljö, har förmågan att förnya och/eller nyskapa dessa system blivit en väsentlig uppgift för studiet av informationssystem och olika grupper av applikationer.

Hundratals verktyg har utvecklats på marknaden som svar på detta behov – verktyg som inriktar sig på olika användningsområden vid migrering och reengineering av system, från programförståelse och remodellering av källkod, till restrukturering av databaser och redokumentation av krav och behov.

Enligt Brodie [10] är det typiskt för arvssystem ("legacy systems") att de är stora (t ex upp mot 10^7 rader kod), geriatriska (t ex mer än 10 år gamla) och att de använder en gammal databastjänst (t ex IMS) eller inget databashanteringssystem alls. Arvssystemen representerar dock vanligen väsentlig information och stora investeringar och har under årens lopp kommit att bli ganska tillförlitliga [9]. De är dessutom kritiska för verksamheten och kan inte stängas av ens för en kort tidsrymd. Dessa informationssystem är mycket kostsamma vad gäller att upprätthålla prestanda och utföra modifieringar, oflexibla (svåra att anpassa till modern miljö) samt "spröda" (kan lätt "brytas itu" när de modifieras). De måste förnyas för att kunna förbättra den verksamhetsprocess de stödjer.

Att "förnya" ett informationssystem genom att utgå från det gamla systemet kan tolkas som att man modifierar eller förbättrar arvssystemet eller att man skapar ett helt nytt informationssystem endast baserat på kraven på dess arvsmotsvarighet. Vilken strategi man väljer beror på kostnadseffektivitet hos reengineeringsmetoden: att köpa nya system eller reengineeringsverktyg.

Bland många olika sätt att "reproducera" arvssystem beskriver vi en typisk process för förnyelse av arvssystem i nästa avsnitt. Syftet med att migrera ett arvssystem är att skapa ett målinformationssystem som bibehåller den funktionalitet som arvssystemet har samt att migrera arvsdata från en arvsdatabas till en måldatabas.

4.1.1 En generell process för reengineering av arvssystem

När ett arvssystem skall förnyas, etableras i allmänhet ett nytt system -"målsystemet" – som bibehåller och bevarar de funktioner och de data som överförs från arvssystemet. Det nya resulterande systemet förväntas täcka eller tillfredsställa alla krav användaren ställde på arvssystemet. Det nya systemet kan naturligtvis implementeras i något annat modernare språk för avancerad programmering, som t ex 4GL, och använda moderna databasutformningsmetoder och frågespråk.

Uppgifterna vid migrering och reengineering av arvssystem kan därför beskrivas på följande sätt:

– Att extrahera de gamla kraven från arvssystemet (här behövs reengineeringsmetoder, så t ex kan metoder för reverse engineering behövas för den händelse att kravdokumenten gått förlorade).

– Att på nytt bedöma och specificera de behov som har fångats i enlighet med moderna systemutvecklingsteknologier och aktuella standards.

– Att utveckla målsystemet i enlighet med de reviderade behoven på ett "forward engineering"-sätt.

– Att överföra data från den gamla databasen till dess nya motsvarighet, samt

– Att testa och värdera målsystemet mot arvsbehoven och de reviderade behoven samt att kontrollera datas konsistens i måldatabasen.

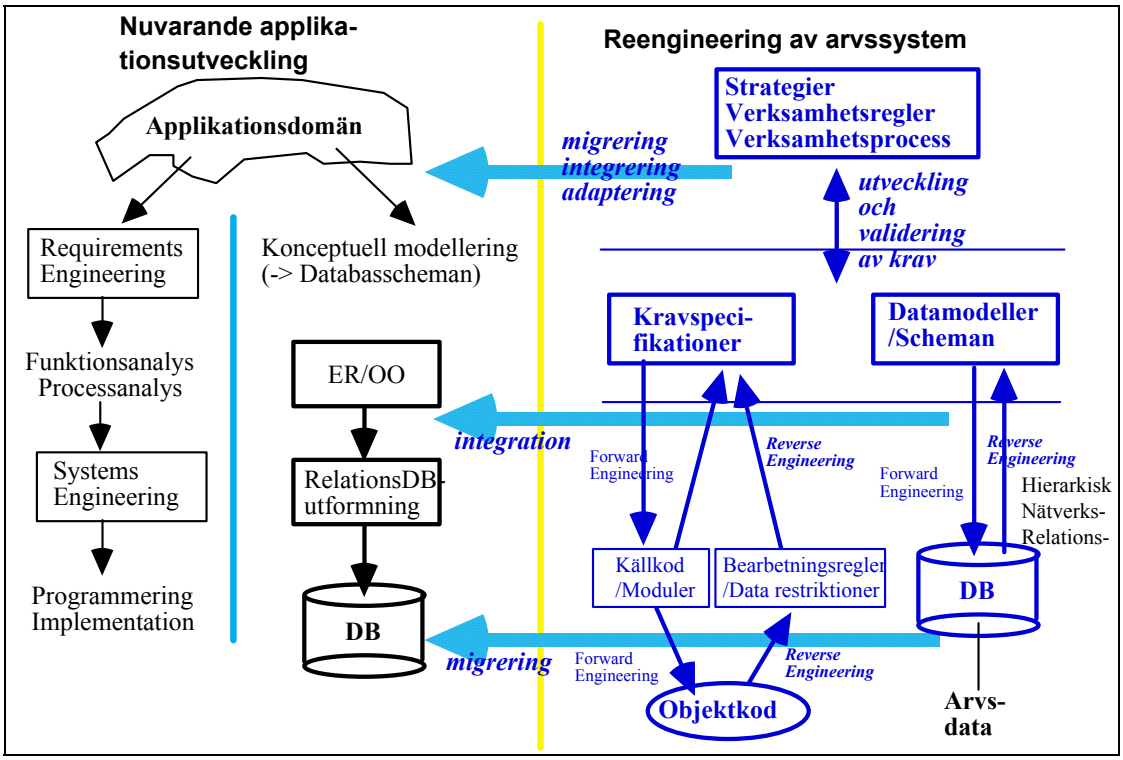
4.1.2 Reengineeringsuppgifter på olika nivåer

Nedanstående figur kan användas för att illustrera olika aspekter på förnyelse av informationssystem som kan visa på olika behov av tekniker, metoder och verktyg för reengineering. Den moderna processen för informationssystemutveckling (till vänster i figuren) anses vanligen markera två typer av sammanflätade livscykler i utveckling: systemutvecklingsfaser och databasutvecklingsfaser. Problem kring reengineering av arvssystem visas till höger i figuren. Vi kan se problemen i tre nivåer enligt följande.

1. Krav- och miljönivån.
Aktiviteterna att migrera och förnya systemarv påverkas av behov och krav att förbättra affärsverksamheten. Det innebär att kraven på arvsinformationssystemen (om de fortfarande existerar eller genom reengineering av systemen) bör analyseras för att kunna möjliggöra att de ombyggda systemen passar nya affärsverksamheter, nya strategier samt affärsregler och -processer som avses beaktas i systemen. Detta är den första aspekten,
d v s nya behov och krav bör beaktas för att förbättra systemen. Å andra sidan bör kraven för de omskapade delarna (systemen) analyseras för att vara kompatibla med kraven på de andra systemen inom samma miljö (d v s inom systemfederationen).

Kraven på arvssystemen, kraven på andra noder i federationen och nya krav för reengineeringsarbetet måste integreras. Visst initialt undersökningsarbete har utförts för att göra det möjligt att integrera krav [23].

2. Schemanivån (konceptuellt schema och funktionellt schema).
Två aspekter bör beaktas avseende konceptuella databasscheman och funktionella scheman resp. Integration av konceptuella scheman i arvssystemsammanhang kan sägas äga rum i följande två situationer. 1) De schemafragment som fås genom dekomponering av arvsschemat under databasmigreringen måste återintegreras i ett globalt schema (ett målschema) för hela den konceptuella databasen. 2) Det så framtagna målschemat bör läggas över de andra konceptuella databasscheman som används för närvarande. Vi antar att de konceptuella databasschemana kan tas fram, antingen är de dokumenterade eller också kan de återskapas genom analys av arvsdatabasen och dess operationer genom reverse modellering av databasen.



3. Data- och applikationskodnivån.
Den databasintegrationsmetod som diskuteras här är enbart ett hjälpmedel för migrering av arvsdatabaser. Det viktiga är att kontrollera konsistensen mellan den gamla och den moderna databasens datastrukturer. På denna nivå ligger integrationsproblemen främst i att eliminera redundans i data, eftersom data kan ha duplicerats ett antal gånger under utvecklandet av arvsdatabasen, i att förena de möjligen olika datastrukturer som uppstått
i arvs- och måldatabaserna, och i andra databaser som används i målmiljön, samt i att upptäcka hur förändringen av arvsdatastrukturerna kommer att påverka operationerna på såväl arvs- som måldata, och på deras strukturer.

Under databasintegrationsprocessen bör speciell uppmärksamhet ägnas åt de operationer som förändrar data i databasen. I vanliga fall är databasmigrerings- och integrations-processen av inkrementell natur. Det betyder att data i databasen migreras och integreras
i måldatabasen bit för bit och att processen kan pågå länge. På grund av inbördes beroenden kan data dupliceras i arvs- och måldatabasen. Denna åtgärd för att bevara dataintegritet kallas "cut overs" [10]. När en förändring görs i en dataenhet D i arvs-databasen, så bör motsvarande data D' i måldatabasen också förändras för att behålla alla data konsistenta. I denna situation bör integrationshjälpmedlet skapa ett antal "mappings" mellan sådana data i arvs- och måldatabaserna och hjälpa till med att styra operationer till båda de data som berörs av operationerna.

4.1.3 Rapportstudie
I följande avsnitt diskuteras de verktyg som används vid reengineering och migrering av arvssystem. Först analyseras de aktiviteter och uppgifter som utförs i reengineeringsprocessen och vi visar på viktiga länkar mellan aktiviteter där reengineeringsverktygen kommer att användas. För det andra föreslår vi ett antal

egenskaper med utgångspunkt från vilka verktygen analyseras. För det tredje görs en enkel utvärdering av verktyg enligt de givna kriterierna.

4.2 Relaterade arbeten om verktygsanalys

I detta avsnitt sammanfattar vi en del angränsande arbete om analys av reverse engineering- och reengineeringsverktyg från litteraturen och belyser sätt eller metoder som använts för sådan analys [8,10,12, 21].

De kategoriseringar som hämtats från litteraturen ger oss en grund för att kategorisera verktygen enligt de funktioner och roller som de spelar i reengineeringsprocessen.

Följande avsnitt beskriver varje kategorisering i detalj. Beroende på aktuella behov är det möjligt att man kommer att välja ett eller flera verktyg från varje kategori.

4.2.1 Brodie och Stonebrakes kategorisering

Enligt [11] kan det vara en något "förvirrande" upplevelse att försöka matcha sina verktygskrav mot befintliga verktyg. Genom att istället organisera verktygen efter funktion kan man underlätta valet av de för de aktuella behoven lämpligaste verktygen.

