Komponentbaserade informationssystem

- arkitektur, livscykel och systemutvecklingsprocess

Abstract

Om en verksamhet väljer att använda ett objektorienterat komponentsynsätt i datorbaserade informationssystem påverkar detta val verksamheten på olika sätt. Syftet med denna rapport är att beskriva och kartlägga hur systemutvecklingsprocessen påverkas. Ett ytterligare syfte är att identifiera de arkitekturer och infrastrukturer som finns i verksamheter som har komponentbaserade informationssystem. Rapporten är i huvudsak baserad på litteraturstudier. Arbetet har bl.a. påvisat att en verksamhets informationssysteminfrastruktur är beroende av den informationssystemarkitektur verksamheten har. Dessutom framgår det att systemutvecklingsprocessen påverkas i samtliga faser ett informationssystem genomgår i livscykeln då ett komponentsynsätt används. Ett mer flexibelt tillvägagångssätt påvisas i framförallt implementerings- samt förvaltningsfaserna genom den flexibilitet användandet av mjukvarukomponenter ger. En del av den problematik som användandet av mjukvarukomponenter medför belyses även som exempelvis hur kan extra 'icke önskvärd' funktionalitet hos en komponent hanteras.

Benneth Christiansson
Institutionen för informationsteknologi
Högskolan i Karlstad

98-10-04

Innehållsförteckning

1. Inledning

I dagens verksamheter ökar behovet av väl fungerande och ekonomiskt försvarbara informationssystem. Då en verksamhet väljer arkitektur för sina informationssystem kommer valet att påverka verksamhetens möjlighet till framgång inom sitt verksamhetsområde. Ett väl fungerande informationssystem kan betraktas som en konkurrensfördel gentemot andra verksamheter i samma bransch. En tänkbar strategi som börjar etablera sig inom systemutvecklingsområdet är att bygga informationssystemen komponentbaserade. Detta är ingen ny tanke utan redan 1968 presenterade Douglas Mcllroy konceptet mjukvarukomponent som byggblock vid mjukvaruproduktion (Vaughn, 1990). Hans vision var en mjukvaruindustri som massproducerade standardkomponenter som kunde köpas över disk. Man önskade en produktkatalog där varje komponent fanns beskriven, som användaren (programmeraren) kunde använda för att beställa de komponenter programmeraren behövde för att sedan foga samman dem till ett skräddarsytt informationssystem. Detta skulle medföra stora förenklingar och rationaliseringar avseende konstruktion och eventuellt även användning av informationssystem.

Motiven till att verksamheter väljer att utveckla och använda komponentbaserade informationssystem kan vara flera. I de flesta fall finns det en uppfattning om att användandet av mjukvarukomponenter skall leda till positiva effekter såsom lägre kostnader, högre effektivitet kortare leveranstider, högre kvalitet, större flexibilitet och anpassningsbarhet (Jacobson et al., 1997b; Steel, 1996a; Sutherland 1996; Szyperski, 1997). Steel (1996a, p. 4) menar att: *"The ideal must be, therefore, to build tailored software from standard components, capitalising on the speed and quality advantages of bought-in-software without suffering the disadvantages of inflexibility*". Frågan är om användandet av ett komponentsynsätt verkligen kommer att uppfylla dessa förväntningar. Vilka krav ställs på verksamhetens utvecklings samt förvaltningsprocesser? Hur påverkar de arkitekturval som görs (implicita eller explicita) en verksamhets informationssystem? Det är dessa frågor som denna rapport skall försöka besvara.

1.1 Syfte & tillvägagångssätt

Syftet med denna rapport är att:

- *Beskriva* och *kartlägga* en systemutvecklingsprocess samt livscykel vid utveckling av komponentbaserade informationssystem.
- *Identifiera* de arkitekturer och infrastrukturer som finns i verksamheter som har komponentbaserade informationssystem.

Denna rapport är en dokumentering av resultaten från fas 2 i forskningsprojektet KOMPASS (komponentbaserade informationssystem). KOMPASS är ett forskningsprojekt som syftar till att undersöka effekter av att använda objektorienterade komponentsynsätt i datorbaserade informationssystem. Projektet bedrivs i samarbete mellan Högskolan i Karlstad, Linköpings Universitet samt samarbetspartners från näringslivet.

Resultaten som rapporten redogör för är baserade på litteraturstudier samt strukturerade samtal med aktörer inom mjukvaruindustrin. Rapporten bygger på och utgår ifrån resultat från projektets första fas. Dessa resultat finns dokumenterade i Christiansson B. (1997). En sammanfattning av resultaten återfinns i kapitel 1.2.

Rapporten bör ha ett värde för personer som är intresserade av:

- objektorienterad systemutveckling
- komponentbaserade informationssystem
- standardsystem (standardiserade informationssystem)
- återanvändning av mjukvara

1.2 Objektorienterade komponentsynsätt

Ett komponentsynsätt kan antingen baseras på ett objektorienterat synsätt eller på de principer som ligger till grund för modulariserad mjukvara. I denna rapport kommer enbart de komponentsynsätt som innefattar ett objektorienterat synsätt att behandlas. Det som utmärker ett objektorienterat komponentsynsätt (OKS) är:

- Att under hela eller delar av ett informationssystems livscykel använda mjukvarukomponenter som uppfyller delar av det objektorienterade tankesättet. Då ett OKS används kan ett informationssystem bestå av köpta, återanvända eller egentillverkade mjukvarukomponenter.

- Att en mjukvarukomponent är en självständig, återanvändbar exekverbar enhet som erbjuder definierad funktionalitet via ett

specificerat kommunikationsgränssnitt. En mjukvarukomponent kan påverka/påverkas av andra mjukvarukomponenter. Varje mjukvarukomponent måste ha en specifikation, kan ha flera implementationer samt exekverbara (binära) former. I relation till objektorientering kan man säga att en mjukvarukomponent kan betraktas som en specialiserad form av objekt.

- Att mjukvarukomponenter kan kommunicera med varandra enligt någon av följande principer:
  - ⇒ Hård bindning, kommunikationsgränssnittet finns inprogrammerat i komponenten
  - ⇒ Mäklare förmedlare, kommunikationen förmedlas via en mäklande programvara

- Att ett OKS minst skall inkludera koncepten inkapsling och kommunikation med meddelanden från det objektorienterade tankesättet.

- Att ett OKS kan ta hänsyn till både den infologiska och/eller den datalogiska delen av ett informationssystem. Langefors (1995) definierar den infologiska delen som bestående av den information som systemet skall hantera och den datalogiska delen som bestående av IT-infrastrukturen för systemet.

- Att ett OKS skall vara tydligt avseende hur komponenter skall återanvändas och anpassas. För att uppnå återanvändning måste verksamheten ha en tydlig organisation för hantering av de resurser som skall återanvändas.

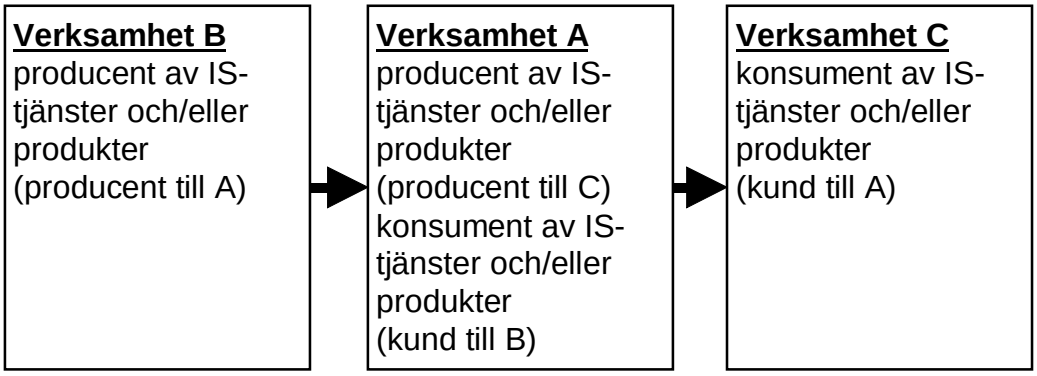
- Att ett OKS har en viss gångbarhet. Med gångbarhet menas synsättets spridning och användning i mjukvaruindustrin samt framtida livskraft. Dessutom menas de underliggande ställningstaganden som synsättet medför. Om synsättet t.ex. är förknippad med någon specifik mjukvarustandard kan detta leda till leverantörs-, maskin-, operativsystems- eller mjukvaruberoenden.

Dessa karaktäristika finns mer utförligt redogjorda och motiverade i Christiansson (1997).

1.3 Verksamhet ur ett informationsystemperspektiv

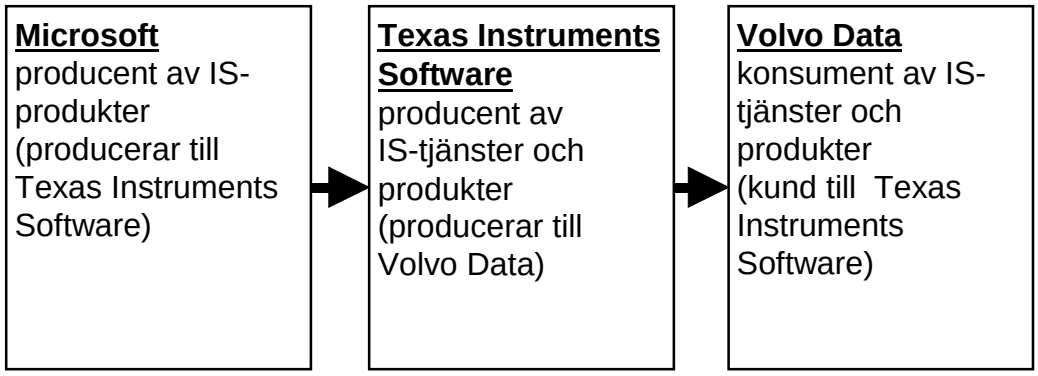
Om man betraktar en verksamhet utifrån ett informationsystemperspektiv kan verksamheten beskrivas utifrån två utgångspunkter: konsumerande

och/eller producerande. En konsumerande verksamhet använder informationssystem som stöd i sin löpande verksamhet. Med begreppet konsumerande menas att verksamheten köper produkter och/eller tjänster till sina informationssystem och inte själva bistår med detta. En producerande verksamhet har som sin löpande verksamhet att tillverka produkter och/eller sälja tjänster till konsumerande verksamheter. Utifrån dessa två utgångspunkter har sedan en given verksamhet ett visst förhållningssätt. En verksamhet kan både producera och konsumera informationssystemsprodukter och/eller tjänster eller kan renodlat vara konsumerande eller producerande. Detta kan dessutom betraktas i flera led. En verksamhet som är producent av informationssystemsprodukter och tjänster kan samtidigt vara konsument hos en annan producerande verksamhet, se figur 1.1.



