EFFEKTIV IT

SYSTEMARVET

RAPPORT NR 17 – OKTOBER 1994

METODIK FÖR REVERSE ENGINEERING/REENGINEERING

– ett eftersatt område

Mats R Gustafsson
Lars-Åke Johansson

SVENSKA INSTITUTET FÖR SYSTEMUTVECKLING
SISU

SISU bedriver ett program för forskning och utveckling inom informationsteknologins tillämpningsområden – Effektiv IT. Grunden till programmet är en förstudie inom detta område som SISU genomfört på uppdrag av Näringsdepartementet och NUTEK. Forskningen koncentreras till områden som har stor ekonomisk relevans för svenskt näringsliv och förvaltning.

Målet med programmet är att svenskt näringsliv och förvaltning ska kunna använda resultaten för att:

- Effektivare styra och utveckla verksamheter
- Minska kostnaderna för informationsförsörjningen
- Bättre utnyttja befintliga informationssystem
- Använda bättre värderings- och kalkyleringsprinciper
- Minska ledtiderna vid införande av nya system
- Förbättra intern och extern kommunikation

Arbetet under första året drivs inom fem forskningsområden: *Systemutvecklingens ledtider och kvalitet*, *Systemarvet*, *Affärskommunikation*, *IT:s ekonomi och management* samt *Verktyg för verksamhetsutveckling*.

Innehåll

1 Inledning

Det råder brist på systematisk metodik och systematiserade arbetssätt som stöd för att utföra reverse engineering och reengineering av informationssystem. Verktyg och andra hjälpmedel för delar av arbetet finns, men helhetsperspektivet och hänsynstagandet till de olika verksamhetsmässiga situationer vari reengineeringarbetet bedrivs är otillräckligt behandlade och utarbetade.

I denna rapport presenterar vi därför några metoder, eller snarare delmetoder, som kan användas som komponenter i en metod som är litet mer heltäckande. Samt vidare ger synpunkter på vad som egentligen skulle krävas, liksom hur man kan tänka sig att gå vidare i metodsammanställning och metodutveckling.

Rapporten är alltså inte någon översikt eller bred sammanställning över befintliga metoder inom området. Istället har vi valt att begränsa oss och valt två delmetoder som vi presenterar litet närmare, främst i syfte att illustrera de olika frågor som kan vara aktuella inom reengineeringsområdet. Det gäller dels metodik för **migrering** av informationssystem och dels metodik för **återanvändning**.

2 Metoder och arbetssätt för reengineering

Framåtriktad systemutveckling (forward engineering) och reverse engineering hänger starkt ihop. Man gör reverse engineering för att ta reda på hur systemet ser ut nu, avgör vad systemet bör göra och på vilket sätt, för att sedan gå "forward" och realisera förändringarna på ett målinriktat sätt. Detta är vad reengineering av informationssystem innebär.

Mycket av den kunskap om metodik och beskrivningstekniker som finns idag kan användas både i reverse- och forward-riktning. Det är en potential som hittills har utnyttjats och fokuserats i alltför liten utsträckning.

2.1 Deltekniker för reverse engineering

Att arbeta med reverse engineering handlar i stor utsträckning om att förstå de system och de data man har. Den egentliga inriktningen på en reverse-engineeringsinsats blir i grunden att förstå dels vad programmen egentligen gör, dels vad de data man har egentligen betyder.

För att ta reda på vad ett program gör studerar man t ex vilka data det hanterar samt vilken behandling programmet utsätter data för. En mängd deltekniker kan användas för att nå dit man vill.

Det svåra är att utföra reverse engineering på ett strukturerat sätt, så att olika åtgärder är adekvata med tanke på vad man vill åstadkomma. Dessa åtgärder måste anpassas beroende på hur den affärsmässiga målbilden ytterst är för insatsen.

Viktiga typer av insatser i reverse engineering kan, på ett grovt plan, vara:

1. Infångning
2. Analys
3. Rapportering
4. Syntetisering
5. Lagring (i nya strukturer under nya begrepp, med nya namn, o s v)

På nästa plan kan man tala om ett antal deltekniker eller aktivitetstyper som ofta blir aktuella vid reverse engineering. T ex:

- dokumentation och diagramgenerering
- dekomponering (t ex "program slicing")
- omstrukturering (med bevarad semantik)
- identifiering av abstraktioner, återanvändningsmöjligheter
- återskapande av design

– integration med CASE-teknologi
– strukturella observationer, likheter med andra konstruktioner
– applicering av domänkunskap
– o s v.

2.2 Principiella steg vid reengineering

Man talar ofta om följande principiella steg när det gäller reverse engineering/ reengineering:

1. En **inventerande analys** genomförs.
2. En **migreringsplan** skapas, baserad både på kortsiktiga och långsiktiga verksamhetsmässiga och informationssystemmässiga behov. Positionering utförs – var är vi idag? Vilken är den nuvarande arkitekturen? Vilken är den arkitektur som vi vill uppnå? Hur ser nuvarande arkitektur ut i förhållande till planerad? Den planerade arkitekturen måste definieras väl.
3. **Transformering**. Hur skall man börja bygga upp den planerade arkitekturen? Vilka är delstegen för att nå dit? Genomförande av delstegen.
4. **Validering**. Kom man fram till den nya arkitekturen? Uppfyller den planerade strukturen de behov som finns?

2.3 Om krav på en migreringsmetod

Vi har i inledningen konstaterat att det är svårt att idag hitta sammanhängande metoder för reengineering av informationssystem.

I denna rapport redovisas några delmetoder som skulle kunna användas som komponenter i en heltäckande metod.

Vi vill dock gå ett steg vidare, just för att området är eftersatt, och även försöka ställa upp ett antal krav på vad en sådan metod bör ha för egenskaper.

Som utgångspunkt har vi tagit de krav som Michael Brodie [4] [5] ställer upp för vad han kallar en migreringsmetod.

En metodik för migrering skall enligt Brodie:

- möjliggöra migrering "på plats",
- skapa förutsättningar för kontinuerlig, säker och tillförlitlig åtkomst till kritiska funktioner och kritisk information och med prestanda som kan matcha verksamhetens arbetsbelastning,
- medföra så få ändringar som möjligt för att reducera komplexitet och risk i migreringen,

- medföra att man ändrar den gamla koden så litet som möjligt för att minimera risk,
- innebära att man åstadkommer så mycket flexibilitet som möjligt så att framtida utveckling underlättas,
- medföra att man minimerar den potentiellt negativa inverkan förändring kan ha för användare och applikationer och inte minst för systemets operation,
- utnyttja fördelarna med modern teknologi och moderna tillvägagångssätt i så stor utsträckning som möjligt.

Vi kan konstatera att dessa krav rör aspekter på informationssystemnivån. Det är informationssystemen som primärt skall migreras.