Verktyg för praktiskt genomförande av migreringsprojekt kan, enligt Brodie, fördelas på fem funktionella kategorier:

1) Bryggor (gateways). En brygga är en programmodul som sätts in mellan operativa programkomponenter för att "medla" mellan dem och för att isolera komponenter från förändringar som företas i andra komponenter.

2) Analysatorer och verktyg för extrahering av specifikationer (avseende såväl data som bearbetning). Dessa verktyg använder ett brett spektrum av tekniker för att fastställa tekniska, funktionella och andra aspekter på applikationer, program eller databaser. På applikationsnivån ger de en översiktsbild av ett helt system och visar flödesdiagram, metrik, kodkvalitet och filanvändning.

3) Migreringsverktyg. Dessa verktyg möjliggör automatisering av olika inkrement i migreringen när målarkitekturen väl är på plats. Det finns verktyg som stödjer delar av gränssnitts-, data- och applikationsmigreringen och andra verktyg som stödjer samtliga dessa.

4) Testverktyg. En grundlig testfas identifierar omfattningen av den enhets- och integrationstestning som behövs för att genomföra migreringsprojektet. En bra testningsprocess testar igenom en stor mängd logik genom att använda en liten mängd data. Övervakning av prestanda bör sättas in tidigt i den testningsprocess som gäller målmiljön, så att alla problem upptäcks så tidigt som möjligt.

5) Konfigurationshanteringsverktyg (Configuration Management Tools). Denna typ av verktyg har många kvaliteter som möjliggör och underlättar styrning och hantering av migreringsprojekt. Dessa verktyg kan förhindra att problem uppstår när många utvecklare arbetar parallellt, vilket ofta är fallet vid migrering. Dessa verktyg kan bl a användas för versionshantering, uppbyggnad av konfigurationer, synkronisering och kontroll.

Observera att denna funktionella kategorisering fokuserar på de arvssystem som går att dekomponera.

4.2.2 En studie av områden för reengineeringsverktyg

Fyra huvudgrupper av reengineeringsverktyg har identifierats i [8]. Det är verktyg för:

- Programanalys
- Kodförbättring
- Kodändring
- Kod-till-kod-transformation.

Programanalysverktyg stödjer analys av programsystem och deras egenskaper i syfte att hjälpa aktuella aktörer att lättare förstå strukturer, uppfatta väsentliga variabler, procedurer och funktioner, o s v, genom att generera och sammanställa olika typer av mått, diagram och rapporter.

Verktyg för kodförbättring (Code enhancement tools) används för att göra källkod och dokumentation lättare att förstå. Häri inkluderas restruktureringsverktyg, som underlättar kodförståelse genom att implementera moderna programmeringskonstruktioner samt omformaterings- och redokumenteringsverktyg med vars hjälp man kan skapa semantiskt ekvivalent representationer på samma relativa abstraktionsnivå.

Verktyg för kodomvandling (Code reversal) eller reverse engineering läser och abstraherar källkod och lagrar design-information i ett repository som i allmänhet är kompatibelt med något CASE-verktyg. Denna information är ofta presenterad i form av dataflödesdiagram, strukturscheman, ER-diagram (Entity-Relationship), etc.

Verktyg för kod-till-kod-transformation översätter källkod från ett språk till ett annat eller från en språkversion till en annan i samma språk.

Rengineeringsprocessen är en komplicerad och diversifierad process. Systemförnyelse-aktiviteter önskas av, beslutas om och drivs av människor, deras verksamhet och de strategier man har för denna verksamhets bedrivande. Målet för reengineering är inte bara arvskod. Här inryms också beslutsfattande, verksamhetsförståelse och processrepresentation, olika stilar för programmering och underhåll/förvaltning av system, o s v.

Den kategorisering av verktyg som beskrivs här fokuserar enbart på klassifikation av reengineeringsverktyg som berör förståelse och omvandling av arvskod. En betydande mängd verktyg, som t ex verktyg för strukturering arvsdatabaser, har inte alls beaktats.

4.2.3 OVUMs undersökning

I "Reverse Engineering: Markets, Methods and Tools" [21] klassificeras reengineeringsverktyg på marknaden i typer enligt följande:

– *Statiska och dynamiska analysatorer* gör det möjligt att åstadkomma bättre kvalitet i den process som skall föregå förändringen. Om koden är av dålig kvalitet måste den kanske förbättras för att göra förnyelsen möjlig. De dynamiska analysatorerna utför också testning genom att löpa igenom koden.

– *Dokumenteringsverktyg* kan producera pappersdokumentation av utformning/design av ett program, ett jobb eller en fil och utföra reverse engineering för att skapa denna information. Skillnaden mellan dokumenteringsverktyg och sanna reverse

engineering-verktyg ligger i att dokumenteringsverktygen inte lagrar resultatet i ett repository.

– *Reformaterare* ändrar ett programs layout primärt genom att utföra indentering av källkoden för att underlätta läsning. Om reverse engineeringverktyg finns tillgängliga för en speciell miljö, fyller reformaterarna ingen direkt uppgift. Om förändringen skall ske manuellt, fyller dessa verktyg en viktig funktion genom att göra koden lättare att uppfatta och förstå.

– *Restruktureringsverktyg* förbättrar processkoden genom att flytta om block av kod i programmet i avsikt att få en bättre anslutning till principerna för strukturerad programmering. De är användbara oavsett om reverse engineeringverktyg finns tillgängliga eller inte. Reverse engineeringverktygen ändrar ofta om kod till en programdesign som bygger på att strukturerade programmeringsprinciper har använts – de förutsätter att någon form av design existerar. Det är sålunda en nödvändig förutsättning vid ändring att dåligt strukturerad kod struktureras om. Om manuell ändring måste till eftersom inget reverse engineeringverktyg finns tillgängligt så fyller restruktureringsverktygen en mycket värdefull uppgift genom att de anpassar koden till ett lättare förstått och strukturerat format.

– *Modulbrytare* bryter ned kod i moduler av anropbara block. De kan användas före restrukturerare eller om sådana saknas i aktuell miljö, i kombination med reformaterare eller reverse engineeringverktyg.

– *Kodkonverterare* konverterar från ett språk till ett annat. De används i reverse engineering-sammanhang för att göra fler verktyg tillgängliga. Konverteringsverktygen är sällan helt "automatiska" i sitt arbete. Kodkonvertering innebär i sig själv ett avsevärt arbete och kräver ett strategiskt beslut om att förändring kommer att ske.

– *Debugging-verktyg* stödjer både utvärdering och förbättring av programkod. Kodens kvalitet kan kontrolleras dynamiskt genom att verktygen löper igenom koden med hjälp av utvalda testdata. De möjliggör även att kodförbättring kan göras on-line. Verktygen ger ofta vägledning beträffande hur koden kan förbättras. Efter det att restruktureringsverktyg har använts, kan debugging-verktygen ofta användas för att ta bort återstående kodproblem.

– *Datastandardiserare* ger möjlighet att söka efter vanliga mönster och avvikelser från standards i programkodens datadefinitioner. De har också funktioner som grundat på detta kan stödja hopslagning och standardanpassning av datadefinitioner. Dessa verktyg är bara till nytta om koden skall ändras manuellt, eftersom det är lättare att standardanpassa och slå ihop data på högre nivå än på kodnivån.

– *Reverse engineering-verktyg* analyserar gamla system för att så långt möjligt härleda designen/utformningen för jobb, program och databaser/filer. Dessa utformningar och interaktion mellan data och processer lagras i ett repository. Ibland kan de också ge stöd i en fortsatt "reverse" process där t ex datamodellen för en databas-design kan härledas i den utsträckning detta är möjligt. De kan i tillägg också ha funktionalitet som gör det möjligt att involvera mänskligt beslutsfattande i denna process.

4.3 Integrationsstöd

När det gäller frågan om integration, kan vi anta, grundat på den generella process för systemmigrering som beskrivits ovan, att tre typer av integration måste beaktas:

1) Integration som avser arvskraven och kraven på det moderna systemet, liksom de omgivande kraven på migreringsprocessen.
2) Integration som utförs för att samordna de gamla och de nya databasschemana.
3) Integration som svarar för hopslagning av arvsdatabaserna och de moderna databaserna.

Att integrera arvssystem med en modern miljö är ett nytt ämne för schemaintegrations-metodik. Det finns därför säkert ett antal frågeställningar avseende olika typer av konflikter som kan uppstå i framtiden.

4.4 Kategorisering av process och uppgifter vid reengineering

I processen att migrera och förnya arvssystem, är det en mycket viktig uppgift att avgöra när och var man skall använda olika reengineeringverktyg. Av figuren ovan kan vi få en översiktsbild av relationerna mellan de förnyade systemen, omgivningen och arvssystemen, och också av relationerna inom arvskomponenterna som resulterat från migrerings- och reengineeringsprocessen. I detta avsnitt, fokuserar vi på reengineeringskraven på denna process och försöker relatera dem till reengineeringsverktygen.

4.4.1 En funktionsorienterad arkitektur för verktygsanalys

Låt oss anta att ett arvssystem kan ses som en mängd av relaterade komponenter, som var och en utför vissa specifika uppgifter och som producerar input till de övriga komponenterna. En reengineeringskonfiguration innehåller huvudsakligen tre delar: ett arvssystem som skall byggas om, en uppsättning reengineeringsverktyg som skall användas för ombyggnationen av systemet, och ett antal repositories för lagring av återanvändbara systemkomponenter. Vidare finns strategier, metoder, och andra hjälpmedel för att driva reengineeringsarbetet.

Ett arvssystem innehåller i sig data, strukturerade i en eller flera databaser, och bearbetningar eller funktioner, realiserade i program som manipulerar data. Databaserna kan ha samma eller olika struktur och är utformade med hjälp av samma eller olika design-verktyg. De är vidare modellerade med samma eller olika konceptuella modelleringsmetoder. Programmen är troligen utformade med ett eller flera datorspråk, har en eller flera programstrukturer, etc. Det kan vara möjligt eller inte möjligt att konvertera kod och data till en annan form av språk eller modelleringsmetod strukturellt. Kod och data kan vara alltför komplexa och speciella metoder kan behöva introduceras för att förenkla problemet med komplexitet [18].

I Systemförnyelse-projektet har vi träffat på ett antal metoder för reengineering av arvssystem. Dessa metoder kan kallas 1) "direkta" metoder, 2) "indirekta" metoder och 3) "återanvändningsmetoder". De kan beskrivas som följer.

Direkta ansatser. När ett målsystem byggs, översätts programmen i arvsapplikationen till målspråket och arvsdata migreras till måldatabaser. Uppgifterna att analysera arvsapplikationer och data utförs under översättningen och migreringen av arvssystemet. Denna metod kräver att arvssystemen är modulariserade och har hanterbar storlek.

Indirekta ansatser. Det första steget vid reengineering ett arvssystem är att analysera program och databaser eller datafiler och generera en uppsättning mellanresultat, t ex flödesdiagram för programstrukturer, "entity/relationship"-scheman för databaser, t o m kravspecifikationer för arvssystemet. Därefter återutvecklas målsystemet och forward engineering bedrivs med utgångspunkt från dessa intermediära diagram och scheman.

