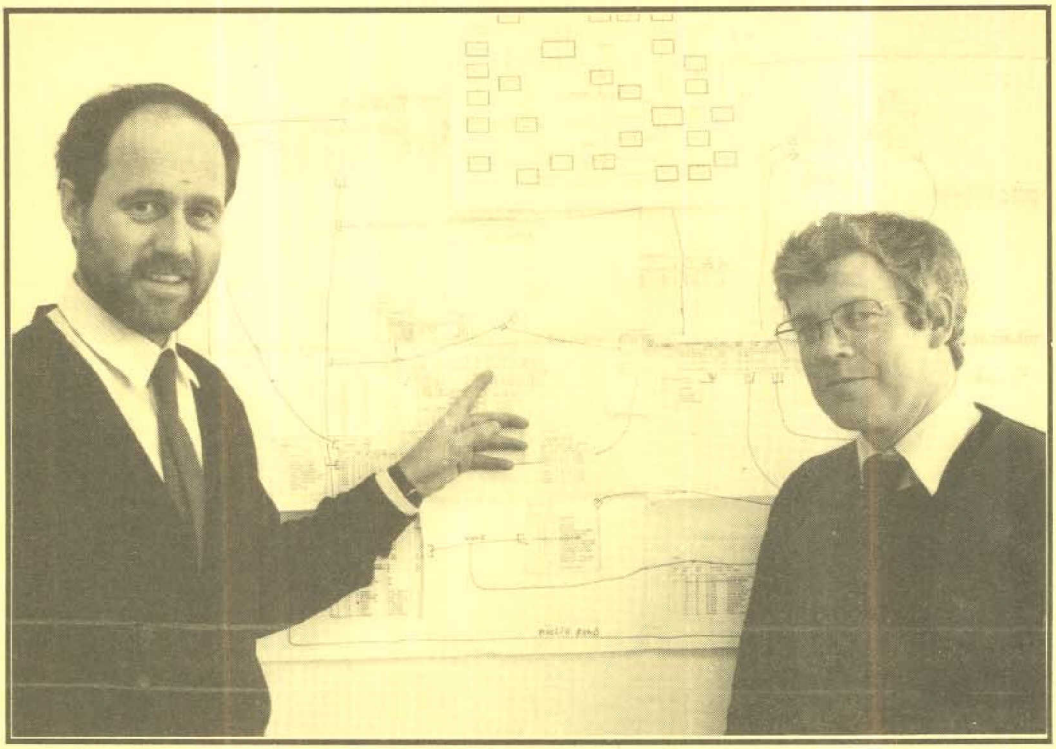
SISU informa

Nr 88/3-4



Claas Åkesson (t.v.) och Eskil Swende startade IRM Consult 1983 tillsammans med Rolf Edgren. Idag räknar man 27 av Sveriges 75 största företag som kunder och har medverkat i ca 300 datamodelleringar. - *Eskil Swende är kontaktman till SISU.*

Innehåll

- I detta nummer -*Lästips för dig* 1
- SISU kalendarium 1
- Metoder och datorstöd i samverkan - konferenspåminnelse 2
- Att välja strategi för CASE - skall metoden styra verktyget eller tvärtom; *Janis Bubenko, SISU* 4
- Kunskapsbaserade användar-gränssnitt; *Erik Knudsen, SISU* 6
- Diagnosticering av konceptuella schemata 8
  Benkt Wangler och Rolf Wohed, SISU
- IRM Consult säljer stadsplanerings-idén; *Lars Bergman, SISU* 12
- Objektorienterad systemutveckling - en översikt; 22
  Matts Ahlsén och StefanBritts, SISU
- Internationellt kalendarium 27
- Integrerad modellering av funktioneroch data i Ericssonmetod; 28
  Mattias Hällström, SISU

SISU informa utges av Svenska Institutet för Systemutveckling.
Ansvarig utgivare: Janis Bubenko jr, tel 752 16 00.
Redaktionen: Lars Bergman & Marianne Sindler

Adress: Box 1250, 164 28 Kista. Besöksadr: Electrum, Kista. Tel. 08- 752 16 00. Fax: 08- 752 68 00

Inledaren

I detta nummer; *lästips för dig*

SISU var väl representerat på NordData 88. I det här numret av Informa finns föredragen med i något redigerad form. Artiklarna ger ett tvärsnitt genom verksamheten inom SISU.

"Att välja strategi för CASE" ger en överblick över problemställningar som man kommer att ställas inför i många organisationer i en kommande period. Dessa problemställningar ligger också i fokus för verksamheten inom SISU nu och framöver.

"Kunskapsbaserade användargränssnitt" presenterar kortfattat ett område som KBS-gruppen inom SISU arbetar intensivt med. Här är t.ex. HSQL-projektet och i viss mån Televerksprojektet aktuella.

"Diagnosticering av konceptuella schemata" ställer lite större krav på läsaren. Där presenteras arbeten som är mera forskningsinriktade. Arbetet i kvalitetsprojektet är närmast grunden för artikeln, men har också direkt koppling till olika projekt där RAMATIC ingår. Diagnosticering med hjälp av datorstöd skall bidra till förbättrad kvalitet i t ex konceptuella modeller som görs med stöd av RAMATIC. Successivt kommer alltså en "intelligent" kvalitetskontroll att byggas in i RAMATIC. Där berörs VDDS-projektet, Ericssonprojektet samt RAMATIC-projektet.

"Objektorienterad systemutveckling - en översikt" tar upp det område som AVANCE-gruppen arbetar med i form av "in-house"-projekt. Artikeln ställer krav på eftertanke hos läsaren, men ger i gengäld en genomarbetad överblick över ett område som drar till sig mer och mer intresse ute i praktikfältet. Artikeln ingår som en del i projektgruppens satsning på att öka sina omvärldskontakter inom praktikfältet.

"Integrerad modellering av funktioner och data i Ericssonmetod" presenterar i översikt den metodik som utvecklats i samarbetet mellan Ericsson och SISU i Ericssonprojektet. Artikeln är hållen på en praktikerorienterad nivå som bör göra den intressant och läsvärd för alla metodintresserade.

IRM Consult presenteras utförligt med förhoppningen att artikeln skall kunna vara användbar både för din egen läsning och för att sprida aktuella idéer i din verksamhet. I artikeln presenteras **IRMA-metodiken** på som jag hoppas ett lättillgängligt sätt.

Lars Bergman

Notiser

Konferensen **"Metoder och datorstöd i samverkan"** presenteras som "annons" på följande uppslag. Vill du ha programmet kan du höra av dig till oss eller till SSI.

Förberedelsena för **"First Nordic Conference on Advanced Systems Engineering"** har satt igång. - En grundtanke i uppläggningen är att vi skall satsa hårt på att påverka "vetenskapliga" föredragshållare till att lägga sig på en för praktiker intressant och förståbar nivå och inriktning. På motsvarande sätt vill vi arbeta så under konferensen att praktikerföredragen bildar grund för diskussion och slutsatser som kopplar till pågående och kommande forskning och utveckling. - **Ambitionen skall vara att åstadkomma broslagning!**

NYA INTRESSENTER

Sedan sist har vi fått två nya intressenter, som vi hälsar varmt välkomna i kretsen.

Arthur Young AB med **Åke Ekström** som kontaktperson.

SPADAB med **Göran Lustig** som kontaktperson.

Kalendarium

SEPTEMBER

7-8 METODER OCH DATORSTÖD I SAMVERKAN.
Konferens och utställning i samarbete mellan SSI Göteborg, VolvoDataSkolan och SISU.

OKTOBER

slutet
Dataadministrationsprojektet håller avrundnings seminarium.

1989 MAJ

9-11 First Nordic Conference on Advanced Systems Engineering,
Konferens och Lab i samarbete med SSI.

Konferens: Metoder och datorstöd i samverkan

Du missar

- AU och ADB
- vägval inför 90-talet
- effekter, behov, teknologi
- utveckling och förvaltning

Metoder och datorstöd i samverkan

En strategisk KONFERENS med demonstrationer och utställning i Volvohallen, Göteborg,

7 - 8 SEPTEMBER

Volvo Data Skolan,
SSI Göteborg
och SISU

Samverkan mellan metoder och datoriserade hjälpmedel är av strategisk betydelse för AU- och ADB-verksamheten. Konferensen ger dig erfarenheter och synsätt i början av en ny epok.

Framtidens arbetssätt för utveckling och förvaltning av system har börjat sitt intåg. Skepsis håller på att gå över i optimism. Utvecklingen från yrkesrollen som programmerare till informationsingjenör är i gång.

Att effektivisera AU-ADB-verksamheten på ett smidigt sätt är en utmaning och kräver eftertanke. Speciellt gäller detta möjligheterna och svårigheterna att förena redan utvecklade informationssystem med nya sätt att arbeta.

För att lyckas med att införa och bruka datoriserade hjälpmedel måste vi ha en *helhetssyn* som tar hänsyn till arbetssätt, arbetsmiljö, organisation, teknik och sambanden mellan dessa. Detta för att utveckla en AU-ADB-verksamhet som svarar upp till de krav på effektivitet och kostnadsmedvetande. Krav som kommer att bli ännu tydligare under 90-talet.

Det finns många exempel på installationer i företag och myndigheter vilka utgått från en teknisk och charmfull produkt som nu slumrar i "törnrosasömn". Att införa och bruka datoriserade hjälpmedel är *endast delvis ett tekniskt problem.*

Vågen av datoriserade hjälpmedel för systemutveckling och -förvaltning har börjat skölja över oss och har med sig budskap som ibland kan uppfattas som spektakulära. Blickar vi tillbaka några år känner vi igen situationen som kännetecknade början av den s-k- fjärde generationens språk (4GL).De hjälpmedel som bjuds ut idag har en mängd etiketter;

Konferens: Metoder och datorstöd i samverkan

väl inte?

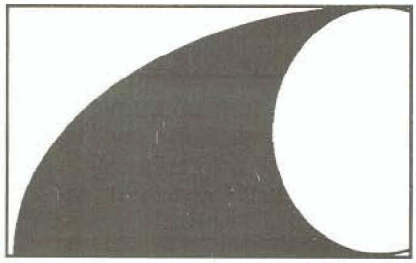
VI SOM ARRANGERAR

SSI GÖTEBORG. Svenska samfundet för Informationsbehandling är en förening för yrkesverksamma inom AU och ADB med syfte att bidra till medlemmarnas professionella utveckling. Föreningen har idag ca 2.700 medlemmar. SSI Göteborg bedriver bl a en omfattande seminarieverksamhet.

SISU, Svenska Institutet för Systemutveckling, bedriver forskning, utveckling och kunskapsöverföring inom området systemutvecklling. Metoder och datorstöd för utveckling av informationssystem är kärnan i verksamheten. Institutet drivs gemensamt av 36 företag och myndigheter. Verksamheten omfattar drygt 25 personer.

Volvo Data Skolan är Volvo Datas enhet för kunskapsöverföring inom Volvokoncernen. Volvo Data är koncernens dataservicebolag och omsätter 570 Mkr och har 700 anställda. Volvo Data Skolan och företagets metodenhet utvecklar, stöder och utbildar i metoder och datorstöd bl a inom området systemutveckling.

En strategisk konferens med demonstrationer om
HUR man gör
VAD man lärt
VAD man vinner



inför 90-talets vägval.

Riktar sig till personer med ansvar och intresse för ändamålsenlig AU och ADB-verksamhet i större företag och myndigheter.

Än är det inte för sent!

Information och bokning!

Ring Eva Lindberg, SSI:s kansli,

08-24 85 55.

Data Dictionary, CASE, I-CASE, IE, CSP, applikationsgeneratiorer, repositories m mm mm m, vilket gör det svårt att göra jämförelser och framförallt att välja bästa satsningen för den egna AU-ADB-verksamheten och den egna metodsynen.

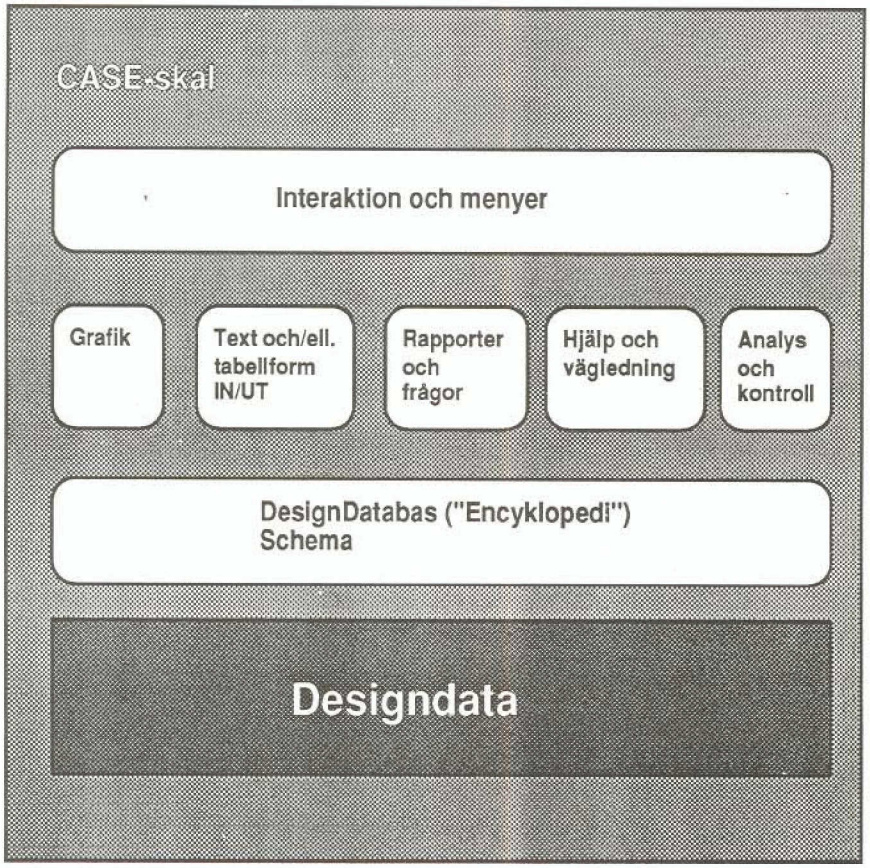
Vår konferens skall hjälpa dig att bemästra området och därmed ge dig kunskap inför vägval för utvecklingen i din egen organisation. Du behöver en heltesbeskrivning och konkreta "bevis" för vad som gjorts och vad som kommer. Vi satsar på professionella föredrag av personer i organisationer, som börjat tackla dessa frågor. Dra nytta av deras kunskaper och erfarenheter.

Vi kommer att erbjuda demonstrationer och praktikfall för de intressantaste datorstöden.

ATT VÄLJA STRATEGI FÖR CASE – SKALL METODEN STYRA VERKTYGET ELLER TVÄRTOM?

Föredraget baseras på en rapport "Selecting a Strategy for Computer-Aided Software Engineering (CASE)" utarbetad vid SISU och SYSLAB, maj, 1988. I det följande ges en sammanfattning av rapporten.

Föredrag vid NordData88,
av Janis A. Bubenko jr



Syftet med rapporten är att belysa och diskutera frågan om att välja en lämplig strategi för att införa CASE i organisationen. Frågan motiveras av det stora antal CASE-verktyg som nu börjar uppträda på marknaden, kanske fler än 100. Det förekommer allt ifrån enkla "leksaksverktyg", som vanligen bara stöder en mindre del av systemutvecklingskedjan, till relativt fullständiga verktyg som stöder en rad av sammanhängade metoder, dvs en metodkedja från de tidiga etapperna till generering av programkod. Kostnaderna för anskaffning varierar mellan några tusen kronor till flera hundra tusen kronor. De allra flesta verktyg baseras på och stöder en viss metodik. Anskaffning av ett sådant innebär att organisationen metodmässigt måste anpassa sig till verktygets metodik. I de allra flesta fall innebär detta i praktiken att organisationen måste byta till en mer eller mindra annorlunda metodik. På senare tid har emellertid s.k. "CASE-skal" dyka på marknaden. I likhet med s.k. "expertsystemskal" blir CASE-skalen verktyg om man "fyller dem" med en viss metodik. Med andra ord, CASE-skal är programmeringsverktyg med vars hjälp man kan, med större eller mindre möda, tillverka ett datorstöd för sin egen metodik. Därigenom uppstår den intressanta frågan huruvida man bör anskaffa ett metodspecifikt verktyg eller försöka själv tillverka - eller "beställa" - ett verktyg som direkt passar organisationens metodik.

Frågan är, givetvis, långt ifrån enkel och kan ej entydigt besvaras. En mängd situationsfaktorer måste beaktas för att välja en lämplig strategi i denna fråga. Rapporten syftar således till att belysa denna komplexa frågeställning. För att överhuvudtaget gå in på frågan måste vi först kasta en blick på dagsläget - state of the art - vad avser

Att välja strategi för CASE

metodutveckling och metodtillämpning och vad som tilldrar sig på foskningsområdet avseende metoder. Vi konstaterar att detta fortfarande är - i praktiken - ett krisområde. Mer strikt, ingenjörsmässig metodik används av mindre än 10% av de professionella syetemerarna. Av de metoder som används är de flesta 70-talsmetoder. Modernare metodik är relativt okänd. Forskningen avs metoder följer olika utvecklingslinjer. De vanligaste baseras på antingen konceptuell modellering, objektorientering eller s.k. kunskapsteknik (regelbaserade system). Viss konvergens bland dessa ansatser kan skönjas men vi är ännu långt ifrån en syntes.

Innan vi går in på aktuella verktyg och utvecklingslinjer presenteras ett antal allmänna, funktionella egenskaper ("features"), som CASE verktyg besitter, eller kan i framtiden komma att besitta. Av speciellt intresse är, tycker vi, egenskaper som ger stöd för verifiering, validering och konstruktion av systemspecifikationer. Stöd för projektarbete, projektplanering och styrning är andra viktiga egenskaper. En skiss över en generaliserad arkitektur av ett (framtida) CASE verktyg ges. Verktygets fundamentala komponent, "design-databasen", kan, generellt sett, innehålla metod- och applikations kunskap på flera nivåer. Metodschemat definierar metodikens "språk" i form av objekttyper, samband och olika typer av begränsningsregler (som kontrollerar att man formulerar en specifikation inom gränsen för vad metoden tillåter). Detta kan kompletteras med metodkunskap som kan vägleda och styra processen och företa olika former av kvalitetskontroller. Mer avancerade framtida system kommer att göra bruk avs s.k. domänkunskap (kunskap om ett visst applikationsområde) som kan vägleda en användare att hålla sig inom applikationsområdets vedertagna begrepp, relationer, mm och verksamt bidra till en bättre kvalitetskontroll av specifikationen. I det generella bgreppet designdatabas kan vi också inkludera återanvändbara komponenter av tidigare gjorda specifikationer. Återanvändning kommer på detta sätt att dramatiskt öka "prestandan" hos systemerare. Med detta i bagaget går vi sedan över till dagens CASE verktyg.