Vi vill dock aktualisera aspekten att en metodik också måste fokusera frågan om att sätta en informationssystemförändring, t ex en migrering, i ett större sammanhang – i ett verksamhetssammanhang.

Vi återkommer till detta i kapitel 4 efter en komprimerad genomgång av den migreringsmetodik som Brodie föreslår.

3 En metod för migrering av informationssystem

3.1 Inledning

Michael L. Brodie presenterar i [4] och [5] en strategi med tillhörande metoder för migrering av existerande system (systemarvet) till modernare målmiljö.

Den engelska termen för systemarvet är "legacy information systems". I definitionen av ett sådant system ligger, förutom att det är gammalt, att det är ett system som är kritiskt för verksamheten och har krav på sig att vara operationellt kontinuerligt över tiden. Detta är alltså viktigt att beakta i en migreringsmetod.

3.2 Migreringsstrategier

Brodie urskiljer 2 principiella strategier för migrering av informationssystem.

Migreringsstrategi 1 ("Cold Turkey")

Denna första strategi för migrering av informationssystem innebär att skriva om ett gammalt system från "scratch" och tillverka det nya systemet med användande av moderna mjukvarutekniker och till önskad hårdvaruplattform.

Detta är en högriskstrategi, anser Brodie, därför att:

- Man måste utlova ett bättre system, bättre i den meningen att det måste innehålla ny och utökad funktionalitet – det räcker inte att det bara är underhållsvänligare
- De verksamhetsmässiga förutsättningarna förändras hela tiden
- Det finns sällan bra specifikationer
- Det finns ofta odokumenterade beroenden
- Det är svårt att styra och administrera stora projekt
- Förseningar tolereras sällan
- Stora projekt tenderar att svälla ytterligare
- Gamla system kan vara alltför stora för att det skall vara möjligt att göra en smidig "cut-over" till det nya systemet.

Migreringsstrategi 2 ("Chicken Little")

Den andra migreringsstrategin går ut på att migrera ett gammalt system i små, inkrementella steg, tills det önskade system uppnåtts som utgör det långsiktiga målet. Systemet finns hela tiden "på plats" och är i drift kontinuerligt och utan avbrott.

Denna strategi innebär reducering av risk och är den strategi som Brodie förordar. Riskerna reduceras bl a genom att:

- Storleken på varje inkrement väljs så att risken minimeras för att det aktuella steget skall misslyckas.
- Om det trots allt misslyckas så finns det alltid en säker reträttposition.

3.3 Migreringsmetod

En migreringsmetod består av ett antal migreringssteg som tillsammans åstadkommer önskad migrering.

Varje steg berör en specifik aspekt av migreringen, t ex databas, applikation eller gränssnitt.

Bland de krav Brodie formulerar för en god migreringsmetod och som vi tidigare återgett betonas faktorer som riskreducering, inkrementalitet och små steg.

En analys förordas, där man iterativt:

- reducerar uppgiften
- dekomponerar funktionalitet
- utformar/designar det nya systemet.

Migreringen utförs också iterativt och avser:

- plattform
- applikation
- data
- gränssnitt.

3.4 Att reducera migreringsuppgiftens komplexitet och omfattning

Att förnya stora informationssystem kan vara en mycket komplex uppgift. En väsentlig fråga är hur man kan reducera komplexiteten och därmed arbetsomfattningen.

I de flesta system finns ofta delar som aldrig utförs. Dessa delar behöver man naturligtvis inte bygga om – det finns ingen anledning att migrera systemdelar som inte längre används. Å andra sidan gäller det naturligtvis att kunna urskilja vilka dessa delar är.

Det kan också vara så att mindre viktiga delar som körs mycket sällan kan avvaras. Detta kan också leda till att man kan hålla komplexitet och migreringskostnader under kontroll.

Vissa gamla system fungerar trots allt bra även om de har dålig struktur. De kan i första läget kapslas in om man inte behöver förändra dem stort. De kan sedan bytas ut efterhand, då man skapar helt nya systemdelar under en längre tidsperiod.

Eliminera alltså så många funktioner och applikationer som möjligt. Ta bara med det absolut nödvändigaste och mest kritiska. Brodies rekommendation på denna punkt är mycket stark. Reducera alltså bort

- kod som inte längre används
- duplicerad kod
- kod som inte är kritisk, samt
- kapsla in koddelar.

3.5 Informationssystemarkitekturer

Brodie påpekar att följande funktioner eller grupper av funktioner återfinns i alla informationssystem:

1. Gränssnitt
2. Applikation
3. Databashantering.

I ett välstrukturerat system är de delar som utför dessa funktioner distinkta och har väldefinierade gränssnitt.

Beroende på hur väl strukturerat ett system är i denna mening kan det mer eller mindre lätt brytas ner i delar. Ett system är

- dekomponerbart om det är välstrukturerat
- semi-dekomponerbart
- ej dekomponerbart om det är ostrukturerat.

3.6 "Gateways"

En gateway är en programmodul som läggs in mellan olika programvarukomponenter och utgör ett mellanled, en "brygga" mellan modulerna.

En "gateway" kan ha olika roller:

- att avskärma komponenter från förändringar som görs i andra komponenter
- att översätta anrop och data mellan komponenter
- att koordinera komponenterna, t ex att uppdatera (båda) komponenterna på ett konsistent sätt.

Placeringen av en "gateway" är kritisk och påverkar komplexiteten hos:

- migreringsarkitekturen
- gateway själv
- migreringsmetoden.

Det finns olika typer av gateways. En databas-gateway kapslar in databashanteringen och databasen sedda från applikationsmodulerna. En applikations-gateway kapslar in applikationerna och nedåt från gränssnittet. En informationssystem-gateway kapslar in hela informationssystemet i det fall vi har ett icke dekomponerbart system.

3.7 Steg i en migreringsmetod

Med bakgrund i ovanstående synsätt, begrepp och faktorer, anger Brodie nu en uppsättning metoder för migrering av ett systemarv. Varje metod består av 5 grundläggande steg:

- Iterativ migrering av datormiljö/plattform
- Iterativ migrering av arvsapplikationer
- Iterativ migrering av data
- Iterativ migrering av användar- och systemgränssnitt
- Iterativ övergång från arvs- till målkomponenter.

Valet av specifik migreringsmetod avgörs av om det system som skall migreras kan betraktas som dekomponerbart, semi-dekomponerbart eller icke dekomponerbart.

4 Utökade krav på en migreringsmetod

Vi kunde konstatera i avsnitt 2.3, att de krav på en migreringsmetod som Brodie har satt upp, och som han haft som utgångspunkt vid utvecklandet av den metodik som kort presenterats i kapitel 3, rör aspekter på informationssystemnivån. Det är informationssystemen som primärt skall migreras.

Men en metodik måste också fokusera frågan om att sätta en informationssystemförändring, t ex en migrering, i ett större sammanhang – i ett verksamhetssammanhang.