Figur 1.1 En verksamhet kan vara både konsument och producent av informationssystemsprodukter och/eller tjänster.

Beroende på om en verksamhet är konsument och/eller producent kommer användandet av ett OKS att påverkas. Ett exempel på hur verksamheter kan förhålla sig till varandra illustreras i figur 1.2.

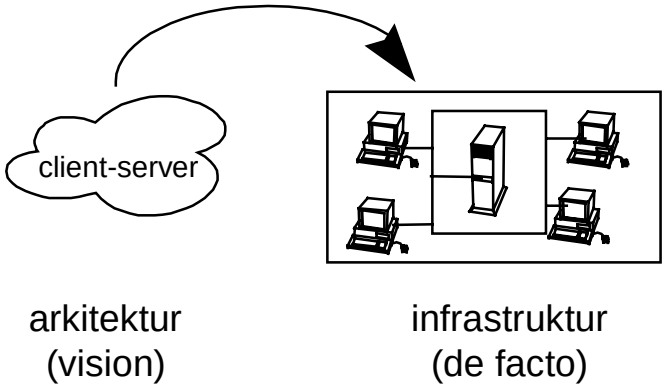


Figur 1.2 Texas Instruments Software är både konsument och producent av informationssystemsprodukter och tjänster.

2. Arkitekturer och infrastrukturer

Begreppet systemarkitektur används i flera olika sammanhang och betydelser. För att klargöra den utgångspunkt som ligger till grund i denna rapport följer nedan en kartläggning, positionering och definition av begreppens innebörd och användning.

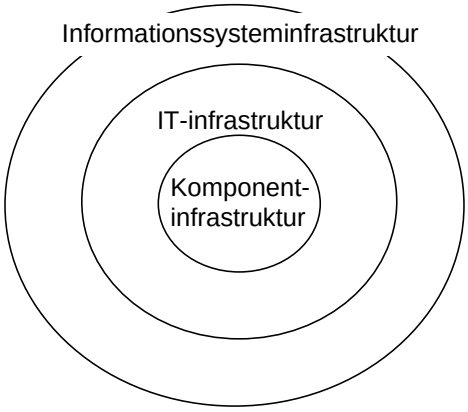
Med arkitektur generellt menar jag den vision som styr hur den artefakt (det fenomen) som arkitekturen avser får sin utformning. Antag exempelvis att en verksamhet har beslutat att använda en client-server-arkitektur i sina informationssystem. Detta skulle innebära att verksamheten strävar efter att skapa en client-server-infrastruktur i sina informationssystem när dessa införs, se figur 2.1. Om verksamheten de facto har en client-server infrastruktur i sina informationssystem beror på hur framgångsrik man förverkligat visionen (arkitekturen).



Figur 2.1 Figuren illustrerar förhållandet mellan en arkitektur och resultatet av att använda en arkitektur.

Szyperski (1997) påpekar att det är viktigt att särskilja på dels arkitekturen och dels den infrastruktur som arkitekturen medför. Dessutom bör infrastrukturens olika delar särskiljas. Då ett OKS används kan man identifiera minst tre olika indelningsnivåer (lager) i den informationssysteminfrastruktur en verksamhet har. Detta illustreras i figur 2.2 genom att den innersta cirkeln beskriver den infrastruktur som finns för att kunna använda mjukvarukomponenter (komponentinfrastruktur). Nästa cirkel beskriver den infrastruktur som finns för att kunna utnyttja de komponentbaserade system som används (IT-infrastruktur). Den yttersta cirkeln beskriver den infrastruktur som finns för att verksamheten skall kunna använda sina informationssystem (informationssysteminfrastruktur). De inneslutna cirklarna representerar

detaljeringsnivåer i en verksamhets totala informationssysteminfrastruktur.



Figur 2.2 Figuren illustrerar de olika nivåerna av infrastruktur som finns då en verksamhet använder ett OKS.

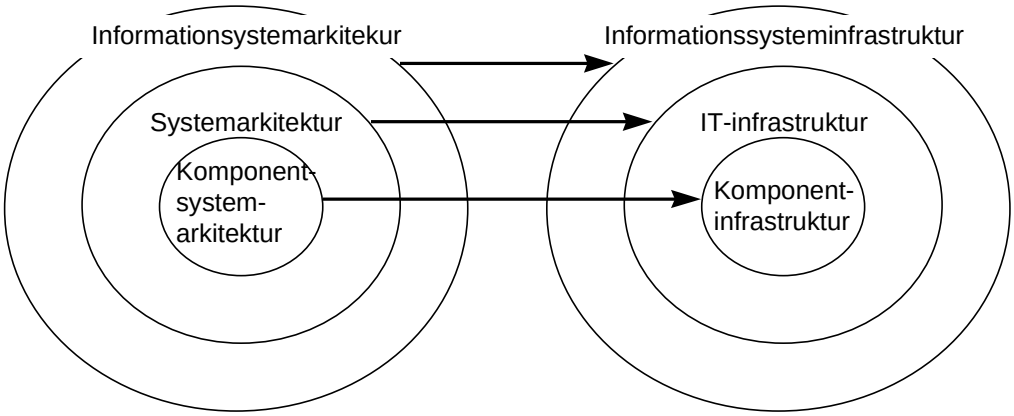
2.1 Begreppet informationsystemarkitektur

Informationssystemarkitektur är ett begrepp som enligt Axelsson (1998, p. 61) beskriver:

” • *fördelning av information och informationsbehandlande uppgifter, och därmed ansvar, mellan olika informationssystem samt mellan system och manuellt utförande*

- *samverkan (kommunikation) som sker mellan olika informationssystem samt mellan system och övrig verksamhet.*”

Det är en verksamhets informationssystemarkitektur som avgör hur verksamhetens totala informationssysteminfrastruktur ser ut där informationssysteminfrastrukturen avser verksamhetens de facto struktur av informationssystem. I figur 2.3 illustreras detta samband, de raka tjocka pilarna illustrerar arkitekturens påverkan på infrastrukturen, arkitekturen är på samma sätt som infrastrukturen uppdelad i olika detaljeringsnivåer. Om verksamheten har en ad-hoc informationssystemarkitektur kommer även informationssysteminfrastrukturen att vara ad-hoc mässig. Arkitekturen fungerar som en styrning av hur strukturen utformas. Med informationssysteminfrastruktur avses de samband som finns mellan interna informationssystem samt samband med informationssystem i omgivande verksamheter (ibid).



Figur 2.3 Figuren illustrerar de samband som finns mellan olika arkitekturer och deras motsvarande infrastrukturer.

2.1.1 Vikten av en informationssystemarkitektur

En verksamhets informationssystemarkitektur blir allt viktigare att definiera, utveckla och underhålla, några orsaker till detta kan vara:

- Flera samband mellan informationssystem då flera informationssystem tillkommer både internt och externt. Detta leder till mer komplexa informationssystem (t.ex. Sims, 1994).

- Spaghettisyndromet, informationssystem som är illa strukturerade är svårare att underhålla och utveckla (t.ex. Goldkuhl et al., 1993).

- Nya tekniska innovationer. Inom IT-området pågår en ständig nyutveckling avseende möjligheter och användning av IT. Dessa nya möjligheter leder till förändringar och utbyte av informationssystem.

- Det ökade strategiska värdet i informationssystem för en verksamhet.

- Då ett OKS används ställs nya krav på verksamhetens totala informationssystemarkitektur bl.a. genom kraven på komponenters kommunicerbarhet och, om standardkomponenter skall användas, en eventuell anpassning till deras funktionalitetsutbud.

2.2 Begreppet systemarkitektur

Enligt Szyperski (1997) är en systemarkitektur en holistisk bild över ett system. Arkitekturen definierar de övergripande oföränderliga parametrar som utmärker alla system byggda med denna arkitektur. En arkitektur är samtidigt basen för oberoendet mellan systemets delar samt basen för samverkan mellan delarna. Oberoendet är nödvändigt för att kunna byta delar. Samverkan är nödvändigt för att möjliggöra arkitekturens syfte: *"…the whole is more than the sum of its parts*" (ibid, p.274). Jacobson et. al. (1997b, p. 171) gör följande definition av begreppet: *"…the* ***software architecture*** *defines the static organization of software into subsystems interconnected through interfaces and defines at a significant level how nodes executing those software subsystems interact with each other*."

Systemarkitekturen kan betraktas som en vision som styr hur verksamhetens IT-infrastruktur utformas (se figur 2.3). En verksamhets IT-infrastruktur kan ses som en instansiering av verksamhetens systemarkitektur. IT-infrastrukturen är den hårdvara och mjukvara som verksamheten har. Systemarkitekturen styrs och påverkas genom de beslut som fattas avseende verksamhetens informationssystemarkitektur.