När det gäller dagens CASE utveckling kan man tala om metodspecifika verktyg, CASE-skal (dvs verktyg för verktygsutveckling) och den forskning som anknyter till CASE. Bland de 100 eller fler verktyg som som finns på marknaden är praktiskt taget alla "metospecifika". Den metodik som de stöder är relativt konventionell, dvs dataflödesmodellering eller

Därigenom uppstår den intressanta frågan huruvida man bör anskaffa ett metodspecifikt verktyg eller försöka själv tillverka - eller "beställa" - ett verktyg som direkt passar organisationens metodik.

enklare former av datamodellering eller en integrerad dataflödes- och datamodelleringsansats. Deras användargränssnitt är ofta färgrika och imponerande. Vad som är mindre imponerande är specifikationsspråkets "kraftfullhet" och den grad av "intelligens" som verktygen besitter. CASE-skalen skall ju kunna programmeras till önskad kraftfullhet och intelligens men här brister det ofta i vårt metodkunnande. De metoder som används i praktiken är i många avseenden alltför "luddiga" för att kunna formellt hanteras av ett datorstöd. För att bygga in "intelligens" i ett datorstöd måste vi i detalj veta hur vi arbetar enligt en viss metodik och hur vi verifierar och kvalitetskontrollerar de lösningar som åstadkommes. Om detta har vi ännu mycket ofullständig kunskap. I varje fall kan vi inte uttrycka den formellt - inte ens för allmänt kända och etablerade metoder. Att i framtiden göra CASE verktygen mer stödjande och intelligenta är hudtemat för ett flertal forskningsprojekt i detta område. Ett exempel på detta är försök att utveckla "expertsystem" som kan analysera systemspecifikationer och diagnosticera felaktigheter och olika kvalitetsmässiga brister eller "tveksamheter" i dessa.

I rapportens slutavsnitt diskuteras ett antal möjliga strategier för CASE. Den enklaste är att vänta och se och låta andra göra de initiala misstagen. Den mest ambitiösa är att bygga ett eget CASE verktyg (med hjälp av ett CASE skal) för den egna metodiken. En mellanvariant är att anskaffa ett metodspecifikt verktyg. Ytterligare varianter diskuteras. Val av strategi beror på ett antal situationsfaktorer. En grupp av faktorer karaktäriserar organisationens systemutveckling, "mognad", policy, attityder och personalens kompetens och kompetensutveckling. En annan grupp av faktorer karaktäriserar organisationens metodutvecklingspolicy, mognad och strategi. Relationer till och tillgång till kunniga CASE leverantörer eller konsulter utgör en ytterligare grupp av faktorer. Avslutningsvis diskuteras olika strategier med hänsyn till nämnda faktorer. Det rekommenderas att inte tillämpa "vänta och se" strategin utan att genom olika satsningar lära sig så mycket om detta i snabb utveckling varande område. De mer ambitiösa strategierna rekommenderas endast till organisationer som har uppnått en hög grad av metodmognad och som har tillgång till för en sådan satsning tillräckligt kompetent personal.

Vad rapporten inte är.

Rapporten presenterar inte en lista av kommersiellt tillgängliga CASE verktyg och dessas egenskaper. Ej heller görs någon ansats att värdera aktuella verktyg och dessas egenskaper. Rapporten ger inte heller entydiga besked och enkla riktlinjer eller tumregler för val av strategi. Rapportens tonvikt ligger på att öka förståelsen för den aktuella och mångfacetterade utvecklingen inom metod- och CASE området. En sådan förtsåelse är en förutsättning för att kunna göra goda avvägningar och utforma den egna strategin för CASE.

Kunskapsbaserade användargränssnitt

Bakgrund

Begreppet informationssystem blir alltmer spritt inom företag och organisationer. I takt med att alltmer information digitaliseras och användargrupperna som utnyttjar dessa informationssystem blir allt större och mer heterogena krävs alltmer sofistikerade verktyg som kan leva upp till de krav som dessa grupper ställer på dessa system i form av användarvänliga och effektiva gränssnitt.

Föredrag vid Norddata88
av Erik Knudsen, SISU

I princip kan informationssystemen sägas ha utvecklats från att ha varit tämligen otympliga i den meningen att nya och förändrade informationskrav har tagit alltifrån dagar till flera månader att realisera till dagens relativt lättåtkomliga system som är bättre anpassade för en miljö med föränderliga informationskrav. Här har de sk fjärde generations (4GL) verktyg spelat en stor roll men även införandet av andra typer av underliggande datamodeller och av slutanvändarna direkt tillgängliga frågespråk tex SQL har haft en stor betydelse.

Ett par allvarliga problem med frågespåk av SQL:s typ är dock att användarna av ett informationssystem måste känna till det underliggande schema som beskriver innehållet i de aktuella databaserna i informationssystemet samt att de är tvingade att lära sig ett formellt språk som vid mer komplexa frågeställningar kan vara nog så svårt att formulera sig med. Dessa problem har varit kända en längre tid och försök har gjorts och görs för att lösa dessa.

Det finns ett antal olika principer som i någon mening kan sägas vara intressanta varav vi i denna rapport tänkte beskriva ett par speciell varianter nämligen grafik och naturligt språkgränssnitt. Dock är det inte den ytliga formen för kommunikation genom dessa typer av gränssnitt i sig som är intressant utan de extra mekanismer och stödsfunktioner som kan realiseras genom att under analysfaserna i hör grad utnyttja en underliggande kunskapsbas. Det mest centrala begreppet här kan sägas vara graden av samarbetsvilja hos informationssystemet eller gränssnittet.

Grafik

Det finns ett antal olika varianter av grafiskt baserade användargränssnitt och det finns inte utrymme att här beskriva dessa. En viss typ är de system som ritar en grafisk bild av det underliggande schema som beskriver innehållet i en aktuell databas. Ofta representeras entiteter som boxar eller cirklar under det att attribut och relationer mellan entiteter anges som bågar eller pilar. Genom att markera olika entiteter och attribut med hjälp av tex ett pekdon skapas dynamiskt en delstruktur av det totala schemat som av systemet tolkas som en fråga när användaren markerat tillräckligt med symboler.

Det finns klara fördelar med ett sådant användargränssnitt varav kan nämnas att användaren varken behöver känna till det fullständiga schemat emedan det finns tillgängligt direkt på skärmen eller behöver lära sig något formellt frågespråk.

Till nackdelarna hör är att om det underliggande schema som skall representeras grafiskt är stort blir det mycket snart svårt att orientera sig i schemat. Vidare om komplexa villkor kopplas till aktuella värden på markerade attribut kan rent semantiska oklarheter uppstå om vad frågan egentligen avser.

Naturligt språk

Med ett naturligt språkgränssnitt avses ett gränssnitt där användaren kommunicerar med informationssystemet på det språk som är användares eget modersmål. Självfallet kan man tänka sig system där kommunikationen görs i något främmande språk tex engelska men poängen här är att de språk som de allra flesta kan bäst är just sitt eget modersmål. Om man har svenska som sitt modersmål och inte kan formulera ett informationsbehov på samma språk är sannolikheten tämligen liten att man skulle lyckas bättre i SQL.

Det finns ett antal system som tillhandahåller en möjlighet att mer eller mindre kommunicera via ett naturligt språkgränssnitt. Dessa kan klassificeras enligt något schema var och ett med sina egenskaper och mekanismer. En mycket förenklad bild är att göra en skärning mellan nyckelordsbaserade och grammatikbaserade system.

Med den första gruppen avses system som mer eller mindre söker efter speciella ord eller termer i en fråga som definierats med utgångspunkt från den aktuella domänen i informationssystemet. Den andra gruppen baserar den initiala analysen med utgångspunkt från en som regel omfattande grammatik som beskriver mängden av korrekta satser i det aktuella språket.

En skillnad i funktionalitet mellan dessa två klasser är att de nyckelordsbaserade systemen oftare klarar av att göra en analys av en fråga men i gengäld har en högre felprocent, dvs feltolkar helt enkelt semantiken hos frågan medan de grammatikbaserade systemen är felkänsliga för syntaktiska fel som egentligen inte påverkar semantiken i en fråga. För att konkretisera resonemanget räcker det med att tänka efter hur vi människor fungerar. Även om talare gör smärre fel i form av felaktig ordföljd, utelämnar ord eller använder fel prepositioner brukar vi ändå som lyssnare kunna förstå innehållet i budskapet.

Vad vi vill hävda är således följande: För att man skall kunna tala om ett reellt naturligt språkgränssnitt räcker det inte med att gränssnittet på en ytlig nivå accepterar strängar i naturligt språk som inmatning. Det måste till en hel del ytterligare mekanismer och egenskaper för detta och det är här som begreppet samarbetsvilja kommer in.

Samarbetsvilja

Samarbetsvilja hos ett gränssnitt kan definieras på olika sätt. I denna framställning avser vi inte ge en fullständig och uttömmande definition utan nöjer oss med att räkna upp ett antal kriterier som vi anser skall ingå och genom att kontrollera hur många av de kriterier som ingår i något befintligt system kan graden av samarbetsvilja hos systemet i fråga bestämmas.

Ett första kriterium avser förmågan att bortse från smärre syntaktiska fel som inte påverkar semantiken hos frågan. Vad som avses med "smärre" kan naturligtvis bli föremål för diskussion och något entydigt svar kan inte ges men en riktlinje att gå efter kan vara att se på problemet ur ett statistiskt perspektiv. Ett exempel kan konkretisera det hela. Satsen "litsa alla personer" borde rimligen kunna accepteras medan det kanske är tveksamt om man kan kräva att satsen "kann dina datr lsa deta" bör accepteras.

Ett andra kriterium avser det semantiska planet. Att kunna göra en korrekt analys vad avser syntaktiska strukturer och element räcker inte. En semantisk analys är i högsta grad nödvändig. En semantisk analys innefattar i sig en rad olika komponenter och för att kunna utföra vissa av dessa måste en koppling till det underliggande schemat i informationssystemet finnas. Likaledes måste en explicit domänkunskap finnas tillgänglig helst uttryckt i form av en kunskapsbas för att möjliggöra vissa analyssteg.

En rad olika problem uppstår i samband med den semantiska analysen och det finns inte utrymmer för att på något sätt ge en uttömmande uppräkning av dessa här utan vi nöjer oss med att räkna upp några få av dessa. Häri ingår bland annat möjligheten att kunna hantera ambiguitet något som ju är kännetecknande för ett naturligt språk till skillnad från formella språk som inte lider av detta. Det normala vid ambiguitet är att presentera de tolkningar som gjorts och sedan låta användaren välja någon av dessa som motsvarar användarens avsikt eller önskan. Vidare måste systemet kunna fungera i en kontext, dvs syftningar både inom en sats och mellan satser måste kunna hanteras på ett korrekt sätt. Ett exempel: En användare frågar "hur många personer är tjänstlediga på avdelning x" och när resultatet visats ställer följdfrågan "hur många kvinnor" är det underförstått "hur många kvinnor är tjänstlediga på avdelning x". Syftningarna skall kunna ske åtskilliga satser tillbaka under en session, dvs precis som vi människor klarar av att föra diskussion med varandra.

Ett tredje kriterium är att initiativet skall kunna vandra mellan användare och system under dialogens gång. Det normala är att användaren ställer frågor och systemet svarar. I vissa lägen tex när systemet behöver ytterligare information eller förtydliganden för att kunna besvara en fråga skall systemet således kunna överta initiativet och fråga begära ytterligare information av användaren.

Ett fjärde kriterium är att systemet skall kunna ge ytterligare eller alternativ information när det kan vara av intresse. Om en användare frågar "När går nästa flyg till Rio" och nästa avgång är med Concorde är det troligt att om svaret bestående av den korrekta tidsuppgiften åtföljs av en notis om att biljettpriset för den avgången är betydligt dyrare än normalt skulle detta upplevas som betydligt mer informativt än om enbart en tidsuppgift ges som svar. Detta gäller i lika hög grad när något svar inte kan ges eller snarare värden som direkt utgör ett svar på en fråga inte kan finnas. På frågan "När ankommer flyget från Oslo" och just den avgången är inställd bör svaret givetvis bli "Den avgången är inställd men nästa ankommer kl x" framför att enbart säga att "uppgift saknas" eller liknande.

Vid detta laget torde det stå helt klart att för att kunna realisera ett användargränssnitt av det slag som skissats ovan måste både den underliggande kunskapsbasen och de olika analysstegen vara tämligen omfattande. Likaledes torde framgå att det finns ett antal kriterier och egenskaper som bör ingå i ett användargränssnitt för att detta skall kunna kallas för naturligt språk. Oss veterligen finns idag inget färdigutvecklat system som uppfyller samtliga kriterier som skissats ovan även om en del uppfyller vissa delkriterier.

Kunskapbasen

Som begreppet kunskapsbas antyder är det frågan om att samla explicitgjord kunskap i någon lämplig form. Meningen är att den kunskap som är definierad skall vara lätt att använda internt i systemet för att kunna användas för slutsatsdragningar och olika resonemang. Den kunskap som är relevant i detta sammanhang kan skiktas i olika nivåer. Det finns ingen klar princip för i vilket skikt en viss typ av kunskap skall placeras utan en bedömning får göras från fall till fall.

Det första skiktet utgörs av domän och applikationsoberoende kunskap. Som inses kan detta skikt bli tämligen brett eftersom det i princip närmar sig det man skulle kunna kalla allmänbildning. Det andra skiktet innehåller domänberoende men applikationsoberoende kunskap. Detta kan exemplifieras med att domänen är biltillverkning men oberoende av om det är Volvo:s eller Ford:s biltillverkning som avses. Ett tredje skikt avser både domän och applikationsberoende kunskap. Med detta avses således kunskap om innehållet i en specifik databas. Ett fjärde skikt avser vad som kan kallas lokal kunskap. Häri ingår kunskap som är specifik för varje användarkategori eller lokal användargrupp.

Varje skikt skall kunna definieras oberoende av de andra skikten samt likaså kunna utökas inkrementellt. På detta sätt erhålls en möjlighet att låta kunskapsbasens kapacitet ständigt utökas på ett administrationsvänligt sätt. En ytterligare fördel är att enstaka eller vissa skikt lätt kan bytas ut när en annan applikation eller domän bli aktuell.

Avslutning

Forskningen inom naturlig språkbearbetning fortgår alltjämt och nya resultat kommer fram fortlöpande. Dock återstår många problem att lösa innan man kan realisera ett användargränssnitt som hanterar naturligt språk enligt de kriterier som angivits ovan. Därmed inte sagt att det är ogörligt. Redan idag finns tämligen avancerade system som förmår både imponera på och förtjusa en användare av informationssystem.

DIAGNOSTICERING AV KONCEPTUELLA SCHEMATA

Inledning

Dagligen utvecklas och dokumenteras hundratals datamodeller, verksamhetsmodeller, begreppsmodeller, dataflödesmodeller, o.s.v. På basis av dessa byggs sedan informations-system. Det finns då stor risk att logiska fel och andra kvalitativa brister hos dessa problemnära modeller och specifikationer fortplantar sig till de implementerade systemen, eller i varje fall försenar och försvårar utvecklingsarbetet.

Föredrag vid NordData88
av
Benkt Wangler, SISU
Rolf Wohed, SISU

Att avhjälpa sådana fel och brister kostar mer ju senare de upptäcks under utvecklings-processen. Om de inte upptäcks förrän systemet tagits i drift kan de dessutom förorsaka direkt skada.

För att komma till rätta med dessa problem har man på många företag infört särskilda rutiner för *kvalitetssäkring* i form av planerade och strukturerade genomgångar av specifikationer tillsammans med 'domänexperter'. Detta angreppssätt syftar, vad gäller tidiga faser av systemutvecklingsprocessen, främst till att stämma av specifikationen i förhållande till domänen, d.v.s. dels att avslöja avvikelser från denna och dels att försäkra sig om att man fått med allt som beställaren avsett.

Några frågor man nu kan ställa är: Kan vi samla och systematisera kunskap och erfarenhet med vars hjälp vi på ett tidigt stadium skulle kunna analysera de problemorienterade specifikationerna i syfte att upptäcka kvalitativa brister? Vad bör f.ö., i detta sammanhang, förstås med *kvalitet*, d.v.s. vilka egenskaper vill vi att den färdiga modellen eller specifikationen skall ha? Vilken ytterligare kunskap krävs för att kunna ställa diagnoser av detta slag? Kan denna samlade kunskap rent av byggas in i ett 'CASE-verktyg', så att eventuella brister upptäcks redan då specifikationerna produceras?

För att undersöka dessa frågor pågår på SISU ett projekt som syftar till att klargöra begreppet kvalitet med avseende på, främst problemorienterade, systemspecifikationer och att, för ett mindre antal typer av sådana, ange explicita 'kvalitetskriterier', om möjligt på en sådan konkretionsnivå att de kan byggas in i ett datorstöd. Eftersom man rimligen inte kan förutsätta att samma slag av kvalitetskriterier gäller för alla typer av specifikationer har vi i en första etapp koncentrerat oss på en bestämd typ, nämligen *konceptuella schemata*. Naturligtvis spelar också den (grafiska och/eller textuella) notation som används stor roll, såtillvida att man inte kan ställa samma krav på en modelleringsansats med små uttrycksmöjligheter som på en med stora. Det modelleringsspråk som vi närmast haft i åtanke vid våra diskussioner inom projektet är det inom SISU utvecklade SIMOL, som är en binär associativ, objektbaserad ansats bestående av såväl en grafisk som en textuell notation. Den fortsatta diskussionen i denna artikel avser med andra ord i första hand kvalitet hos konceptuella schemata beskrivna i SIMOL. De kvalitetskriterier som anförs är dock generella i den meningen att de kan sägas gälla för alla modelleringsspråk som har motsvarande uttryckskraft.

Parallellt med denna verksamhet har också pågått utveckling av ett expertstöd för diagnosticering av konceptuella schemata, avsett att komplettera det SISU-utvecklade CASE-verktyget RAMATIC. Sektion 4 i denna artikel behandlar frågor som är aktuella i detta sammanhang.

Kvalitetsbegreppet

Med *kvalitet* hos en produkt eller tjänst avses enligt internationella standardise-ringsorganisationen, ISO, dess " ...förmåga att tillfredställa kundens eller användarens behov och förväntningar". Man skulle också kunna säga att någonting har hög kvalitet i samma mån som det fyller sitt syfte.

Den senare formuleringen ställer det tillämnade användningsområdet i centrum. För att kunna avgöra om någonting har hög kvalitet bör man alltså känna till vad det skall användas till. Detta innebär alltså även, att om någonting kan användas till mer än en sak, kan det också ha olika kvalitet med avseende på de olika användingsom-

rådena. Till skillnad från många andra slag av systemspecifikationer kan konceptuella schemata användas i en rad olika sammanhang, förutom för informationssystemutveckling, t.ex. för affärs- och organisations-utveckling.

Av detta följer, för det första, att man vid en analys av kvalitetsbegreppet med avseende på konceptuella schemata, först borde undersöka vilka de potentiella användningsområdena för konceptuell modellering är och därefter för vart och ett lista kvalitetskrav. För det andra innebär det att man vid kvalitetsdiagnos av ett specifikt schema strängt taget bör veta vilket det avsedda användningsområdet är.

Ett konceptuellt schema kan göra nytta på flera olika sätt. Det man först tänker på är antagligen att det skall utgöra ett bra underlag för utvecklingsaktiviteter, t.ex. databas-utformning, där det utgör en del av utgångsmaterialet. Det är härvid viktigt att schemat innehåller så mycket som möjligt av den information som krävs. Detta ställer dels krav på modelleringsansatsens uttrycksmöjligheter, men också på att schemat utformats på ett sätt som underlättar transformationen från konceptuellt schema till t.ex. databas-struktur.