Återanvändningsansatser. I denna metod antas arvssystemet vara oförändrat och kapslas in eller "wrappas" som en enhet eller som en uppsättning av isolerade oberoende enheter och kommunicerar med andra system inom samma miljö via ett speciellt gränssnitt.

För varje ansats finns det reengineringsverktyg som kan användas och som är lämpliga för resp metod. Som sägs i [11], är det för många verktyg på marknaden inte helt klart hur man skall använda dem i praktiska applikationer. Att utföra en funktionell analys av reengineeringprocessen borde kunna hjälpa oss att bestämma vilka krav som behövs för att välja lämpliga reengineerigsverktyg.

4.4.2 Processer i reengineeringsaktiviteter

Baserat på de tre ansatserna ovan som kan användas vid förnyelseaktiviteter, diskuterar vi först möjliga uppgifter, funktioner, och process för reengineeringsverktyg. Därefter föreslås i nästa avsnitt en kategorisering av reengineeringsverktyg.

I den direkta ansatsen behövs översättnings- och migreringsverktyg. För det första översätts arvsapplikationsprogram till målsystem från omoderna till moderna språk. Sådana översättningar eller transformationer förändrar i allmänhet inte de funktioner som arvskoden utför. Från perspektivet testning och validering, utför det resulterande målsystemet samma funktioner som dess arvsmotsvarighet. När översättningsarbetet närmar sig sitt slutförande, börjar migreringsverktygen att flytta data från arvsdatabaserna till måldatabaserna. De relaterade databashanteringsfunktionerna migreras också in i de moderna databasapplikationerna.

För att kunna klara av översättnings- och migreringsuppgifterna, är analys av arvskod och data eller databasstruktur säkert en första uppgift i reengineeringsprocessen. Därefter skall arvssystemet dekomponeras i delar eller komponenter av hanterbar storlek. Om systemet är väl modulariserat (som strukturerade program eller relationsdatabaser) så kan dekomponeringen vara mycket enkel. Om systemet är svårare att dekomponera måste speciella verktyg användas för detta.

Om man använder denna ansats så kan arvssystemen inte vara alltför stora och komplexa, eftersom ansatsen vanligen innebär att man förnyar arvssystemet i ett svep, något som kallas "cold turkey" i [11].

I den indirekta ansatsen består förnyelseprocessen huvudsakligen av två delprocesser: reverse engineering och forward engineering. I reverse engineering-processen analyseras arvskod och data i syfte att komma fram till olika diagram och strukturer, och till deras funktioner och databasscheman.

Denna process kan stödjas med reverse engineeringverktyg. Resultaten används sedan för att utveckla målsystemet på ett forward engineeringsätt.

Utöver analys- och dekomponeringsverktygen som fortfarande är behövliga, så är viktiga verktyg som används i denna ansats reverse engineering- och forward engineering-verktyg. I denna ansats måste en hel del arbete utföras under analyssteget. Fokus måste läggas på sättet att "parsa" koden och strukturera arvsdata. Reverse engineeringverktyg hjälper till att analysera arvssystemen och genererar en del intermediära resultat, som senare kommer att användas för att utveckla målsystemet.

I återanvändningsansatsen hålls arvssystemet oförändrat. Ett gränssnitt upprättas för att koppla arvssystemkomponenterna med systemen utanför inom samma miljö eller så inkapslas det som en enhet och kommunicerar med de andra systemen och omgivningen som en helhet. Under denna omständighet är gränssnitt- och inkapslingsverktygen mycket viktiga, eftersom arvsfunktionerna och arvsdata överförs och utbyts med omgivningen enbart genom denna brygga. Fullständigheten i den överförda informationen och funktionerna som utförs är starkt beroende på den "gateway" som utformas av gränssnitts- och inkapslingsverktygen.

Från analysen av möjliga reengineeringsprocesser ovan, kan vi få fram en skiss till en klassificering av reengineeringsverktyg att användas i reengineeringsprocessen. Man bör också beakta andra typer av verktyg, t ex verktyg som används för att hantera konfigurationer och repositories.

4.4.3 Funktioner i reengineeringsprocessen

Funktionerna vid reengineering av arvssystem kan sammanfattningsvis sägas inrymma följande uppgifter:

- att förstå och analysera arvskod och arvsdata,
- att översätta arvsapplikationerna och migrera arvsdata till nya databassystem,
- att bedriva reverse engineering av arvskod och arvsdata,
- att bedriva forward engineering utgående från mellanresultaten från reverse engineering-aktiviteterna,
- att kapsla in arvssystem ("encapsulation or wrapping") och att skapa gränssnitten till andra noder i omgivningen, och
- att stödja reengineeringsuppgifterna.

Denna kategorisering antas innehålla alla nödvändiga uppgifter som skall utföras i reengineeringsprocessen.

4.5 Några exempel på verktyg

Existing Systems Workbench (ESW), Viasoft Inc.
Innehåller många delverktyg t ex Via/Insight (interaktivt analysverktyg för COBOL-program), Via/Alliance (Application Understanding), Via/SmartEdit, Via/SmartDoc (programdokumentation), Via/Recap (mätning komplexitet), Via/Renaissance (Repackaging), etc.

Bachman toolset, Bachman Information Systems Ltd.
För reverse och forward engineering. Innehåller bl a remodelleringsstöd.

Maestro, Softlab.

Revolve, Micro Focus.

Ett applikations- och databasanalysverktyg.

Visual Quality ToolSet och Visual Reengineering ToolSet inkl bl a Battlemap (BAT), Codebreaker, Instrumentation, McCabe& Associates.

4.6 Sammanfattning

Vi har i detta avsnitt diskuterat verktyg för förnyelse av informationssystem och presenterat en uppsättning karakteristika för klassificering av sådana verktyg.

Följande två aspekter är mycket viktiga för en lyckosam migreringsansats:

– Att välja en uppsättning reengineeringsverktyg
– Att använda verktygen på rätt sätt i en reengineeringsplan eller process.

Detta avsnitt är ett försök att göra en sådan analys av reengineeringsverktyg för dessa två syften.

5. Genomförande

Efter det att man har definierat en strategi enligt vilken migreringsarbetet skall bedrivas kan arbetet gå in i en genomförandefas. Man har genomfört en riktad verksamhetsanalys och olika aktiviteter i migreringsarbetet har definierats på basis av vald strategi.

Det kan t ex innebära att olika former av reverse engineeringsarbete skall genomföras. Det kan vara fråga om remodellering av informationssystem och att olika systemdelar skall förbättras. Nyutveckling av delar kan också var aktuell.

Ofta blir det fråga om att en arkitektur skall definieras, som skall vara basen för förnyelsearbetet och mot vilken olika förbättrade och nya systemdelar skall kommunicera. Det är viktigt att denna arkitektur är klar i sin definition så att olika aktiviteter i form av implementeringsarbete får en klar riktning.

Olika modelleringsarbeten avseende nya systemdelar skall genomföras. Dessa arbeten måste ta sin utgångspunkt i beskrivningar som verksamhetspersoner upprättar och som bedrivs med benägen assistans från modelleringsexperter m fl.

Det är viktigt att utarbeta en plan enligt vilken genomförandet skall bedrivas. Denna plan utgör grund för projektstyrning avseende genomförandet. En projektledare för detta arbete skall utses. Denna projektledare måste vara väl insatt i vad migreringsarbete och olika typer av strategier kräver.

Styrningen av genomförandet skall ske med utgångspunkt från de resultat som har uppnåtts i verksamhetsanalysen samt den strategi för migrering som har definierats. Det kan t ex innebära att om man skall gå mot ökad samverkan mellan relativt självständiga verksamhetsdelar så måste man se till att denna målbild också realiseras.

Ofta kan genomförandet av olika aktiviteter parallelliseras så att man får ett ”concurrent” arbetssätt, olika aktiviteter kan genomföras samtidigt. T ex kan olika insatser för reverse engineering av olika systemdelar ske samtidigt. Vidare kan remodellering av system ske samtidigt som man börjar att skapa modeller utifrån

önskemål och synpunkter från verksamheten. Dessa modellresultat kan sedan med fördel kopplas samman (integreras) så att man får en god bild av vad det förnyade systemstödet skall innehålla.

Genomförandet av olika delaktiviteter, som var och en kan vara ganska komplex, kräver kompetens inom en mängd områden. Det krävs t ex kompetens inom modelleringsområdet såväl som om hur gamla system kan vara uppbyggda ända över till kunskap om hur den nya moderna arkitektur kan vara uppbyggd som är målet för förnyelse.

Genomförandet kräver också att olika programvaror och verktyg måste anskaffas så att de kan utnyttjas för reverse engineering och reengineering. Denna anskaffning bör ha initierats i tid och olika typer av utbildning kan behöva startas så att man har kompetens att hantera olika typer av verktyg.

Genomförandet kräver ett starkt ledarskap och en medvetenhet om syftena bakom förnyelsearbetetet. Dessa syften skall snabbt kunna återges då man är i tvivelsmål om varför man är inne i en viss fas.

Olika former av samordning krävs för att genomföra delinsatser över ett brett fält.

Ett intressant hjälpmedel för att samordna resultat är ett utbyggt repository. Där kan delresultat från olika delinsatser lagras, resultat som avser gamla system eller modeller som är framtagna för att beskriva den framtida verksamheten. Om man har olika resultat lagrade på detta sätt kan man lättare upptäcka skillnader och fel i olika delresultat.

Har man inte erforderlig kompetens för olika delsteg, får man hyra in sådan kompetens eller alternativt tillse att det finns handledare tillgängliga för att personer som genomför olika delaktiviteter skall få stöd.

6. Successiv reviewing

Det är väsentligt att genomförandet sker på ett strukturerat och planerat sätt. Detta för att processen skall bli så styrd och målinriktad som möjligt. Denna målstyrning sker på basis av att förutsättningarna inte ändras.

För att ta reda på om förutsättningarna har ändrats eller att annan relevant kunskap har dykt upp, måste man vid vissa punkter under genomförandet granska så att det inte finns anledning att ändra på insatsernas inriktning – alltså på uppbyggnaden av strategin.

Antag att en strategi har formulerats enligt följande:

- Verksamhetsanalys avseende de affärsmässiga förutsättningarna och målen för verksamheten samt avseende önskad styrning av verksamheten. Vilka nya affärstjänster skall stödjas? Hur skall motsvarande uppgifter utföras i verksamheten?

- Begreppsmodellering avseende berörda verksamhetsdelar.

- Granskning av befintliga system avseende om de kan utvidgas med nya systemkomponenter på grund av krav om nya verksamhetstjänster eller om man måste bygga om dessa delar helt från grunden.

- Val av typ och väg för att bygga upp en ny arkitektur som skall användas för olika system att kommunicera (samverka) kring så att ledtider i verksamheten kan minskas och effektiva tjänstepaket levereras med korta ledtider i verksamheten.

- Reverse engineering av system som måste utvidgas med nya systemdelar i syfte att avgöra hur de är uppbyggda och om de kan utvidgas med nya systemkomponenter.

- Reverse modelling avseende de gamla systemdelar som skall användas och sammankoppling av dessa modeller med de modeller som har kommit fram från modellering av hur verksamheterna vill se sina begrepp.