2.2.1 Vikten av en systemarkitektur

Asker (1994, p. 21) påpekar att *"En genomtänkt och hållbar arkitektur är kanske den enskilt viktigaste faktorn för en långsiktigt framgångsrik IT-produkt*". Vidare säger Szyperski (1997, p.273) *"Only where an overall architecture is defined and maintained do evolution and maintenance of components and systems find the firm foundation they require.*" Framtagandet av och utformandet av en verksamhets systemarkitektur är alltså mycket centralt. Arkitekturen styr den IT-infrastruktur som verksamheten har (se figur 2.3) och därigenom direkt påverkar hur stor nytta verksamheten har av sina datorbaserade informationssystem.

Det är dessutom mycket viktigt att systemarkitekturen skapar bra möjligheter till förvaltning och utveckling av verksamhetens datorbaserade informationssystem. Jacobson et. al. (1997b, p. 171) lyfter fram detta faktum: *"Software architecture is important in order to maintain the integrity of systems so that development and maintenance do not result in a patchwork of uncoordinated fixes. A well-articulated software architecture is also key to managing the complexity of software systems, allowing large organizations to work on parts in parallel*".

Det ställs ytterligare krav på en systemarkitektur i komponentbaserade informationssystem eftersom de består av köpta, återanvända eller

egentillverkade mjukvarukomponenter. För att kunna inhandla mjukvarukomponenter måste systemarkitekturen ge möjlighet att använda externt tillverkad mjukvara. Denna mjukvara skall dessutom kunna vara standardiserad. Detta leder till behovet av en komponentsystemarkitektur som definierar hur komponenter skall kunna installeras, exekveras och kommunicera med andra komponenter i existerande IT-infrastruktur (Jacobson et. al, 1997b; Szyperski, 1997; Clements & Northrop, 1996).

Skall verksamheten tillverka egna mjukvarukomponenter måste även hänsyn tas till utvecklingsverktyg som möjliggör tillverkning av komponenter som passar i verksamhetens IT-infrastruktur. För att uppnå återanvändning av mjukvarukomponenter behövs dessutom någon form av komponentlager där komponenterna kan förvaras efter anskaffning/utveckling och varifrån komponenterna kan distribueras internt eller externt. Även denna aspekt måste tas med i systemarkitekturen.

Det finns idag ett flertal datorbaserade verktyg för att skapa 'komponentlager'; exempelvis Texas Instruments Software AB har ett verktyg som heter Arranger som fungerar som ett komponentlager (Texas Instruments Incorporated & Microsoft, 1995b). Gemensamt för dessa komponentlager är att de understödjer en viss typ av mjukvarukomponenter. Detta faktum bör tas med i beräkningen då verksamhetens komponentsystemarkitektur definieras.

2.3 Begreppet komponentsystemarkitektur

Enligt Szyperski (1997) är komponentsystemarkitekturen den uppsättning beslut som styr hur mjukvarukomponenter kan installeras och exekveras. Dessutom avgör komponentsystemarkitekturen hur mjukvarukomponenter kommunicerar med varandra. Komponentsystemarkitekturen styrs och påverkas genom de beslut som fattas avseende verksamhetens systemarkitektur. Det är komponentsystemarkitekturen som avgör hur verksamhetens komponentinfrastruktur ser ut.

2.3.1 Vikten av en komponentsystemarkitektur

Komponentsystemarkitekturen styr hur en verksamhets komponentinfrastruktur ser ut och kommer därigenom att styra hur väl verksamheten kommer att kunna dra nytta av ett OKS. Då en komponentsystemarkitektur definieras är det centralt att verksamheten undersöker existerande komponentsynsätt och därefter väljer den som anses mest lämplig för given situation (se ex. Christiansson, 1997; Fagerström, 1995; Szyperski, 1997, Jacobson et. al, 1997b). Det är svårt att skapa en egen

komponentsystemarkitektur då möjligheten att köpa standardiserade komponenter reduceras. För att avgöra om en komponentsystemarkitektur är lämplig bör hänsyn tas till aspekter såsom arkitekturens gångbarhet, kommunikationsprincip, distributionsformer och återanvändnings-principer (Christiansson, 1997). Steel (1996b, p. 7) håller med om detta: *"A prerequisite to distributed component-based computing is a consistent infrastructure for component communication and interoperation."* Sims (1994, p. 223) påpekar dessutom vikten av kopplingar till informationssystem som inte har använt samma systemarkitektur. ”*Clearly what is needed is a single infrastructure that can be used by all application development projects, and that is capable of providing connections to other applications built without it…*”

Om verksamheten är IS-konsumerande måste ytterligare hänsyn tas vid definition av komponentsystemarkitektur. Man bör exempelvis inte låsa sig till en leverantör av komponenter eller komponentinfrastruktur och bli helt beroende av denna (Asker, 1994). Ett exempel på en kriterielista baserad på Asker (1994) för val av leverantör är att:

1. Leverantören använder rätt teknologi. Teknologin skall passa in i och kunna integreras med övriga system både på kort och lång sikt.
2. Leverantören bör ha möjlighet att vidareutveckla och underhålla de produkter som levereras.
3. Leverantören bör erbjuda bra relationer avseende stöd och assistans till levererade produkter.
4. Leverantören skall arbeta under rätta kommersiella villkor. Med rätta kommersiella villkor menas hur ‘öppen’ den systemarkitektur man väljer är. Finns det flera verksamheter som konkurrerar om konsumenterna eller är arkitekturen låst till en enskild leverantör?

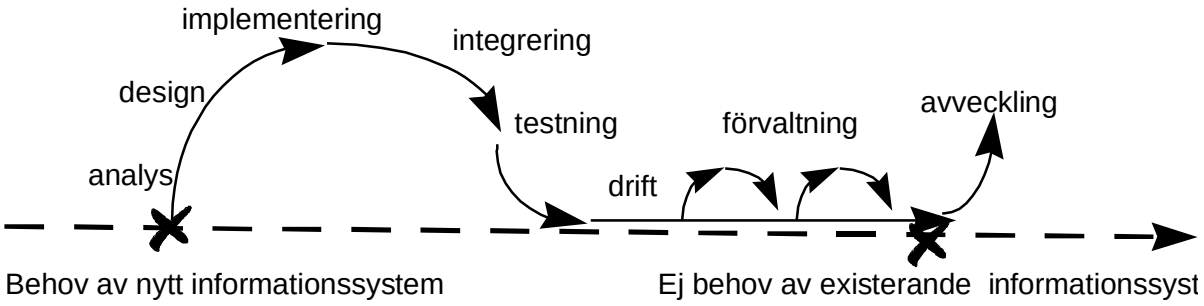
3. Ett informationssystems livscykel

Ett informationssystems livscykel kan betraktas ur flera perspektiv. Jag har genom att granska ett flertal olika beskrivningar av livscykler kommit fram till att den på något vis (sekventiellt, iterativt, [1]rekursivt eller parallellt) beskrivs genom följande faser/processer, se exempelvis (Christiansson, 1997; Goldberg & Rubin, 1995, Andersen, 1991, Fagerström, 1995).

Analys	Att nå förståelse för den verksamhet som informationssystemet skall stödja
Design	Att utveckla en detaljerad beskrivning av informationssystemet
Implementering	Att uttrycka designen på ett för datorn exekverbart sätt
Integrering	Att anpassa implementeringen till existerande miljö och system
Testning	Att identifiera och eliminera felaktigheter och icke önskvärda effekter i systemet och verifiera systemets funktionalitet
Drift	Att hålla det implementerade systemet i drift på en daglig basis
Förvaltning	Att följa upp driften och genomföra löpande korrigeringar i det implementerade systemet
Avveckling	Att avveckla det implementerade systemet (helt eller delvis) och tillvarata informationen i systemet

[1] rekursiv systemutveckling innebär att utvecklingen av ett informationssystem genomförs genom att systemet delas upp i delar som i sin tur delas upp i delar osv. Utvecklingen bedrivs sedan genom att de 'minsta' delarna färdigställs för att de större delarna skall kunna färdigställas osv. Reksursiv utveckling är intressant att beakta vid utveckling av komponentbaserade informationssystem och dess betydelse kommer att vidareutvecklas i ett senare skede i forskningsprojektet KOMPASS.

Ett informationssystem existerar för att lösa ett eller flera problem/behov i en verksamhet. Det är händelser och behov i denna verksamhet som initierar skapandet av ett informationssystem med dess livscykel. Det är under arbetet med att analysera/utveckla en verksamhet som behovet av ett informationssystem uppstår, se figur 3.1. Figuren illustrerar dessutom hur förvaltningsinsatser genereras utifrån ett informationssystems drift. Verksamheten har enbart 'nytta' av informationssystemet där livscykelns pilar ligger på verkamhetens tidsaxel. Under utvecklingsfasen ex. bidrar inte informationssystemet med någon 'nytta' till verksamheten.



Figur 3.1 Ett informationssystem livscykel sett i ett tidsperspektiv. Den streckade pilen representerar en verksamhets förlopp över tiden.