En annan generell nyttoeffekt är att begrepp som används i en verksamhet får en klarare definition. Detta medför att andra, relaterade, specifikationer får en tydligare innebörd, vilket är bra oavsett vad de skall användas till.

Också andra slag av positiva effekter förekommer, t.ex skapande av konsensus omkring en begreppsapparat och skapande av nya användbara begrepp. Här är det dock snarast 'kvaliteten' hos själva modelleringsprocessen. som är avgörande. I detta sammanhang får man inte heller glömma den ubildningseffekt som blir resultatet för de som deltar i en modellerings-process.
I vårt projekt är dock syftet att enbart granska schemat som sådant, oavsett hur det har producerats. Modelleringsprocessen och de kvalitetsaspekter som kan läggas på denna därför inte av primärt intresse och lämnas därför i fortsättningen därhän.

En genomgång av olika användningsområden har dock övertygat oss om att kvalitetskraven inte varierar särskilt mycket med hänsyn till dessa, utan på det hela taget är generella. Rent allmänt kan dock sägas, att om schemat riktar sig till slutanvändare så bör man betona enkelhet och om det skall utgöra underlag för utformning av implementerbara system-komponenter så krävs en hög detaljeringsgrad. Detta talar för att det borde finnas möjlighet att ha schemata på olika detaljerings- och abstraktionsnivåer.

Bland viktiga, generella kvalitetskriterier kan nämnas följande:

- **syntaktisk och semantisk korrekthet**
- **god begriplighet, låg komplexitet**
- **målinriktning, relevans**
- **stabilitet, flexibilitet**
- **hög fullständighet vad gäller täckning av områdets verksamhetsregler**

Vi avstår här från en mer djupgående diskussion av innebörden i, och bakgrunden till, dessa kvalitetskrav för att istället diskutera vilka generella egenskaper hos schemat som är av betydelse i sammanhanget och som alltså kan utgöra utgångspunkt i en kvalitetsdiagnos.

Analyserade egenskaper

I det följande listas ett antal väsentliga och grundläggande egenskaper hos konceptuella schemata, som är, åtminstone i princip, möjliga att bygga in i ett datorstöd. Egenskaperna tänkes utgöra utgångspunkt för ett på heuristiska regler byggt diagnosticeringshjälpmedel, som interaktivt kan användas som stöd vid schemadesign.

De uppräknade egenskaperna är inte oberoende av varandra. Man kan dock dela upp dem i två klasser, en klass för egenskaper hos schemat som sådant, dvs oavsett den avbildade domänen, och en för egenskaper som har med schemats sätt attt avbilda den specifika domänen att göra.

En kort beskrivning ges för varje egenskap tillsammans med en indikation av på vilket sätt en analys av denna egenskap skulle kunna användas i det nämnda datorstödet.

Egenskaper hos schemat oberoende av domän

a) Syntaktisk korrekthet

Med detta avses här huruvida schemat är *syntaktiskt korrekt* och *konsistent*, d.v.s. internt motsägelsefritt. Kontroll kan utföras automatiskt, antingen i samband med att modellen skapas, eller i samband med att någon annan analys utföres. Felaktigheter påpekas.

Till syntaktisk korrekthet kan också räknas *trohet mot benämningskonventioner* inom modelleringsansatsen, i den mån sådana finns. Kontroll av detta kan ske vid behov. Avvikelser påpekas.

b) Presentationskvalitet

Här avses i första hand grafisk kvalitet, dvs sådant som överskådlighet, undvikande av korsande pilar, antal symboler per dm^2, etc. Denna egenskap kan inte avgöras enbart utifrån modellens konceptuella innehåll (CDB i RAMATIC), utan kräver tillgång till representation av grafiken. Ett sätt att styra den grafiska kvaliteten är att låta datorstödet generera bilden på ett optimalt sätt utifrån en 'konceptuell representation' .

c) Modellstorlek

Schemats omfång har naturligtvis betydelse för hur lätt det är att överblicka. Med omfång eller storlek avses här i första hand antalet begrepp, d.v.s. antalet namngivna företeelser. Exakt hur ett sådant värde skall användas eller vilken storlek som är kritisk är dock tills vidare oklar.

d) Objektsamband

Med detta avses antal och karaktär hos samband av olika slag, t.ex. 'attributsamband' och generiska samband. Egenskapen kan analyseras dels genom beräkning av kvantifierbara storheter, som t.ex. antal och frekvens av olika slags samband och dels genom att man detaljgranskar vissa kritiska typer.

Värden på de kvantifierbara storheterna utgör en sorts mått på omfång eller komplexitet. Hur de skall användas är t.v. oklart. Ytterligare en analysaspekt gäller i vilken utsträck-

ning sambanden är obligatoriska eller valfria. Samband som är misstänkta eller kritiska i vissa sammanhang är rena 1-1-förhållanden (1:1,1:1) mellan entitetstyper, ickefunktionella förhållanden (-:M,-:1) eller många till många-förhållanden.

Vad gäller generiska samband kan man tänka sig att verktyget ger förslag till möjliga generaliseringar eller specialiseringar utifrån attributlikhet eller partialitet i attribut resp.

Egenskaper i schemats förhållande till sin domän

Då vi som människor bedömer riktigheten eller kvaliteten hos t.ex. ett konceptuellt schema, gör vi det utifrån den tolkning vi gör av schemat mot bakgrund av kunskap vi har om den begreppsvärld som används. Domänkunskap är med andra ord av central betydelse vid kvalitetsdiagnostik. Framför allt är det kanske fråga om huruvida de begrepp som förekommer i schemat används på det sätt som är normalt inom verksamhetsområdet.

e) Semantisk korrekthet.

Här är det fråga om huruvida schemat avbildar det avsedda verksamhetsområdet på ett korrekt sätt, dvs om det ansluter till vad som är brukligt inom området eller vad som tidigare befunnits/fastställts vara rätt och riktigt. För att detta skall kunna avgöras av ett datorstöd, krävs att domänkunskap, t.ex. i form av tidigare producerade schemata, finns representerad i dess kunskapsbas, liksom att det finns möjlighet att koppla nya begrepp till redan kända. Det senare kan kanske ske i form av generiska kopplingar till de tidigare kända begreppen. Avvikelser från dessa kan då upptäckas och ifrågasättas av verktyget.

f) Fullständighet.

Här gäller det i vilken utsträckning schemats innehåll täcker objektsystemet, dvs hur stor del av verksamhetens regler (business rules) som finns representerade i schemat. Ett möjligt sätt att komma åt detta skulle vara att jämföra schemat med någon annan beskrivning av objektsystemet. T.ex. genom att låta verktyget generera all möjliga utsagor som schemat kan ge upphov till i naturligt språk. Dessa elementära utsagor skulle sedan kunna jämföras med en beskrivning av objektsystemet, lämpligen även den i naturligt språk.

g) Redundans

Det vi i första hand tänker på i detta sammanhang är härledbara attribut, dvs egenskaper och samband i schemat som kan härledas ur andra. I sådana modelleringsansatser som ger möjlighet att uttrycka härledningsregler är det naturligt att beskriva även härledbara attribut. Frågan är dock hur sådan redundans som införts oavsiktligt skall upptäckas.

g) Stabilitet/flexibilitet

Naturligtvis vill man också att införda begrepp skall vara stabila och inte förändras över tiden liksom att det skall vara lätt att införa förändringar när det blir nödvändigt, d.v.s. att en förändring i omvärlden föranleder små förändringar i schemat. Att bedöma stabilitet över tiden förutsätter dock vissa överväganden beträffande den framtida utvecklingen inom verksamhetsområdet. Hur detta skall kunna automatiseras är svårt att se. Rent allmänt torde dock gälla att objekttyper på hög generaliseringsnivå är stabilare än mer specialisera-de. Vid en stabilitetsanalys skulle därför objekt kunna klassificeras efter generaliserings-nivå. De mest specialiserade objekten skulle därefter kunna granskas mer i detalj. Man kan också tänka sig att ha scheman på olika abstraktionsnivåer.

Hög flexibilitet innebär att ändringar och tillägg är lätta att göra. Allmänt kan väl anses att tillägg av ett attribut är en mindre ändring än tillägg av en ny objekttyp. Detta skulle tala för att man redan från början inför objekttyper mer eller mindre överallt där det möjligen kan komma ifråga, t.ex. att man objektifierar M-M-förhållanden och specialiserar eller dekomponerar objekt utöver vad som från början förefaller nödvändigt.

Expertstödets uppbyggnad

Idé och önskade egenskaper

Ett viktigt syfte med analys av kvalitetsbegreppet, sett ur diagnosstödets synvinkel, är att få en grov uppskattning av vilka typer av fel eller tveksamheter som är mest talrika. Med hjälp av denna uppskattning kan man sedan fokusera diagnosen på just dessa aspekter i syfte att avlägsna de värsta felen först. Förutom att detta leder till högre effektivitet vid konstruktion av schemat, innebär det också en kvalitetsvinst för användaren, genom att denne tidigt kan upptäcka att han är på fel spår och därigenom slipper onödigt specificeringsarbete.

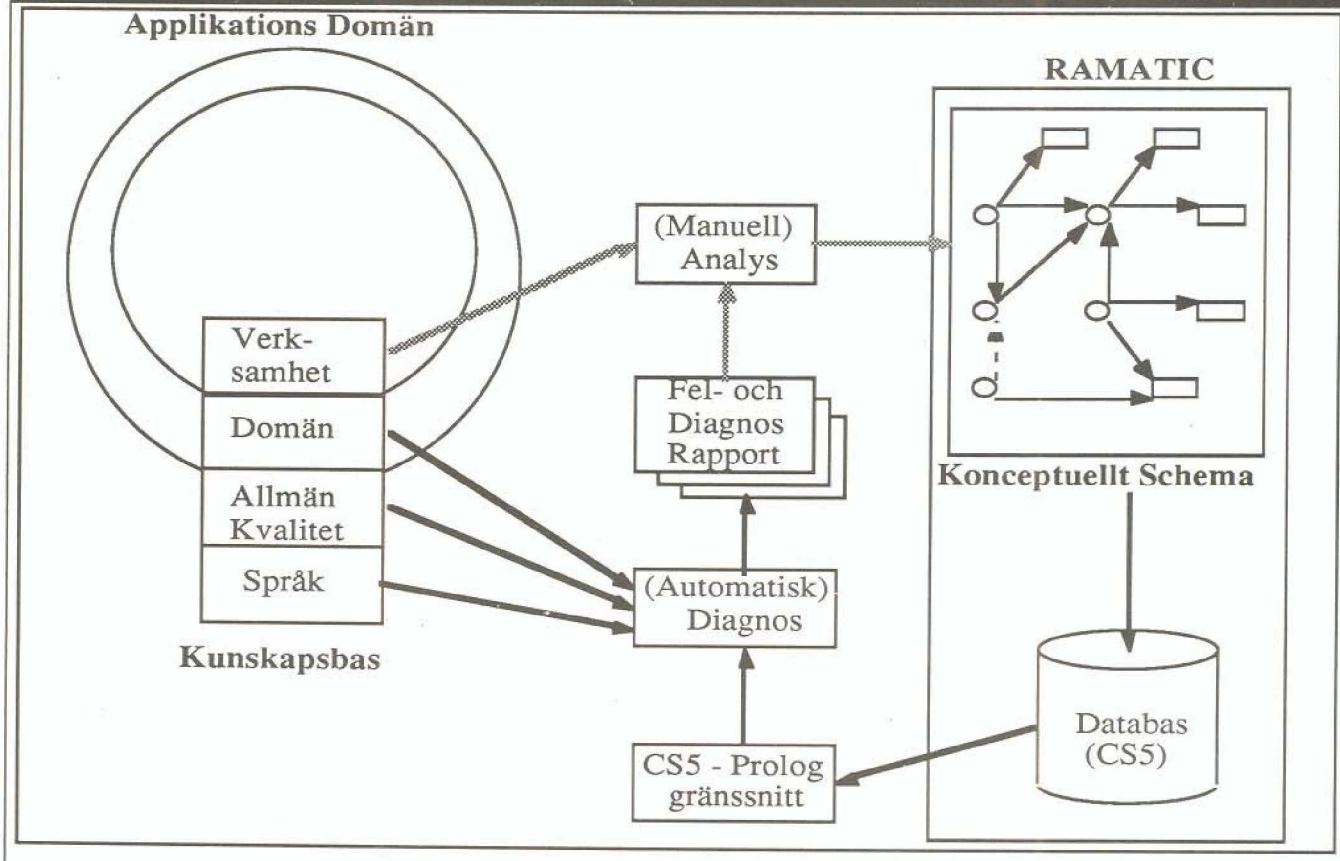
Diagnosticering av schemat tänkes ske utifrån de egenskaper som nämns ovan. Många av dessa låter sig dessvärre inte datoriseras så enkelt. Detta gäller framför allt de som har att göra med kunskaper om domänen. En möjlighet är här att systemet, då det inte har tillräcklig information om domänen, frågar användaren. Redundans i schemat är ett exempel på en sådan egenskap.

Egenskapsanalysen kan i vissa fall utgöra underlag för heuristiska regler som resulterar i förslag till förbättringar. Dessa heurismer kan delas in i grupper, där varje grupp avser en viss typ av misstänkt fel. Varje grupp får sedan en prioritet som beror på hur fundamental den anses vara. Avgörande för detta kan t.ex. vara att man först vill få bort de fel som har de värsta konsekvenserna eller de som är flest till antalet. De grupper av heurismer som har den högsta prioriteten blir sedan de som invokeras först vid diagnosen.

Expertstödets utseende i dag

Expertsystemet är implementerat i Prolog som ett produktionssystem med ett antal regler. Varje heurism motsvaras av minst en regel. För att minska tolkningssvårigheter, har syntaxregler alltid den högsta prioriteten. Domänregler avses kontrollera att de domänspecifika konventionerna följs. Av skäl som nämns nedan har vi, för närvarande inte implementerat denna typ av regler. Allmänna kvalitetsregler representerar kunskap om vad som allmänt kan anses vara bra kvalitet hos ett schema.

Diagnosticering av konceptuella schemata



Vid utvecklingen av diagnosstödet har vi hittills koncentrerat oss på konstruktionen av heurismerna eftersom det är dessa som upptäcker brister i schemat, vilket ju är diagnosens huvudsakliga syfte. Nuvarande implementering läser ett schema ur RAMATIC's databas och översätter detta till Prologpredikat. Utifrån dessa arbetar sedan heurismerna.

I nuvarande version av systemet har vi inte lagt in någon interaktion med användaren. Innan vi gör detta vill vi först lösa problemet att representera domänkunskap, så att detta kan tas med i diagnosen. Bland många tekniker för att bygga upp domänkunskap, studerar vi speciellt induktion som ett medel att åstadkomma hierarkiska kunskapsstrukturer. Dessa kan i sin tur användas för att applicera generell domänkunskap på ett specifikt schema.

Reglerna i expertsystemet har en relativt restriktiv syntax. Det har fördelen att även andra än systemkonstruktören kan förstå dem, eftersom man inte behöver ha någon djupare kunskap om diagnossystemets uppbyggnad. Som exempel kan vi ta en regel som upptäcker ett syntaxfel:

```
"här skall det stå en lämplig text att skriva ut till användaren"
Rule Id A23'
  If
        Name Of Attribute Is Containing_upper_case_letter
  Then
        Name Of Attribute Violates_this_rule.
```

Den här regeln kommer att upptäcka alla attributnamn som innehåller en eller flera stora bokstäver och etablera detta som faktum i diagnosen.

Naturligtvis kan det också vara motiverat att etablera andra fakta än sådana som rör rena felaktigheter. Det kan t.ex. vara så att vissa företeelser i schemat förutsätter andra. Detta innebär att en slutsats i en regel måste ingå som villkor i en annan. Sådana kedjor av regler medför att också mer komplicerade kontroller kan utföras. I framtida versioner av verktyget kan slutsatsdelen även tänkas innehålla operationer som t.ex. att rita förslag till användaren på dennes skärm.

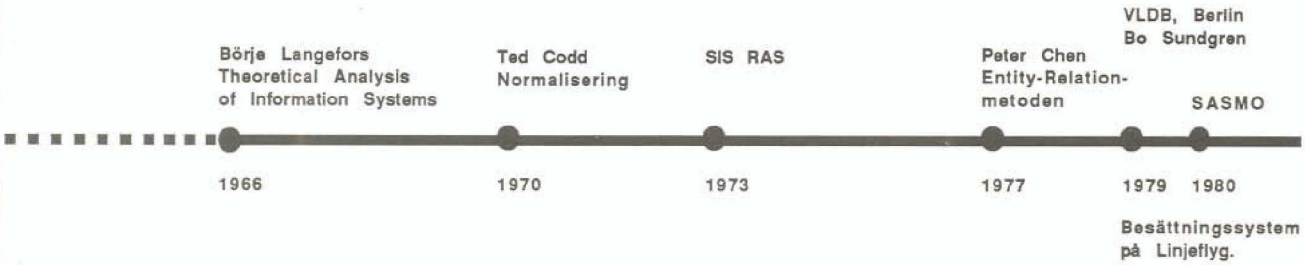
Slutsatser

Det är vår uppfattning att verktygsanknutna analys/diagnoshjälpmedel av den typ som här diskuteras är möjliga att konstruera och att de kan vara till stor hjälp vid utveckling av informationssystem. Särskilt viktig anser vi den semantiska aspekten vara. Dessvärre är det också den som ställer sig svårast att implementera på ett från databehandlings- och användningssynpunkt effektivt sätt. Att lösa detta problem är en betydande utmaning för framtiden.

Som nämnts har vårt arbete hittills varit inriktat mot konceptuell modellering. Vi tror dock att mycket av våra resultat kan generaliseras eller modifieras att gälla också andra slag av specifikationer, t.ex. verksamhetsbeskrivningar eller programspecifikationer. Naturligtvis spelar även här den använda formalismen, t.ex. MBI eller ISAC för verksamhetsgrafer, eller "Structured Design" för programspecifikationer, en stor roll. Programstrukturdiagram enligt "Structured Design" är f.ö. ett av de få områden där det finns något så när välformulerade och väletablerade kvalitetsregler.

I det fortsatta arbetet med det ovannämnda SISU-projektet kommer vi, förutom konceptuella schemata, också att penetrera ett mindra antal typer av systemspecifikationer, liknande de som nämns närmast här ovan.

IRM Consult säljer stadsplaneringsidén



Vad är det som håller på att hända ute bland de ADB-användande organisationerna? Vilka behov och möjligheter som informationsteknologin skapar är viktiga?

Informationsteknologin som konkurrensmedel

Eskil Swende ger några exempel:

American Hospital Supply
Ett mindre, tillverkande företag, som genom att sätta ut terminaler på sjukhusen fick kunderna att även beställa andra företags produkter via systemet. Idag är företaget USA´s största grossist för sjukvårdsartiklar.

Citibank
Tidigt ute med världsomspännande kommunikationsnät. Idag är man helt dominerande inom världens valutahandel. Det finns alltid en öppen valutahandel någonstans i världen tillgänglig genom deras system. Därigenom kan banken handla valuta 24 timmar per dygn.

American Airlines
Genom att under tio års tid investera över 2 miljarder SEK i världens mest sofistikerade bokningssystem har man nått en marknadsandel på 45 %. Systemet får användas av andra flygbolag mot en väl tilltagen avgift och självklart får de egna flygningarna en framträdande plats. Dominansen hotar nu andra flygbolag. Under senaste året tjänade bolaget mer pengar på att sälja sina ADB-tjänster än på sin egen flygverksamhet.