Man kan på detta sätt definiera migrering så att den rör en successiv process, som måste genomföras i ett verksamhetssammanhang och i anslutning till ett affärsutvecklingsprogram för en verksamhet.

Vi utvidgar därför Brodies krav med följande punkter:

Identifiering av behov i verksamhet och affärsutveckling

Metoden måste kunna ange ett sätt för att bedriva en verksamhetsanalys där man identifierar verksamhetens mål, inklusive affärsmålen. Det är fundamentalt att all reengineering och migrering av informationssystem måste göras mot bakgrund av vad man egentligen skall göra i verksamheten. Detta har sin orsak i att en reengineeringsinsats kan göras på en mängd olika sätt och i hög grad vara betingad av vad man vill åstadkomma inte minst verksamhetsmässigt.

Eventuellt kan man peka ut en metod som finns i någon forward engineering-metodik och ange hur denna skall användas och kopplas in i ett reengineeringssammanhang.

Identifiering av förutsättningar, effekter och val av reengineeringsstrategi

Om man vet vilka målen för en verksamhet är och vilka uppgifter som måste finnas i verksamheten, har man skapat förutsättningar för att hitta en strategi enligt vilken man kan identifiera konkreta åtgärder för reengineering.

Utformningen av dessa åtgärder måste också definieras på basis av vilka förutsättningar som finns i verksamheten. Ett exempel är den nivå från vilken man måste lyfta standarden på ett visst system. "Från vad och till vad"?

En metod måste kunna ge vägledning om hur man gör detta.

Identifiering av önskad arkitektur på det migrerade systemet

När man genomför reengineering av informationssystem måste man också ha klart för sig vilken typ av systemarkitektur man siktar mot. På vilket sätt skall systemet struktureras? Vill man bygga på vissa standards? Är man ute efter någon form av client/serverlösning, skall databasdelarna respektive gränssnittsdelarna vara tydligt separerade, skall systemet kunna köras på olika maskinplattformar, vill man åstadkomma någon typ av meddelandesamverkan mellan olika systemdelar, skall man ha någon typ av EDI-gränssnitt i sina system?

Man kan också använda sig av vissa arkitekturformer för att genomföra själva migreringsarbetet. Exempelvis kan en modern CORBA-baserad client/serverlösning vara målet för arkitekturen (CORBA – Common Object Request Broker Architecture) [17]. I denna arkitektur lägger man sedan vissa gamla systemdelar som har ett gränssnitt till CORBA och som fungerar så att CORBA hämtar objekt från de gamla systemen så att de inifrån CORBA-miljön ser ut som en server. Men att man i gränssnittet sörjer för en översättning av anrop av tjänster till något som det gamla systemet känner igen.

Efterhand kan man sedan bygga om det gamla systemet utan att det stör den övriga miljöns sätt att fungera.

Det är viktigt att en metod ställer frågor och ger handledning i beslut kring dessa aspekter.

Avgränsning av delinsatser

Skall man göra en större migreringsinsats är det väsentligt att ett sådant arbete görs i delar. Detta arbetssätt bör fokuseras av en välutvecklad metod. Det finns många exempel på att alltför stora projekt startats, vilket resulterat i att systemet aldrig kunnat realiseras. Uppgifterna och problemen blir allt för stora. Man klarar inte av att genomföra och styra uppgifterna i projektet.

Det är viktigt att en metod fokuserar på att urskilja mindre system och systemdelar som man kan angripa var för sig och att varje del skall ha ett mål vad avser hur det skall det se ut när det är färdigt. Vidare skall man också av varje del kunna uppfatta effekter relevanta för verksamheten. Varje delsteg måste kunna motiveras genom sina effekter. Samband skall finnas till efterföljande delinsatser.

Uppdelningen i delinsatser kan med fördel grundas i olika delar av verksamheten. Till exempel: ett nytt databasgränssnitt till en viss typ av databashanterare för ett visst system, en generaliserad moduls användning i andra system, ett nytt EDI-gränssnitt från existerande systemkomponent, ett frifrågestöd för en viss databas, o s v.

Identifiering av strategi för att genomföra migreringen

Genomförandet av de olika delinsatserna måste också sättas samman så att de ligger i lämplig ordning och att man använder principer som stämmer överens med de verksamhetsmässiga mål som finns ultimärt och med den arkitektur man siktar mot.

Strategin, och hur den hänger ihop med målen, måste också presenteras och motiveras på olika sätt.

Ansats för hantering på managementnivå

Det har visat sig att reengineeringsinsatser är svåra att klara ur management-synvinkel. Man betraktar en reengineeringsinsats som en teknisk fråga. Management anser sig inte behöva engageras i denna typ av insatser.

I själva verket är det så att det är väsentligt att det finns åtminstone en person på toppledningsnivå som känner till och kan försvara varför en viss form av reengineeringsarbete bedrivs.

Detta av flera skäl. Dels inåt i ledningsgruppen som sådan, dels gentemot de personer som ingår i och medverkar i reengineeringsarbetet. Dessa personer måste motiveras, entusiasmeras och uppmärksammas på hur de olika insatserna är viktiga och vad som särskilt bör prioriteras. Olika insatser tenderar annars att bli okoncisa och med oengagerade medverkande.

Beskrivningsformer

Metoden måste också kunna peka ut beskrivningssätt för verksamhets-funktioner, -processer, -begrepp och -regler. Dessa skall kunna användas i olika metodsteg. De skall dels kunna användas för att beskriva resultatet av den verksamhetsanalys som skall ligga till grund för definitionen av en migreringsinsats. De skall dels kunna användas för att beskriva resultaten från en reverse engineeringsinsats i form av begreppsbeskrivningar och regelbeskrivningar från vad som, ur verksamhetssynvinkel, egenligen görs i en uppsättning program.

5 Metodik för återanvändning

5.1 Inledning

Den nya plattform som man vill skapa vid en reengineeringsinsats kan innehålla inslag av återanvändbara komponenter. Man vill alltså göra en investering för framtiden och skapa en plattform som skall underlätta framtida utveckling, vidareutveckling och förändring av system [15].

Det finns 2 steg i detta:

- att skapa komponenter som är återanvändbara och därmed lägga grunden för att återanvända (*design for reuse*),
- att löpande använda sig av återanvändbara komponenter när man omformar och vidareutvecklar system (*design by reuse*).

Vilken typ av komponenter är det som man vill kunna återanvända? Det kan vara komponenter som ligger på olika nivåer, som t ex system, procedurer, paket, datatypdefinitioner, varabeldefinitioner, programbeskrivningar, program och programdelar. Men också systemarkitekturer, verksamhetsmodeller av olika slag såsom funktionsmodeller, begreppsmodeller och processmodeller.

