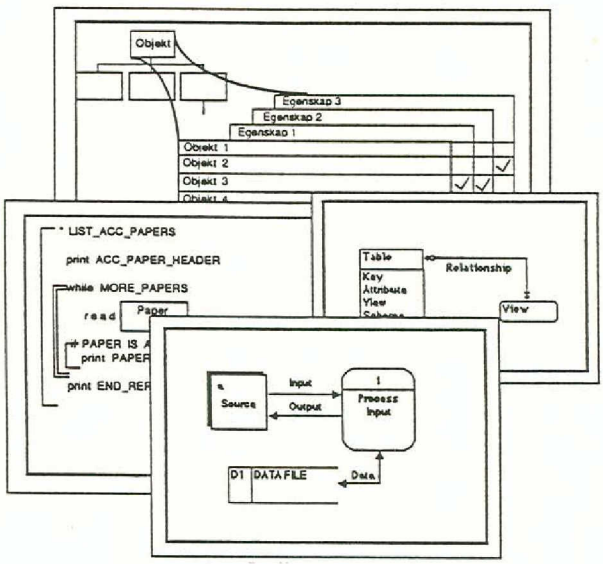
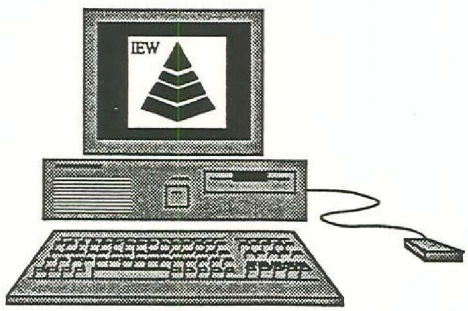
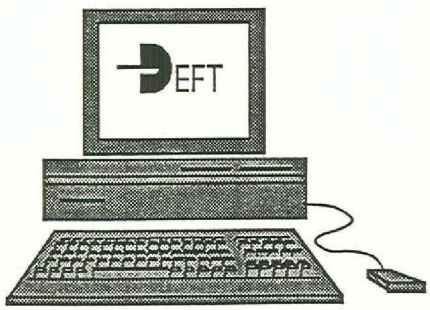
SISU

En Experimentell Studie Av Case-verktygen Deft Och IEW/WS

Tapani Kinnula & Jalal Matini







ISSN: 0282-9924

Innehållsförteckning

EXPERIMENTELL STUDIE AV CASE-VERKTYGEN DEFT OCH IEW

Förord

Denna skrift har utarbetats som examensarbete av de undertecknade från datatekniklinjen på Kunliga Tekniska Högskolan i Stockholm under sommaren 1988. Arbetet bedrevs i SISU:s regi. SISU (Svenska Institutet för SystemUtveckling) bedriver forskning inom området utveckling av informationssystem.
Syftet med arbetet var att skapa ett underlag för noggranna studier av datorstödda verktyg för analys och design av informationssystem samt med detta underlag som grund göra en studie av två kommersiella verktyg. Resultatet av studien skulle i första hand användas som underlag för den som tänker utföra en liknande studie och i andra hand kunna användas som informationskälla för den som behöver kunskap om de studerade verktygen. En typisk situation då sådan kunskap behövs är när ett företag överväger att skaffa sig datorstödda hjälpmedel för underlätta och effektivisera utvecklingen och underhållet av nya informationssystem. En objektiv och saklig rapport om observationer på olika verktygs egenskaper vid tillämpning på ett praktikfall kan då ge information om vilket verktyg som bäst lämpar sig för företaget i fråga.

Inom SISU planerar man att i samarbete med SISUs intressenter göra en liknande studie av de vanligaste på marknaden förekommande verktygen för analys och design av informationssystem (s.k. CASE-verktyg). Underlaget för studier av dessa skulle därför utformas så att man i framtiden skulle ha en enhetlig och detaljerad beskrivning om hur studierna skall bedrivas och vilka aspekter man skall lägga vikt på, dvs något slags granskningsprotokoll.

Vad ingår i denna skrift

För att den tidspressade läsaren skall få en god uppfattning om rapportens innehåll börjar vi med några sidors genomgående sammanfattning. Resten av rapporten har delats in i fem tämligen oberoende delar. I den första delen berättar vi litet om CASE-begreppet och CASE-verktyg. Den andra delen innehåller en detaljerad beskrivning av underlaget för studien. I underlaget ingår ett fiktivt praktikfall för simulering av en verklig situation och ett granskningsprotokoll, som tar upp vilka aspekter vi har funnit att man bör titta på vid studier av CASE-verktyg. Granskningsprotokollet kan även ses som ett skelett för en rapport för beskrivning av CASE-verktyg. Del tre och fyra innehåller en beskrivning av Deft och IEW respektive, vilka var de verktyg vi granskade. Dessa beskrivningar kan ses som separata rapporter om dessa verktyg egenskaper i enlighet med vårt granskningsprotokoll. I den femte delen återfinns en kort sammanfattning av verktygen samt hur de relateras till varandra. Som avslutning ger vi en kort beskrivning om vårt arbete och några synpunkter och råd till dem som tänker företa sig motsvarande uppgifter. För den intresserade finns i appendixet en del av dokumentationen till "implementationerna" av de praktikfall vi använde oss av vid det utvärderingsarbetet.