Samarbetet mellan SAS, Lufthansa, Air France och Iberia i *Amadeusprojektet* skall ses mot den bakgrunden. Hotet utifrån får arga konkurrenter att samarbeta för att försöka undvika att hamna i samma situation som de amerikanska flygbolagen. Det är fråga om miljardinvesteringar som de enskilda flygbolagen inte anser sig kunna bära själva.

Wasa
Wasa lanserade nyligen en ny pensionsförsäkring, där storleken kan bestämmas när försäkringstagaren vet hur mycket pengar han får över. Tidigare innebar pensionsförsäkring ett mycket långsiktigt och stelbent åtagande. Produkten ställer helt nya krav på ADB-systemen. Konkurrenterna klarade inte att snabbt kunna kopiera produkten.

Reuter
Reuter är troligen den som tjänar mest på valutahandeln. Reuters ADB-system används av bankerna i deras handel. Reuter har kunnat uppnå monopol i Sverige och kan styra priset. Andra företag har otroligt svårt att konkurrera. Reuter bygger barriärer genom att binda sina kunder med flerårskontrakt.

Volvo
I ledarskap dec -87 redovisas att Volvo behövde sju år för att utveckla en ny bilmodell. Honda klarar det på fyra. Den som vill vinna näst rond i den matchen måste utnyttja informationsteknologin effektivare. Det är genomgående för all tillverkningsindustri att produkternas livslängd minskar. Det blir allt viktigare för lönsamheten att komma först med en ny produkt.

Kontentan av dessa exempel är att företag som utnyttjar informationsteknologin har ett redskap för konkurrens och affärsutveckling. För den som inte utnyttjar möjligheterna blir informationsteknologin ett hot.

IRM Consult säljer stadsplaneringsidén

Artikeln baseras i stor utsträckning på artiklar och annan skriven dokumentation från IRM Consult samt på samtal med Eskil Swende. VD i IRM Consult AB och kontaktman för SISU. - Mycket av IRM Consults idéer och argumentation för dessa uppfattar jag som allmänt giltiga för den som vill sprida budskapet om nya synsätt och metoder i sin organisation. - Eftersom IRM Consult konsekvent framför sitt budskap på ett målmedvetet pedagogiskt sätt har det känns motiverat att försöka ge en relativt utförlig resumé av materialet.

Lars Bergman, SISU

Fakta om IRM Consult

Kunder
Kunderna återfinns bland de stora börsbolagen, t.ex. Volvo, SKF, Ericsson, Saab-Scania, men också på statliga verk som Televerket, Tullverket och Luftfartsverket.

Omsättning
1985: 3 milj SEK
1987: 5,4 milj SEK

Anställda
10 personer varav 8 konsulter.

SVEA-modellen utges av Kommundata

1982 1983 1985 1987 1988 1990 2000

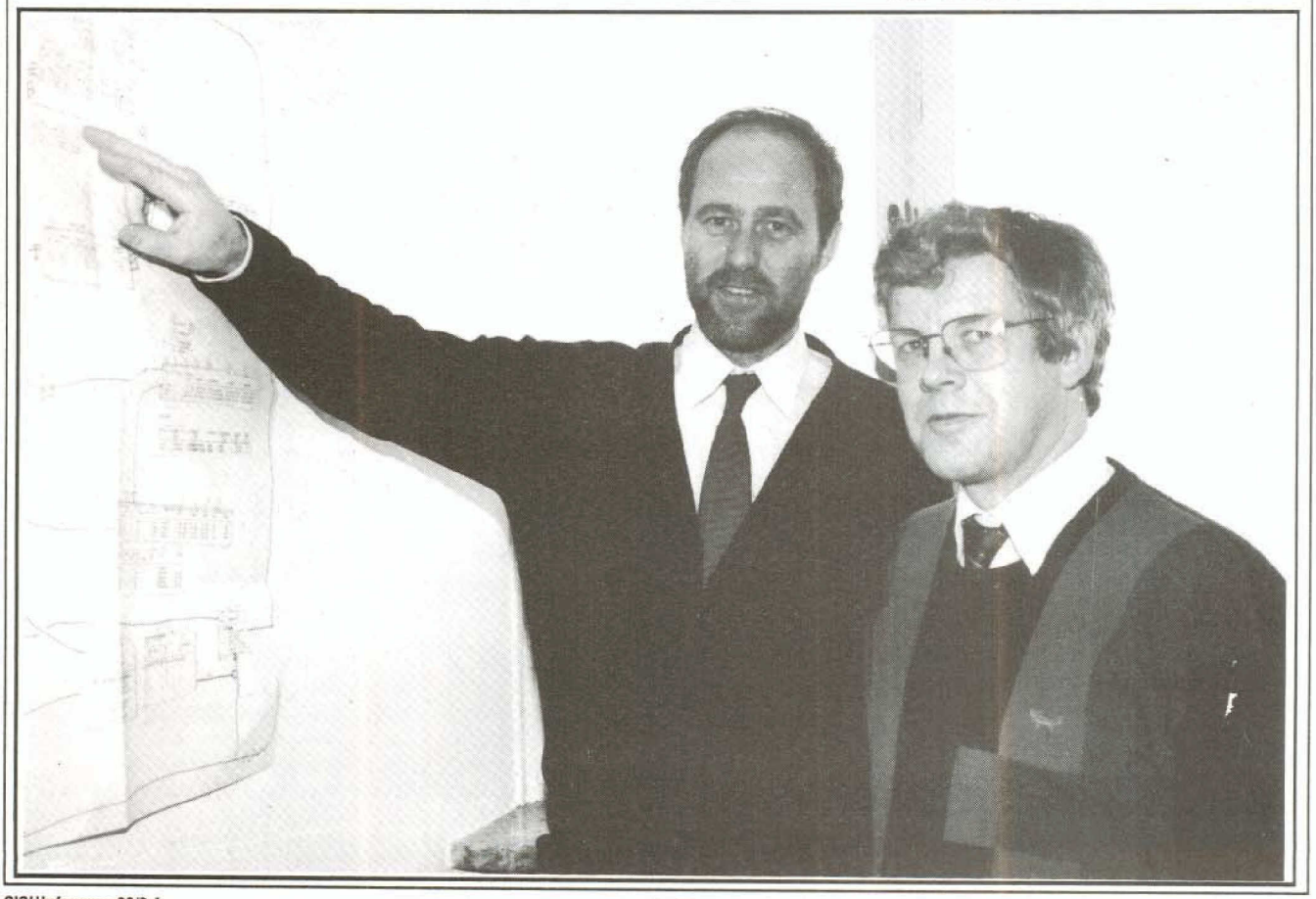
IRM Consult AB bildas

Eskil Swende, Claas Åkesson och Rolf Edgren, alla tre från SAS, nu på heltid i IRM-Consult.

Lars Axelsson kommer från Kommundata

Idag ett 10-tal anställda varav 8 konsulter.

Claas Åkesson pekar ut en del av stadsplanen från ett aktuellt uppdrag för Eskil Swende, VD i IRM Consult AB och kontaktman till SISU.



IRM Consult säljer stadsplaneringsidén

Informationsresursen – en ledningsangelägenhet

Eskil nämner SAS, Televerket och Posten som tre organisationer där man lyft upp styrningen av informationresursen till ledningsnivå. Man har skapat vad som i USA kallas CIO, Chief Information Officer. Denna styrning kan inte utföras av ermer eller mindre bolagiserad dataavdelning. Man ska inte vara både stadsplanerare och byggmästare anser Eskil.

I grunden för IRM-Consultkonceptet ligger amerikanen Ted Codd's normaliseringsteori som omsattes till praktisk tillämpning på SAS med hjälp av professor Bo Sundgren.

Grundidén bakom planeringen är att betrakta information som en resurs precis på samma sätt som andra resurser t ex personal, material och pengar. En resurs ska anskaffas, lagras, behandlas och användas samt avvecklas när den inte längre behövs. Vi försöker styra resursen genom att planera och följa upphanteringen av den i verksamheten. Vi kallar detta synsätt IRM, Information Resource Management.

En mognadsprocess

Den amerikanske professorn Richard L Nolan vid Harvard Business School, har presenterat en modell. Den beskriver olika stadier i ADB-verksamhetens mognadsprocess.

Många organisationer i Sverige liksom utomlands kan sägas ligga i övergångsskedet från integrationsfas till dataadministrationsfas.

Integrationsfasen kännetecknas av att man när man bygger nya system försöker utnyttja data som redan finns i andra system genom att göra olika kopplingar. Man knyter så ihop system "ad hoc" utan långsiktiga hänsyn och planering. Efter hand blir systemstrukturen mycket komplex och därigenom systemen svåra att ändra och underhålla.

Dataadministrationsfasen, den femte fasen, kännetecknas av att information ses som en gemensam och värdefull resurs som skall styras på samma sätt som andra resurser.

Övergången aktualiserar behovet av att skapa en överblick och utgångspunkt i verksamheten och dess information och planera för infrastrukturutveckling och för strukturförändring i systemportföljen.

ADB-verksamhetens situation idag

Vad menar du med "spagettisyndromet"? - En konsekvens av bristande samordning är vad vi kallar spagettisyndromet. Det vill säga att olika system efter hand har kommit att kopplas samman på ett så oöverskådligt sätt att nästan ingen har överblick över systemstrukturen. Spagettin har många besvärliga konsekvenser. Det är kostsamt, resurskrävande och svårt att ändra i systemen. En stor del av informationen som finns lagrad är ändå inte tillgänglig när den behvös.

Teknologins utveckling de senaste 10 åren

En annan faktor som påverkar utvecklingen kraftigt inom området är teknologins utveckling med dess åtföljande möjligheter. Eskil Swende ger sin uppfattning såhär:

För tio år sedan var

- kommunikationsnäten dåligt utbyggda och standardiserade och det var därför svårt att överföra data från ett system till ett annat
- det fanns inga bra databashanterare som gjorde att samma data enkelt kunde användas i flera system
- data-dictionary, dvs ett verktyg som höll reda på var data fanns och hur data skulle tolkas, fanns inte heller
- programmeringsspråken, som Cobol och Fortran, var bara lämpliga för dataspecialisterna.

Idag är situationen snabbt på väg att förändras

- utvecklingen på kommunikationssidan är snabb både vad gäller standard och prestanda
- på databassidan har vi fått relationsdatabaser som är flexibla när det gäller att kombinera data på nya sätt
- data-dictionary har utvecklats även om det är en bra bit kvar innan vi kan få fram bra datakataloger, som är bra för användaren
- de s.k. 4:e generationens språk har drastiskt underlättat för icke specialister att utföra sin egen databehandling.

En ny infrastruktur behövs - med en god stadsplan som grund

VD har ett ansvar för uppbyggnaden av företagets infrastruktur i informationssamhället. Detta ansvar kan inte bara delegeras till dataavdelningen eller decentraliseras till användare. Däremot behöver man en funktion som hjälper till med planering och genomförande av en infrastruktur. Dataadministration är det etabelerade namnet på en sådan funktion, men vi anser att ordet "administration" dåligt beskriver vad det är fråga om. Det engelska ordet management anger mycket bättre vad det gäller. Att medverka till uppbyggnaden av en ny infrastruktur är inget passivt administrationsjobb utan kräver en synnerligen aktiv styrning med företagsledningens fulla stöd om det skall lyckas. Denna funktion, IRM-funktionen, skall koncentrera sig på att planera data och kommunikation, som kräver lång framförhållning.

Stadsplanen är, enligt IRM-Consult, det gemensamma underlaget för att bygga en ny infrastruktur för verksamhetens informationsbehandling. Infrastrukturen skall vara sådan att den gör informationen tillgänglig och underlättar affärsutvecklingen i verksamheten.

Normalt kan de som arbetar med affärsutveckling för lite om informationteknologin, och de som kan teknologin vet för lite om affärsidéerna. Sätt dem tillsammans i en arbetsgrupp och låt dem kläcka idéer, studera andra företag och lära av varandra. Testa idéerna i mindre skala, etablera utvecklingsprojekt med begränsade försök, utveckla vidare.

IRM Consult – ett kunskapsföretag?

Är kunskapsföretag rätt etikett? Vi utvecklar, tillämpar och lär ut våra metoder. Däremot sysslar vi inte med "resursförstärkning". Vår verksamhet är nog just kunskapsförstärkning, så kunskapsföretag tycker jag stämmer bra med min bild av vad vi är.

Vi jobbar med seminarier och med projektinsatser. I seminarierna tillämpar och utbildar vi i arbetsmetodik och synsätt samtidigt som vi arbetar fram konkreta resultat i samarbete med kunden. Det gäller också när vi utbildar personer från kundorganisationen till att fungera som interna handledare och metodstödjare. När det gäller projektinsatser är dessa lika fördelade på SVEA-uppdrag i enskilda projekt och IRMA-uppdrag för stadsplanering.

IRM Consult säljer seminarier och utbildar instruktörer i kundföre-

IRM Consult säljer stadsplaneringsidén

tagen. Till seminarierna samlas användare som 'kan' verksamheten. Tillsammans bygger man en datamodell kring begrepp som inte förändras med verksamheten. Fasta begrepp kan vara produkt, kund, konto, flight, artikel, etc. Vi jämför dataplaneringen med hur samhället planeras. Idag är det självklart med en genomtänkt infrastruktur, att man lägger bostäder i vissa områden, planerar var vägar skall dras och hur folk skall transporteras. Om du bygger ett hus får du el och vatten från samhällsnätet.

Tre produkter utgör utbudet i huvudsak. IRM-metodiken med SASMO i centrum utgör den traditionella basen i verksamheten. Till detta har lagts "stadsplanering" enligt IRMA. Nu har också SVEA-metodiken förstärkts genom att Lars Axelsson gått över till IRM Consult.

Metoden datamodellering (SASMO) har nu använts ca 300 gånger i ett 50-tal olika skandinaviska verksamheter. Den ingår som del i SVEA- och IRMA-metodiken.

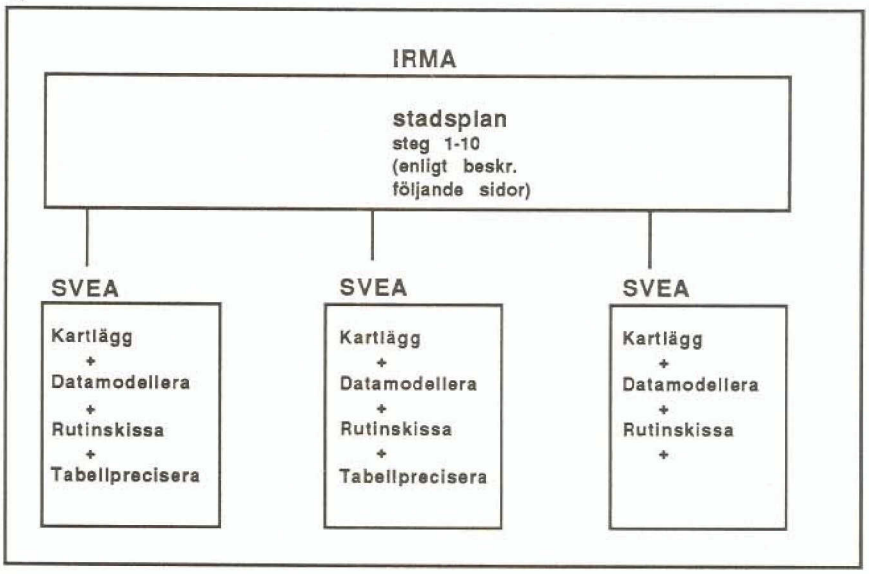
SVEA-metodiken sätts an för utveckling av enskild tillämpning. Den kommer alltså in när planeringen skall realiseras i form av nya system.

IRMA

Det finns en beprövad metodik, IRMA, Information Resource Management Architecture, för att planera och genomföra uppbyggnaden av infrastrukturen. Den utgår från en noggrann genomgång av själva data och betecknas därför som "data-driven". Därigenom uppnås en stabilitet i strukturen som är nödvändig, dvs den ändras inte av omorganisationer eller nya rutiner.

När själva kartan är ritad är det dags att fundera på hur stadsplanen skall se ut och hur den skall förverkligas. En metod för detta; IRMA, har utvecklats av IRM Consult och praktiserats vid ett tiotal företag. Arbetet bedrivs som ett projekt under 3 månader med deltagande av personer som väl känner verksamheten. ADB-folk deltager med sin kunskap om befintliga system.

Arbetsstegen i IRMA-metoden presenteras på de följande sidorna med exempel från posten Helsingborg, ett projekt inom ramen för Postens informatikstrategi.



"Styrkan i IRM Consults affärsidé är kombinationen SVEA/IRMA. Att kunna följa stadsplanen ända ner tilldatabas och program. En datamodell av hög kvalité är grunden till effektiv informationsbehandling."

SVEA finns beskriven i boken: Utvecklingshandboken, SVEA, Strukturerad Verksamhetsinriktad Arbetsmodell, utgiven av Kommundata.

Kritiska framgångsfaktorer för IRM-arbetet

Det finns idag erfarenheter från IRM och datamodellering från ett 50-tal verksamheter i Sverige, däribland 27 av Sveriges 75 största företag. SAS och Linjeflyg var pionjärer på området med Vattenfall och Volvo PV som tidiga efterföljare.

Utifrån sina erfarenheter formulerar Eskil följande checkpunkter för den som skall arbeta enligt IRM-konceptet:

Problemmedvetandet
Problemmedvetandet varierar starkt men ökar efterhand som spagettistrukturen förvärras och användars krav på snabb information ökar.

Eldsjäl på data- eller användarsida
I de företag som hittills startat har det funnits en eldsjäl antingen på data- eller användarsidan. I de företag som nu kommit en bit på väg har användare och datafolk bestämt sig för att arbeta gemensamt.

Ledningens engagemang
Hittills är det bara i några få företag som ledningen verkligen engagerat sig. Det beror nog på ett bristande problemmedvetande. En tilltrasslad spagettistruktur är inget som syns vid ett besök i datorhallen på det sätt som t.ex. en dålig materialstyrning syns på lager och verkstadsgolv.

Ideförsäljningen är avgörande.
För att lyckas är idéförsäljningen en avgörande faktor. Här kan man förstås dra nytta av erfarenheter från andra företag, men det är nog viktigare att skapa ett problemmedvetande med utgångspunkt i den egna verksamhetens problem. Dessutom att samtidigt skaffa egna erfarenheter av data-driven utveckling enl. SVEA från begränsade utvecklingsprojekt. Vi rekommenderar då starkt att *börja med ett operativt system* och inte ett ekonomi- eller statistiksystem. Anledningen till det är att operativa data skall lagras i stabila datalager, som sedan kan användas för ekonomiska och andra analyser, och då är det viktigt att börja i rätt ända.

Den teknologiska utvecklingen stöder
Den teknologiska utvecklingen stöder en IRM-lösning även om det fortfarande finns en del brister vad gäller t.ex. informationskatalog och data-dictionary vilka berörts tidigare.

Tålamod
Det absolut viktigaste för att lyckas! Massor av tålamod!

Arbetsstegen i IRMA

1 2

Illustrationerna har hämtats från ett projekt vid Posten i Helsingborg, med godkännande från Postens informatikstrategienhet. Tack!