Några motiv till varför man önskar återanvända modeller av olika slag kan vara att:

- modellerna representerar eget verksamhetskunnande,
- modellerna representerar andras verksamhetskunnande (t ex vara "best practice"-modeller),
- man vill hitta skillnader mellan egna och andras modeller, dels för att urskilja det unika med den egna verksamheten, dels för att ta tillvara andras kunskap,
- man vill åstadkomma viss kvalitet på kortare tid,
- man vill komma snabbare till färdigt system,
- man vill introducera en viss begreppsapparat.

För att exemplifiera vad denna problematik rör sig om har vi i detta avsnitt valt att litet mer detaljerat beskriva de olika frågeställningar som kan vara aktuella när det gäller att skapa komponenter lämpliga för återanvändning och sedan återanvända dem, genom att presentera en speciell ansats.

Det finns flera ansatser när det gäller återanvändning eller återbruk ("reuse") inom informationssystemområdet. Vi har dock inte haft någon möjlighet att göra en uttömmande analys och genomgång av alla dessa, utan istället valt att presentera **ett** synsätt.

Vad vi vill illustrera är **vidden av problematiken**, de svårigheter och kanske fällor som man kan hamna i när man utformar och utnyttjar återanvändbara komponenter.

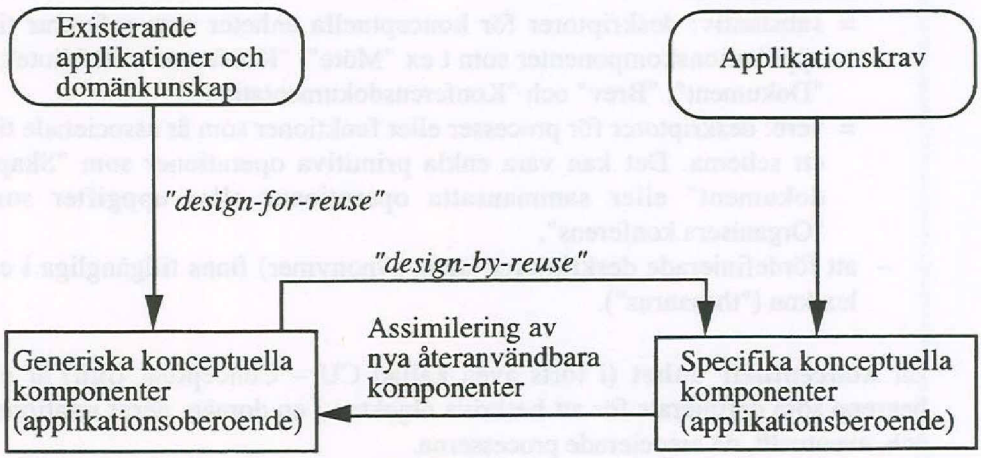
5.2 En återanvändningsbaserad applikationsutvecklingsprocess

Den ansats som vi beskriver är utvecklad vid Politecnico di Milano (PdM), Italien. Man har där arbetat med återanvändningsmetodik ganska länge och har publicerat ett stort antal artiklar i ämnet, se t ex [2], [6] och [10]. Den rapport som följande framställning bygger på och varifrån illustrationsexemplet har hämtats, är [2]. Exemplet har justerats lätt och försvenskats av oss.

5.2.1 Huvudsaklig inriktning

De typer av modeller som man har studerat vid PdM är i första hand s k ER (Entity-Relationship)-modeller, d v s modeller av informations-/ datamodell-typ, i vilka verksamhetens/databasens objekt (entiteter), relationer och attribut beskrivs. Man har också i någon mån behandlat processmodeller av dataflödes-diagramtyp. PdM, liksom för övrigt SISU, ingår också som part i det europeiska samarbetsprojektet F3 ("From Fuzzy to Formal") i vilket aktualiserats problem som återanvändning av målmodeller, aktörsmodeller och kravmodeller för informationssystem.

Figur 1 ger en översikt över processen.



Figur 1: Översikt över återanvändningsprocessen

5.2.2 Faser i återanvändningsansatsen (*"design for reuse"*)

Metoden anger följande fem faser vid utformning av återanvändbara komponenter:

1. Urval av kandidatscheman
2. Klassificering av kandidatscheman
3. Urval och klassificering av kandidatkomponenter
4. Utformning av återanvändbara komponenter
5. Assimilering av återanvändbara komponenter.

Fas 1: Urval av kandidatscheman

Denna fas avser processen att välja ut scheman som kan antas vara relevanta för återanvändningssyften från existerande applikationsscheman.

Grundproblemet här är alltså att bland de scheman som existerar kunna identifiera dem som bäst kan tänkas matcha krav och egenskaper hos framtida applikationer som kan komma att konstrueras med hjälp av återanvändning. Detta är naturligtvis ingen lätt bedömningsuppgift.

För att kunna göra detta måste schemana beskrivas med hjälp av speciella **schemadeskriptorer**. För dessa gäller:

- att de beskriver ett schema,
- att de uteslutande väljs bland olika "konceptuella enheter" som förekommer i ett schema,
- att de kan vara
  - = substantiv: deskriptorer för konceptuella enheter som refererar till applikationskomponenter som t ex "Möte", "Konferens", "Bibliotek", "Dokument", "Brev" och "Konferensdokumentation".
  - = verb: deskriptorer för processer eller funktioner som är associerade till ett schema. Det kan vara enkla primitiva operationer som "Skapa dokument" eller sammansatta operationer eller uppgifter som "Organisera konferens".
- att fördefinierade deskriptorer (samt synonymer) finns tillgängliga i ett lexikon ("thesaurus").

En **konceptuell enhet** (i forts även kallad CU – Conceptual Unit) är ett begrepp som definierats för att beskriva objekten i en domän, deras relationer och, eventuellt, de associerade processerna.

Ett **konceptuellt schema** är en mängd relaterade konceptuella enheter uttryckta i lämplig konceptuell modell (t ex i en Entity-Relationship-modell eller i en objektorienterad modell).

Hur bär man sig då åt för att välja sina deskriptorer? Varje konceptuell enhet i ett schema tilldelas först en **vikt.** Vikten för en konceptuell enhet beräknas som summan av:

- antalet strukturella egenskaper (SPs – "Structural Properties")
- antalet beteendemässiga egenskaper (BPs – "Behavioral Properties")
- antalet intilliggande CUs ("Conceptual Units") d v s konceptuella enheter som har direkt relation till aktuell enhet.

För de konceptuella enheter som ingår i en s k is-a, eller generaliserings-hierarki, beaktas vidare:

- hierarkisk nivå (HL – "Hierarchical Level") för enheten i hierarkin: 1 för roten, 2 o s v för efterföljande nivåer
- antalet enheter som är direkta efterföljare (subobjekt) till den aktuella enheten
- antalet enheter som ligger på samma specialiseringsnivå som enheten.