Det arbete som bedrivs för att utveckla verksamheten, samt den existerande verksamhetskulturen kommer att påverka ett informationssystems livscykel genom exempelvis organisationsformer och synsätt på IT. Frågor som om man på verksamhetsnivå tagit beslut om de systemarkitekturer som skall användas eller om varje informationssystem och dess relationer med andra system konstrueras separat för varje informationssystem. Andra relevanta frågor som ex. hur hanteras möjligheten till återanvändning, sker återanvändning på verksamhetsnivå eller får varje projekt där utveckling av informationssystem sker hantera detta själv. Svaren på dessa frågor kommer att ha stor påverkan på hur ett informationssystems livscykel ser ut och hur användningen av ett OKS påverkar den. Att använda ett OKS ger möjlighet att utnyttja standardiserade mjukvarukomponenter, detta innebär enligt min mening stora likheter med användande av standardsystem. Inom området standardsystem finns mycket forskning genomförd avseende anskaffning, anpassning samt förvaltning av standardsystem (Anveskog et al, 1984; Anveskog et al, 1983; Brandt et al, 1998; Nilsson, 1991; Andersson & Nilsson, 1998)

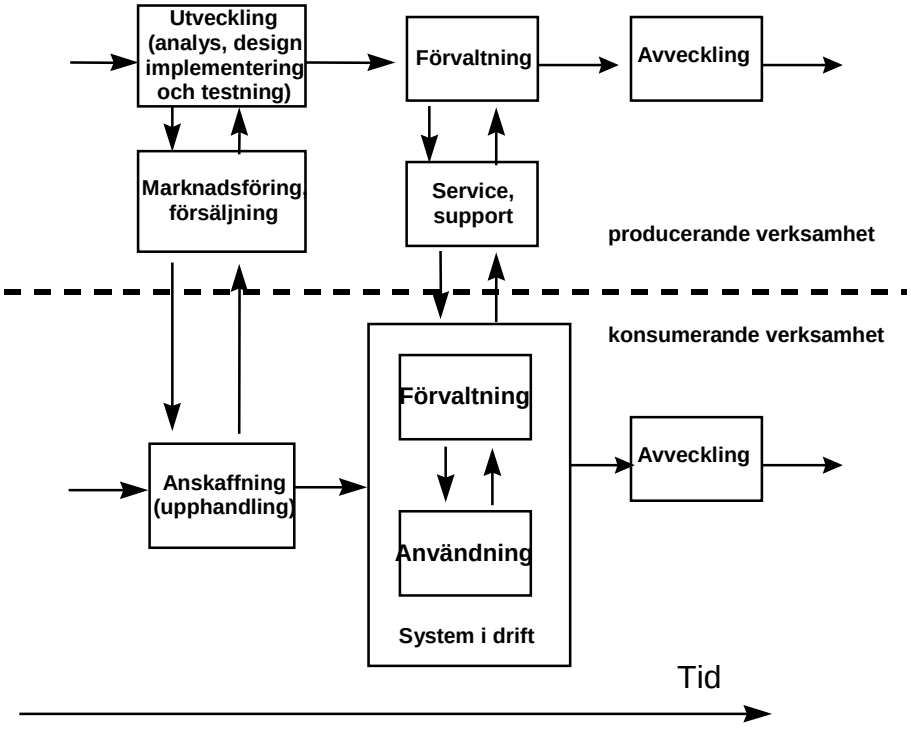
3.1 Livscykel hos ett standardsystem

Andersson och Nilsson (1996, p. 58) definierar ett standardsystem som: ”*... en mer eller mindre färdig programvara som kan tas i bruk direkt i ett företags verksamhet, till skillnad från egenutvecklade system som måste byggas upp från grunden.*” Användandet av ett OKS innebär möjligheten att anskaffa och använda standardkomponenter i informationssystem, detta medför att livscykeln hos ett informationssystem där ett komponentsynsätt används borde ha likheter med livscykeln hos ett standardsystem.

Nilsson 1991 menar att ett standardsystems livscykel innebär en växelverkan mellan en producerande samt en konsumerande verksamhet. Samma standardsystem har en livscykel i den producerande verksamheten samt en annan livscykel i den konsumerande verksamheten. Dessa två separata livscykler har dock beröringspunkter och beroendeförhållanden sinsemellan. Livscykeln för ett standardsystem i en producerande verksamhet kan jämställas med ett generellt informationssystems livscykel med tillägg av följande faser/processer:

Marknadsföring, försäljning	Ett standardsystem måste marknadsföras samt säljas till konsumerande verksamheter.
Service, support	En producerande verksamhet kan ansvara för delar av förvaltningen och driften av standardsystemet hos den konsumerande verksamheten.

För en konsumerande verksamhet ersätts analys-,design-,implementerings- samt testningsfasen med en anskaffningsfas. Där standardsystem väljs och upphandlas. Dessutom kan förvaltning- samt driftsfaserna hanteras av den verksamhet som har producerat standardsystemet. Dessa faser och deras växelverkan beskrivs i figur 3.2.



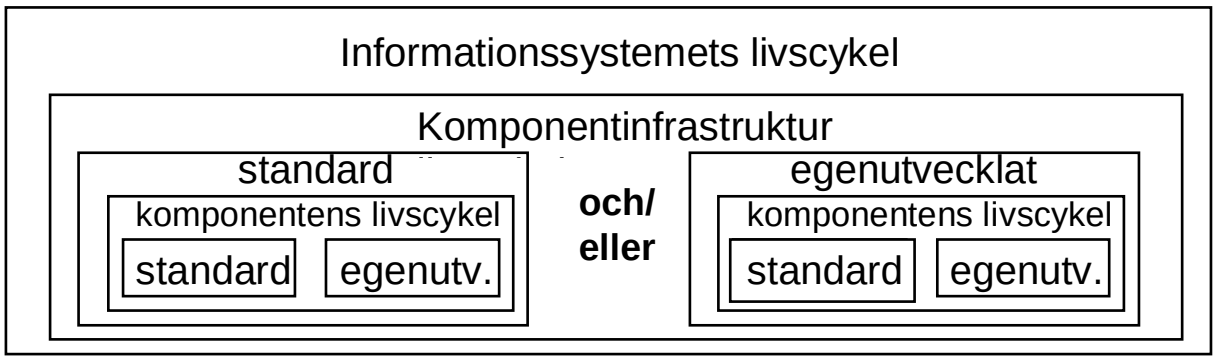
Figur 3.2 Ett standardsystems livscykel (fritt efter Nilsson 1973, p. 66).

Det finns ytterligare en viktig fas att nämna avseende livscykeln hos ett standardsystem. Om en verksamhet är producent av standardsystem måste även systemen distribueras till kunden. Distributionen kan eventuellt betraktas som ingående i marknadsföring och försäljning, men jag väljer ändå att särskilt belysa denna fas. Att distribuera ett standardsystem innebär att på lämpligt sätt förflytta nödvändig information samt mjukvara till den kund som anskaffat systemet. Denna förflyttning kan ske på olika sätt exempelvis via disketter, CD-rom eller direkt nedladdning via ett nätverk (Internet). Distrubitionsfasen inträder efter att producerande verksamhet implementerat systemet och efter konsumerande verksamhets anskaffningsfas.

3.2 Livscykel hos ett informationssystem då ett OKS används

Då ett OKS används går det inte att enbart betrakta det aktuella informationssystemets livscykel. Informationssystemet består av (se figur 3.3.) mjukvarukomponenter som var och en har sin egen livscykel som dessutom måste beaktas. Dessa komponenter kan antingen vara standardkomponenter eller egenutvecklade komponenter vilket även det får inverkan på den individuella komponentens livscykel enligt samma resonemang som fördes avseende standardsystem i tidigare avsnitt. Vidare har den komponentinfrastruktur som bl.a. ligger till grund för

komponenternas kommunicerbarhet en egen livscykel, även komponent-infrastrukturen kan vara egenutvecklad eller standardiserad.



Figur 3.3 De livscykler som existerar i ett informationssystem då ett komponentbaserat synsätt används.

Detta leder till en komplex struktur i ett informationssystem avseende förvaltning och nyutveckling där enskilda komponenters individuella livscykler måste kartläggas och följas. "*When COTS components, the maintenance activity must be expanded to include upgrading and replacing of components. As new versions of components are released by the software developers, and as superior components become available in the marketplace, system maintainers must evaluate the costs and benefits of integrating newer versions of the component into the system.*"[2] (Dean & Vigder, 1997, p. 5). Det faktum att livscyklerna är olika för en egenutvecklad och en standardkomponent ökar dessutom komplexiteten.

[2] COTS är en förkortning för Commercial Off-The-Shelf och refererar till en mjukvarukomponents ursprung, i detta fallet standardkomponent.

4. Systemutvecklingsprocessen

En verksamhet som producerar och/eller konsumerar informationssystemsprodukter och/eller tjänster bedriver en eller flera systemutvecklingsprocesser. Dessa processer kan vara mer eller mindre väl definierade och explicit uttryckta. Då en verksamhet använder ett OKS kommer detta att få effekter i verksamhetens systemutvecklingsprocesser.

4.1 Ett OKS påverkan på systemutvecklingsprocessen

Att använda ett komponentsynsätt påverkar systemutvecklingsprocessen. Grundtanken är att man skall gå från en fullständig konstruktion alternativt inköp av ett informationssystems alla delar till att anskaffa komponenter som sedan fogas samman till ett komplett informationssystem ”*Rather than buying into a complete and perhaps less-than-perfect packaged solution an organization can purchase only the required components, then combine them into a customized solution*.” (Chappell, 1997, p. 5). En jämförelse med bilindustrins utveckling är att från början byggde varje biltillverkare alla delar själv och satte sedan samman dem till den färdiga bilen. Idag köper en biltillverkare delar av andra leverantörer som sedan fogas samman till en färdig bil.

Att använda ett OKS skulle också kunna innebära att man minskar glappet som finns mellan analys- och designfasen, det som Stolterman (1991) identifierar som ‘det glömda språnget’. Dessutom finns det ett glapp mellan design- och implementeringsfasen då man gå från systemets infologiska del till dess datalogiska del (Langefors, 1995; Christiansson, 1996). Glappet skulle minska genom det faktum att en komponent alltid har en specifikation, en eller flera implementationer samt exekverbara former (Christiansson, 1997), detta innebär att samma komponent kan beskrivas och identifieras i alla faser i ett informationssystems livscykel genom dess tre former. Beroende på vilken fas i livscykeln informationssystemet befinner sig i används den form av komponenten som passar bäst, se tabell 4.1. För infologiska problem använder vi specifikationen och för datalogiska problem använder vi implementationen samt den exekverbara formen.