- Uppbyggnad av sammanhängande modeller inklusive domänbeskrivningar.

- Utveckling av nya systemkomponenter och koppling av dessa till de äldre programmen så att de arbetar väl mot den nya arkitekturen med avseende på systemsamverkan.

- Provkörning av de nya systemkomponenterna.

- Successivt införande av de nya systemkomponenterna i fungerande system och sammankoppling med existerande och preparerade systemdelar.

- Utbildning.

- Uppföljning mot de ursprungliga verksamhetsmässiga kraven avseende att stödja nya verksamhetstjänster.

- Kontroll av att modeller finns i uppbyggt repository.

Med avseende på att denna strategi gäller kan det inträffa att nya förutsättningar uppkommer i verksamheten.

Det kan t ex inträffa att ytterligare nya kundorienterade tjänster måste kunna produceras. Det kan då vara relevant att se till att dessa ytterligare krav tillgodoses i arbetet med de olika aktiviteterna.

De olika aktiviteterna är i detta fall inriktade på de typer av arbete som också är aktuella i det utvidgade fallet och arbetet kommer inte till sin inriktning att förändras i så stor utsträckning. Däremot blir omfattningen ofta större. Det viktiga blir att få med förändringar i förutsättningarna i tid.

Ett successivt reviewing-förfarande kan medföra att denna typ av förändringar i förutsättningarna för migreringsarbete beaktas, så att de olika aktiviteterna i migreringsarbetet blir korrekt inriktade och omfattar de saker som är relevanta. Det är viktigt att se till, när man arbetar med de små detaljerna i varje aktivitet, att man vet att aktiviteten som helhet är inriktad på rätt sätt i enlighet med verksamhetsintentionerna.

7. Sammanfattning

Förnyelse av informationssystem är ett aktivt förändringsarbete och har alltid ett mer eller mindre uttalat syfte.

Man bör inrikta sig på att förnyelse av informationssystem verkligen kommer att få önskade effekter i verksamheten, att effekterna är uttryckta i verksamhetstermer. Det är lätt att endast fokusera tekniska aspekter av informationssystemet vid förnyelsearbetet.

Många organisationer har upplevt att det saknas kunskap om att förnya informationssystem. Man kan konstatera att det saknas sammanhängande metoder för systemförnyelse.

I denna skrift föreslås en metodram som kan användas för att, utifrån förutsättningar och mål, formulera en sammanhängande strategi som har önskade effekter.

Detta bygger på ett antagande om att det inte går att framställa en metod som gäller för alla förnyelseprocesser. Förutsättningar och målbilder är så olika att de kommer att fordra sammansatta angreppssätt som ser olika ut beroende på situation. Vissa situationer innebär att vissa typer av åtgärder är helt inadekvata, i andra situationer kan de vara ytterst väsentliga.

Utgångspunkten är att skapa en successiv migreringsprocess snarare än att förändra system eller systemkluster i ett enda steg. Detta innebär att man kommer att köra gamla och nya systemdelar tillsammans under perioder.

En annan utgångspunkt har varit att utnyttja modern teknik och moderna metoder för att förnya informationssystem. Det gäller att snabbt flytta upp systemarvet på en ny modern nivå så att man inte får leva med gamla arv alltför länge. Det innebär också att man kan använda moderna objektorienterade miljöer tillsammans med gamla systemdelar. Moderna repository-verktyg är också viktiga delverktyg för att förnya system på ett offensivt sätt.

Det innebär alltså att förnyelsearbetet kräver stor kunskap om modern metodik och informationsteknologi om det skall ske på ett offensivt sätt.

Systemförnyelsearbetet måste ta utgångspunkt i vad verksamheten vill och vilka affärsmöjligheter som står till buds. Hela systemförnyelsearbetet skall ha som utgångspunkt att med kunskap förändra systemarv så att man kan ta vara på nya affärsmässiga möjligheter. Det är ofta detta som har varit problemet idag.

En av de stora utmaningarna för en organisation när det gäller migreringsarbete är att människorna i organisationen måste lära om och lära nytt ifråga om informationsteknologi. Har man inte människorna med sig i förnyelsearbetet kommer det att misslyckas eftersom motivation saknas. I förnyelseprocessen måste därför ingå att man bygger in att ny kunskap tillförs för att de personer som har äldre kunskapsramar och som har sysslat med äldre teknologier skall ha en chans att bygga upp nya referensramar. Detta så att man ser möjligheter med ny teknik och nya lösningsplattformar.

En normativ utsaga är också att detta förändringsarbete skall ha ett medvetet syfte i verksamhetstermer.

8. Referenser del II

Verksamhetsanalysdelen

[1] Bubenko jr, J.A: Extending the Scope of Information Modelling, 4th Int. Workshop on the Deductive Approach to Information Systems and Databases, Costa Brava, Catelonia, 1993.

[2] Bubenko jr, J.A. & Wangler, B: Objectives Driven Capture of Business Rules and of Information Systems Requirements, IEEE Systems Man. and Cybernetics'93 Conf, Le Touquet, France, 1993.

[3] Bubenko jr,J.A.et al: Facilitating "Fuzzy to Formal" Requirements Modelling, IEEE Int'l Conf. on Req. Eng. (ICRE94), Colorado, USA, 1994.

[4] Bubenko jr, J.A. & Kirikova, M: "Worlds" in Requirements Acqusition and Modelling, 4th European-Japanese Seminar on Information Modelling and Knowledge Bases, Stockholm, 1994.

[5] Hugoson, M-Å, Hesselmark, O & Grubbström, A: MBI-metoden. En metod för verksamhetsanalys, Studentlitteratur/Programator, 1983.

[6] Kirikova, M & Bubenko jr, J.A: Enterprise Modelling: Improving the Quality of Requirements Specifications, Dept of Computer and Systems Science, Stockholm University, 1994 (IRIS 17).

[7] Nellborn, C. & Holm, P: Capturing Information Systems Requirements Through Enterprise and Speech Act Modelling, CAiSE-94, Utrecht, Netherlands, June 1994.

Verktygsvalsdelen

[8] Arnold, R. S: Software Reengineering. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[9] Berztiss, A: Software Methods for Business Reengineering, Dept. of Computer Science, Univ. of Pittsburgh, Pittsburgh, 1994.

[10] Brodie, M. & Stonebrake, M: DARWIN: On the Incremental Migration of Legacy Information Systems, GTE Laboratories, DARWIN Project TR-0222-10-92-165, March 1993.

[11] Brodie, M. & Stonebrake, M: Migrating Legacy Systems, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1995.

[12] ERCIM. Proc. of ERCIM Workshop on Methods and Tools for Software Reuse, Heraklion, Crete, Greece, Oct. 29-30, 1992.

[13] F3-Consortium: Requirements Engineering Handbook. ESPRIT Project – F3, 1994.

[14] Falk, T. et al: Effektiv IT: En Förstudie, SISU, Stockholm, 1993.

[15] Frazer, J. A: Reverse Engineering: Hype, Hope or Here?, The Institute of Software Engineering, Belfast, 1991.

[16] Gustafsson, M. R. & Johansson, L.-Å: Metodik för reengineering av informations-system – ett eftersatt område, SISU, Effektiv IT Systemarvet, Okt. 1994.

[17] Kalman, K: Reverse Modeling from Relational Database Schemata to Entity-Relationship Schemata, Stockholm University, Master Thesis, 1990.

[18] McCabe, T: A Complexity Measure, IEEE TOSE. Vol. 2. No. 4, 1976.

[19] Mosley, V: How to Assess Tools Efficiently and Quantitatively, IEEE Software Vol. 9. No. 3, 1992

[20] Poston, R. & Sexton, M: Evaluating and Selecting Testing Tools, IEEE Software. Vol. 9. No. 3, 1992.

[21] Rock-Evans, R. & Hales, K: Reverse Engineering: Markets, Methods and Tools, Ovum Ltd, London, 1990.

[22] Ruhl, M. K. & Gunn, M. T: Software Reengineering: A Case Study and Lessons Learned, Software Reengineering, Arnold, R. S. ed(s), IEEE Computer Society Press, Los Alamitos, California, 173-198, 1993.

[23] Song, W. W: OFEM: A Structural Representation to Support Objectives Modelling Integration, The International Conference on Concurrent Engineering, Washington, USA, 1995

[24] Song, W. W: An Analysis of the Case Studies on the Legacy System Migration and Reengineering**,** SISU, ESPRIT Project F3 (P6612), SISU, March 1995.

[25] Song, W. W: Apply the Schema Integration Techniques to the Legacy System Migration**,** DSV Report 95-015, March 1995.

[26] Song, W. W: Integration Issues in Information System Reengineering, The 20th Annual International Computer Software and Applications Conference, Seoul, South Korea, 1996.

[27] Waters, R. C. & Chikofsky, E. J: Working Conference on Reverse Engineering, Baltimore, MD, 1993.

[28] Wong, K, Tilley, S. R, Muller, H. A. & Storey, M. D: Structural Redocumentation: A Case Study, IEEE Software, Jan. 1995.

III Samverkande informationssystem och migrering mot objektorienterad teknik

1. Samverkande informationssystem

Många organisationer vill förnya sina informationssystem av anledningen att man vill samverka mer mellan olika delar i verksamheten. Detta måste oftast ske genom att man skapar en ny grundarkitektur för informationssystemen som tillåter en bättre samverkan mellan olika system. I Bilaga A tas olika principer upp för hur samverkan mellan informationssystem kan skapas.

2. CORBA vid stegvis migrering till objektorienterad miljö

2.1 Inledning

Många företag och organisationer har i dag stora problem med sina gamla system. De är oftast mycket stora och komplexa, kostsamma att underhålla, svåra att ändra i, skrivna i otidsenliga programspråk, dyrbara att köra på stordatorsystem, gjorda att använda specialdatabassystem etc. Om man tänker migrera ett sådant system till en modernare miljö med exempelvis fönstergränssnitt, modulariserad uppbyggnad enligt client-/server- eller tre-nivå-arkitektur och moderna databashanterare är det i många fall inte praktiskt att migrera hela systemet på en gång utan det är lämpligare att dela upp migreringen i flera mindre steg. Denna uppdelning i mindre steg kallas i [1] för ”Chicken Little” jämfört med ”Cold Turkey”, som den konventionella icke uppdelade metoden kallas.