IRMA -tillämpningar

Eurocard, Götabanken, KTAS, PK-banken, Postverket, RKA, SAS, SSAB. Statoil, Televerket och Åke Larsson.

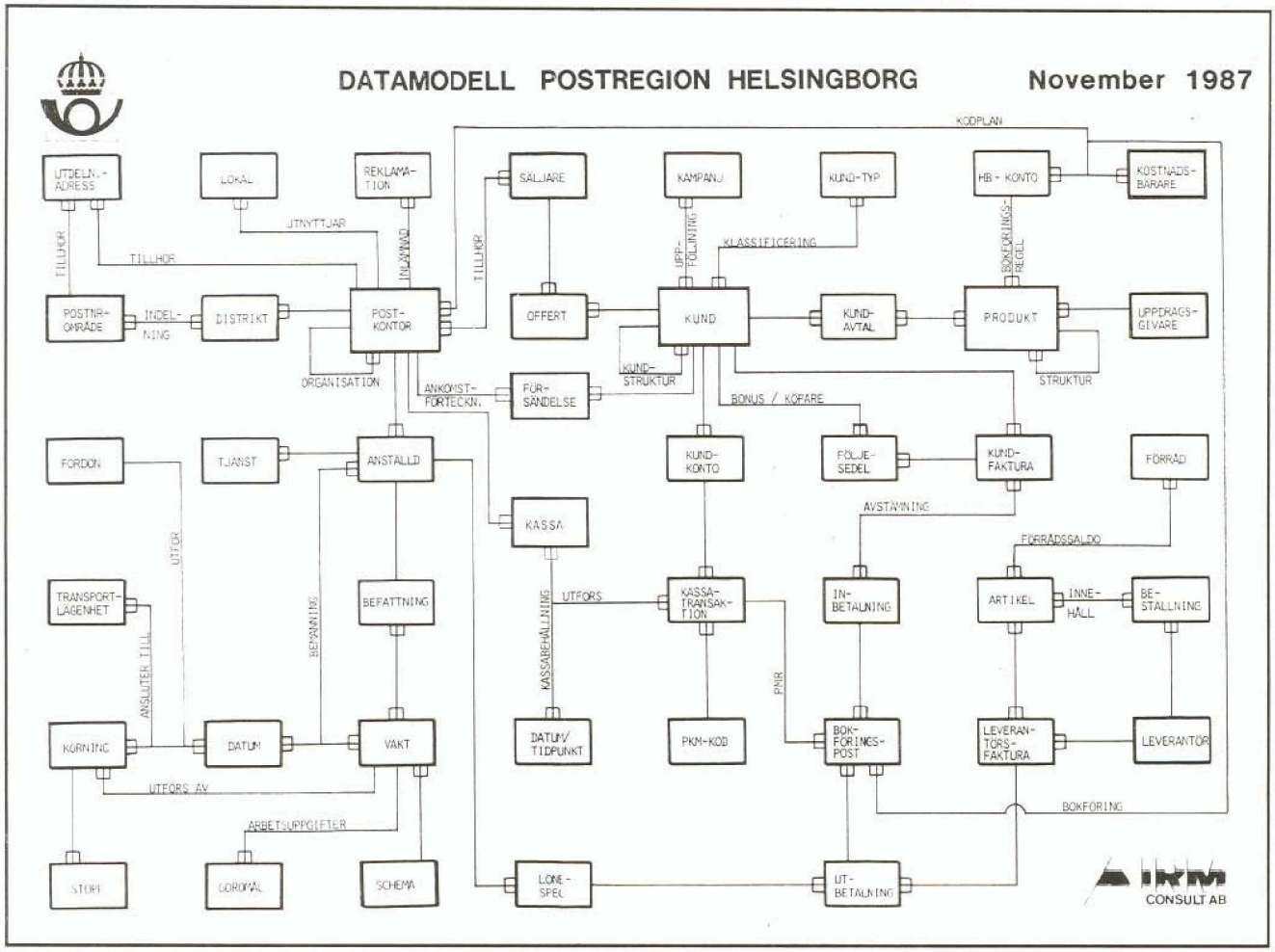
1. Förankra och planera

Innan arbetet startas är det viktigt att det har en ordenlig förankring hos ledningen, verksamhetsfolk och dataavdelning och att en projektledare utses.
Därefter detaljplaneras arbetet. Deltagare utses. Ett tiotal personer med god kunskap om den operativa verksamheten och några personer från dataavdelningen. Aktiviteterna tidsplaneras och datum för presentation av slutrapporten fastställs ca tre månader från start.

2. Datamodellering

Grunden för stadsplanering är en så kallad datamodell som utarbetas under ett tvådagars seminarium med samtliga deltagare.
Då etableras en enhetlig begreppsapparat och sambanden mellan data i verksamheten beskrivs. Denna beskrivning är stabil och påverkas inte av förändringar i verksamhetens organisation eller rutiner.

Datamodell



IRM Consult säljer stadsplaneringsidén

3

3. Funktioner

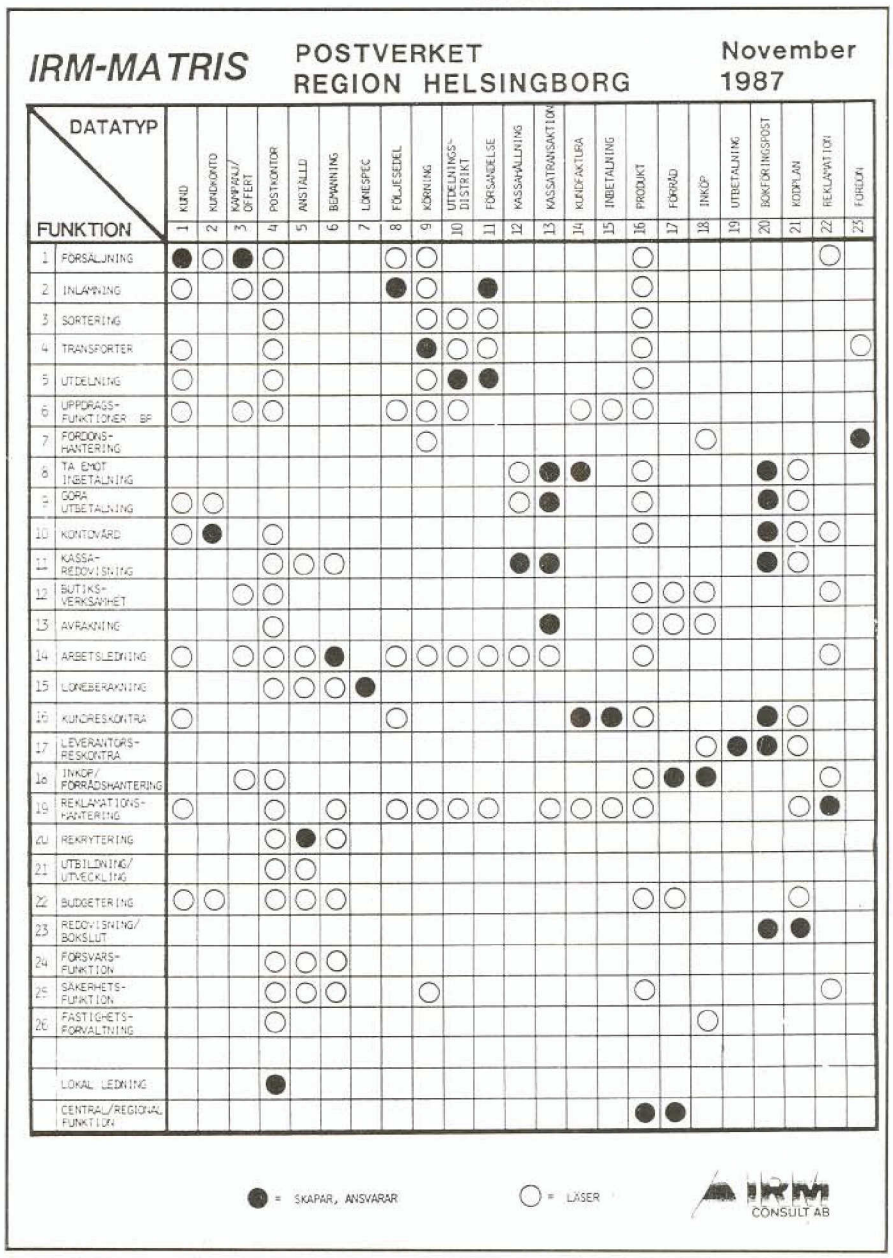
Här indelas verksamheten i ett tjugotal huvudfunktioner med tyngdpunkten på den operativa verksamheten.
Det är i de operativa funktionerna som data uppstår och här som ansvaret för att anskaffa data bör placeras. Grundregeln är att anskaffa data vid källan.

4

4. IRM-matris

En matris där raderna utgöres av funktionerna och kolumnerna utgörs av data från datamodellen. Den utarbetas också i seminarieform där samtliga deltagare medverkar aktivt. *IRM-matrisen utgör sedan själva kartan,* där det framgår vilka data som används i varje funktion. Här fastställs också dataansvaret; det vill säga vilken funktion som har ansvar för olika data om t.ex. kunder, produkter, konto.

IRM-matris



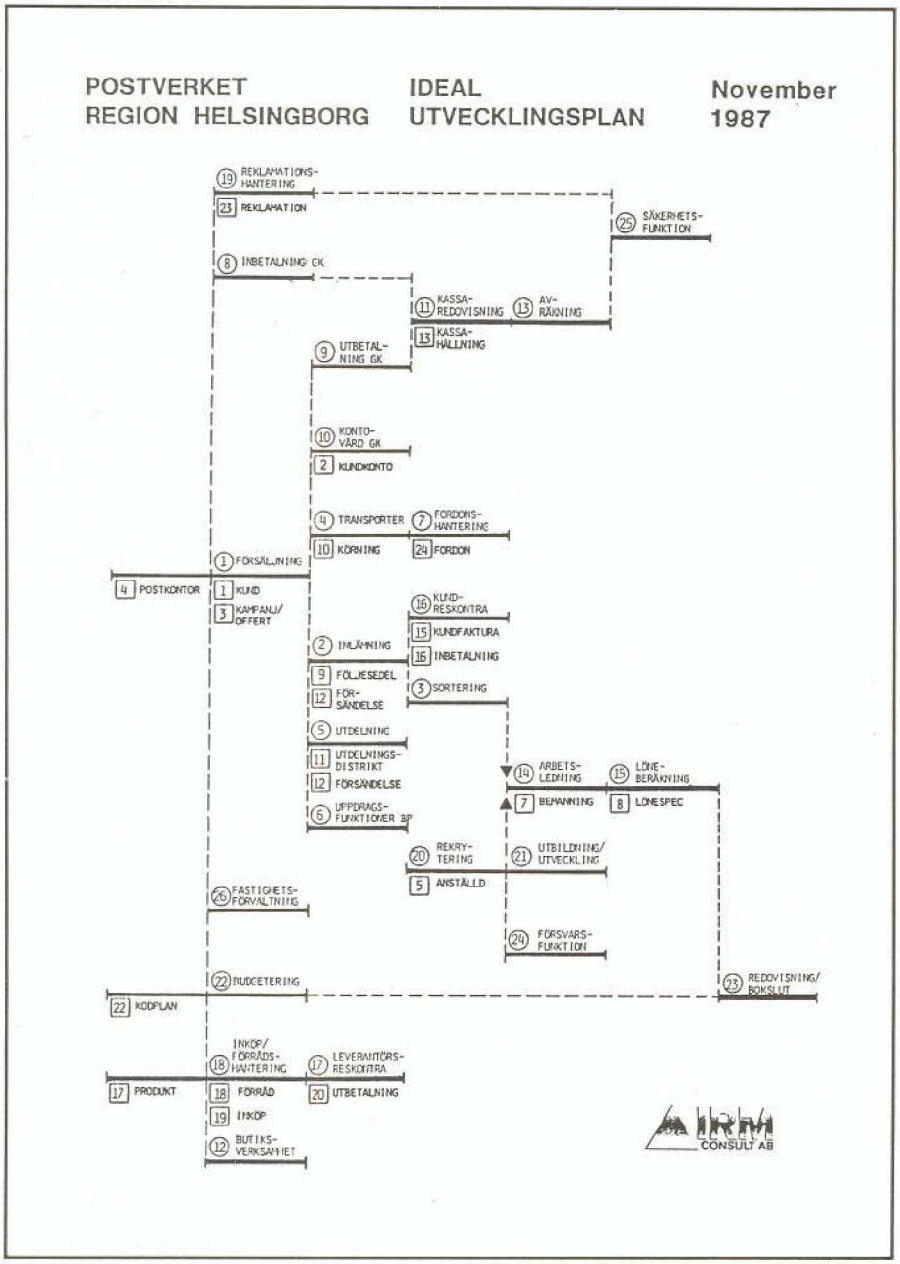
IRM Consult säljer stadsplaneringsidén

5

5. Ideal utvecklingsplan

Baserat på en analys av IRM-matrisen utarbetas sedan en ideal utvecklingsplan. Den beskriver i vilken ordning system och databaser skulle utvecklas, om alla system var lika viktiga och vi inte behövde ta hänsyn till de befintliga system som finns.

Ideal utvecklingsplan



6

6. Kritiska faktorer

Nu är det dags att engagera ledningen igen. Detta görs under ett halvdagsmöte, som inleds med en statusrapport där arbetet fram till idealplanen presenteras.
Därefter utarbetas tillsammans med ledningen en lista över kritiska faktorer (KFF). Det vill säga de faktorer som är kritiska för verksamhetens framgång.
Dessa faktorer utgör sedan grunden för att bedöma hur den ideala planen eventuellt behöver justeras.

IRM Consult säljer stadsplaneringsidén

7

7. Nuvarande system

Nu görs ytterligare en matris, där raderna utgörs av respektive system och kolumnerna , som tidigare, utgörs av data från datamodellen. Denna matris visar att samma data behandlas likartat i flera system; d.v.s. visar på den bristande samordningen i systemstrukturen. Matrisen är också grunden för att planera avvecklingen av strukturen. Det vill säga vilka system, som berörs när nya databaser och system skal utvecklas.

Nuvarande system

IRM-MATRIS POSTVERKET REGION HELSINGBORG November 1987

SYSTEM \ DATA-TYPER	1 KUND	2 KUNDKONTO	3 KVITTENS/ OFFERT	4 POSTKONTOR	5 ANSTÄLLD	6 BEFATTNING	7 LÖNESPEC	8 FÖRSÄLJNINGSDEL	9 KÖPNING	10 UTDELNINGS-DISTRIKT	11 FÖRSÄNDELSE	12 KASSAHÅLLNING	13 KASSATRANSAKTION	14 KUNDFAKTURA	15 INBETALNING	16 PRODUKT	17 ARTIKEL	18 INKÖP	19 UTBETALNING	20 BOKFÖRINGSPOST	21 KODPLAN	22 REKLAMATION	23 FORDON
1 SAFE	○									●													
2 DAFO				○													○	●		○			
3 ADE-EKO	●	●	●	●				●						●	●			○	●	●	○		
4 PKM				○								●	●			○							
5 PDB				●	●	●																	
6 BER				●	●		●																
7 RAPP				●	●	●																	
8 PBA				●		●	○													●	●		
9 BEF				●		●																	
10 VALBORG				●	●	●																	
11 START				●	●																		
12 PERSONALSTATISTIKSYSTEM				●	●		○																
13 POSTKONTORS-REGISTRET				●																			

● = SKAPAR, UPPDATERAR ○ = LÄSER

IRM CONSULT AB

8

8. Rapporten

Nu är det dags att sammanställa stadsplanerapporten och diskutera fortsatt arbete. Dessa förslag rör sådana aktiviteter som att etablera funktion för att genomföra stadsplanen, vilken teknologi ska användas (datarer, databashanterare, kommunikationsnät etc), hur ska säkerheten hanteras och hur / när ska det första projektet startas.
Rapporten diskuteras och godkänns av samtliga deltagare.

9

9. Avrapportering och beslut

Arbetet avrapporteras för ledningen och beslut fattas om det forftsatta arbetet.

10

10. Idéförsäljning

Stadsplanearbetet och tankarna bakom innebär förändringar mot dagens sätt att arbeta. Att informera även personal, som inte direkt deltager i arbetet, är därför en viktig aktivitet under hela arbetet.
Det är också så att information är en abstrakt resurs och därför svår att hantera och beskriva. Så fort konkreta resultat uppnås ges därför nya möjligheter att bättre föra ut budskapet i verksamheten.

Objektorienterad systemutveckling - en översikt

Inledning

Objektorienterade ansatser har visat sig vara kraftfulla hjälpmedel för specificering, konstruktion, implementering och underhåll av informationssystem. Många nya metoder, tekniker och verktyg baserar sig därför på objektbegreppet. Tyvärr tillämpas dessa ansatser dock än så länge i begränsad omfattning. En oklar begreppsapparat försvårar dessutom förståelsen för området.

Föredrag vid NordData88 av
Matts Ahlsén, SISU
Stefan Britts, SISU

Artikeln belyser "objektorienterad systemutveckling" inom ramen för en enkel begreppsapparat. Exempel på tillämpningar av det objektorienterade synsättet ges från olika områden såsom programmering, databashantering, konceptuell modellering och artificiell intelligens. Vidare belyses aktuella utvecklingstrender samt några för- och nackdelar med det objektorienterade synsättet.

Viktiga begrepp

Någon formell definition av begreppet objektorientering existerar inte. Däremot finns det ett antal centrala begrepp som i olika omfattning förekommer inom områden där en objektorienterad ansats tillämpas. Vi skall här försöka ge informella definitioner av dessa begrepp. Definitionerna bygger på etablerad terminologi från forskningen inom området. Avsikten med definitionerna är att ringa in de viktigaste egenskaperna hos dessa ansatser. Vi vill understryka att alla begrepp ej behöver vara representerade för att en ansats skall kunna kallas objektorienterad. Listan av begrepp är heller inte fullständig. Vissa begrepp belyses dessutom med hjälp av några förenklade exempel.

Objekt

Objektbegreppet förekommer i många och varierande betydelser inom allti från systemutvecklingsmetodik, programspråk, operativsystem till datorarkitektur. Det objektbegrepp vi här främst avser har sitt ursprung från programmeringsmetodik, speciellt sk "objektorienterad programmering". Vi skall senare titta närmare på vad denna metodik specifikt innebär. Först görs emellertid en generalisering av objektbegreppet. Ett "objekt" är en logisk kontext eller *modul* ("behållare") dels för information med en viss *struktur* (data) dels för ett *beteende* enligt bearbetningsregler (operationer). Objekt har vidare ett internt tillstånd eller *minne* och skall kunna ha en *oberoende existens*, t ex i en datamodell eller i ett programspråk. Det utgör en slags grundläggande byggklots eller komponent. Förutom modul förekommer också individ och entitet som synonymer till objekt.