Livscykelns faser	Mjukvarukomponentens former
Analys	Specifikation
Design	Specifikation och implementation
Implementering	Implementation och binär form
Integrering	Implementation och binär form
Testning	Specifikation, implementation och binär form
Drift	Specifikation, implementation och binär form
Förvaltning	Specifikation, implementation och binär form
Avveckling	Specifikation, implementation och binär form

Tabell 4.1 Tabellen visar vilka av komponentens former som används under de olika faserna i livscykeln.

Om en verksamhet är producent av informationsystemprodukter och/eller tjänster kan systemutvecklingsprocessen dessutom innehålla de ytterligare faser som finns beskrivna i kapitel 3.1.

4.1.1 Analys

Analysfasens syfte är att identifiera och ge förståelse för de problem som informationssystemet skall lösa. Att använda ett komponentsynsätt under analysfasen kan innebära att analysera problemen utifrån ett komponentperspektiv. De komponenter som identifieras beskrivs på specifikationsformen. Utifrån denna specifikationsform går arbetet vidare med att anskaffa en komponent som uppfyller denna specifikation. Viktiga frågeställningar förutom komponentens specifikation är hur komponenterna hänger ihop och vilken funktionalitet de skall erbjuda. Detta medför att det sätt som en komponents specifikation beskrivs bör vara väl definierat i verksamheten. Om vi vill uppnå återanvändning måste det dessutom vara standardiserat. Beroende på graden av återanvändning kanske specifikationen skall vara standardiserad även över verksamhetsgränser, för att på så sätt kunna identifiera standardkomponenter. Ett standardiserat sätt att uttrycka komponenters specifikation ökar möjligheten att kontrollera om en befintlig komponent kan användas och

på så sätt uppnå återanvändning. Detta medför möjligheten att 'söka' i komponentlager efter en existerande komponent som har samma eller liknande specifikation som den man har behov av. Se ex. Christiansson (1997); Leach (1997); McClure (1997) för vidare resonemang om en komponents återanvändbarhet.

Ett OKS kan dessutom bidra med en möjlighet till ökad förståelse av en problemsituation. Genom kunskap om de tjänster som befintliga komponenter erbjuder kan detta vid analysen ge en ökad förståelse för problemsituationen och förmåga att definiera denna. Kommunikationen mellan analytiker och aktörer i problemsituationen kan förbättras genom att en 'statisk' miljö existerar i form av befintliga komponenter och deras tjänsteutbud. Detta innebär att man kan minska antalet okända faktorer i analysarbetet.

Dean & Vigder (1997) påpekar att då ett OKS används tenderar analysfasen att generera en mer abstrakt och kvalitativ kravspecifikation då existerande komponenter sätter de kvantitativa gränserna för vilka komponenter som kan användas. De använder sig av ett exempel där ett krav var att det skulle gå att skicka data mellan två system i en sekvens (batch) genom användande av Internet. Detta menar de är ett exempel på ett mera kvalitativt formulerat krav, om ett OKS inte använts skulle kravet innehållit, formatet på datan som skulle skickas samt protokollet som skulle använts för kommunikationen. Vidare påpekar Dean & Vigder (1997, p. 5) att användarmedverkan vid analys kan öka då ett OKS används: "*The major goal is to attempt to describe the real world functionality of the system as opposed to describing the technical aspects. This leads us to conclude that the needs of the end user take on a much more important role in COTS software based development ...*".

4.1.2 Design

Under designfasen skall en arkitektur och detaljer för lösningen av problemsituationen utvecklas. Om ett OKS används är det mycket centralt att under designfasen ha stor kunskap om den komponentinfrastruktur som används. För att uppnå de positiva effekter som förväntas med ett komponentsynsätt bör man ha skapat en given komponentsystem-arkitektur som spänner över alla de informationssystem som finns i verksamheten. Det kan krävas att denna komponentsystemarkitektur följer en marknadsstandard för att kunna samarbeta och utnyttja leverantörers och kunders informationssystem. Arbetet med att skapa en komponentinfrastruktur bör inte bedrivas för varje problemsituation separat utan bedrivas parallellt på en verksamhetsnivå. Paradoxalt nog så leder användandet av ett komponentsynsätt till att komponentinfra-

strukturen bör styras och utvecklas centralt. Detta står i direkt motsats till en av fördelarna med att använda ett komponentsynsätt nämligen att de komponenter som bygger upp ett informationssystem kan existera och förvaltas distribuerat (lokalt) för att minska verksamhetens centralisation.

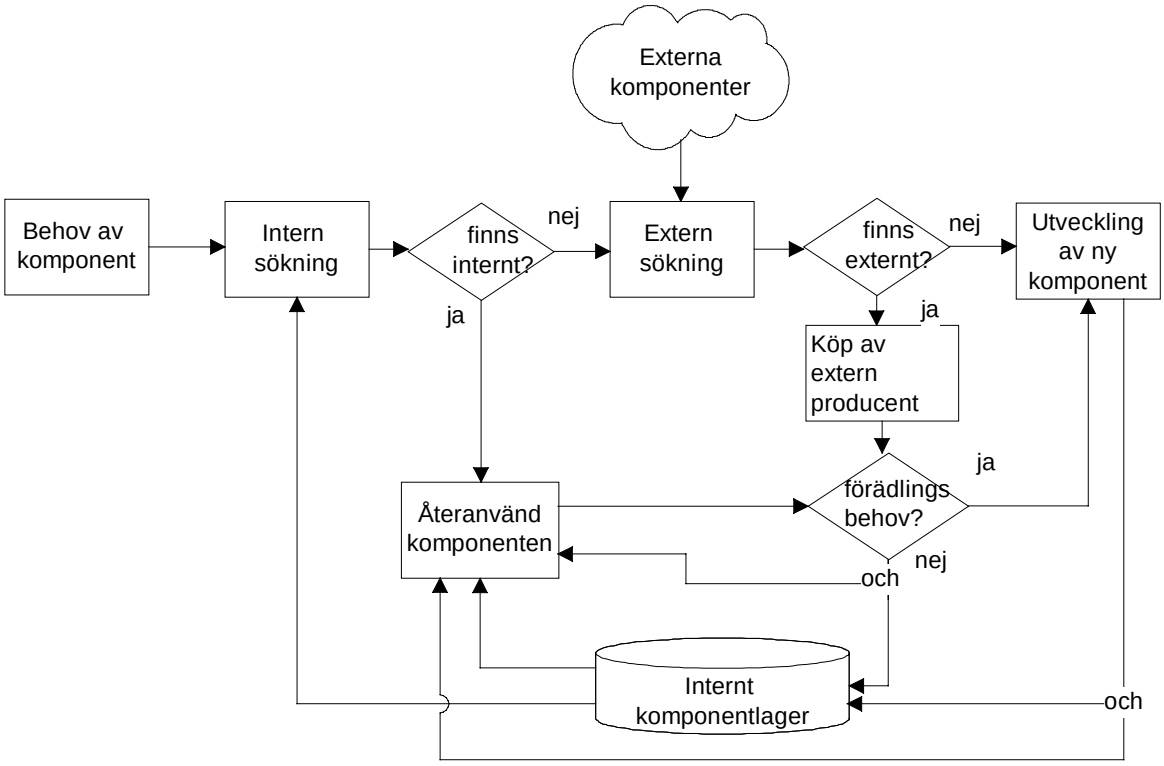
Då ett OKS används kan (om verksamheten enbart köper in existerande komponenter) designfasen ersätts med en anskaffningsfas där anskaffning av de komponenter som har identifierats under analysfasen sker. Här förstärks ytterligare vikten av ett standardiserat sätt att beskriva en komponents specifikation. Om en standard används, kan man i anskaffningsfasen leta efter en komponent som helt eller delvis uppfyller begärd specifikation. Dean & Vigder (1997, p. 5) uttrycker detta i följande citat: *"A survey of COTS components available in the marketplace must be performed, and criteria established for selecting the appropriate components. The Criteria range across a broad spectrum including run-time characteristics, documentation, vendor support, etc.*"

Om komponenten enbart delvis uppfyller specifikationen finns två alternativ, antingen anpassas komponenten efter egen lösning eller så anpassas lösningen efter komponentens specifikation. Det senare alternativet kan upplevas som radikalt men det finns förespråkare och experiment genomförda som pekar på fördelar med detta angreppssätt.

Det bedrivs ett forskningsprojekt i Kanadensiska försvarsmakten under namn ”COTS Software Integration State of the art” och deras forskningsresultat indikerar detta angreppssätt som det mest fruktbara. ”*Therefore, in order to realize the benefits of COTS software a procurement process must be in place that defines requirements according to what is available in the marketplace, and that is flexible enough to accept COTS solutions when they are proposed*” (Vigder et al., 1996, p. 13). En intressant iakttagelse till detta angreppssätt är att utifrån det sätt som Nurminen (1988) beskriver utvecklingen av informationssystemutvecklingen så är detta ett 'steg tillbaka' till samma nivå som vi befann oss på då datorn först började användas i informationssystem. Det är informationssystemet som avgör vad som är möjligt och människan anpassar sig till detta.

Anskaffningsfasen kan beskrivas genom ett flödesdiagram, se figur 4.1. Figuren beskriver en idealsituation där man kan tänka sig flera möjliga anpassningar för en specifik situation. Under analysfasen identifieras ett behov av en komponent, komponenten beskrivs och definieras på specifikationsnivå. Under anskaffningsfasen undersöks först det interna komponentlagret. Finns inte komponenten där så söker man externt hos komponentproducenter. Finns inte komponenten där så beställs en tillverkning av komponenten som antingen kan ske via en förädling av

existerande komponenter eller helt ny tillverkning. Under denna process är det komponentens specifikation (existerande eller önskvärd) som möjliggör anskaffningen.