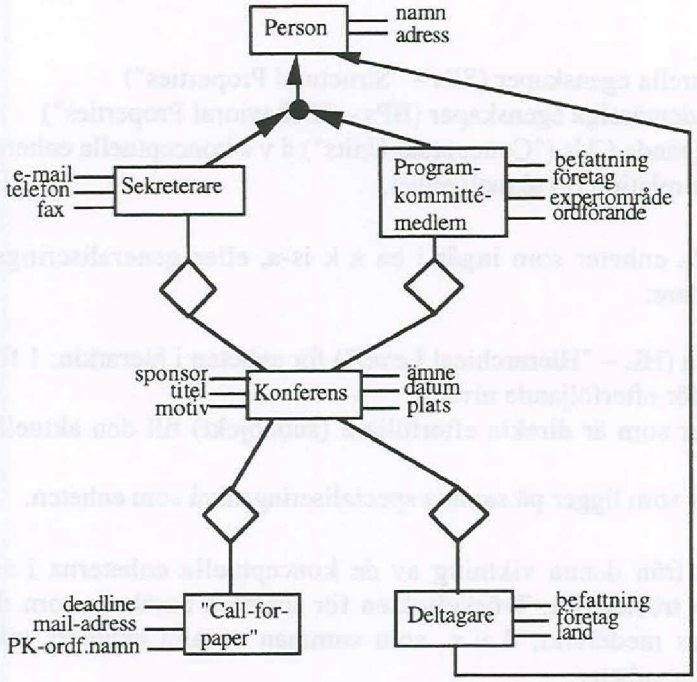
Med utgångspunkt från denna viktning av de konceptuella enheterna i ett schema väljs sedan tröskelvikt. Tröskelvikten för schemat beräknas som de ingående enheternas medelvikt, d v s som summan av alla enheters vikt dividerad med antalet enheter.

De enheter vars vikt är större än eller lika med tröskelvikten väljs nu i utgångsläget som deskriptorer. Man kan naturligtvis modifiera och korrigera det urval som man nu kommit fram till på olika sätt. T ex kan man göra de enheter som ingår i en is-a-hierarki till schemadeskriptorer, eller kan det vara mänskliga bedömningar av semantisk eller konceptuell natur som man vill låta påverka valet av deskriptorer.

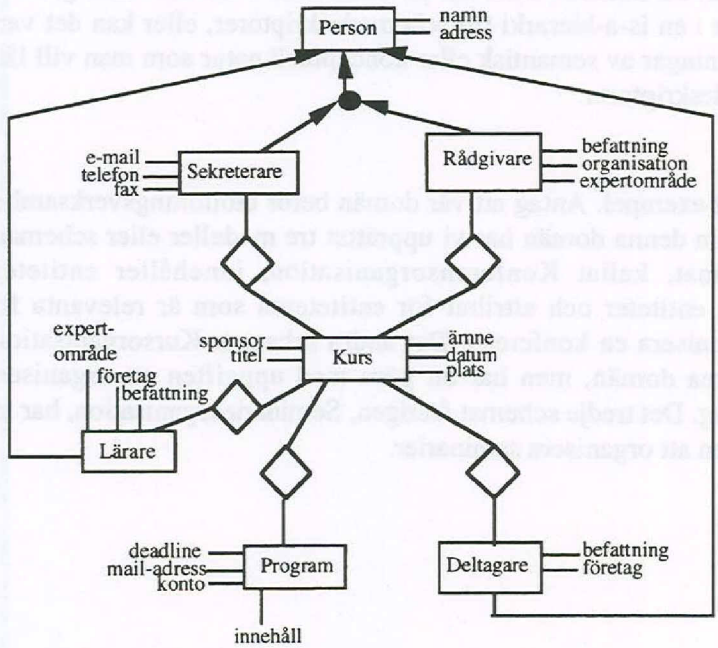
Ett exempel

Låt oss titta på ett exempel. Antag att vår domän berör utbildningsverksamhet av olika slag. Inom denna domän har vi upprättat tre modeller eller scheman. Det första schemat, kallat Konferensorganisation, innehåller entiteter, relationer mellan entiteter och attribut för entiteterna som är relevanta för uppgiften att organisera en konferens. Det andra schemat, Kursorganisation, ligger inom samma domän, men har att göra med uppgiften att organisera kurser av olika slag. Det tredje schemat återigen, Seminarieorganisation, har att göra med uppgiften att organisera seminarier.

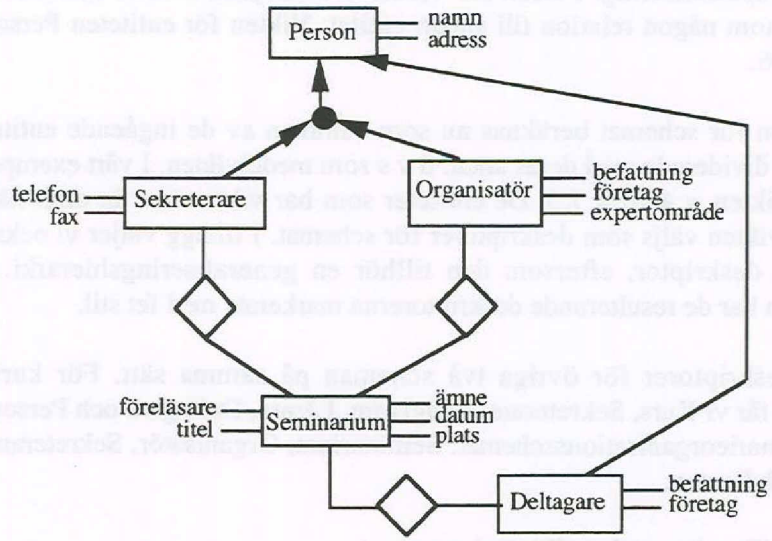
Samtliga tre scheman återfinns nedan grafiskt beskrivna i en E/R-liknande notation.



Figur 2: Schema för konferensorganisation



Figur 3: Schema för kursorganisation



Figur 4: Schema för seminarieorganisation

Om vi nu beräknar vikt för de konceptuella enheter som ingår i schemat avseende konferensorganisation, kan vi ställa upp följande tabell:

	ENTITET	SP	HL	DC	DD	AD	W
1.	**Person ***	2	1	-	3	-	6
2.	**Sekreterare**	3	2	2	-	1	8
3.	**PK-medlem**	4	2	2	-	1	9
4.	**Konferens**	6	-	-	-	4	10
5.	Call-for-paper	3	-	-	-	1	4
6.	**Deltagare**	3	2	2	-	1	8

Tabell 1: Vikter för enheterna i konferensschemat

SP = antal strukturella egenskaper
HL = hierarkisk nivå
DC = antal andra subobjekt på samma nivå
DD = antal direkta efterföljare (subobjekt)
AD = antal grannar med direkta samband

Entiteten Person, t ex, har 2 attribut, ligger överst i en is-a-struktur och ingår inte i någon specialisering. Person har vidare 3 subobjekt, men är inte direkt relaterad genom någon relation till annan entitet. Vikten för entiteten Person blir därför = 6.