Objekt

```
OBJEKT Brev =
    VARIABLER författare, innehåll : Text; postat : Datum;
    OPERATION Editera(); innehåll := "Hej";

Brev.innehåll := "Hej";
Brev.Editera();
```

```
OBJEKTTYP Brev =
    INSTANSVARIABLER författare, innehåll : Text; postat : Datum;
    OPERATION Editera(); innehåll := "Hej";

VARIABLER x, y, z : Brev;

x.innehåll := "Hej";
x.Editera();
```

Objekttyper och klasser

Objekttyper och klasser

Vill vi dynamiskt kunna skapa och förstöra godtyckligt många förekomster av objekt med liknande egenskaper, kan vi generalisera objektbegreppet genom att skilja på ett objekts beskrivning och en viss representation eller förekomst av det. Beskrivningen av beteende och struktur utgör en *objekttyp* utifrån vilken förekomster eller instanser av objekt kan genereras. Mängden av objekt skapade enligt en given objekttyp sägs utgöra en *klass*. Med en klass avser vi alltså en mängd instanser av någon typ emedan vi med objekttyp menar egenskaperna hos objekten i denna mängd. Dessa två begrepp används dock ibland synonymt. Andra synonymer är entitetstyp, metadata och mönster.

Inkapsling

Inkapsling eller dataabstraktion. Att "abstrahera" innebär möjligheten att kunna bortse från detaljer som anses vara irrelevanta för de egentliga egenskaperna hos en viss företeelse. Vid utveckling av informationssystem är naturligtvis detta synnerligen viktigt. Dataabstraktion är också den egenskap som kanske mest präglar objektorientering. Det betyder att vi främst är intresserade av ett objekts beteende och vad vi kan göra med det, snarare än hur detta beteende är realiserat och hur objektets interna struktur ser ut. Med detta synsätt kommer en objekttyp att bestå av två delar, dels ett *gränssnitt* som definierar beteendet hos medlemmar av objektklassen (objektens "abstrakta egenskaper") i form av en uppsättning operationer, samt en *realisering* i termer av struktur och bearbetningsregler. Gränssnittet kallas objekttypens *specifikation*, och realiseringen dess *implementering*. Specifikationen utgör sålunda kontaktytan mot det omgivande systemet, dvs andra objekt samt användarna. Om språket omöjliggör åtkomst till interna datastrukturer annat än via gränsnittets operationer talar man om inkapsling. Inkapsling kan kombineras både med objekt (moduler) och typer. I det senare fallet erhåller vi abstrakta datatyper.

Inkapsling

```
OBJEKTTYP Brev =
    SPECIFIKATION
        OPERATION Editera();
    IMPLEMENTERING
        INSTANSVARIABLER författare, innehåll : Text; postat : Datum;
        OPERATION Editera(); innehåll := "Hej";

VARIABLER x, y, z : Brev;

x.Editera();
```

Egenskapsarv

Objektklasser kan vara överlappande vilket innbär att ett visst objekt kan ingå i flera klasser. Detta medför att objekt kan ha egenskaper från mer än en objekttyp. Ett vanligt sätt att åstadkomma detta är att relatera flera objekttyper genom egenskapsarv. Detta utgör ett kraftfullt hjälpmedel för att strukturera informationssystem genom successiva specialiseringar av objekttyper. Samtidigt som detta ger möjlighet till återanvändning av existerande beskrivningar kan egenskapsarv också ses som ett grundläggande modelleringskoncept baserat på specialisering/generalisering. Mer specifikt sägs två objekttyper A och B ha relationen *supertyp - subtyp*, om B ärver egenskaperna hos A. Om B också tillför ytterligare egenskaper utgör den en *specialisering* av A.

En subtyps objektklass utgör en delmängd av sin supertyps objektklass. Det finns dock olika former av egenskapsarv. Vi skall här nämna två principer som påverkar dessa former, nämligen *hur* man ärver och *vad* som ärvs. *Hur* arvet går till bestäms av om *enkelt* eller *multipelt* arv tillämpas. Det förra innebär att varje objekttyp enbart kan ha en omedelbar supertyp i arvskedjan vilket resulterar i en arvshierarki i form av ett träd. Vid multipelt arv kan däremot en objekttyp ha flera omedelbara supertyper vilket gör att vi får en arvshierarki i form av en riktad graf utan cykler (en objekttyp kan inte vara supertyp till sig själv). *Vad* som ärvs bestäms av huruvida enbart specifikationen eller även implementeringen i en supertyp blir "tillgänglig" vid realisering av en subtyp, dvs om vi i implementeringen av en subtyp enbart kan använda oss av komponenter från supertypens specifikation eller även dess implementering.

```
OBJEKTTYP Dokument =
    SPECIFIKATION
        OPERATION Editera();
    IMPLEMENTERING
        INSTANSVARIABLER författare, innehåll : Text;
        OPERATION Editera(); innehåll := "Hej";

OBJEKTTYP Brev =
    SPECIFIKATION
        SUPERTYP Dokument;
    IMPLEMENTERING
        INSTANSVARIABLER postat : Datum;

VARIABLER x, y, z : Brev;

x.Editera();
```

Egenskapsarv

Objektorienterad systemutveckling - en översikt

Utbyggbarhet

Alla datamodeller är uppbyggda kring något typsystem där en viss uppsättning bastyper på olika sätt och i varierande grad kan anpassas till hur vi vill använda modellen. Flexibiliteten hos typsystemet bestämmer graden av anpassningsbarhet. En modell med relativt hög anpassningsbarhet bör omfatta *användardefinierade objekttyper* (användare i betydelsen systemutvecklare, programmerare). Detta innebär att vi utifrån bastyperna och med hjälp av modellens grundläggande primitiver ges möjlighet till utvidgning och anpassning av typsystemet med användardefinierade typer som kan existera tillsammans med, och har likvärdig status som, de ursprungliga bastyperna. Detta kan t ex åstadkommas med egenskapsarv och/eller olika former av parametrisering.

Generisk programmering

Detta innebär att programmeraren skall kunna skriva kod som opererar på objekt utan att objektets exakta typ vid exekveringen behöver vara känd. Det räcker med att känna till objektets gränssnitt. Objektet kan sedan vara av godtycklig typ som uppfyller specifikationen. Mekanismen för att hantera detta kallas "dynamisk bindning", dvs vilken implementering av en operation som skall användas avgörs först vid exekveringen baserat på det aktuella objektets typ. En begränsad form av generisk programmering är också möjlig genom parametrisering. Fördelen med generisk programmering är bl a att "nya" typer kan skapas och användas av "gamla" typer utan att de gamla typerna behöver modifieras. Genom generisk programmering (särskilt dynamisk bindning) ökar flexibiliteten avsevärt. Detta kan dock ske till priset av en viss effektivitetsförlust då bindningen måste göras dynamiskt under exekveringen varje gång anrop sker, och inte som vid statisk bindning, en gång för alla vid kompileringen.

Generisk programmering

```
VARIABLER x, y, z : Dokument;

x.Editera();
```

Stark typning

Med stark typning menas att alla uttryck (objekt) i språket skall kunna typbestämmas ur den statiska representationen av språket (källkoden). Typbestämningen används för att avgöra om ett uttryck är korrekt eller ej. I starkt typade språk är det därför omöjligt att utföra något på ett objekt som inte finns definierat för dess typ. Exempel på omöjliga uttryck är: multiplikation av ett heltal med en text och höjning av lönen för en traktor (vi förutsätter här att typerna är definierade med en "naturlig" semantik). Stark typning gör det möjligt att på ett tidigt stadium upptäcka alla fel av detta slag. Typkontroll av ett uttryck kan ske antingen vid kompilering eller exekvering av kod. Stark typning medger typkontroll redan vid kompileringen. Många obehagliga överraskningar kan därför undvikas vid exekveringen. Priset för stark typning och den därmed ökade "säkerheten" är något minskad flexibilitet.

Stark typning

```
VARIABLER x, y, z : Dokument;

x.Editera();     /* Ok */
x.Addera(5);     /* Fel - addera ej definierad för dokument */
```

Beständighet

I objektorienterade programspråk existerar objekten enbart under programmets exekvering. Det objektorienterade synsättet kan också utgöra en grund för konstruktion av databassystem. Objekten måste här kunna sparas (mer eller mindre) permanent för senare återsökning och bearbetning. Datamodeller, omfattande definitions- och manipuleringsspråk, baserade på objekt är betydligt mer utrycksfulla än traditionella post-orienterade datamodeller. Priset för detta är dock i viss mån minskad flexibilitet vid utsökningar.

Parallellitet

Styrkan hos ett objektorienterat programspråk eller en databashanterare ökar om dessa understöder parallellitet (sann parallellitet eller pseudoparallellitet). Detta innebär vanligen också att objekt skall kunna delas mellan flera exekverande program (processer). Vidare, om uppdateringar av delade objekt skall vara möjliga med bibehållen konsistens måste åtkomsten synkroniseras genom ett transaktionshanteringsprotokoll. Protokollet måste t ex kunna förhindra samtidiga uppdateringar, backa ur transaktioner och återstarta systemet vid allvarliga fel.

Homogenitet

I en objektorienterad miljö bör "allting" vara objekt. Det skall inte göras någon skillnad mellan program, data och meta-data (typer) i detta avseende. Då objekt vanligen byggs upp av mer elementära byggstenar är det en fördel om byggstenarna också utgörs av objekt. Objekten blir därmed enhetligt uppbyggda. Objekt av typen "Anställd" kan t ex vara uppbyggda av objekten namn, anställningsnummer och befattning. Inte bara program och data utan också typer kan representeras som objekt. En typ, t ex "Anställd", kan representeras som ett objekt av typen "Typ". Objekt av typen "Typ" har egenskapen att kunna beskriva typer, inklusive typen "Typ" själv. Det övergripande målet med kravet på homogenitet är att åstadkomma en renodlad och enhetlig miljö inom vilken olika slags objekt kan definieras och bearbetas. Användaren slipper därmed att lära sig olika språk för att hantera olika slags objekt. Dessutom kan både språk och system i regel göras mindre men kraftfullare än tidigare.

Av ovan uppräknade begrepp anses de sex första - objekt, typer, inkapsling, egenskapsarv, utbyggbarhet och generisk programmering - vara de viktigaste. Övriga begrepp tillför ytterligare intressanta egenskaper till grundmodellen.

Tillämpningsområden

Vi skall nedan kort exemplifiera begreppen genom att försöka relatera dessa till några olika tillämpningsområden. Vi vill understryka att detta inte är frågan om någon klassificering eller bedömning av huruvida de språk och modeller som nämns kan kallas "objektorienterade" eller ej, utan snarare en illustration av de generella begreppen. Diskussionen bygger på en enkel korsreferenstabell mellan å ena sidan begrepp och å andra sidan språk/modeller. I de fall då det är ett gränsfall huruvida ett begrepp skall anses som representerat, eller om det kan ingå/tillämpas villkorligt, har detta markerats med parenteser kring X-en.

Begrepp \ Språk	Ada	Clu	Simula	Smalltalk-80	Actors
Objekt	x	x	x	x	x
Typer/klasser		x	x	x	
Inkapsling	(x)	x	(x)	x	(x)
Egenskapsarv			x	x	
Utbyggbarhet	x	x	x	x	
Generisk progr.	x	x	x	x	
Stark typning	x	x	x		
Beständighet				(x)	
Parallellitet	x		(x)	(x)	x
Homogenitet				x	x

Objektorienterad programmering

Objektorieterad programmering representerar en ansats till programutveckling som gradvis vuxit fram ur olika konventioner och tekniker vilka under senare år börjat konvergera och resulterat i en mängd programspåk och miljöer. En bra programmeringsmetod kan dock inte enbart baseras på konventioner och informella regler. Det krävs också formellt stöd för dessa i form av primitiver i programspråken. Vårt val av exempelspråk är på intet sätt uttömmande. Utvecklingen och användningen av dessa språk har dock i hög grad påverkat och bidragit till utvecklingen av objektorienterad programmering samtidigt som deras ursprungliga syften och mål varit olika. Vi skall diskutera de definierade begreppen i relation till språken utifrån matrisen nedan.

Samtliga språk inbegriper inte oväntat objektbegreppet. Detta realiseras t ex i Ada genom Package-moduler och i Simula som instanser skapade utifrån klasser. Objektförekomsternas interna datastruktur definieras av sk *instansvariabler* baserade på språkens inbyggda typer. I många språk finns dock skillnader i hur man ser på objektens beskrivningar, huruvida de kan betraktas som objekttyper vilka är integrerade i språkets typsystem och om även dessa i sig betraktas som objekt. För att en objektbeskrivning skall kunna hanteras som en genrell typ i språket måste det åtminstone vara möjligt att deklarera variabler på objekttypen och använda dessa i generella uttryck. En ytterligare restriktion är att det ej bör vara möjligt att manipulera objektförekomster på annat sätt än via de operationer som ges av objekttypens specifikation (inkapsling). Utifrån detta strikta synsätt är det endast i Clu och Smalltalk som objektbeskrivningar också utgör abstrakta datatyper. I Ada är Package-moduler i princip frikopplade från det egentliga typ-

systemet ochses som statiska (textuella) block. Objekten är här instanser av datatyper som deklarerats inom dessa block. Ej heller i Simula utgör objektbeskrivningar abstrakta datatyper i strikt bemärkelse, däremot gäller både för Ada och Simula att objektförekomster mycket väl kan användas som om de vore en del av det inbyggda typsystemet.

Smalltalk är i detta sammanhang det enda språket där objektbeskrivningar också representeras som objekt i språket, dvs är förekomster av en meta-objektbeskrivning (en "objektbeskrivning för objektbeskrivningar" eller typen "Typ"). Detta är ett av sätten som begreppet homogenitet är representerat i Smalltalk. Objekt i Smalltalk kan också i viss mån betraktas som beständiga då man har möjlighet att spara alla aktiva objekt mellan sessioner.

De flesta språk ger ofta någon möjlighet till att realisera inkapsling men det är bara i vissa språk, som t ex Clu och Smalltalk, där denna är ovillkorlig. Att ha ovillkorlig dataabstraktion får anses vara enbart av godo med tanke på den säkerhet som detta medför. I kombination med textuell/syntaktisk separering av objektbeskrivningars specifikation och implementering får vi också ökad överskådlighet. Denna separering gör det också möjligt att utveckla hela applikationer enbart i termer av specifikationer av objekttyper. Därefter kan objekttyperna kompletteras med implementeringsdetaljer och successivt förbättras. Konsistenskontroller kan hela tiden göras mot de ursprungliga specifikationerna vilket garanterar att inga felaktigheter smyger sig in.

Av våra exempelspråk är det endast Simula och Smalltalk som har egenskapsarv i strikt mening. I bägge fallen rör det sig om enkelt arv och subtyper ärver såväl specifikation som implementering från supertypen. Det senare innebär att subtyper, förutom att ha tillgång till supertypens specifikation, även direkt kan nå både instansvariabler och eventuella interna funktioner. Det senare kan vara mindre bra ur säkerhetssynpunkt men å andra sidan ger det viss flexibilitet. I språk som saknar egenskapsarv kan vi i vissa fall uppnå en liknande effekt genom parametrisering av objekttyper eller objektbeskrivningar. I Ada ges t ex möjlighet att specificera sk generiska programmoduler. Både egenskapsarv och parametrisering utgör en grund för utbyggbarhet av språkens typsystem (datamodell).

Stark typning i ett språk ger oss möjlighet att fånga upp många inkonsistenser redan på ett tidigt stadium (vid kompilering). Ada, Clu, och Simula är alla starkt typade medan Smalltalk väsentligen är ett otypat språk. Vid utveckling av komplex programvara under produktionsförhållanden är stark typning nästan ett måste. Det finns dock många fördelar även med otypade eller svagt typade språk, främst ur flexibilitetssynpunkt. Det kan i ett designskede vara önskvärt att låta berörda objekt själva (mha run-time systemet) avgöra om de kan utföra det som begärs av dem. Detta är en fördel ex vis under prototyputveckling när objektens gränssnitt ännu inte tagit fast form.

Möjlighet till generisk programmering kan åstadkommas på flera sätt. Detta handlar, som vi tidigare nämnt, om att producera generell programkod. Som ett exempel kan vi tänka oss en generell (eller "generisk") typ "Dokument" som i sin specifikation har operationen "editera". Dokument kan specialiseras i bl a typerna "Rapport" och "Brev". Representationen för dessa typer kan emellertid skilja sig åt avsevärt, vilket dock inte återspeglar sig i specifikationen (editera). Varje subtyp måste därför ha sin egen implementering av "editera". Om vi nu från en applikation vill kunna editera olika typer av dokument räcker det med att man i koden i applikationen använder sig av den generella operationen "editera dokument". Beroende på vilken typ objektet egentligen är av anropas rätt implementering av "editera" dynamiskt. Nya subtyper till dokument kan också skapas, t ex "Memo", utan att applikationen påverkas. Implementering av denna typ av programkod underlättas betydligt av språk som har egenskapsarv i kombination med möjligheter till dynamisk bindning, tex Simula. Det går i princip att uppnå motsvarande effekt i t ex Ada genom att använda parametrisering och generiska deklarationer.

Stöd för hantering av parallellitet i programspråk kan dels omfatta definition av transaktioner och dels språkprimitiver för kommunikation och synkronisering av processer. Actors representerar härvidlag ett språk (eller snarare en språktyp) och en formalism av ren forskningskaraktär som speciellt utvecklats för att studera system av parallellt exekverande och kommunicerande objekt. Av våra exempelspråk är Actors det mest speciella i och med att man här har renodlat objektbegreppet och koncentrerat sig på att kunna uttrycka och och hantera parallellitet. I detta avseende kan det ses som en slags grund på vilken man kan bygga mer generella språk som omfattar flera av begreppen som karakteriserar objektorientering. Simula och Smalltalk innehåller språkprimitiver för parallellitet baserade på co-rutinprincipen, detta är en form av pseudoparallellitet vilket innebär att vi kan skapa och använda processer som ger sken av att exekvera parallellt. Ada är däremot att betrakta som ett sant parallellt språk där parallellitet hanteras via sk Task-moduler som resulterar i processer vilka alltid betraktas såsom exekverande parallellt.

Objektorienterad systemutveckling - en översikt

Begrepp \ System	E/R	SIM	GemStone	VBASE
Objekt	x	x	x	x
Typer/klasser	x	x	x	x
Inkapsling			x	x
Egenskapsarv		x	x	x
Utbyggbarhet			x	x
Generisk progr.			x	x
Stark typning	(x)	(x)	x	x
Beständighet	x	x	x	x
Parallellitet	x	x	x	x
Homogenitet			(x)	x

Objektorienterade databashanteringssystem

Objektorienterade databassystem har bl a växt fram ur behovet att tillföra mer semantik till "data" än vad som är möjligt med traditionella datamodeller. Diskrepanserna mellan existerande konceptuella modeller och de datamodeller (relations-, hierarkiska- och nätverksmodeller) som dagens DBHS bygger på har blivit mycket stora. I motsats till programspråk har databassystem en starkt begränsad uttryckskraft, dvs datadefinitions- och datamanipuleringsspråk (DDL+DML). Komplexa datastrukturer låter sig inte enkelt byggas på databasernas post-begrepp. Till de största nackdelarna hör att datamodellen är statisk och att data inte kan ses som objekt. Andra nackdelar är att relationer (1:1, M:M, etc) och egenskapsarv mellan objekt inte kan modelleras explicit.

Många olika typer av objektorienterade databassystem har därför utvecklats. Gemensamt för dessa är beständiga objekt, typer och parallellitet (transaktionshantering). Till de första och enklaste hör implementeringar av E/R-modellen (Entity/Relationship). Dessa omfattar både objekt (entiteter) och typer i svag mening, men saknar viktiga komponenter såsom egenskapsarv, dataabstraktion, en utbyggbar datamodell samt generisk programmering.