Figur 4.1 Flödesdiagram över anskaffningsfasen av mjukvaru komponenter

Om verksamheten producerar IS-tjänster och/eller produkter kommer det under designfasen att ske en design av de komponenter som skall konstrueras eller förädlas. Denna process kan jämställas med en traditionell design av IS men dock måste vissa skillnader påpekas

- En komponent kommer att vara mindre komplex att konstruera då den erbjuder definierad funktionalitet via ett specificerat kommunikations-gränssnitt. Detta benämns vanligen med modularisering av mjukvara och är en erkänd teknik för att hantera komplexitet (Carrano, 1995).
- Vid design finns alltid en uttryckt specifikation av komponenten (Christiansson, 1997).
- Att designa komponenter kan bedrivas självständigt och parallellt med design av andra komponenter (Eklund, 1993).
- Designen skall genomföras på ett sådant sätt att återanvändning kan uppnås.
- Vid design av komponenter måste komponenten anpassas till given komponentsystemarkitektur.

4.1.3 Implementering

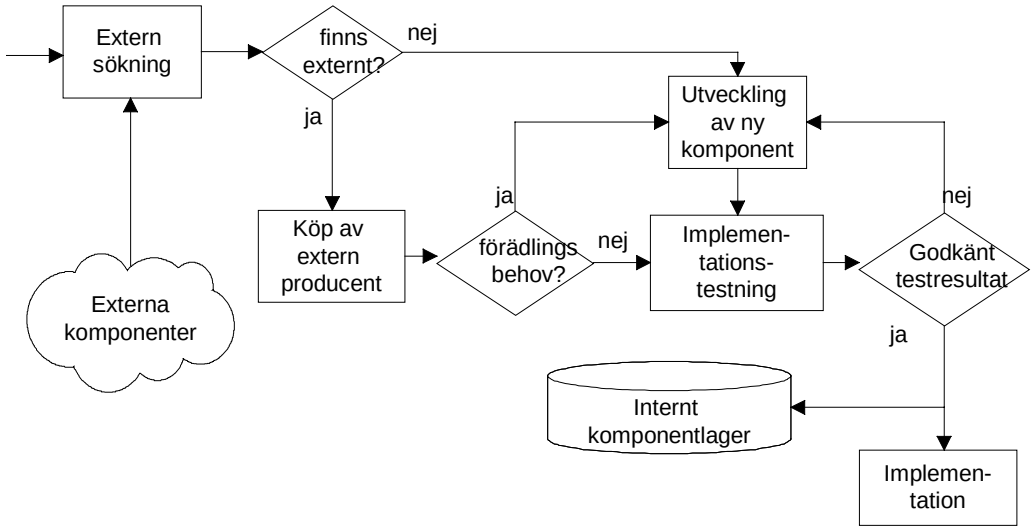
I en implementering uttrycks designen på ett för datorn exekverbart sätt. Detta görs i traditionell mening genom att mjukvaran som krävs konstrueras eller köps in av leverantör. Om ett OKS används innebär det att implementeringen övergår från ett konstruktionsarbete till ett monteringsarbete. Istället för att mjukvaran skapas i ett traditionellt programutvecklingsverktyg så monteras färdiga mjukvarukomponenter ihop till det system som beskrivits i designfasen. ”*…the notion of building system by writing code has been replaced with building a system by assembling and integrating existing software components …*” (Software Engineering Institute, 1997, p. 1) Detta medför enligt förespråkarna av ett OKS en stor reducering i arbetsbörda då programmerare inte längre måste konstruera hela system från början utan istället kan sätta ihop färdiga delar till hela system. Man måste dock tänka på att fortfarande kommer vissa komponenter att behöva konstrueras, vissa komponenter är verksamhetsspecifika eller unika för en viss situation. En del komponenter kommer att kräva en förädling för att passa in i en given lösning.

Det kommer att behövas en omfattande initial testfas. En ny komponent måste testas för att man skall vara säker på att komponenten utför det som specifikationen beskriver samt att den fungerar. Det kanske till och med behövs testning för att förstå vad komponenten gör och hur den skall implementeras. Detta har uppmärksammats i forskningsprojektet ”COTS Software Integration State of the art” där man identifierat ett flertal orsaker till varför en komponent måste testa för att vara möjlig att förstå:

”*Understanding the behavior of complex components is an extremely difficult task for a number of reasons:*
The documentation is incomplete or wrong…
The interface may be very complex…
There are bugs in software…” (Vigder et al., 1996, p.14)

Detta innebär att till skillnad från traditionell systemutveckling, där testfasen kommer efter implementering, måste det vid användande av ett OKS genomgås en testfas innan implementering. Det skall dock påpekas att då en komponent testats och visat sig uppfylla sin specifikation kan verksamheten återanvända den utan att genomföra denna initiala testning igen. En sammanfattning av implementeringsfasen skulle kunna vara att först genomgå en anskaffningsfas som kan leda till behovet av förädling eller konstruktion av komponenter. Därefter genomgås en initial testfas för att undersöka att komponenten utför det som specifikationen specificerar och att den dessutom har en tillräcklig tillförlitlighet. Därefter

kan komponenten implementeras samt placeras i verksamhetens komponentlager för att uppnå återanvändning. Detta förlopp beskrivs i figur 4.2.



Figur 4.2 Flödesdiagram som beskriver hur en implementeringsfas kan se ut då ett OKS används.

4.1.4 Integrering

Integrering innebär att implementeringen skall anpassas till existerande miljö och system. Detta blir en mycket central roll då ett OKS används. Det är under denna fas som systemets komponenter sammanfogas till det färdiga systemet. Arbetet i integreringsfasen förenklas av att det finns en väl definierad systemarkitektur som tas hänsyn till under analys, design och implementering. Enligt förespråkarna för användning av ett OKS skall detta arbete inte vara resurskrävande utan enbart bestå av en 'plug in' process. Detta givetvis under förutsättning att man har en grundläggande systemarkitektur som möjliggör detta. "*...software vendors can create their own innovative, unique, and value-added components that can plug into other vendors' components. The different software vendors don't even need to communicate with each other to exchange specifications or in any way coordinate the design and assembly of their specialized software components*" (Leake 1994, p. 2). Detta stämmer dock inte riktigt med andra erfarenheter som finns beskrivna inom ämnet. Inom forskningsprojektet "COTS Software Integration State of the art" påpekas att det oftast krävs någon form av sammanfogningsteknik som inte har med komponenterna att göra utan snarare med den systemarkitektur som finns. "*This model usually depends on the use of some "gluing technology", which may be*

unrelated to the components, to provide an interface between components." (Vigder et al., 1996, p. 3)

Software Engineering Institute (1997) påpekar vidare att integrationsfasen har blivit huvuddelen i realiseringen av IS. "*In contrast to traditional development, where system integration is often the tail end of an implementation effort, component integration is the centerpiece of the approach; thus implementation has given way to integration as the focus of system construction*" (ibid, p. 1).

En annan effekt på integreringsfasen är att vissa problem med komponenter inte kan upptäckas förrän integrationen görs (ibid). Detta innebär att även integrationsfasen kommer att till stor del bestå av testning, men här kommer testningen att ske utifrån komponenters påverkan av och på varandra. Att detta arbete skulle kunna genomföras med ett 'plug and play' tillvägagångssätt kan tyckas något oreflekterat.

4.1.5 Testning

Testfasen innebär att det integrerade informationssystemet genomgår en testning för att identifiera och eliminera de felaktigheter och icke önskvärda effekter som systemet har samt att verifiera och kontrollera systemets kvalitet. En testfas leder aldrig till att alla fel och brister elimineras utan innebär alltid att man får nöja sig med en viss nivå av tillförlitlighet. Denna nivå är starkt beroende av den problemsituation som systemet löser (Eklund, 1993).

Då ett OKS används kommer testfasen att påverkas främst avseende när i systemutvecklingsprocessen testen genomförs. Som tidigare nämnts kommer test att ske både i implementerings- samt integrationsfasen. Testfasen kan snarare kopplas ihop med en given komponent än ett givet informationssystem. Då en komponent anskaffas skall den testas innan implementering sker. Dessutom skall, då en komponent integreras med andra komponenter, ett integrationstest ske (Vigder et al., 1996; Software Engineering Institute, 1997).

Testning kommer att ske av delvis nya orsaker jämfört med traditionell systemutveckling där test genomförs för att identifiera och eliminera de felaktigheter och icke önskvärda effekter som ett IS har. Då ett OKS används kommer test även ske för att skapa förståelse för och kunskap om en given komponent. Då en komponent anskaffats måste man kontrollera att komponenten verkligen uppfyller det den anskaffats för och har en tillräcklig nivå av tillförlitlighet.

För att kunna genomföra ett test måste man veta vad som skall testas och vad systemet skall klara av att uppfylla. Detta innebär i traditionell systemutveckling att man använder sig av resultaten av analys- och designfaserna för att konstruera de testfall som krävs. Ett problem med att åstadkomma de testfall som krävs är att en översättning av analysresultaten till konkreta test i implementeringsfasen är nödvändig. Om man använder ett OKS förenklas detta genom att varje komponent har en specifikationsnivå, den beskriver exakt vad komponenten skall utföra och testfall kan konstrueras utifrån denna.

4.1.6 Drift

Ett informationssystem byggt på mjukvarukomponenter kommer att genomgå en driftsfas i sin livscykel. Förhoppningsvis är denna fas den största avseende kalendertid och resursåtgång. Det är i driftsfasen som nyttan av systemet fås. En driftsfas innehåller enligt Brandt et al (1998) bl.a. följande moment:

- Övervakning, kontrollera att systemet används som det är tänkt.
- Övervakning av driftsmiljön, mät och kontrollera prestanda och tillgänglighet
- Kompetensbevakning, se till att kompetens om systemets tekniska såväl som användarmässiga aspekter bibehålls i verksamheten.