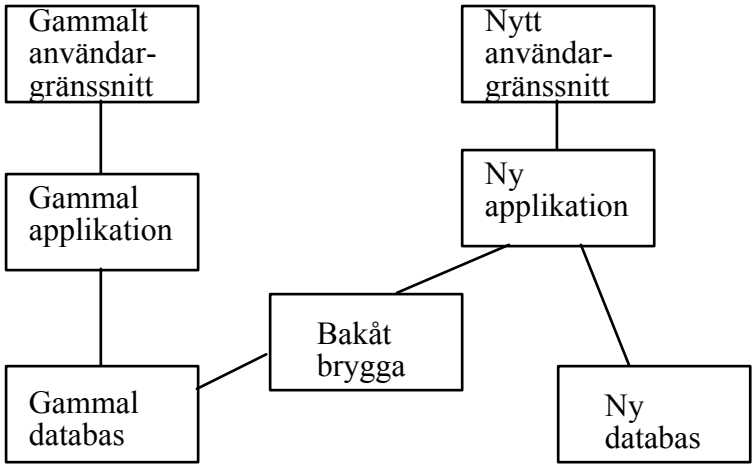
För att minska behovet av migrering i framtiden för de system, som skapas i dag, så är det viktigt att man följer moderna principer för systems uppbyggnad. Sålunda vill man att system skall vara modulariserade med uppdelning i en client-/server- eller tre-nivå-arkitektur. Vidare skall systemen kunna samverka med andra system i en heterogen distribuerad datormiljö. Detta sista krav kan man åstadkomma med en objektorienterad CORBA-miljö [2]. Samtidigt drar man nytta av fördelarna med objektorientering: en effektivare modellering av verkligheten, inkapsling av både data och tjänster, möjlighet till återanvändning etc.

Vare sig man vill migrera ett gammalt system med företrädesvis stegvis migrering till ett nytt system eller man vill erbjuda data och tjänster från ett gammalt system i en objektorienterad miljö, så förefaller tekniken att kapsla in gamla system i ett CORBA-skal mycket lovande. Genom att exempelvis från äldre system erbjuda nya tjänster i en distribuerad objektorienterad miljö, kan man dra nytta av redan existerande funktionalitet samtidigt som det äldre systemet lever kvar eller samtidigt migreras. Detta tillåter företag och organisationer att gradvis övergå till objektorienterad teknik i sin existerande miljö utan att för den skull behöva återskapa alla sina existerande applikationer på en gång.

CORBA är emellertid en ganska ny arkitektur för distribution av objekt. Är den mogen att användas i praktiken? Dessutom har OMG (Object Management Group) många andra aktiviteter än att just specificera CORBA. De specificerar bl a också olika behov av bastjänster att användas tillsammans med CORBA. Finns det användbara implementeringar, som stöder dessa bastjänster? Denna rapport försöker ge några svar på dessa frågor genom att dels genom att beskriva CORBA och dess olika standardiserade bastjänster, dels genom att ange vilken service man kan förvänta sig av CORBA-implementeringar, som finns i dag. Som typiskt exempel på hur långt man har kommit idag, exemplifierar vi med Orbix, en CORBA-implementering från IONA Technologies, Dublin (se http://www.iona.ie), som av många betraktas som ledande inom området och som vi själva har stor erfarenhet av. Man kan förvänta sig att finna ungefär motsvarande funktionalitet hos andra implementeringar av CORBA men knappast mycket mer.

2.2 Stegvis migrering

I [1] beskrivs en metod för stegvis migrering av gamla system till nya system (Chicken Little). För äldre stora, komplexa system, där kontinuerlig drift dessutom är livsviktig för företagets verksamhet, är det helt enkelt inte möjligt att försöka konvertera hela det gamla systemet på en gång (Cold Turkey). Ju svårare det är att dekomponera det gamla systemet i separata moduler, desto svårare är det dessutom att migrera det gamla systemet. I figur 1 nedan visas hur det skulle kunna se ut efter ett migreringssteg för ett dekomponerbart äldre system. Den gamla och nya applikationen lever tillsammans, där den nya applikationen lagrar och hämtar data från den gamla databasen via en bakåtbrygga (Reverse Database Gateway).



Figur 1. Tillstånd hos systemarkitektur vid stegvis migrering.

Exempelvis skulle den gamla databasen kunna vara en hierarkisk databas, medan den nya kan vara en relationsdatabas. Den nya applikationen skulle då också kunna se databasen som en ren relationsdatabas med ett SQL-gränssnitt. Bryggan måste då förstå SQL-koden och översätta den till gränssnittet mot den gamla hierarkiska databasen.

Ett alternativt sätt är att lyfta upp gränssnittet, som den nya applikationen ser, till en högre abstraktionsnivå, företrädesvis ett objektorienterat CORBA-gränssnitt. I stället för att bryggan skall förstå SQL, så måste bryggan i stället implementera CORBA-gränssnittet. CORBA medger sålunda att på ett distribuerat objektorienterat sätt integrera gamla applikationer med nya genom att kapsla in det gamla systemen i CORBA-skal. Den service, som det äldre systemet skall lämna, beskrivs med en IDL-specifikation och när någon operation anropas, så görs lämpliga anrop till det gamla systemet.

2.3 Introduktion till CORBA

2.3.1 OMG

OMG är en intressentorganisation, som bildades år 1989 av 8 stora datorföretag för att ta fram standarder för att olika applikationer skall kunna samverka i en heterogen datormiljö på ett smidigt sätt. OMG har vuxit mycket starkt sedan starten och har numera över 500 medlemsföretag.

OMG valde följande tre principer för sitt arbete:

– Objektorienterad teknik som den bästa tekniken för att bygga samverkande system.

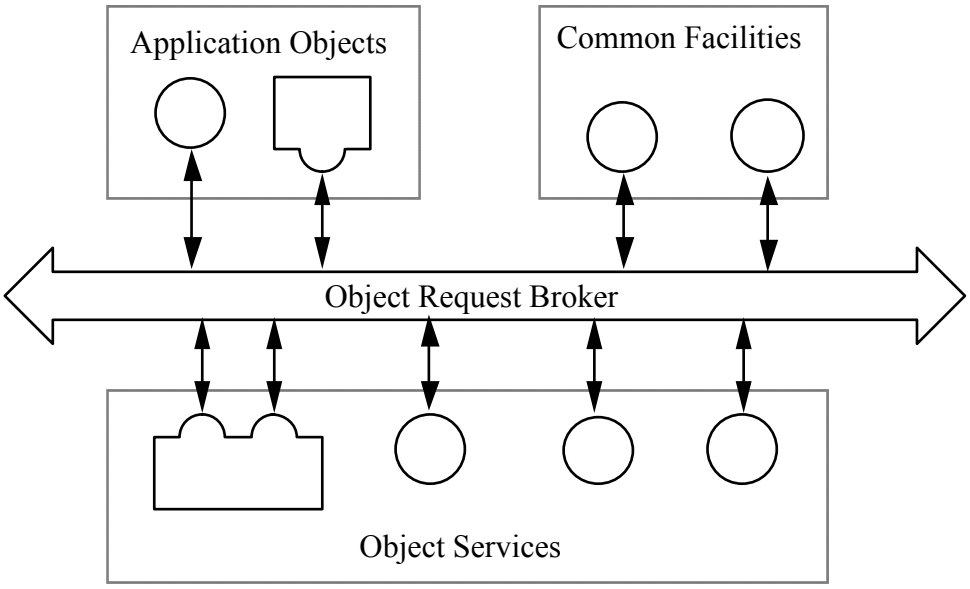
– Endast föreslå standards som samtidigt har testats i någon implementering.

– Att OMG självt inte skall utveckla och sälja programvara.

OMGs första arbete var att 1990 ta fram en objektorienterad arkitektur beskriven i Object Management Architecture (OMA) Guide [3]. Denna arkitektur innehåller bl a en konceptuell objektmodell kallad ”the core object model” och en referensarkitektur kallad Object Management Architecture (OMA), vilken alla applikationer skall anpassas till. OMA beskrivs i följande avsnitt.

2.3.2 OMA Referensmodell

OMAs referensmodell består av fyra delar: Object Request Broker (ORB), Object Services, Common Facilities och Application Objects enligt figur 2.



Figur 2. OMA Referensmodell.

De fyra delarna i referensmodellen utgörs av

– en Object Request Broker (ORB), vilken möjliggör att objekt kan utföra operationer hos andra objekt i en distribuerad, heterogen miljö.

– Object Services är en samling bastjänster hos vissa förutbestämda objekt. Exempel på sådana tjänster är händelsehantering och distribuerad transaktionshantering.

– Common Facilities är en samling objekt och tjänster, som är mera allmänna till sin karaktär.

– Application Objects är objekt, som är speciella för applikationerna.

Alla objekt utom Application Objects ligger under OMGs ansvarsområde.

Den centrala delen är ORB-en, som kan betraktas som en gemensam kommunikations-buss för objekt, med vilken objekt i de tre andra delarna kan samverka med varandra. Ett objekt har typiskt tillgång till en objektreferens till ett annat objekt. Med denna objektreferens kan objektet begära att få utfört en operation hos det andra objektet och få tillbaka resultatet från operationen. ORB-en ser till att med hjälp av objektreferensen finna det anropade objektet, var det än befinner sig (”location transparency”), och få operationen utförd. På vilken typ av dator det anropade objektet finns, i vilket programspråk objektet är implementerat eller under vilket operativsystem det befinner sig är också helt transparent.

2.3.3 CORBA

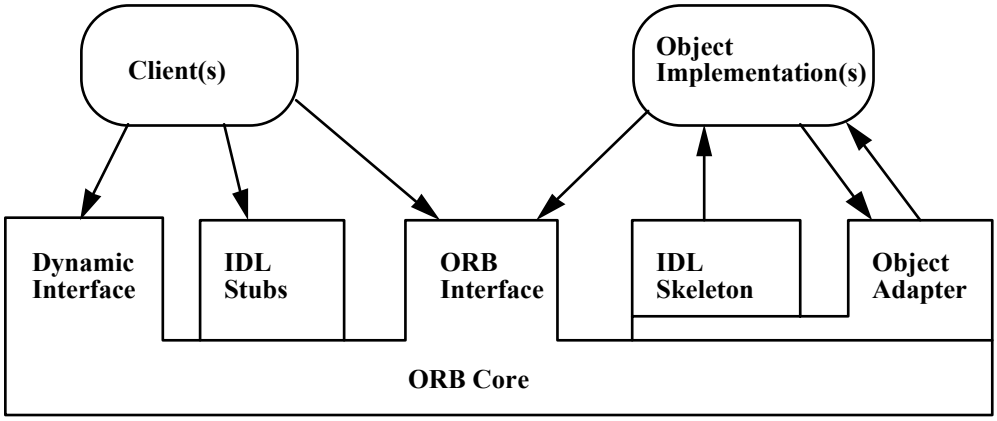
Under 1991 antog OMG specifikationen för CORBA (Common Object Request Broker Architecture) [2] som ORB-komponent i sin OMA.

CORBA kännetecknas av följande egenskaper:

– Separation av gränssnitt och implementering. En klient till CORBA-objekt ser endast det definierade gränssnittet, ej hur objektet är implementerat (maskin- och språk-oberoende).
– Gränssnittet definieras med hjälp av ett speciellt objektorienterat språk IDL (Interface Definition Language). Språket har en syntax, som är tagen från C++.
– Objektreferenser är typade i gränssnittet.
– Multipelt arv.
– Subtypning för att utöka och specialisera funktionalitet.

Eftersom alla gränssnitt definieras med ett och samma språk (IDL), även naturligtvis objekten i Object Services och Common Services, så får man en enhetlig objektmodell i sin CORBA-miljö.

Strukturen hos en ORB enligt CORBA (version 1.x) ser ut som i figur 3.



Figur 3. Strukturen hos CORBA.