Utan egenskapsarv ökar dataredundansen samtidigt som viss semantik (specialisering/generalisering) ej kan representeras i databasen. Utan inkapsling reduceras objekttyperna till datastrukturdefinitioner och instansvariablerna kan därför inte skyddas på samma sätt som annars. Detta ökar i sin tur risken för inkonsistenser i databasen och minskar flexibiliteten vid kodningen. Flexibiliteten påverkas också starkt negativt genom avsaknaden av generisk programmering. Slutligen, utan en utbyggbar datamodell begränsas användaren till ett fördefinierat antal datatyper och operatorer på dessa. I system av denna typ måste man nästan alltid tillgripa applikationsprogram för att utföra mer komplicerade bearbetningar av objekt i databasen.

Databassystem av senare datum har försökt gå runt dessa brister. Så har t ex SIM (Unisys) infört egenskapsarv medan GemStone (Servio Logic) och VBASE (Ontologic) dessutom stöder dataabstraktion, en utbyggbar datamodell och generisk programmering. Faktum är att synen på objekt starkt påminner om den i programspråken ovan. Skillnaden är att här kan objekten sparas permanent. Parallellitet med transaktionshantering är en annan skillnad mot flertalet exemplifierade programspråk ovan.

Alla system, utom VBASE, är uppbyggda för att i huvudsak fungera som en databas-"server". En klar distinktion görs mao mellan applikationsprogram och databas. Applikationsprogrammen anropar databasen för t ex lagring och återsökning av objekt. Vid transport av objekt mellan databas och applikationsprogram måste emellertid objekten konverteras från databasens till applikationsprogrammets datamodell, och omvänt. Beroende på värdspråk blir avståndet mellan de två datamodellerna olika långt vilket påverkar komplexiteten hos konverteringsproceduren. Detta är en brist i kravet på homogenitet. Applikationsprogrammen behöver f ö inte vara skrivna i något objektorienterat språk.

VBASE har drivit kravet på homogenitet så långt att programspråk och databasspråk integrerats inom ramen för en och samma datamodell. Detta kallas ibland för ett "singel-level store". Systemet ser automatiskt till att objekten

hämtas från, respektive sparas på, sekundärminne vid behov. Användarna ser då ingen skillnad mellan program- och databasdata och behöver därför bara lära sig en enda datamodell för att hantera både program och databas. Ett och samma objektorienterade språk kan därigenom används både som programspråk och som databasspråk. System av denna typ brukar kallas för objekthanteringssystem (OHS). Dessa kräver bl a att också typer betraktas som objekt (gäller f ö även GemStone). Vid SISU bedrivs bl a utveckling av en OHS-prototyp kallad AVANCE (tidigare OPAL).

Problem som ännu inte är lösta för objektorienterade databaser som stöder dataabstraktion är effektiv hantering av utsökningar och hur ett generellt frågespråk skall se ut.

Andra tillämpningsområden

Tyngdpunkten i exemplifieringen av objektorientering har legat på programspråk och databassystem. Inom dessa områden sker idag en mycket intensiv utveckling. Vi vill emellertid ge ytterligare ett par exempel från angränsande områden.

Konceptuell modellering är ett område som närmat sig det objektorienterade synsättet men som också påverkat detta starkt. Syftet med en konceptuell modell är att avbilda "informationsobjekt" i stället för informationsflödet i en verksamhet. Betoningen ligger på begreppsanalys och förståelse av hur begrepp används i organisationen.

Flera centrala modelleringsbegrepp i koncetuell modellering såsom klassificering, aggregering, generalisering och semantiska relationer mellan objekt, återfinns även i objektorienterade system. Klassificering motsvaras av typbegreppet som grupperar ett antal objekt (instanser) med gemensamma egenskaper. Aggregering innebär att olika attribut grupperas till ett objekt, t ex Person är ett aggregat av namn och adress. Generalisering motsvaras av egenskapsarv. Semantiska relationer mellan objekt, slutligen, motsvaras av referenser mellan objekt.

Styrkan hos konceptuella modeller ligger i att avbilda objekt och relationer mellan dessa. Svagheterna är att de ofta saknar stöd för dataabstraktion och ett renodlat typbegrepp. Typ här motsvarar närmast en klass. Dessutom är flertalet konceptuella modeller inte exekverbara.

Inom AI-området återfinns många exempel på objektorienterade ansatser. I synnerhet inom området expertsystem försöker man hitta sätt att representera kunskap som sedan kan användas för slutsatsdragning. För att åstadkomma detta används semantiska nät, "frames" och andra kraftfulla modellerings- och representationstekniker. Många begrepp känns här igen ifrån konceptuell modellering. Svagheterna är dock att stöd oftast saknas för dataabstraktion, beständiga objekt och parallellitet. Ibland görs heller ingen klar distinktion mellan typ och objekt.

Samlingen av exempel på tillämpningsområden för det objektorienterade synsätttet skulle kunna göras mycket mer omfattande: konstruktion av användargränssnitt, operativsystem, hårdvara, metoder för konstruktion av mjukvara etc. Av utrymmesskäl nöjer vi oss emellertid med detta.

Objektorientering och systemutvecklingsprocessen

Vi skall avsluta denna artikel med att relatera ideerna kring objektorientering till potentiella effekter på systemutvecklingsprocessen, både vad avser specificering, konstruktion, implementering och underhåll/vidareutveckling.

Modellnärhet är den kanske viktigaste aspekten. Objekttyper avbildar verkligheten på ett naturligt sätt. Vidare är steget ifrån en objektorienterad konceptuell modell till en implementering i ett objektorienterat språk / databassystem avsevärt kortare än till en implementering i t ex COBOL. Därigenom slipper man en mängd fel introducerade under konstruktions- och implementeringsfaserna.

Återanvändbarhet av både specifikationer, programkod och data åstadkommes genom konstruktion av ett antal generella objekttyper samt genom egenskapsarv. Tiden och kostnaderna för nyutveckling av system reduceras därvid avsevärt.

Förändringsbarhet och *portabilitet* är effekter av dataabstraktion, främst separeringen mellan specifikation och implementering. Implementeringen av en typ (interna datastrukturer och kod) kan ändras fritt utan att specifikationen (operationsgränssnittet) påverkas. Andra typer berörs därför inte av interna förändringar. Detta kan ses som en form av "dataoberoende". Även om specifikationen ändras kan berörda typer spåras upp och anpassas till förändringarna. Tiden och kostnaderna för underhåll och vidareutveckling kan därför minskas.

Tillförlitlighet och *konsistens* är effekter av stark typning och kraftfulla mekanismer för transaktionshantering. Genom stark typning upptäcks nästan alla "otillåtna" bearbetningar av objekt på ett tidigt stadium. Transaktionshanteringen hjälper till att bibehålla systemet i ett konsistent tillstånd även vid allvarliga dator- eller programfel. Behörighetmekanismer och övervakningsfunktioner ("triggers") kan ytterligare öka tillförlitligheten.

Det objektorienterade synsättet utgör, som vi sett ovan, en generell grundteknik som kan bidra till att utveckla många andra tillämpningsområden. Objektorientering kan i sig ses som ett stöd för goda konventioner för systemutveckling. Inom forskningsområdet har de formella grundvalarna och mer eller mindre forskningsbetonade produkter utvecklats i mer än 20 år. Den kommersiella utvecklingen av området befinner sig dock ännu i sin linda. Att döma av de forskningsinsatser som görs och de erfarenheter man idag börjat vinna kommer objektorienterade ansatser att utvecklas till ett intressant alternativ. Är du och ditt företag redo ?

Internationellt kalendarium

CRIS 88
COMPUTERIZED ASSISTANCE DURING THE INFORMATION SYSTEMS LIFE CYCLE
19 - 22 September 1988
Royal Holloway and Bedford New College,Egham, Surrey, England

IFIP WG 8.1 conference in the CRIS series.

CRIS 88 is the fourth in the series of international conferences on the theme of **´Comparative Review of Information Systems Methodologies´**.
Previous conferences in the series have dealt with the review, comparison and assessment of a number of state-of-the-art Information Systems Methodologies. The forthcoming conference extends the theme by focusing on automated aids for assisting various stages of the Information Systems development life cycle.

The programme has been structured into two related themes:

- Theme A: **Automated aids for analysis and design** - this will cover aids for facilitating the Business analysis and systems design phases.
- Theme B: **Integrated and automated analysis, design and construction** - this will cover integrated approaches which provide assistance from the analysis through to construction.

To enable comparative evaluation of the approaches presented, each presenter will illustrate the procedures and tools using one of two pre-specified case studies.

Who Should Attend?

- **Technical managers** responsible for selection, evaluation and introduction of information systems methodologies and supporting tools
- **Information systems designers, managers and those marketing systems,** who are interested in new concepts and products, coming to the marketplace
- **Researchers** investigation or developing products and approaches to information systems

All queries should be addressed to:
CRIS 88
The Conference Department (BISL)
The British Computer Society
13 Mansfield Street
London
W1M 0BP
Tel: +44 1 637 0471
Tfx: +44 1 631 1049

INTERNATIONAL CONFERENCE ON DESIGNING SUPPORT SYSTEMS

Dutch Decision Support Systems Research Group
Noordwijkerhout
The Netherlands
November 14 and 15, 1988

MOTIVATION
Support systems like decision support systems, expert systems and office systems play an increasingly important role in organizations. However, it is still not clear **how to design such systems in an effective and efficient way.**

Effective and efficient design of support systems requires answers to questions like:
- What is the added value of support systems?
- How can we recognize this added value?
- What methodology can be used in designing support systems?
- How can we overcome software engineering bottlenecks in building support systems?

Professor Henk Sol
Dept. of Information Systems
Delft University of Technology
P.O. Box 356
2600 AJ Delft
The Netherlands

VDM-Europe Symposium ´88

VDM: the way ahead

12-16 September 1988
Trinity College, Dublin, Ireland

VDM is the acronym for the Vienna Development Method. VDM is applied industrialla and commercially in the systematic development of large software systems by a growing number of European companies. The CEC and several Community Member States are sponsoring a growing number of projects which apply or develop various aspects of VDM.

The CEC has established a VDM-Europe Group which meets regularly to discuss issues of VDM Experience: use & applications, education and training, tools, foundations, and standardization.

Organised by VDM-Europe in cooperation with the Commission of the European Communities.
12-13 September: tutorial
14-16 September: symposium.

Registration:
Karel De Vriendt
Commission of the European Communities
200, rue de la Loi - A25 7/3
B-1049 BRUSSELS
Tel +32 2 235 77 69
Tlx COMEU B 21877
Tfx +32 2 235 65 02
E-mail: karel_ de_ vriendt@eurokom.ucd.ie

Referenser

ACM, *Object-Oriented Programming Systems, Languages and Applications* (OOPSLA´86), Conference Proceedings, Portland, Oregon, Nov. 1986.

ACM, *Object-Oriented Programming Systems, Languages and Applications* (OOPSLA´87), Conference Proceedings, Baltimore, 1987.

Cox, B.J., *Object Oriented Programming: An Evolutionary Approach*, Addison-Wesley 1986.

Goldberg, A. and Robson, D., *Smalltalk-80 The Language and its Implementation*, Addison-Wesley, 1983.

IEEE, Proceedings: *1986 International Workshop on Object-Oriented Database Systems*, Sep. 1986.

IEEE, Tutorial: *Object-Oriented Computing. Volume 1: Concepts*, Computer Society Press of the IEEE, Washington, 1987.

IEEE, Tutorial: *Object-Oriented Computing. Volume 2: Implementations*, Computer Society Press of the IEEE, Washington, 1987.

Schmucker, K.J., *Object-Oriented Programming for the Macintosh*, Hayden, 1986.

Shriver, B. and Wegner, P. (eds), *Research Directions in Object-Oriented Programming*, MIT Press, Cambridge, Massachusetts, 1987.

Integrerad modellering av funktioner och data i Ericssonmetod

Ericsson och SISU har tagit fram en systemutvecklingsmetod avsedd för stora interaktiva informationssystem. Metoden kan betraktas som pragmatisk, då man har valt att göra avkall på metodens teoretiska underbyggnad för att istället göra den enklare. Resultatet har blivit en metod som integrerar funktions- och datamodellering.

Föredrag vid NordData88
av Mattias Hällström, SISU

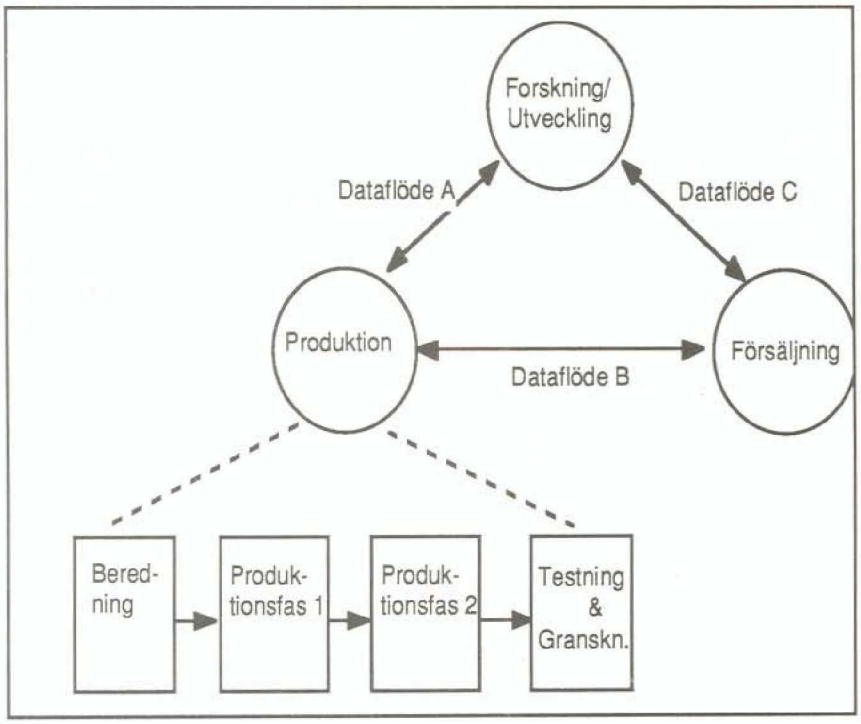


Fig. 1

Bakgrund

Ett företags verksamhet kan beskrivas på flera sätt. Det är vanligt vid utveckling av informationssystem, att man utgående från företagets organisation försöker identifiera och beskriva olika delaktiviteter, dvs beskriva vad som försiggår inom de olika ansvarsområden man valt att organisera företaget i.

I fig 1. beskrivs verksamheten m.h.a. tre delaktiviteter/processer: Forskning/Utveckling, Produktion och Försäljning. Mellan de olika delaktiviteterna sker ett utbyte av information, dvs dataflöde A, B, C. Genom att i detalj bryta ned de olika aktiviteterna, tex Produktion, kan man få en uppfattning om vilka behov av informationsutbyte det finns dem emellan. Den färdiga dataflödesmodellen kan utgöra ett underlag för bedömning av om och i så fall var det i verksamheten finns motiv för att införa datorstöd, dvs om informationsbehovet hos de olika delaktiviteterna kan tillgodoses av ett eller flera datoriserade informationssystem.

Dataflödesmodellering i olika former är sedan länge en vanlig teknik när det gäller att specificera informationssystem.

Integrerad modellering av funktioner och data

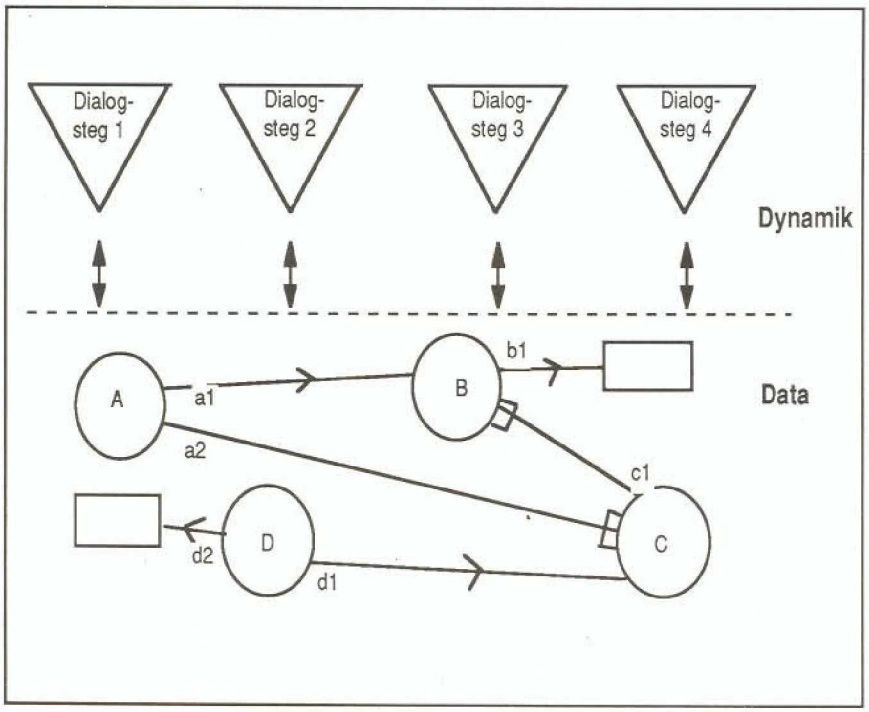


Fig. 2 En systemspecifikation innehåller både en beskrivningav informationssystemets data och dess beteende (dynamik).

Ett informationssystem som till stora delar består av meddelandehantering, kommunikation osv kan på ett bra sätt beskrivas med en dataflödesmodell. Det centrala i ett sådant system är just informationsflödet.

Men låt oss betrakta ett stort homogent interaktivt informationssystem. Det enda egentliga informationsflödet i ett sådant system är i dialogen mellan användarna och systemets databas(Fig. 2).

Vi ska här visa en ansats till en systemutvecklingsmetodik, där vi har valt att helt utelämna dataflöden när vi analyserar och specificerar informationssystemet. Denna metodik har tagits fram i samarbete med Ericsson och är tänkt att användas för stora interaktiva informationssystem.

Ericsson/SISU-metoden

När vi specificerar ett system är det systemets funktionalitet, d.v.s. vad systemet skall göra och inte verksamheten vi vill beskriva. Detta ställer krav både på den systemutvecklingsmetodik vi använder och den specifikation den leder fram till. Systemutvecklingsmetodiken ska hjälpa systemerarna att utreda och dokumentera hur informationssystemet ska bete sig och vilka data det ska hantera. Specifikationen ska sedan utgöra ett bra underlag för realisering av systemet.

Idag och än mer framöver i tiden kan vi säga att en systemutvecklingsmetod består av tre komponenter:

- En eller flera beskrivningstekniker
- Ett systematiskt tillvägagångssätt (en process)
- Verktyg i form av datorstöd.

I SISU/Ericsson-metoden används datamodellering, enligt en egen ansats kallad EMOL, Ericsson Modeling Language, för att beskriva data, och funktions- och dialogbeskrivningar för att specificera systemets beteende (dynamik).

Själva systemutvecklingsprocessen kan här delas in i tre faser:

- Funktionsnedbrytning och identifiering av informationsbehov
- Dialogbeskrivning
- Databasdesign.

Stöd för beskrivningsteknikerna EMOL respektive funktionsbeskrivningar har implementerats i RAMATIC, som är SISU:s datorstöd för analys och specificering.