Avseende driftsfasen i komponentbaserade informationssystem finns det inte mycket material att ta del av. Detta kan bero på att användandet av komponenter i informationssystem fortfarande är en relativt ny teknik och det inte finns så många komponentbaserade system i drift ännu. Det går dock att dra ett antal slutsatser utifrån ovanstående moment i en driftsfas.

Att övervaka ett komponentbaserat system kommer att vara komplicerat. Såväl informationssysteminfrastrukturen, komponentinfrastrukturen samt individuella komponenter bör övervakas avseende användning. Används standardkomponenter kan detta ytterligare komplicera övervakningen då användningen av en standardkomponent kan vara mer eller mindre väl specificerad.

Avseende övervakning av driftsmiljön kommer det faktum att komponenter kan existera distribuerat avsevärt försvåra övervakningen. I ett komponentbaserat informationssystem kan ingående komponenter existera på olika geografiska platser och ändå samarbeta via den komponentinfrastruktur som finns. Detta kan medföra att en verksamhet kan använda en komponent men inte ha några möjligheter att övervaka den avseende prestanda, tillförlitlighet osv.

4.1.7 Förvaltning

Förvaltning innebär att det implementerade och integrerade systemet skall underhållas och förändras allteftersom verksamhetens behov och krav förändras under den tid systemet befinner sig i drift. Det är mycket viktigt att iaktta att ett IS är under förändring då verksamheten är det.

En fördel med att använda ett OKS är möjligheten att byta ut eller förändra en eller flera komponenter för att svara mot de krav verksamheten ställer på förändring av ett IS. Detta förfaringssätt skulle betydligt förenkla förvaltningen av ett system. "*...changing an application to meet new needs is easier because a new component from the same vendor can be purchased and plugged into the application to extend its functionality. Similarly, a component from a different vendor with new capabilities can be substituted for an older component.*" (Leake, 1994, p. 3) I anslutning bör det dock nämnas att ett utbyte av en komponent måste förbigås av en utförlig initial testfas (både isolerat och integrerat) och en eventuell anpassning av komponentens kommunikationssätt för att få den att passa i en given systemarkitektur (Vigder et al., 1996; Software Engineering Institute, 1997).

Ytterligare en viktig aspekt att ta hänsyn till vid användandet av ett OKS är att, då en verksamhet köper komponenter från leverantörer kan ett leverantörsberoende uppstå. Leverantören står kanske för utvecklingen och eventuellt även underhållet av komponenter (Vigder et al., 1996). Detta ökar ytterligare komplexiteten i drifts- och förvaltningsfasen. Det kan dessutom skapa icke önskvärda beroendeförhållanden till en extern leverantör. "*The source code for these components is not available to the system developer nor does the system developer control the specification, release schedule and evolution of the components*" (Dean & Vigder, 1997, p. 1)

4.1.8 Avveckling

Då ett informationssystem befinner sig i en avvecklingsfas är tanken att den information som finns i systemet skall flyttas in i det nya system som bör vara redo att ta över efter det system som avvecklas (Andersen, 1993). Då informationssystemen är komponentbaserade kan avvecklingsfasen delas upp i att behandla avveckling av informationssystem, komponent-infrastruktur eller avveckling av enskild komponent. Att avveckla ett

informationssystem kommer att innebära att en viss sammansättning och användning av komponenter kommer att upphöra. Detta innebär inga stora effekter på varken komponentinfrastrukturen eller individuella komponenter (förutom att en komponent som är unik för ett visst system kan bli överflödig och avvecklas). Individuella komponenter kan avvecklas och ersättas med nya, även detta kommer att ge relativt små effekter. Om däremot komponentinfrastrukturen skall avvecklas kommer detta att ge stora effekter på verksamhetens informationssystem då systemen består av komponenter som fungerar enbart genom den komponentinfrastruktur som finns. Generellt kan sägas att komponentbaserade informations-system innebär ett nytt flexiblare sätt att avveckla system, då komponenter och komponentinfrastruktur inte behöver avvecklas utan kan fylla en ny roll i ett nytt informationssystem.

5. Slutsatser

Då ett OKS används påverkas systemutvecklingsprocessen i flera viktiga avseenden. Kortfattat kan den systemutvecklingsprocess som behövs då ett OKS används beskrivas enligt följande:

Analys	Att uppnå förståelse för problemen som informationssystemet skall lösa och identifiera dem utifrån ett komponent perspektiv.
Anskaffning	Att anskaffa de komponenter som man har identifierat under analysfasen.
Design	Att anskaffade komponenter eventuellt skall utvecklas eller att anskaffningen misslyckades och komponenten då designas innan implementering kan ske.
Implementeringstestning	Att en komponent måste testas för att man skall vara säker på att komponenten utför det som specifikationen beskriver samt hur och att den fungerar.
Implementering	Att mjukvarukomponenter monteras ihop till det system som beskrivits i anskaffnings- eller designfasen.
Integrering	Att implementeringen anpassas till existerande systemarkitektur och miljö.
Integrationstestning	Att testa komponenterna utifrån deras påverkan av och på varandra.
Drift	Att hålla det implementerade systemet i drift på en daglig basis.

Förvaltning	Att följa upp driften och genomföra löpande korrigeringar i det implementerade systemet.
Avveckling	Att avveckla det implementerade systemet (helt eller delvis) och tillvarata informationen i systemet.

Testning får en ny och utökad betydelse då ett OKS används till att användas för att uppnå förståelse och inte enbart för att kontrollera korrekthet som var fallet i traditionell systemutveckling. *För att uppnå återanvändning av mjukvarukomponenter behövs dessutom någon form av komponentlager* där komponenterna kan förvaras efter anskaffning/-utveckling och varifrån komponenterna kan distribueras internt eller externt
Om verksamheten är producent av informationssystemsprodukter och/eller tjänster kan dessutom följande faser tillkomma i systemutvecklingsprocessen:

Marknadsföring, försäljning och distribution	Mjukvarukomponenter måste marknadsföras samt säljas till konsumerande verksamheter. Därefter distribueras komponenterna till köparen.
Service, support	En producerande verksamhet kan ansvara för förvaltningen och driften av komponenter hos den konsumerande verksamheten.

Organisationen behöver en *ny typ av mjukvaruverktyg* både avseende de tidiga och sena faserna i systemutvecklingsprocessen. Det behövs mjukvara för att organisera och förvalta verksamhetens komponentlager samt distributionsmöjligheter för att uppnå återanvändning, inom den egna organisationen. Om verksamheten är producent av IS-tjänster och/eller produkter kanske man även måste kunna distribuera sina komponenter externt. De verktyg som används för implementering kommer också att påverkas då en verksamhet som använder ett OKS är mer beroende av integrations- och testningsverktyg snarare än mjukvaruutvecklingsverktyg.

Vikten av *ett formaliserat och standardiserat sätt att uttrycka en komponents specifikation på*. Detta är nödvändigt för att kunna uppnå återanvändning och möjliggöra en minskning av glappet som finns mellan den infologiska och den datalogiska delen vid utveckling av ett IS.

Med arkitektur menas generellt den vision som styr hur det fenomen som arkitekturen avser får sin utformning. Det är en verksamhets informationssystemarkitektur som avgör hur verksamhetens totala informationssysteminfrastruktur ser ut där informationssysteminfrastrukturen avser verksamhetens de-facto struktur över sina informationssystem. *Då ett OKS används kan man identifiera minst tre olika indelningsnivåer i den informationssysteminfrastruktur en verksamhet har:*

- informationssysteminfrastruktur, avser de samband som finns mellan interna informationssystem samt samband med informationssystem i omgivande verksamheter.
- IT-infrastruktur, den hårdvara och mjukvara som verksamheten har.
- komponentinfrastruktur, den mjukvara som används för att möjliggöra installation, exekvering samt kommunikation mellan komponenter.

För varje infrastruktur finns det en motsvarande arkitektur som styr hur infrastrukturen utformas:

- informationssystemarkitektur styr informationssysteminfrastruktur
- systemarkitektur styr IT-infrastruktur
- komponentsystemarkitektur styr komponentinfrastruktur

För att kunna använda standardkomponenter måste komponentinfrastrukturen ge möjlighet att använda externt tillverkad mjukvara. Denna mjukvara bör dessutom kunna vara standardiserad. Detta leder till *behovet av en komponentsystemarkitektur som definierar hur komponenter skall kunna installeras, exekveras och kommunicera med andra komponenter i existerande IT-infrastruktur.*

En verksamhet kan beroende på det sätt man väljer för att anskaffa och integrera komponenter bli mer eller mindre *beroende av leverantörer*. Ansvaret för de komponenter som organisationens IS består av flyttas från den egna verksamheten till externa leverantörer. Detta medför en minskad möjlighet att påverka vilka komponenter som tillverkas och när detta sker. Det blir mycket viktigt att arbeta med *öppna standarder* för att undvika alltför stora leverantörs-beroenden.

6. Omnämnanden

Följande personer har starkt bidragit till denna rapport med synpunkter, ideér och kritik: fil lic. Marie-Therese Christiansson (Högskolan i Karlstad), prof. Anders G. Nilsson (Högskolan i Karlstad; Institut V), Gösta Steneskog (Handelshögskolan i Stockholm, Institut V), prof. Bo Sundgren (Handelshögskolan i Stockholm; SCB), seniorkonsult Lars Wiktorin (ITplan) samt seniorkonsult Claes-Göran Lindström (ITplan).

7. Referenser

Andersson R. & Nilsson A. G. (1996). *Standardsystemmarknaden-en bransch i omvandling?* In: Lundeberg, M. & Sundgren, B. (Eds.1996). Att föra Verksamheten framåt, Studenlitteratur, Lund.

Anveskog L., Järperud J., Lundeberg M., Melin S. & Nilsson A. (1983) *Verksamhetsutveckling Att anpassa standardsystem*. Studentlitteratur, Lund.