En distribuerad CORBA-miljö utgörs av klienter och servrar, där servrarna i detta sammanhang kallas för objektimplementeringar. Det gränssnitt mot objekten, som specificeras i en IDL-specifikation, kompileras med en IDL-kompilator till en IDL-stubb, som klienterna kan använda sig av och till en IDL-skelett, som ORB-en använder sig av för att anropa de faktiska objektimplementeringarna. I exempelvis en C++-miljö utgörs stubbarna och skeletten av C++-kod, men det är inget som hindrar, att de är skrivna i andra programspråk och stubbarna behöver inte heller vara skrivna i samma programspråk som skeletten.

För att de olika objekten skall kunna anropa varandra har det hittills krävts att samma CORBA-implementering (t ex Orbix) har körts på de olika maskinerna. Dessa implementeringar av CORBA uppfyller specifikationen för CORBA 1.x. Den senaste versionen av CORBA-specifikationen, CORBA 2.0 [4], definierar bl a hur CORBA-implementeringar från olika leverantörer skall kunna samarbeta. Dels finns det ett generellt protokoll GIOP (General Interoperability Protocol) definierat för kommunikation mellan olika ORB-ar, dels finns det ett från GIOP specialiserat protokoll IIOP (Internet Interoperability Protocol), som bygger på TCP/IP. För att en ORB skall kunna sägas uppfylla specifikationen för CORBA 2.0, så måste den åtminstone klara IIOP. Orbix 2.x från IONA uppfyller CORBA 2.0.

En klient kan nå andra objekt och deras funktioner (operationer, metoder) med hjälp av objektreferenser till objekten. Om klienten exempelvis är skriven i C++, utgörs objekt-referenserna av normala pekare till objekt. Det normala är att klienten anropar den aktuella funktionen via en IDL-stubb. Genom att konsultera ett ”Interface Repository” kan klienten dynamiskt ta reda på vilka gränssnitt som existerar och alternativt göra anrop
via ”Dynamic Invocation”. Denna möjlighet är betydelsefull för applikationer av typ ”browsers”, som i förhand inte skall behöva veta vilka gränssnitt, som existerar, utan dynamiskt kan arbeta mot godtyckliga objektstrukturer. Dynamic Invocation kräver dock mer programmeringsarbete.

Från och med CORBA 2.0 finns även på serversidan en DSI (Dynamic Skeleton Interface”), som möjliggör dynamisk hantering av objektanrop. Denna DSI utgör serverssidans motsvarighet till klientsidans DII.

När en klient har gjort ett anrop av en funktion hos ett objekt, är det sedan ORB-ens uppgift att lokalisera lämplig implementeringskod för objektet och att anropa funktionen via ett IDL-skelett. När anropet är klart, lämnas returparametrar och funktionsvärde tillbaka till klienten. Vissa av dessa returparametrar och funktionsvärdet kan vara nya objektreferenser, som klienten kan använda för att anropa ytterligare objekt. För att utföra funktionen kan objektimplementeringen behöva anropa ORB-en direkt eller funktioner hos en ”Object Adaptor”. En Object Adaptor tillhandahåller diverse tjänster som exempelvis generering och tolkning av objektreferenser, funktionsanrop via IDL-skelettet, aktivering av objektimplementationer samt registrering av objektimplementationer i ett ”Implementation Repository”. Det kan finnas behov av olika typer av Object Adaptors, som t ex för integrering med objektdatabaser eller integrering med gamla (”legacy”) system. I CORBA-specifikationen definieras endast en generell sådan, nämligen ”Basic Object Adapter” (BOA), som skall finnas i alla ORB-ar.

2.3.4 Object Services

För att en distribuerad objektorienterad miljö enligt OMGs OMA skall bli praktiskt användbar i någon större omfattning, fordras utöver CORBA olika servicefunktioner som exempelvis händelsehantering och transaktionshantering. I OMG-dokumentet Object Services Architecture [5] har man identifierat vilka sådana servicefunktioner, som kan anses nödvändiga. Vissa av funktionerna är viktigare än andra och har därför prioriterats. Hittills har ett antal servicefunktioner antagits av OMG och specificerats i dokumenten Common Object Services Specification, I och II (COSS1 [6, 8] och COSS2 [7, 8]). Övriga servicefunktioner är under bearbetning i så kallade OS RPFs (Object Services Request for Proposal) och väntas antas senare. Under senare tid har också OMG börjat arbetet med att standardisera Common Facilities [9].

Följande servicefunktioner finns specificerade i COSS1:

– Object Event Notification (asynkron händelsehantering)

– Object Lifecycle (service för att skapa, radera, kopiera och flytta objekt)

– Object Name (service för namngivning)

– Persistent Object (allmänt gränssnitt för att permanent lagra objekt)

Följande servicefunktioner finns specificerade i COSS2:

– Object Concurrency Control (upprätthålla konsistens vid samtidig access)

– Object Externalization (att lagra ett objekt i en vanlig fil)

– Object Relationships (hantering av relationer mellan objekt)

– Object Transaction (transaktionshantering i ett distribuerat system)

På tur att specificeras finns servicefunktionerna Object Security, Object Time, Object Licensing, Object Properties, Object Query, Object Change Management, Collections Object och Object Trader. Dessa tjänster är mer eller mindre färdigspecificerade och uppräkningen här följer en ungefärlig prioritetsordning.

Ett gemensamt betydelsefullt drag hos dessa servicefunktioner är att gränssnitten är definierade med IDL. Detta innebär, att alla objekt som existerar i vår distribuerade värld utgörs av CORBA-objekt, vare sig objekten härstammar från servicefunktioner eller från våra egna definierade applikationsobjekt.

Många av dessa servicefunktioner är naturligtvis mycket viktiga komponenter i ett system av praktisk natur. Tyvärr finns det inte ännu så många kommersiellt existerande servicefunktioner tillgängliga på marknaden.

IONA har plockat ut några viktiga servicefunktioner för implementering. Sålunda finns namnservice och händelsehantering enligt COSS1 implementerade i Orbix 2.x, medan transaktionshantering planerades tidigare att släppas under 1:a kvartalet 1996, men har ännu inte sett dagens ljus.

Dessutom har IONA gjort en anpassning mellan Orbix och den objektorienterade databashanteraren ObjectStore motsvarande området COSS2 Persistent Object. Med hjälp av denna anpassning kan man låta Orbix- och ObjectStore-objekten vara samma objekt. Man slipper att skapa kopior av objekten i Orbix och att skriva egna funktioner för exempelvis ladda objekten till Orbix. Man kan dessutom utnyttja ObjectStores faciliteter för transaktioner, frågespråk och backup/recovery, vilka av naturliga skäl saknas i Orbix. Effekten av anpassningen är att man får en förstärkande kombination av Orbix' distribution av objekt med ObjectStores kraftfulla stöd för permanentlagring av objekt.

I följande avsnitt beskrivs översiktligt några viktiga servicefunktioner. Dessa funktioner, nämligen namnservice, händelsehantering och transaktionshantering, kan tas som exempel på hur servicefunktionerna ser ut.

2.3.4.1 Namnservice

Med namnservice kan man namnge objekt med en association, som kallas namnbindning (name binding). Denna namnbindning är alltid definierad inom ett visst namnområde (name context). Detta namnområde utgörs av ett objekt, som innehåller alla unika namnbindningar för området. Med hjälp an operationerna bind och unbind kan man sätta namn och ta bort namn hos objekt. Med hjälp av operationen lookup får man en objektreferens tillbaka för ett namngivet objekt. Med hjälp av denna

objektreferens kan man sedan nå och skapa andra objekt. Typiskt är därför de namngivna objekten grova objekt (serverobjekt), som i sig innehåller många andra mer finfördelade objekt.

2.3.4.2 Händelsehantering

Ett standard CORBA-anrop innebär en synkron exekvering av en operation hos det anropade objektet. Parametrar och returvärde transporteras mellan klient och server. Om anropet misslyckas av någon anledning, får klienten ett felavbrott (exception) tillbaka och måste vidta lämpliga åtgärder.

I många sammanhang behövs i stället en asynkron kommunikation mellan objekt. Exempelvis skulle ett objekt tänkas vilja ha reda på, när attributvärden ändras för vissa andra objekt. Detta sköts med skapade händelser.

Servicefunktionen händelsehantering (Event Notification) är mycket generell till sin karaktär och bygger på att objekt kan anta rollerna leverantörsroll och mottagarroll. Leverantörsobjektet skapar händelser, medan mottagarobjektet konsumerar händelsen. Initiativet för att ta hand om händelser kan komma antingen från leverantören (push model) eller från mottagaren (pull model).

Händelsehantering kan antingen ske med direkt kommunikation mellan leverantör och mottagare eller också indirekt genom att skapa särskilda händelsekanalobjekt, som leverantörsobjekt och mottagarobjekt kan registrera sig hos. Händelsekanalobjekten tillåter att flera leverantörer kan kommunicera med flera mottagare. För övrigt är händelsekanalobjekten precis som alla andra serviceobjekt standard CORBA-objekt och anropen av funktioner hos dessa objekt sker med standard CORBA-anrop. Med ett CORBA-objekt menas det abstrakta objekt, som man ser via IDL-gränssnittet och som anropas via en objektreferens. Hur det verkliga objektet ser ut och hur det är implementerat är dolt.

Händelsetjänsten kan implementeras på många olika sätt beroende på olika krav och olika operativsystemmiljöer. Exempelvis kan trådar (threads) stödjas i operativsystem som stöder trådar, annars inte.

IONA har relativt nyligen släppt sin version av CORBA-servicen händelsehantering. Den heter OrbixTalk och realiserar såväl gränssnittet i CORBA-servicen som en egen påbyggnad, som gör det lättare för programmeraren att hantera händelser.

2.3.4.3 Transaktionshantering

Servicefunktionen Object Transaction är antagen och specificerad i COSS2 [7, 8]. Hur transaktionshanteringen är tänkt att se ut mera i detalj finns specificerat i ett särskilt dokument [10].

I databashanterare är transaktioner ett viktigt instrument för att skapa konsistenta data. I en distribuerad miljö tillkommer problemet att en transaktion kan omfatta flera databashanterare samtidigt.

Transaktionshanteringen, som den är specificerad i [10], kännetecknas av följande egenskaper:

– Stöd för såväl en flat som en grupperad transaktionsmodell.

– En transaktion kan innefatta både ORB- och icke-ORB-applikationer och resurser.

– Samverkan mellan X/Opens Distributed Model (DTP) [11].

– Både klienter och servrar kan propagera en transaktion vidare.

– Implementeringar kan utnyttja existerande TP-monitorer.

Det finns en stor frihet i vad man vill ta med i en viss implementering av transaktionsservicen. Exempelvis behöver man inte stödja en grupperad transaktionsmodell utan enbart en flat sådan.