Tröskelvikten för schemat beräknas nu som summan av de ingående entiteternas vikter dividerade med deras antal, d v s som medelvikten. I vårt exempel blir tröskelvikten = 45/6 = 7.5. De entiteter som har vikt större än eller lika med tröskelvikten väljs som deskriptorer för schemat. I tillägg väljer vi också Person som deskriptor, eftersom den tillhör en generaliseringshierarki. I tabellen ovan har de resulterande deskriptorerna markerats med fet stil.

Vi väljer deskriptorer för övriga två scheman på samma sätt. För kursorganisation får vi Kurs, Sekreterare, Rådgivare, Lärare, Deltagare och Person, och för seminarieorganisationsschemat: Seminarium, Organisatör, Sekreterare, Deltagare och Person.

Fas 2: Klassificering av kandidatscheman

För att underlätta identifiering av konceptuella enheter lämpliga att abstrahera återanvändbara komponenter ifrån, grupperas de konceptuella scheman som valts som kandidater i likhetskluster ("similarity clusters") på grundval av hur pass lika de är.

Två schemans likhet beräknas på grundval av antalet schemadeskriptorer som de har gemensamma.

Likhetskoefficienten för två scheman beräknas enligt:

$$Sim(S_1,S_2)=(\#\text{ of SDs} \in (S_1 \wedge S_2))/((\#\text{ of SDs} \in S_1) + (\#\text{ of SDs} \in S_2))$$

$$0 \leq Sim(S_1,S_2) \leq 1$$

Klustring av konceptuella scheman sker på basis av:

- likhetskoefficienter och
- hänsyn till synonymer.

Likhetsmatrisen för de tre schemana i exemplet blir (efter det att hänsyn har tagits till de synonymlänkar som finns definierade i lexikonet):

	Schema: Konf.org.	Schema: Sem.org.	Schema: Kursorg.
Schema: Konf.org.	-	0.6	0.6
Schema: Sem.org.	0.6	-	0.6
Schema: Kursorg.	0.6	0.6	-

Tabell 2: Likhetsmatris för exemplets tre scheman

Vi bestämmer oss för att skapa ett enda kluster som innehåller de tre schemana och som alltså har likhetskoefficienten = 0.6.

Att först skapa kluster av likartade scheman underlättar nästa uppgift, som innebär att urskilja kandidater till återanvändbara komponenter.

Fas 3: Urval och klassificering av kandidatkomponenter

Givet ett kluster jämförs nu schemadeskriptorerna för i klustret ingående scheman i syfte att välja ut de konceptuella enheter som uppvisar hög grad av inbördes "semantisk affinitet".

Semantisk affinitet råder mellan konceptuella enheter i olika scheman om de beskriver mängder av objekt som har ett antal gemensamma egenskaper.

Man arbetar med ett antal s k **affinitetskoefficienter** avseende:

- strukturell och beteendemässig affinitet,
- hierarkisk affinitet och
- grannaffinitet ("adjacency affinity")

Strukturell och beteendemässig affinitet:

$$SA(CU_1,CU_2) = \frac{2(\text{SPs gemensamma för } CU_1 \text{ och } CU_2)}{\sum_{i=1}^{2} \text{SPs för } CU_i}$$

$$BA(CU_1,CU_2) = \frac{2(\text{BPs gemensamma för } CU_1 \text{ och } CU_2)}{\sum_{i=1}^{2} \text{BPs för } CU_i}$$

Hierarkisk affinitet:

$$HA(CU_1,CU_2) = \frac{2(\text{gemensamma föregångare och efterföljare till } CU_1 \text{ och } CU_2)}{\sum_{i=1}^{2} \text{föregångare och efterföljare till } CU_i}$$

Grann-affinitet ("adjacency affinity"):

$$AA(CU_1,CU_2) = \frac{2(\text{närliggande CUs gemensamma för } CU_1 \text{ and } CU_2)}{\sum_{i=1}^{2} \text{närliggande CUs för } CU_i}$$

Varje koefficient kan anta värden mellan 0 och 1. Värdet 0 utsäger att de två konceptuella enheterna inte uppvisar någon affinitet d v s att det finns inga gemensamheter för de två enheterna. Värdet 1 utsäger naturligtvis, i motsats till detta, att de två enheterna är starkt besläktade – de är i själva verket helt ekvivalent definierade i de två schemana.

På grundval av dessa affiniteter kan sedan den **globala affiniteten** beräknas som:

$$GA(CU_1,CU_2) = w\ SA(CU_1,CU_2) + w\ BA(CU_1,CU_2) + HA(CU_1,CU_2) + AA(CU_1,CU_2)$$

Vikten w, där w>1, införs för att ge **större vikt till de strukturella och beteendemässiga** affinitetskoefficienterna, jämförda med hierarki- och "grann"- koefficienterna. I vårt exempel har w = 3 valts. De framräknade värdena kan nu sammanställas i en matris, en s k affinitetsmatris.

CUs	Kurs	Sekre-terare	Råd-givare	Lärare	Delta-gare	Kon-ferens	Sekre-terare	PK medlem	Delta-gare	Semi-narium	Sekre-terare	Orga-nisatör	Delta-gare
Kurs		0	0	0	0	**2.9**	0	0	0	**2.6**	0	0	0
Sekre-terare	0		1.2	1.2	1.3	0	**3**	1.1	1.3	0	**2.7**	1.2	1.3
Råd-givare	0	1.2		**4**	**3.7**	0	1.2	**2.7**	**2.7**	0	1.3	**3**	**2.7**
Lärare	0	1.2	4		**3.7**	0	1.2	**2.7**	**2.7**	0	1.3	**3**	**2.7**
Del-tagare	0	1.3	3.7	3.7		0	1.3	2.4	**2.7**	0	1.5	**2.7**	**3**
Konfe-rens	2.9	0	0	0	0		0	0	0	**2.8**	0	0	0
Sekre-terare	0	3	1.2	1.2	1.3	0		2.1	2.2	0	**2.7**	1.2	1.3
PK-medlem	0	1.1	2.7	2.7	2.4	0	2.1		**3.1**	0	1.2	**2.7**	2.4
Delta-gare	0	1.3	2.7	2.7	2.7	0	2.2	3.1		0	1.3	2.4	**2.7**
Semi-narium	2.6	0	0	0	0	2.8	0	0	0		0	0	0
Sekre-terare	0	2.7	1.3	1.3	1.5	0	2.7	1.2	1.3	0		2.3	2.4
Organi-satör	0	1.2	3	3	2.7	0	1.2	2.7	2.4	0	2.3		**3.7**
Delta-gare	0	1.3	2.7	2.7	3	0	1.3	2.4	2.7	0	2.4	3.7	

Tabell 3: Affinitetsmatris för exemplet

Från dessa värden kan indelning ske i **affinitetsmängder** (affinity sets), där varje mängd innehåller de konceptuella enheter som är "starkast inbördes kopplade" till varandra. Hur pass starkt enheterna skall vara kopplade för att vara intressanta att ta med i det vidare arbetet avgörs genom att en tröskelvikt sätts.