Anveskog L., Nilsson A. & Nord I. (1984) *Verksamhetsutveckling Att välja standardsystem*. Studentlitteratur, Lund.

Asker B. (1994) *Att bygga produkter med köpt programvara.* Sveriges verkstadsindustrier, Stockholm.

Asker B. (1995) *Arkitektur och systembygge för programvara.* Sveriges verkstadsindustrier, Stockholm.

Asker B., Nilsson M., Söderström P. & Wiktorin L. (1996) *Återanvändning i verkligheten.* Studentlitteratur, Lund.

Axelsson K. (1994). *Informationssystemstrategiers betydelse vid förändring av informationssystem*. Institutionen för datavetenskap, Linköpings universitet.

Axelsson K. (1998). *Metodisk systemstrukturering att skapa samstämmighet mellan informationssystemarkitektur och verksamhet.* (doktorsavhandling), Linköpings Universitet.

Bassett P G. (1997) *Framing software reuse.* Prentice-Hall inc., UK, London.

Bergvall M. & Welander T. (1996) *Affärsmässig systemförvaltning.* Studentlitteratur, Lund.

Brandt P., Carlsson R. & Nilsson A. G. (1998). *Välja och Förvalta Standardsystem*. Studentlitteratur, Lund.

Brown A. W. & Wallnau K. C. (1996*). Engineering of Component-Based Systems.* In: Brown A. W (Eds.1996). Component-Based Software Engineering Selected Papers from the Software Engineering Institute. IEEE Computer Soceity Press, California, Los Alamitos.

Carrano F. M. (1995) *Data Abstraction and Problem Solving with C++.* Addison-Wesley Publishing Company, Menlo Park.

Casanve C. (1995) *Business-Object Architectures and Standards.* Data Access Coorporation, USA, Miami.

Chappell D. (1997). *The next Wave Component Software Enters The Mainstream.* Chappel & Associates, USA, Minneapolis.

Christiansson B. (1996) *Kunskapsprojektering -Effekter av en objektorienterad komponentstrategi vid utveckling och förvaltning av datorbaserade informationssystem.* Högskolan i Karlstad.

Christiansson B. (1997) *Objektorienterade mjukvarukomponenter i datorbaserade informationssystem. Arbetsrapport 97:7,* Högskolan i Karlstad.

Clements P. C. (1995). *From Subroutines to Subsystems: Component-Based Software Engineering.* In: Brown A. W (Eds.1996). Component-Based Software Engineering Selected Papers from the Software Engineering Institute. IEEE Computer Soceity Press, California, Los Alamitos.

Coad P. (1995) *Object Models Strategies, Patterns, & Applications.* Prentice-Hall, Inc., NJ, Englewood Cliffs.

Digre T. (1995) *Business Application Components.* Texas Instruments inc, USA, Plano.

Dean J. C. & Vigder M. R. (1997*) System Implementation Using Off-the-shelf Software.* National Research Council of Canada, Toronto.

Eklund S. (1993) *Programkonstruktion och projekthantering.* Studentlitteratur, Lund.

Fagerström J. (1995) *Objektorenterad analys och design -en andra generationens metod.* Studentlitteratur, Lund.

Garlan D. & Shaw M. (1994). *An Introduction to Software Arhitecture.* In: Ambriola V. & Tortora G. (Eds. 1993). Advances in Software Engineering and Knowledge Engineering, Volume I, World Scientific Publishing Company, USA, New Jersey.

Goldberg A. & Rubin K S. (1995*) Succeeding with Objects, Decision Frameworks for Project Management.* Addison-Wesley Publishing Company, California, Menlo Park.

Goldkuhl G. (1994). *Några problem vid datadriven strukturering av informationssystem*. Institutionen för datavetenskap, Linköpings universitet.

Graham I. (1994) *Migrating to Object Technology*. Addison-Wesley Publishing Company, Inc., California, Menlo Park.

Hertha W F., Bennett J E., Post F J. & Page I M. (1995) *An Architecture Framework: From Business Strategies to Implementation.* Canadian Imperial Bank of Commerce, Canada, Toronto.

Hung K. (1996) *A dynamic Business Object Architecture in an Iterative Life-cycle Environment for Information System Development.* (november), http://www.scism.sbu.ac.uk/cios/hungks/.

I-Kinetics, Inc. (1995). *Realizing a Virtual Application Warehouse using ComponentWare.* I-Kinetics, Inc., USA, Burlington.

ISO/IEC 12207 (1995) *Information technology - software life cycle processes.* ISO/IEC, Switzerland, Geneve.

Jacobson I. (1992) *Object-oriented Software Engineering a Use Case Driven Approach*. Addison Wesley Publishing Company, California, Menlo Park.

Jacobson I., Griss M. & Jonsson P. (1997a) *Making the Reuse Business Work.* In: Parrish E. A. (Ed.1997). Computer Innovative technology for computer professionals (October)IEEE Computer Press, California, Los Alamitos, vol 30, Number 10, 36-42.

Jacobson I., Griss M. & Jonsson P. (1997b). *Software Reuse Arcitecture, Process and Organization for Business Success.* Addison Wesley Longman, Inc., California, Menlo Park.

Jazayeri M. (1996) *Component Programming- A fresh look at software components.* Technical university of Vienna, Austria, Wien.

Langefors B. (1973) *Theoretical Analysis of Information Systems.* Studentlitteratur, Lund.

Langefors B. (1995) Essays on Infology. Department of Information Systems, Studentlitteratur, Lund.

Leach R., J. (1997) *Software reuse -methods, models and costs.* McGraw-Hill, USA, New York.

Leake G. (1994) *The Benefits of Component Software. How OLE Can Help Move Software from the Object-Oriented Age to the Component Era.* Microsoft Corporation, USA, Seattle.

Lundeberg M. & Sundgren B. (1996) *Att föra verksamheten framåt. Människor och informationssystem i samverkan.* Studentlitteratur, Lund.

Mattison R. & Sipolt M J. (1994) *The Object-Oriented Enterprice making Corporate Information Systems Work.* MCGraw Hill McGraw-Hill, USA, New York.

McClure C. (1997) *Software Reuse Techniques.* Prentice-Hall Inc.,New Jersey, Upper Saddle River.

Nilsson A. G. (1991). *Anskaffning av standardsystem för att utveckla verksamheter.* (doktorsavhandling) EFI, Handelshögskolan i Stockholm.

Nurminen M. L. (1988). *People or computers: Three Ways of Looking at Information Systems.* Studentlitteratur, Lund.

Pettersson K. & Goldkuhl G. (1994). *A Comparison between two strategies for Information Systems Architecture.* Dept. of Computer Science, Sweden, Linköpings University.

Pettersson K. (1994). *Några problem vid verksamhetsbaserad systemstrukturering.* Institutionen för datavetenskap, Linköpings universitet.

Ramackers G. & Clegg D. (1995*) Business Object Modelling, requirements and approach*. Oracle Corporation, UK.

Shelton R E. (1995a) *Business Objects Buying Objects Off the-Shelf*. Data Management Review, (november), http.//www.openeng.com.

Shelton R E. (1995b*) Business Objects Storage Strategies for Business Objects*. Data Management Review, (september), http://www.openeng.com.

Shelton R E. (1996a) *Business Objects Storage Strategier for Re-Use*. Data Management Review, (februari), http://www.openeng.com.

Shelton R E. (1996b) *Business Objects Workflow*. Data Management Review, (mars), http://www.openeng.com.

Sims O. (1994) *Business Objects Delivering Cooperative Objects for Client-Server*. Mcgraw-Hill Book Company, UK, London.

Software Engineering Institute (1997) *Component-Based Software development / COTS integration*. Carnegie Mellon University. USA, Pittsburgh.

Steel J. (1996a). *Component Technology Part I An IDC White Paper*. International Data Corporation, UK, London.

Steel J. (1996b). *Component Technology Part II An IDC White Paper*. International Data Corporation, UK, London.

Stolterman, E. (1991). *Designarbetets dolda Rationalitet - En studie av metodik och praktik inom systemutveckling.* (doktorsavhandling) Institutionen för informationsbehandling, Umeå universitet.

Sundgren B. (1996). *Vi vill ha ett användarvänligt och flexibelt system! - Att utforma system med delvis motstridiga och okända syften*. In: Lundeberg, M. & Sundgren, B. (Red.1996). Att föra Verksamheten framåt, Studenlitteratur, Lund.

Sutherland J. (1996) *The Object Technology architecture: Business Objects for Corporate Information Systems*. OOPSLA'95 Workshop on Business Object Design and Implementation, Springer-Verlag, Berlin.

Szyperski C. (1997). *Component Software Beyond Object-Oriented Programming*. Addison Wesley Longman, Inc., California, Menlo Park.

Texas instruments & Microsoft. (1995a*) Re-Engineering Application Development.* USA, Seattle.

Texas Instruments Incorporated & Microsoft. (1995b) *Repository Technology A Key Enabler for Business Application Development.* USA, Seattle.

Vaughn T. (1990) *Issues in Standards for Reusable Software Components*. (november, 1996) http://ruff.cs.umbc.edu.

Veryard R. (1997). *Software Component Quality*. (september), http://www.user.globalnet.co.uk/~rxv/CBDmain/DIPQUE.htm.

Veryard R. (1998a). *Component-Based Development & Open Distributed Processing.* (september), http://www.scipio.org.

Veryard R. (1998b). *Making CBD effective in your organization.* (september), http://www.scipio.org.

Vigder M. R, Gentleman W. M. & Dean J. (1996). *COTS Software Integration: State of the art.* National Research Council of Canada, Toronto.

Vigder M. R. & Dean J. C. (1998). *Managing Long-Lived COTS Based Systems.* National Research Council of Canada, Toronto.

Wang G., Klawitter D. & Smith S. (1997) *Component-Based Software Developmet: an Architectural View*. The Boeing Company, USA, Seattle.