För att en transaktion skall uppnå ACID-egenskap (Atomic, Consistent, Isolated, Durable) med många distribuerade objekt inblandade, används ”2-phase commit”. De objekt, som ingår i en transaktion och som samtidigt förändrar sitt tillstånd, registrerar ett resursobjekt hos transaktionsservicen. Detta resursobjekt har sedan ansvaret att se till att protokollet för 2-phase commit fungerar. Resursobjektet måste bl a ha operationer för prepare, commit och rollback. Om exekveringen av resursobjektet försvinner p g a något fel, så skall objektet kunna återskapas och skall enligt 2-phase commit-protokollet kunna utföra riktiga åtgärder för varje möjlig situation. Detta innebär bl a att vissa tillståndsdata måste ha lagrats på ett ”oförstörbart” medium.

2.3.5 COM/CORBA

Microsofts COM (Component Object Model) utgör basen för Object Linking and Embedding (OLE). Den nya ”Distributed COM”, som förväntas att se dagens ljus under 1996, är i princip en konkurrent till CORBA. Största nackdelen med COM är emellertid att den är knuten till PC-miljö och är framtagen av ett enda företag, medan CORBA är en generell arkitektur, som ej är knuten till en viss datormiljö. Nu kommer det lika fullt att utvecklas en hel del applikationer, som använder sig av Microsofts COM. Därför är det mycket intressant att kunna koppla samman en COM-miljö med en CORBA-dito. OMG har därför för närvarande en RFP (Request For Proposal) ute för att få synpunkter på hur en sådan koppling skulle kunna se ut.

IONA har föregått den kommande förslaget för koppling mellan COM/OLE och CORBA och har skapat en version av Orbix för Windows NT och Windows 95, där man från Windows-applikationer kan nå inte bara Windows-objekt utan även CORBA-objekt i andra miljöer. En naturlig uppdelning i en tredelad arkitektur är att Windows-applikationen tar hand om användargränssnittet, medan CORBA handhar delarna med logik och datalagring.

Medan Microsofts COM använder globalt unika identifierare (128 bitars heltal), identifieras objekt i CORBA med objektreferenser, en implementationsberoende typ garanterad att identifiera samma objekt varje gång referensen används i en operation. Däremot är inte objektreferenserna enligt CORBA-specifikationen garanterade att vara unika.

2.4 CORBA på olika plattformar

I migreringssammanhang är det mycket vanligt att träffa på gamla system, som körs på stordatorer och som man vill migrera till ny teknik och miljö. I enlighet med principen

för stegvis migrering enligt ”Chicken Little”, kan det då vara en intressant framkomlig väg att kapsla in delar av det gamla systemet i CORBA-skal. Då är det också en stor fördel, om det finns CORBA-implementeringar, som är körbara direkt i stordatormiljön.

Hittills har det i första hand funnits implementeringar för UNIX, men även för PC Windows (95 och NT) och MAC. Vad som är intressant nu, är att det även börjar dyka upp implementeringar för IBM stordatormiljöer. Exempelvis har IONA annonserat ut sin Orbix/Enterprise för ”Early Participation”, där man har möjlighet att testa deras beta-version. Den första versionen är körbar på MVS. Stöd för CICS är planerat att komma under 1997.

Ytterligare intressant är det att det kommer fler och fler CORBA-implementeringar, som stöder CORBA 2.0, vilket innebär, åtminstone i teorin, att olika ORB-ar kan prata med varandra. Vi närmar oss därmed möjligheten till miljöer med helt integrerade applikationer skrivna i olika språk och som körs på olika plattformar och operativsystem.

3. Referenser del III

[1] M. Brodie, M. Stonebraker. Migration Legacy Systems, Gateways, Interfaces & The Incremental Approach, Morgan Kaufmann Publishers, 1995.

[2] Object Management Group. The Common Object Request Broker: Architecture and Specification, John Riley & Sons Inc., 1992.

[3] Object Management Group. Object Management Architecture Guide (OMA Guide), Second Revision, Object Management Group, Framingham, MA., Sept. 1992.

[4] Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 2.0, Object Management Group, Framingham, MA., July 1995.

[5] Object Management Group. Common Object Services Architecture, Object Management Group, Framingham, MA., Januari 1995. OMG Document 95-1-47.

[6] Object Management Group. Common Object Services Specification, Volume I. John Wiley & Sons, Inc, 1994.

[7] Object Management Group. Common Object Services Specification, Volume II. John Wiley & Sons, Inc, 1994.

[8] Object Management Group. CORBAservices: Common Object Service Specification, Object Management Group, Framingham, MA., Mars 1995.

[9] Object Management Group. Common Facilities Architecture, Revision 4.0, Object Management Group, Framingham, MA., Jan. 1995. OMG Document 95-1-2.

[10] Object Management Group. Object Transaction Services, Object Management Group, Framingham, MA., Jan. 1995. OMG Document 94-8-4.

[11] Distributed Transaction Services: The XA Specification, X/Open Document XO/CAE/91/300, December 1991.

IV Fallbeskrivningar

1. Migrering inom sjukvården – ett fall

1.1 Inledning

Informationssystemarv förekommer i stort sett i alla branscher. I följande avsnitt skall vi litet mer handgripligt studera ett exempel på hur man kan förnya ett systemarv. Exemplet kommer från en konkret verksamhet – en sjukvårdsverksamhet.

Inom sjukvården står man inför stora uppgifter när det gäller att skapa en mer sammanhängande informationssystemmiljö med en genomtänkt arkitektur.

Hitintills har man datoriserat på ett ganska fragmenterat sätt genom att man låtit systemdelar och enskilda system växa upp på ett spontant sätt. Det innebär att olika system såsom journalsystem, patientadministrativa system, röntgensystem, etc tillåtits att etablera sig utan att olika krav avseende den arkitektur på vilka de skall vara byggda har ställts. Speciellt märker man detta då man vill knyta samman olika system så att dessa kan kommunicera.

I fallet som skall presenteras har man en situation där man vill skapa en bättre samverkan mellan olika sjukvårdsenheter. Man vill också att system skall vara byggda på sådant sätt och ha sådana egenskaper att de kan medverka till detta.

1.2 Bakgrund

Inom svensk sjukvård, och även på andra håll i bl a Europa, är man intresserad att realisera ett koncept ofta refererat till som vårdkedjor. Det innebär att olika sjukvårdsenheter kan fokusera på ett antal vårduppgifter som sammanhänger i en vårdprocess. Ofta kan denna process börja i primärvården där man konstaterar att patienten behöver fortsatt vård, varför han/hon slussas vidare till andra enheter som utför olika typer av behandlingsinsatser och åtgärder tills patienten blir färdigbehandlad och kommer in i ett uppföljningsskede.

Målet med dessa processer är att de skall löpa effektivt och att behandlingsarbetet skall uppfattas sammanhängande för patienten. Informationsflödet är väsentligt för att denna process skall kunna löpa smidigt.

Det innebär att olika typer av information skall kunna flyta mellan de olika sjukvårdsenheterna för att t ex boka tider, medicinska uppgifter skall kunna utbytas, olika typer av provsvar skall kunna kommuniceras, och att olika typer av remisser skall kunna sändas iväg och följas upp.

De informationssystem som idag används av sjukvårdsverksamheterna är i mindre omfattning inriktade på att kommunicera. De flesta systemen är inriktade på att stödja de enskilda delarna i processen, mer än att sörja för sambanden mellan enheterna.

1.3 Ansats

I förevarande fall är det egentligen två frågor som måste lösas på informationssystem-nivån och som har sin utgångspunkt i mål på verksamhetsnivån:

1) Man vill skapa samverkan mellan informationssystem som baseras på en samverkan mellan sjukvårdsenheter på verksamhetsnivån.

2) Man måste ta hänsyn till ett systemarv på så sätt att dessa system är delar varifrån man hämtar data som sedan utgör samband med system från andra delar. Det är i detta fall inte realistiskt att man bygger nya system för de olika systemdelarna utan arvssystemen måste vara delar i en samverkansmiljö.

Uppgiften för systemförändringen blir följaktligen att hitta en arkitektur för samverkan mellan sjukvårdsenheterna och deras system så att informationen bättre kan flyta mellan enheterna och deras resp system.

Det finns flera sätt att skapa samverkan mellan informationssystem. T ex kan olika sätt att skapa meddelandesamverkan mellan informationssystem användas. Ett sätt är att skapa en meddelandesamverkan mellan system och verksamheter.

Strävan bör vara att få en arkitektur som innebär att varje sjukvårdsenhet har sitt eget ansvar baserat på sina egna uppgifter i så hög utsträckning som möjligt.

Finns det andra arkitekturer som realiserar detta mål? Projektet beslöt sig att prova på ytterligare en arkitektur som baserar sig på en standard som just nu är under framväxt.

System och systemarkitekturer kommer troligen i framtiden alltmer att bygga på komponenter. Dessa har troligen sin utgångspunkt i objekt som man uppfattar i verksamheten och i hur dessa objekt beter sig i verksamheten.

Situationen blir mer komplex om man också beaktar att dessa objekt kan bli aktuella för verksamheter som samarbetar inom ramen för olika processer som samverkar.

1.4 Genomförande

Det första steget när det gäller att få upp en struktur och skapa vissa primära kopplingar till arvssystem håller just nu på att genomföras.

Det är viktigt att de steg som tas i ett migreringsarbete är delar av en större migrerings-insats. Mot bakgrund av förutsättningarna i STAR har därför följande delaktiviteter påbörjats:

a) En studie har bedrivits avseende hur de aktuella sjukvårdsverksamheterna ser ut och hur man vill att de skall se ut. På vilket sätt vill man skapa en förbättrad samverkan? Hur ser de delverksamheter ut som skall förses med nytt informationssystemstöd och vad skall de ha för mål?

b) En studie har bedrivits avseende målet för STAR. Vad är det för typ av samverkan man vill stödja? Hur vill man att verksamhetsdelarna skall samverka?

c) Baserat på hur man har avsett att samverka, dels i de aktuella verksamheterna och dels synsätt man har ha inom STAR, finns det några tänkbara systemarkitekturer som kan vara lämpliga och som kan realisera denna form av samverkan?

d) Vilka av arvssystemen måste man ha kvar åtminstone på kort sikt för att lösa de lokala uppgifterna för verksamhetsdelarna? Hur skall man åstadkomma att de också skall stödja den avsedda formen av samverkan mellan verksamhetsdelarna?

e) Studier av de befintliga arvssystemen och avgöranden av huruvida man kan förnya dem så att de både löser de lokala verksamhetsuppgifterna bättre och stödjer de nya samverkanskraven. Eller skall man bygga nya sådana system?

f) Baserat på den nya arkitekturen för samverkan och baserat på att man vill förnya åtminstone någon del av de nya systemen – hur kan man ansluta de utpekade arvssystemen till den nya samverkansarkitekturen? Hur skall den gamla strukturen hos arvssystemen anslutas till den nya samverkansarkitekturens sätt att fungera?

g) Studier av arvssystemen och hur de uppbyggda.

h) Utformning av en noggrannare specifikation avseende vad som skall ingå i samverkan mellan sjukvårdsenheter samt hur de enskilda sjukvårdsenheterna skall arbeta med avseende på detta.

i) Noggrannare beskrivning av varje steg i hela samverkansprocessen avseende remiss- och svarsprocessen. Vad är det som skall göras i varje arbetssteg? Vilka beslut tas och vad skall finnas tillgängligt för att ta varje beslut?

j) Utformning av en detaljerad begreppsmodell avseende det om vilket man skall samverka mellan enheterna.

k) Utformning av IDL-modeller (IDL=Interface Definition Language, se del III avsnitt 2). Dessa modeller definierar hur samverkan mellan delsystemen skall fungera.

l) Beskrivning av operationer avseende händelser som man vill att systemet skall kunna avbilda ifrån verksamheten.

m) Baserat på vad som skall kommuniceras, utformas ett gränssnitt mot arvssystemen som skall ingå i miljön.

n) Beskrivning av hur man kan kommunicera direkt med arvssystemens databaser där så är adekvat.

o) Uppbyggnad av gränssnitt på lägre nivå mot ett av arvssystemen.

p) Skapande av klientdelar som skall kommunicera med kommunikationsnivån.

q) Kontroll av att samverkansnivån fungerar enligt prov.

r) Annat arvssystem provas i mindre skala.