1 Funktionsnedbrytning och identifiering av informationsbehov

Ett stort informationssystems komplexa funktionalitet kräver en uppdelning av systemet i delfunktioner. Det är däremot inte säkert att den indelning man gör vid specificeringen av systemet kommer att vara densamma som vid realiseringen. Man är helt enkelt tvungen att dela upp systemet i delfunktioner för att få grepp om vad det hela handlar om. Detta är ett vanligt tillvägagångsätt för att lösa stora problem, vilket återspeglas i de flesta systemutvecklingsmetoder.

Det är också vanligt att man gör en funktionsnedbrytning helt fristående från beskrivningen av de data som systemet ska hantera, dvs de informationsbehov som finns hos varje delfunktion. I SISU/Ericsson-metoden har vi däremot valt att integrera funktionsnedbrytningen med datamodelleringen.

Integrerad modellering av funktioner och data

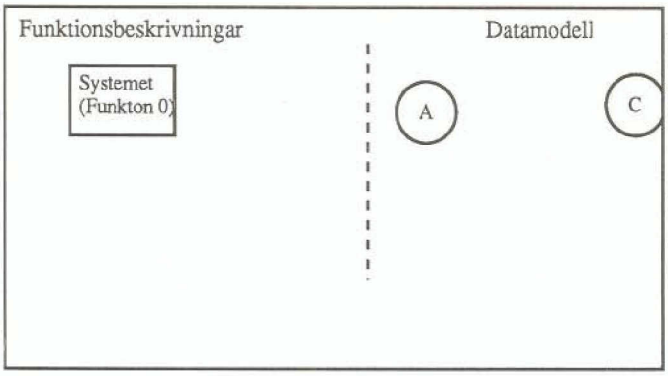


Fig 3.1 En översiktlig beskrivning av själva systemet innehåller referenser till för systemet centrala objekt. Datamodellen uppdateras med dessa.

Metoden förutsätter att det gjorts en inledande analys av den del av verksamheten informationssystemet ska stödja. Funktionsnedbrytningen och datamodelleringen är en iterativ process där datamodellen växer fram under det att man bryter ned och specificerar funktioner. Varje funktionsbeskrivning innehåller en enkel verbal beskrivning samt referenser till datamodellen. Varje element som införs i datamodellen svarar mot ett specifikt informationsbehov hos en funktion.

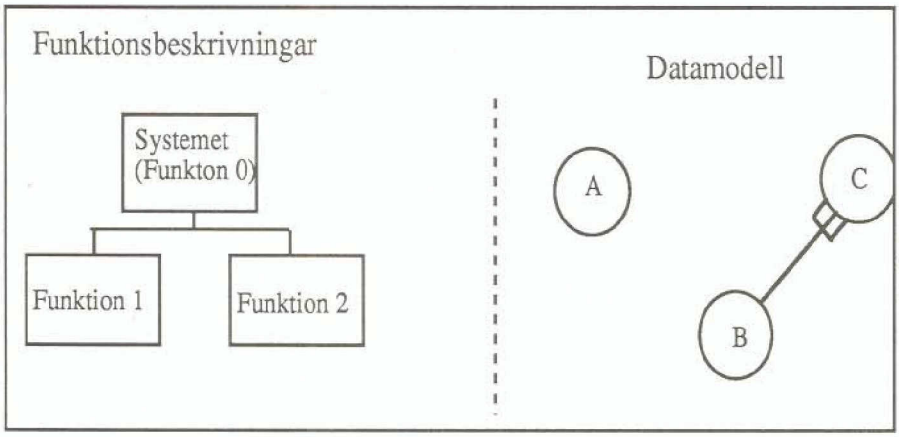


Fig 3.2 Systemet bryts ned i delfunktioner. När dessa beskrivs preciseras informationsbehoven ytterligare. Datamodellen uppdateras med nya objekt och attribut.

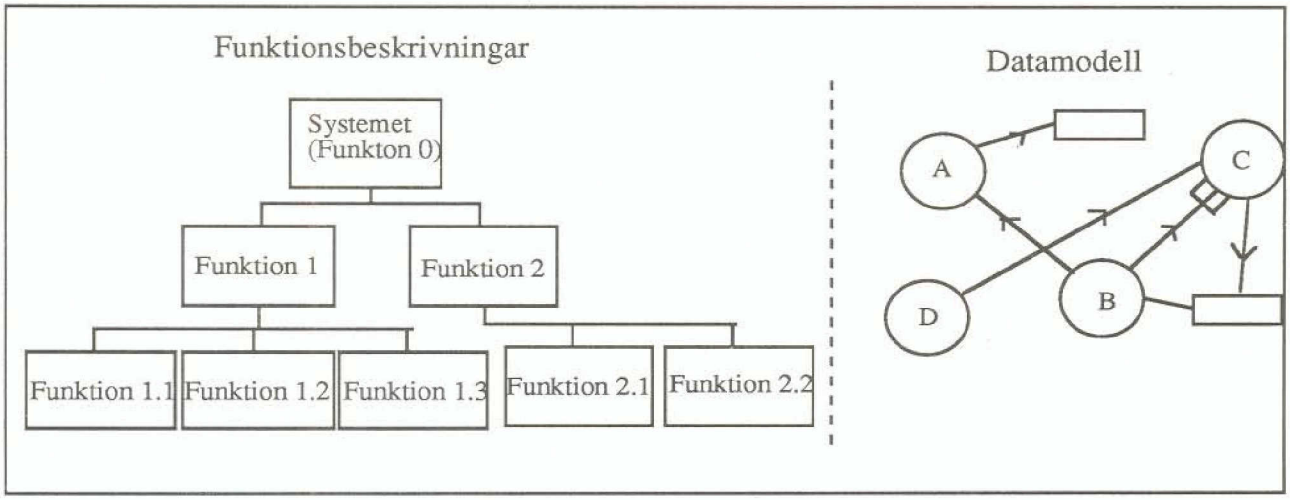


Fig 3.3 Funktionsnedbrytning fortsätter och datamodellen blir allt mer komplex. Observera att funktionsnedbrytningen görs på "bredden först". Detta leder till färre ändringar i datamodellen under arbetets gång.

Funktionsnedbrytningen fortgår till dess att delfunktionerna blivit möjliga att beskriva med hjälp av ett eller flera **dialogsteg**. En delfunktion som det inte är meningsfullt att fortsätta bryta ned benämns **basfunktion**.

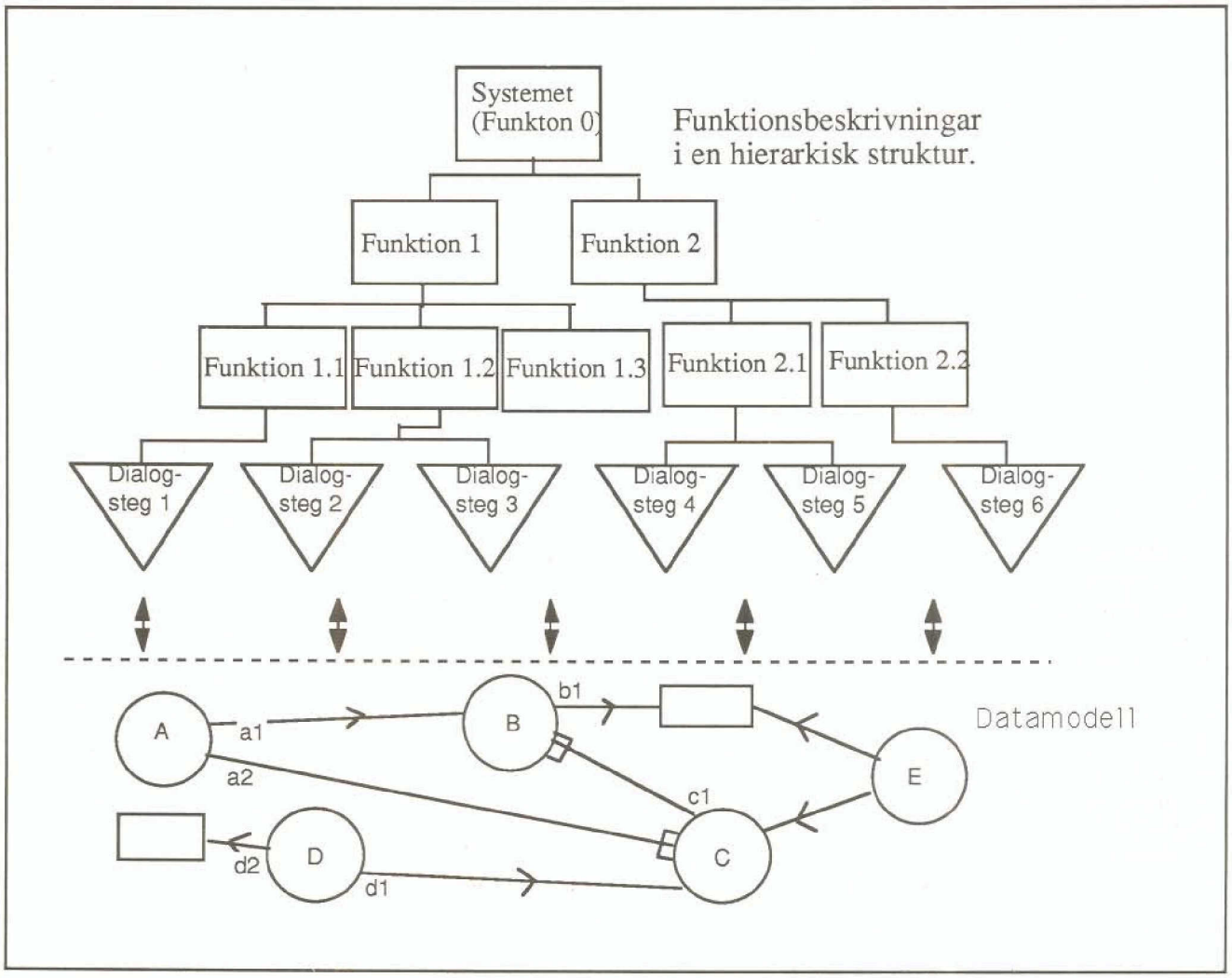


Fig. 4 Varje basfunktion beskrivs av ett eller flera dialogsteg. Både funktionsbeskrivningar och dialogsteg innehåller referenser till modellen.

2 Dialogbeskrivning

Varje basfunktion beskrivs av ett eller flera dialogsteg (se fig. 4) samt en dialogstruktur. Ett dialogsteg innehåller en detaljerad verbal beskrivning samt preciserade referenser till datamodellen.

Det finns tre typer av dialogsteg:

- formulär,
- rapport och
- import/export av data.

Dialogsteg av typ formulär:
Är den vanligaste typen av dialogsteg. Beskriver en möjlig dialog mellan användare och informationssystemet. Vid referenser till datamodellen skiljer man här på att läsa resp. uppdatera Objekt/Attribut. Dialogsteget kan också innehålla eventuell layoutbeskrivning av det formulär som ska användas i dialogen.

Dialogsteg av typ rapport:
Innehåller en beskrivning av en icke-interaktiv rapportfunktion. Interaktiva rapportfunktioner beskrivs som formulär. Också här skiljer man på att läsa resp. uppdatera Objekt/Attribut.

Dialogsteg av typ Import/export av data:
Denna typ av dialogsteg beskriver gränssnitt mot andra system.

Dialogstruktur:
För varje basfunktion finns en dialogstruktur. Med grafisk notation beskrivs villkorliga och ovillkorliga övergångar mellan basfuktionens olika dialogsteg.

3 Databasdesign

De data och relationer mellan data som informationssystemet ska hantera beskrivs med hjälp av en konceptuell datamodell. I samarbete med Ericsson har EMOL, en grafisk notation utvecklats. Datamodellen består dels av en graf och dels av kompletterande beskrivningsformulär.

Exempel på modelleringsbegrepp i EMOL:

Objektmängd:
Representeras grafiskt av en cirkel. Objektmängden representerar de objekt man i systemet vill lagra information om.

Attribut:
Representeras grafiskt av en pil. Attributet anger den information man vill lagra om varje objektmängd. Attributet kan vara kopplat antingen till en annan objektmängd eller till en värdemängd.

Värdemängd:
Representeras grafiskt av en rektangel. Värdemämgden representerar den mängd värden ett attribut kan ha.

Genom att varje objekt och attribut i datamodellen är motiverade av ett informationsbehov hos en funktion eller ett dialogsteg, räknar vi med att datamodellen är av så god kvalité att den direkt kan översättas till ett databasschema.

I de utvärderingar av metoden som pågår, kommer den konceptuella datamodellen först att mekaniskt översättas till en **relationsmodell** i tredje normalform. Datorstöd för detta är under utveckling. Relationsmodellen kallas efter den direkta översättningen för ett **preliminärt databasschema.**
Undan för undan optimeras databasschemat med avseende på prestanda i sin tillämpning. Det färdiga schemat kallas för **optimerat databasschema.**

Sammanfattning

Metoden kan i grova drag beskrivas som en iterativ process där datamodellen växer fram under nedbrytning och specificering av de funktioner i verksamheten som informationssystemet skall stödja. Viktigt är den direkta koppling som finns mellan datamodellen och funktionsspecifikationerna. Datamodellen måste " motiveras", dvs varje element som förs in i datamodellen måste svara mot ett informationsbehov i någon funktion.

Vi har hittills inte funnit någon anledning att i denna systemutvecklingsmetod modellera dataflöden. Det är lättare för användare och systemerare att kommunicera, om specificeringen av systemets funktionalitet görs i form av dialogsteg.Vi tycker dataflödesmodellen kan vara en något "onaturlig" beskrivningsteknik, som i många fall kan leda till svårförståeliga specifikationer.

Metoden kommer att provas hos Ericsson i ett stort projekt kallat PAM (Produkt Assurance and Modification handling system). Erfarenheter från detta projekt kommer säkert att leda till en hel del ändringar och modifieringar. Närmast i tiden ligger en utökning av ansatsen för att möjliggöra specifikation av distribuerade informationsssystem.

SISU-matrikel

AR-BOLAGET
Anders Bohman
AR-Bolaget AB,
Box 5156, 102 44 Stockholm,
Tel: 08/63 03 60

ARTHUR YOUNG AB
Åke Ekström
Box 3143, 103 62 Stockholm
Tel: 08/796 33 00

ASEA
Gunnar Nilsson
ASEA DATA AB, 721 80 Västerås
Tel. 021/32 33 00

AU-GRUPPEN
Sven-Bertil Wallin
AU-Gruppen AB,
Kungsg. 53, 111 22 Stockholm,
Tel: 08/24 34 20

DATA LOGIC
Örjan Odelhög
Data Logic AB,
Fröfästeg. 125, 421 31 Västra Frölunda,
Tel: 031/45 03 40

DIGITAL
Staffan Westbeck
Digital Equipment AB,
Allen 6, 172 89 Sundbyberg
Tel: 08/733 80 00

ENEA
Bo Steinholtz
ENEA DATA Svenska AB,
Box 232, 183 23 Täby
Tel: 08/756 72 20

ERICSSON
Christer Dahlgren
HF/DT ERICSSON, 126 25 Stockholm
Tel: 08/719 07 53

FFV ELEKTRONIK
Hans Holmberg
FFV Elektronik AB,
Box 1381, 171 27 Solna
Tel: 08/730 50 00

FÖRSVARETS RATIONALISERINGSINSTITUT
Stig Åke Nilsson
FRI, Box 80008, 104 50 Stockholm
Tel: 08/788 75 00

FÖRSVARSSTABEN
Torleif Olhede
Försvarsstaben,
Box 80001, 104 50 Stockholm,
Tel: 08/788 78 67

IBM
Lars Arosenius
IBM Svenska AB, 163 92 Stockholm
Tel: 08/793 40 60

INFOLOGICS
Dick Eriksson
SU TVT Infologics AB,
Chalmers Teknikpark
Sven Hultins g. 9, 9A, 412 88 Göteborg
Tel: 031/72 42 60

IRM CONSULT
Eskil Swende,
IRM Consult AB
Box 100, 161 26 Bromma,
Tel: 08/26 93 10

KOMMUNDATA
Karl-Erik Lennartsson
Kommundata AB, 125 86 Älvsjö
Tel: 08/749 80 00

MIMER SOFTWARE AB
Lars-Erik Jansson
Box 1713, 751 47 Uppsala
Tel: 018/18 50 00

PARALOG
Mats Löfström
Paralog AB, Box 2284, 103 17 Stockholm
Tel: 08/14 41 90

PEAB
Stellan Borg
Philips Kistaindustrier AB
Dream, Box 33, 164 93 Kista
Tel: 08/703 10 00

POSTEN
Gert Persson
Posten,
Koncernstab KP, 105 00 Stockholm
Tel: 08/781 10 00

PROGRAMATOR
Håkan Friberg o Karl-Olof Wigander
AB Programator,
Box 20072, 161 20 Bromma,
Tel: 08/799 35 00

RIKSSKATTEVERKET
Lars Olsson,
Riksskatteverket, 171 94 Solna
Tel: 08/764 80 00

SAAB-SCANIA
Sven Yngvell
Saab, Flygdivisionen Dataservice
581 88 Linköping
Tel: 013/18 23 86

SAS DATA
Ove Lundvall
SAS Data, 161 87 Stockholm
Tel: 08/797 10 18

S-E-BANKEN
Peter Söderström
S-E-banken, SMD M4, 106 40 Stockholm
Tel: 08/763 50 00

SKANDIA
Anders Fungdal
Skandia-Data, 103 50 Stockholm
Tel: 08/788 17 26

SKF
Bo Lindahl
SKF Group Headquarters
415 50 Göteborg,
Tel: 031/372626

SPADAB
Göran Lustig
Box 341, 101 24 Stockholm
Tel: 08/13 41 54

STATSKONSULT
Per-Olof Hultman
Statskonsult Admin. Utveckl. AB
Box 4040, 171 04 Solna
Tel: 08/730 03 00

STATSKONTORET
Kerstin Norrby, Lars Hellberg
Statskontoret,
Box 34107, 100 26 Stockholm
Tel: 08/738 45 94, 08/738 47 77

SÖDRA SKOGSÄGARNA
Jerry Nilsson
Södra Data AB,
Box 832, 264 00 Klippan,
Tel: 0435/12090

TELEVERKET
Henry Samuelson
Televerket, ADB-Service
Q24:05, Box 164, 136 23 Haninge
Tel: 08/713 58 00

TELEVERKET
Avd f Grundteknik inkl dotterbolag
Karl-Erik Carlsson
Telelogic AB
Box 883, 851 24 Sundsvall
Tel: 060/16 14 44

UNISYS
Inge Dahlberg
Unisys AB, 171 91 Solna
Tel: 08/55 15 00

VATTENFALL
Georg Karlén
Statens Vattenfallsverk,
Vattenfall Data, 162 87 Vällingby
Tel: 08/739 50 00

VOLVO-DATA
Kenneth Pettersson o Anders Persson
AB Volvo-Data,
Avd 2800, 405 08 Göteborg,
Tel: 031/66 76 48,66 56 48

VOLVO LASTVAGNAR
Tore Altenstedt
Volvo Lastvagnar AB,
Avd 24170 BC4, 405 08 Göteborg,
Tel: 031/66 68 81

VOLVO PERSONVAGNAR
Uno Eriksson
Volvo PV AB,
Avd. 50820 AU, 405 08 Göteborg,
Tel: 031/592074