I vårt exempel har tröskelvikten satts till 2.5 och 4 stycken affinitetsmängder urskilts. Mängderna är:

AM1: Kurs, Konferens, Seminarium

AM2: PK-medlem, Deltagare (konferens)

AM3: Rådgivare, Lärare, Deltagare (kurs), Organisatör, Deltagare (seminarium)

AM4: Sekreterare (kurs), Sekreterare (konferens), Sekreterare (seminarium).

Fas 4: Utformning av återanvändbara komponenter

Med underlag i definierade affinitetsmängder samt naturligtvis **mänsklig bedömning, utvärdering och beslut,** abstraheras nu en motsvarande generisk resurs och riklinjer för återanvändning skapas. I detta läge har vi skapat en **återanvändbar komponent**.

Statiska konceptuella enheter ger upphov till återanvändbara **resurs**-komponenter och dynamiska konceptuella enheter till återanvändbara **process**komponenter.

Generiska konceptuella enheter ges namn som så väl som möjligt uttrycker deras semantik och som härleds från semantik och roller för de enheter från vilka de abstraheras.

I exemplet kan t ex följande abstraktioner resultera:

(Konferens, Seminarium, Kurs) = **Möte**

(Sekreterare-konferens, Sekreterare-seminarium, Sekreterare-kurs) = **Sekreterare**

(PK-medlem, Organisatör, Rådgivare, Lärare, Deltagare-kurs, Deltagare-seminarium) = **Deltagare/Organisatör.**

För att en generisk konceptuell enhet skall få kallas återanvändbar komponent/resurs, **måste den vara försedd med riktlinjer som anger hur den kan återanvändas.**

Den generiska, konceptuella enhetens olika **återanvändningsroller** definieras. Rollen anger typ av återanvändning, t ex Möte i rollen av Konferens. För varje roll anges förslag till **åtgärder** för att anpassa och modifiera den generiska enheten avseende samband och specialiseringsberoenden samt **regler** för hur man kan berika den konceptuella enheten med tillägg av mer specifika strukturella och beteendemässiga egenskaper typiska för vald återanvändningsroll.

T ex för Möte i rollen av Konferens:

(M) <definiera en relation med **Sekreterare** CU>

välj **Sekreterare** guideline-CU för mer information>

(0) <definiera en relation med **Lärare** CU>

se **Deltagare/Organisatör** specialiseringar för att härleda Lärare> OR <definiera ett attribut **Lärare** för **Möte** CU> ...

(M anger "mandatory" och O "optional").

Fas 5: Assimilering av återanvändbara komponenter

Återanvändbara komponenter definieras som **generiska konceptuella enheter** plus **riktlinjer** (guidelines) för hur den generiska enheten kan anpassas till ett speciellt sammanhang/en speciell applikation.

Komponenterna lagras nu i ett återanvändningsrepository. Denna fas kallas assimilering, eftersom det är viktigt att **nya** återanvändbara komponenter integreras väl med de komponenter som redan finns lagrade i repositoriet, d v s de komponenter som tidigare tagits fram som återanvändbara.

I assimileringsfasen ingår följande delarbetssteg:

1. Förberedande integration
2. Schemajämförelse
3. "Conforming" av schema.

Förberedande integration

Förberedande integration handlar om att välja ut vilka scheman som skall integreras och att avgöra i vilken ordning de skall integreras. Därmed definieras den integrationsstrategi som skall användas.

Schemajämförelse

Schemajämförelse innebär att analysera och jämföra de scheman som skall integreras i syfte att finna motsvarigheter mellan begrepp samt att upptäcka olika typer av möjliga konflikter såsom namnkonflikter, typkonflikter och beroendekonflikter.

"Conforming" av schema

Schema "conforming" handlar om att lösa konflikter och avser konceptuella enheter och riktlinjer.

Representationer av samma begrepp i olika scheman klassificeras t ex som:

- identiska,
- ekvivalenta,
- kompatibla eller
- icke kompatibla.

Detta arbetssteg avslutar design-for-reuse-processen. De återanvändbara komponenterna har skapats, noggrant beskrivits och lagrats i ett speciellt repository. Avsikten är att de nu skall kunna utsökas och användas vid olika ny- och vidareutvecklingsinsatser.

5.2.3 Verktyg

Vid PdM har man utvecklat verktyg som stödjer applikationsutveckling med återanvändning (kallat RECAST) och verktyg som assisterar i arbetet med att konstruera återanvändbara komponenter enligt ovanstående metodik (kallad EXTRACT). De interagerar med ett återanvändningsrepository.

6 Sammanfattning och slutsatser

Vi konstaterade inledningsvis att det råder brist på systematisk metodik och genomarbetade arbetssätt som stöd för att utföra reverse engineering och reengineering av informationssystem.

Verktyg och andra hjälpmedel för delar av arbetet finns i viss utsträckning och vi har presenterat några metoder/delmetoder som skulle kunna användas som komponenter i en mer heltäckande metod. Metoderna berör två viktiga delområden inom reengineering: migrering av arvssystem och utformning och användning av återanvändbara komponenter.

Vi har vidare försökt betona att en bra metod också måste fokusera frågan om att sätta en informationssystemförändring, t ex en migrering, i ett större sammanhang – i ett verksamhetssammanhang. I tillägg till de krav på en migreringsmetod som återgetts i avsnitt 2.3, har vi därför i kapitel 4 formulerat ytterligare ett antal krav avsedda att beakta detta.

I syfte att underlätta framtida utveckling, vidareutveckling och förändring av system kan den nya systemplattform som man vill skapa vid reengineeringsinsats innehålla inslag av återanvändbara komponenter.

Man vill alltså både kunna skapa komponenter som är återanvändbara (*design for reuse*) och använda sig av dessa komponenter när man omformar och vidareutvecklar system (*design by reuse*).

Komponenterna – som kan ligga på olika nivåer – kräver i båda fallen omsorgsfull semantisk/konceptuell analys och beskrivning. Den presenterade ansatsen betonar att en "konceptuell enhet" inte kan betraktas som en återanvändbar resurs förrän den har försetts med riklinjer för hur den skall eller kan tolkas och anpassas/förfinas vid återanvändning.

Metoduppbyggnaden inom reengineeringsområdet bör intensifieras. Möjligheter till samverkan mellan projekt i Europa och USA bör tillvaratas. Många metodkomponenter och beskrivningssätt kan hämtas från metodik för forward engineering. Denna typ av metodik bör också inriktas mot successiv systemutveckling där löpande förändringar och förbättringar är sättet att skapa de informationsstöd som "affären" behöver.