Det bakomliggande syftet har varit att de olika sjukvårdsenheterna skall bli effektiva och autonoma i sina uppgifter och i sitt ansvar, men att de också effektivt måste kunna kommunicera med de andra sjukvårdsenheterna i vårdkedjan så att den totala vårdprocessen kan bli effektiv.

Sjukvården har i liten utsträckning arbetat med system som har samverkat mellan olika sjukvårdsenheter. För att sjukvården skall kunna bli effektiv, måste man ha informationssystem som kan kommunicera och spegla hur olika objekt, t ex patienter, går från de ena delprocessen till den andra.

I det aktuella fallet har man valt ut remissen som en viktig aspekt som man måste kunna samverka effektivt om inom ramen för en vårdkedja. En remiss är en begäran om att få ett antal tjänster utförda för att patienten skall bli diagnosticerad på ett korrekt sätt och få en adekvat behandling.

Vissa undersökningar kan inte göras inom ramen för den lokala sjukvårdsenheten utan utförs alltså genom en begäran mot en annan enhet. De tjänster som man behöver anlita kan finnas i flera sjukvårdsenheter. Innan dessa tjänster genomförs, utförs ofta en kontroll av huruvida begäran är korrekt och om det verkligen finns fog för den utställda begäran om få aktuell tjänst utförd i förhållande till noterade symptom och till åtgärder som har vidtagits i form av förberedande undersökningar o s v. Olika andra förutsättningar och informationskomponenter måste också finnas tillgängliga för att tjänsterna skall kunna utföras på ett korrekt sätt.

Den utställande enheten behöver dessutom, och detta är viktigt, följa upp hur det går med de utställda remisserna. Man frågar och söker: Var befinner sig remissen just nu? Har det utfärdats något svar och var befinner sig i så fall svaret?

I det förevarande fallet har en speciell ansats för objektsamverkan byggande på det s k Object Brokering-konceptet studerats och provats.

Olika sjukvårdsenheter utför sina uppgifter och behöver för detta betrakta olika objekt som patient, undersökningsresultat o s v. Samverkande objekt fungerar på ett sådant sätt att man t ex på vårdcentralen skapar ett objekt som motsvarar den skapade remissen. När denna sänds iväg till ett specialistsjukhus som skall utföra tjänsten, skapas att nytt objekt som motsvarar den mottagna remissen på kliniksidan. Remissen kopplas också till patientjournalen på den remitterande sidan.

Vårdcentralen kan sedan fråga efter hur status är med avseende på den ivägsända remissen genom att fråga objektvärlden på kliniksidan om vilket status aktuell remiss har uppnått där. Detta får man göra om man har uppgiften att göra en sådan uppföljning. Kliniken blir tvungen att tillhandahålla information avseende var de mottagna remisserna befinner sig eftersom det är vårdcentralens uppgift att följa upp detta.

När specialistenheten har utfärdat ett svar med avseende på remissen, skapas ett objekt som motsvarar detta svar och sänds iväg till den vårdcentral som ställt ut remissen. När svaret anländer till vårdcentralen skapas ett svarsobjekt som kan införlivas med patientjournalen.

2. Systemförnyelse inom en bankverksamhet

Den aktuella bankverksamheten har gått in i en intensiv period för att förnya sina informationssystem.

Verksamheten har en mängd stora viktiga system som utgör de stora delarna av tjänster i kärnverksamheten. Dessa system har hunnit bli ett antal år gamla och har använts längre än vad som ursprungligen planerades. De stora tunga systemen som används för betalningsförmedlingstjänster är idag av batch-typ. Det innebär att de inte på kort tid kan omvandlas till en systemtyp som t ex erbjuder on-line transaktionshantering.

Det har på senare tid inträffat saker som gör att systemförnyelsefrågorna har fått stor aktualitet.

Dels måste bankverksamheten gå in i en hårdare konkurrenssituation där nya aktörer levererar tjänster som ligger nära de tjänster som har varit stommen i bankverksamhetens kärnverksamhet. Olika informationssystem måste kompletteras och byggas om för att man skall kunna realisera de nya kompletteringstjänster som konkurrensläget kräver. Det gäller bl a vissa kompletteringstjänster som kommer att finnas i anslutning till de viktiga betalningstjänsterna som har varit viktiga för bankverksamheten under en mängd år.

Det andra skälet till att man vill att systemen skall förändras är att man upplever att de äldre informationssystemen är svåra att ändra och bygga ut. Eftersom man har, och bedömer att man alltmer kommer att få en situation där man måste ändra i befintliga system, vill man se till att de har en sådan status att de tillåter förändring med bibehållen tillförlitlighet.

De system som skall kunna förändras hanterar idag mycket stora betalningsbelopp varje dygn.

Man har insett att det är orealistiskt att bygga om de äldre systemen på kort tid och ersätta dem med nya system som snabbt skall sättas in i produktion och där man samtidigt kan påräkna att de nya systemen har god tillförlitlighet med få fel. Detta har inneburit att man har gått in för att söka kunskap om att kunna migrera och förnya sina system på ett successivt sätt.

2.1 Strategi för bankverksamheten

Verksamheten har också fattat ett beslut om hur förnyelse av informationssystem skall gå till. Det skall ske genom att ett s k integrerat CASE-verktyg skall användas. Utifrån specifikationer som skall byggas upp för systemdelar som skall förnyas via denna nya miljö skall man kunna få hjälp från verktyget att få fungerande system.

Detta innebär att man måste arbeta på ett annat sätt i verksamheten när det gäller att förnya system. Specifikationer för informationssystem måste göras på ett bättre sätt än hittills. Detta kräver nytt kunnande som måste ställas till förfogande i verksamheten.

CASE-verktygsmiljön skall användas för att realisera nya systemdelar på ett nytt sätt. En modellbas skall byggas så att man lättare hittar information om hur existerande system är realiserade. Där skall finnas verksamhetsnära begrepp, som t ex

"Försändelse", kopplat till hur olika tabeller och koddelar ha realiserat begreppets intention och operationer maskinmässigt.

2.2 Några observationer

Det integrerade CASE-verktyget ställer en del krav som kan vara svåra att klara för verksamheten och förnyelsearbetet.

Organisationen måste i hög grad lära sig att skapa sådana modeller som CASE-verktyget måste ha för att generera fungerande system. Alla egenskaper som systemet skall ha måste kunna uttryckas av den användande organisationen i enlighet med de uttrycksformer som verktyget ställer till buds.

Man kan konstatera att många av de systemutvecklare som arbetar i en organisation som har en stor mängd arvssystem i stor grad är utbildade på konventionella programmeringsspråk. De har svårigheter att lära sig nya former av specifikationsspråk som moderna CASE-verktyg bygger sin generering på.

Varje CASE-verktyg har i viss mån egna uttrycksformer som man måste sätta sig in i. Vissa grundmodelleringstekniker finns, men man måste också bygga upp kunskap om den teori som dessa beskrivningstekniker bygger på .

Man kan också konstatera, att CASE-verktyg som bl a försöker skapa en genereringsmiljö i vilken man skall kunna generera fungerande system baserade på speciella språk, inte i nämnvärd grad har fokuserat på frågan att kunna koppla nybyggda systemdelar i verktyget med delar från gamla system som man vill kunna köra fortsättningsvis.

Integrerade CASE-verktyg har också andra egenheter som grundas på att de har uppfattningen att man ofta bygger interaktiva system och inte batch-system. I en systemförnyelse är det inte alltid uppenbart klart om man skall gå ifrån en satsvis bearbetning till en realtidsapplikation.

Den studie som SISU har bedrivit tillsammans med bankverksamheten har därför koncentrerat sig på hur man kan koppla befintliga system till systemdelar genererade i det aktuella CASE-verktyget.

Detta kan man klara på olika sätt. I CASE-verktyget finns faciliteter, som innebär att ett gränssnitt skall kunna användas för att utväxla data för systemdelar genererade i CASE-verktyget och t ex en databas som ingår i ett system och som det är orealistiskt att byta ut i ett enda slag.

I bankverksamhetsfallet är direktivet att kunna förbereda existerande system så att man skall kunna bygga ut dem på olika sätt i enlighet med nya affärskrav som man vet kommer.

Man har inom bankverksamheten tagit ett beslut att det integrerade CASE-verktyget skall användas. Det skapar restriktioner och målvariabler som inte går att påverka.

Integrerade CASE-verktyg kanske inte har den öppenhet som man skulle önska sig i olika avseenden. Man har emellertid under gång fört fram möjlighet att kunna

kommunicera med systemdelar genererade enligt koncepten för det aktuella CASE-verktyget.

Det finns dock anledning att varna för att användande av dagens integrerade CASE-verktyg kan medföra att man drar på sig ett framtida arv av system genererade på ett visst sätt. Man blir hänvisad till att utrycka systemegenskaper i ett specifikationsspråk och i en realiseringsmiljö som leverantören har utformat och till att leverantören fortsätter att underhålla miljön.

Leverantörer och underleverantörer kan dra sig ut ur marknaden, vilket i värsta fall kan innebära att nya systemversioner inte kan skapas när genereringsmiljöerna inte längre underhålls.

2.3 Föreslagen strategi för bankverksamheten

Följande kombinationer kan ses som delar av en strategi, som föreslagits i verksamheten efter det att studier bedrivits av vad som skulle kunna vara lämpligt:

a) En viss mängd av systemportföljen förnyas i ny CASE-miljö.
   En stor mängd existerande system som inte är alltför prestandakritiska skapas i den nya CASE-miljön.

b) Stora system – konventionell programmering.
   De stora transaktionstunga systemen bedöms som svåra att få goda prestanda på om man använder sig av en CASE-miljö för generering. Dessa miljöer programmeras därför i konventionella språk.

c) Nya system – objektorienterade system.
   Vissa nya system är inte prestandakritiska, men måste vara attraktiva och anpassade i enlighet med nya affärsmässiga krav i nya tjänster. Dessa skapas i nya utvecklingsmiljöer.

d) Uppbyggnad av repository.
   Förnyade systemdelar beskrivs i ett samlat repository. Där finns såväl begrepps-modeller på verksamhetsnivån som modeller av hur olika systemdelar ligger realiserade.