7 Referenser

Referenser och litteraturförteckning.

[1] Arnold, Robert S (Ed). "Software Reengineering", IEEE Computer Society Press, 1993

[2] Bellinzona, R. et al: "Methodology for Reuse", Politecnico di Milano, F3.PdM.2-1-3-R2, April 1993

[3] Biggerstaff, T.J. "Design Recovery for Maintenance and Reuse", IEEE Computer, July 1989

[4] Brodie, Michael L. and Stonebraker, M. "DARWIN: On the Incremental Migration of Legacy Information Systems", GTE Laboratories Technical Report TR-0222-10-92-165, March 1993

[5] Brodie, Michael L. "Interoperable Information Systems: Motivations, Challenges, Approaches, and Status", Documentation at the Nordic Symposium on Interoperability and Legacy Systems, April 20-21, 1994, SISU, Stockholm

[6] Castano, S, De Antonellis, V. "A Model for Reusable Requirements", Politecnico di Milano, F3.PdM.2-1-3-R1, November 1992

[7] Chikofsky, E, and Cross II, J. "Reverse Engineering and Design Recovery: A Taxonomy", IEEE Software, Jan 1990

[8] Dahl, R & Johansson, L-Å: "Referat och reflektioner från konferensen 3rd Reverse Engineering Forum", SISU, 1992

[9] Forte, G. "Re-engineering Tools: A Spectrum of Objectives and Capabilities", CASE Outlook, Vol 6, No 3, 1992

[10] Francalanci, C, and Pernici, B. "Hierarchical classification of reusable conceptual components", Politecnico di Milano, Italien, 1994

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

[12] Hagwall, H. "Återanvändning av programvara – utvecklingen i USA", Utlands-rapport, Sveriges Tekniska Attachéer, 1992

[13] IEEE Software (ISSN 0740-7459), January 1990. Temanummer om Software Maintenance, Reverse Engineering & Design Recovery.

[14] IT 2000, Effektiv IT, Förutsättningar för ett nytt utvecklingsprogram inom informationsteknologins tillämpningsområden, En förstudie, Ds 1993:43, Näringsdepartementet, Regeringskansliets offsetcentral, 1993.

[15] Johansson, L-Å och Gustafsson, M.R. "Affärsmässiga scenarier som bakgrund till reengineering av informationssystem", Effektiv-IT, Systemarvet, Rapport Nr 3, SISU, 1994

[16] Johansson, L-Å, Dahl, R, Gustafsson, M.R. "Kunskap för hantering av systemarvet – en första systematisering", Effektiv-IT, Systemarvet, Rapport Nr 8, SISU, 1994

[17] Object Management Group and X/Open. "The Common Object Request Broker: Architecture and Specification", OMG Document Number 91.12.1, Revision 1.1, 1992

[18] Proceedings of ERCIM Workshop on Methods and Tools for Software Reuse, Institute of Computer Science, Heraklion, Crete, Greece, October 29-30, 1992

[19] Ulrich, W. "Re-development Engineering: Formulating and Information Blueprint for the 1990's", CASE Outlook, No 2, 1990, pp 15-23

[20] "Understanding enables Reengineering", Proceedings of the 3rd Reverse Engineering Forum, Sept. 15-17, 1992, Northeastern University, Burlington, Massachusetts, USA, 1992.

Effektiv IT-rapporter

Nr 1 Att Mäta Informationsteknologi – Data om IT i Sverige och utomlands,
Mattias Hällström, december 1993. *IT:s Ekonomi & Management*

Nr 2 Mätning för Effektiv Systemutveckling,
Tapani Kinnula, mars 1994. *Systemutvecklingens Ledtider & Kvalitet*

Nr 3 Affärsmässiga Scenarier som bakgrund till Reengineering av Informationssystem,
Lars-Åke Johansson, Mats R Gustafsson, mars 1994. *Systemarvet*

Nr 4 Concepts and Notations for Open-edi Scenarios,
Matts Ahlsén, mars 1994. *Affärskommunikation*

Nr 5 Business Process Reengineering – vad är det?
Mattias Hällström, april 1994. *Verktyg för Verksamhetsutveckling*

Nr 6 Managing Information Technology: The Capital Budgeting Process,
Thomas Falk, Nils-Göran Olve, maj 1994. *IT:s Ekonomi & Management*

Nr 7 Integrerad Systemutveckling – lärdomar från industrin tillämpade på
systemutveckling, Sten-Erik Öhlund, Lars Bergman, maj 1994.
Systemutvecklingens Ledtider & Kvalitet

Nr 8 Kunskap för hantering av systemarvet – en första systematisering,
Lars-Åke Johansson, Mats R Gustafsson, Roland Dahl, juni 1994. *Systemarvet*

Nr 9 Metoder för Business Process Reengineering,
Mattias Hellström. *Verktyg för Verksamhetsutveckling*

Nr 10 GIATs modell för integrering av logiskt underhåll med utveckling av produktsystem,
Lars Bergman. *Systemutvecklingens Ledtider & Kvalitet*

Nr 11 Ekonomisk värdering av IT-satsningar,
Nils-Göran Olve. *IT:s Ekonomi & Management*

Nr 12 IT i årsredovisningen,
Nils-Göran Olve. *IT:s Ekonomi & Management*

Nr 13 Kalkylmodeller för reverse engineering/reengineering-insatser – vad finns idag?
Lars-Åke Johansson, Mats R Gustafsson. *Systemarvet*

Nr 14 Om arkitektur för samverkande informationssystem,
Matts Ahlsén. *Affärskommunikation*

Nr 15 Enkät- och intervjuundersökning om värderingsinstrument för IT-investeringar,
Mats Waltré. *IT:s Ekonomi & Management*

Nr 16 Gruppdatorn – ett verktyg för verksamhetsutveckling,
Mattias Hellström. *Verktyg för Verksamhetsutveckling*

Nr 17 Metodik för reverse engineering/reengineering – ett eftersatt område.
Lars-Åke Johansson, Mats R Gustafsson. *Systemarvet*

Nr 18 Processförbättring för införande av ISU – några exempel på effektiv produktframtagning
Sten-Erik Öhlund, Lars Bergman. *Systemutvecklingens Ledtider & Kvalitet*

Nr 19 Datorstöd för Integrerad Systemutveckling,
Sten-Erik Öhlund, Lars Bergman. *Systemutvecklingens Ledtider & Kvalitet*

Svenska Institutet för Systemutveckling, SISU, bedriver forskning, följer utvecklingen och förmedlar kunskap om informationsteknologins tillämpning på informationsanvändning och informationsförsörjning i företag, myndigheter och andra organisationer. Institutet verkar inom detta område som ett opartiskt nationellt kompetenscentrum.

SISU

Electrum 212, 164 40 Kista
Isafjordsgatan 26
Telefon 08-752 16 00 Telefax 08-752 68 00