Tillkännagivelser

Givetvis skulle vi inte kunnat klara av allt det omfattande arbete som föregick den här skriftens tillkomst helt på egen hand. Nej, vi behövde hjälp. Och all den hjälp vi behövde fick vi också. Därför ser vi oss föranledda att uttrycka vår tacksamhet till de personer och företag som medverkade i att vårt examensarbete blev en mycket intressant och lärorik upplevelse i en vänlig och trevlig omgivning.

Janis Bubenko jr, SISU, gav oss inledande undervisning och såg till att allt flöt som det skulle. Han stod dessutom till förfogande som vår "slutanvändare" för konferenssystemet - som var vårt praktikfall - och gav oss värdefull kunskap vilka krav och problem man stöter på i ett verkligt fall. Mattias Hällström, vår handledare på SISU, såg till att vi fick de resurser vi behövde och gav oss genom många och långa diskussioner ökade kunskaper med nya perspektiv på metoder och verktyg för datorstödd systemutveckling. Benkt Wangler ställde upp och svarade på våra frågor samt erbjöd sig att hjälpa oss vid anskaffning av litteratur. Överhuvudtaget var personalen på SISU vänlig och trevlig, vilket gjorde att vår vistelse där blev mycket angenäm.

Naturligtvis behövde vi också verktyg att göra studierna på. Athur Young AB ställde upp med Information Engineering Workbench®/WS (IEW/WS) och Åke Ekström på Arthur Young bistod med hjälp och information närhelst det behövdes. Infotool AB ställde upp med Deft och var mycket hjälpsamma. Åke Nyberg på Infotool gav oss dessutom en förhandstitt på en helt ny version av Deft och såg till att vi fick tillgång till ännu ej publicerat material om hur Deft kan användas vid systemutvecklingsprojekt.

Reservationer

Vi poängterar att syftet med en studie enligt denna skrift är just - en studie. Granskningen av verktyg utgör inte någon utvärdering av dessa eller ett försök att betygsätta dem. Meningen är att ge en informativ beskrivning av dem. Vi tar heller ingen ställning till huruvida ett verktyg är bättre eller sämre än något annat. Detta vore faktiskt högst tvivelaktig. Det är ju hur verktyget uppfyller företagets krav på ett systemeringshjälpmedel och hur det skall användas i företaget, som avgör om man skall satsa på det. Det är alltså i högsta grad frågan om en individuell bedömning där verktygets kvalitéer och företagets behov måste sammanvägas.

Vi vill också framhålla de resultat och slutsatser vi kommit fram till inte är resultat från en omfattande vetenskaplig undersökning, utan till stor del baserar på våra egna bedömningar ocf erfarenheter under vårt arbete. Studien kan ej heller sägas vara heltäckande i den mening att den tar upp alla relevanta aspekter i systemutvecklingsprocessen. Vi har utgått från ett fiktivt praktikfall som grund för vår undersökning. Syftet med praktifall är ju att få ett realistiskt och konkret intryck av verktygets egenskaper, men ett idealiserat praktikfall kan inte avspegla alla de faser och problem som kan förekommma vid tillämpning av verktygen på verkliga praktikfall.

Tapani Kinnula
Jalal Matini

SAMMANFATTNING

Denna sammanfattning ger en kort genomgång av rapportens innehåll och följer kapitelindelninegn i rapporten. För att underätta orienteringen har de viktigaste aspekterna återgivits som marginalanteckningar.

CASE

CASE (Computer Aided Software Engineering) är ett begrepp som man allt oftare hör i samband med systemutveckling. CASE-verktyg är datorstödda hjälpmedel för utveckling av informationssystem, och används oftast för analys och design av system, men många verktyg ger stöd även för andra moment i systemutvecklingsarbetet. Vanligtvis innehåller de ett antal ritverktyg som i kombination med en databas (datakatalog) används för att skapa systemspecifikationer. Ritverktygen brukar stödja vanligt förkommande grafiska beskrivningstekniker, som dataflödesdiagram, entity-relationshipdiagram och olika dekompositionsdiagram. Systemspecifikationerna lagras i datakatalogen för analys, dokumentation och eventuell vidareanvändning (t.ex. automatisk kodgenerering).

Användning av CASE-verktyg kan medföra besparingar i både tid och besvär samtidigt som kvalitén på specifikationer och dokumention ökar. Kan verkyget dessutom kombineras med 4GL-system eller har automatisk kodgenering kan besparingarna bli avsevärda på grund av kraftigt reducerat programmeringsarbete.

UNDERLAG FÖR EN STUDIE AV CASE-VERKTYG

För att få verklighetsanknytning till användning av de studerade verktygen använde vi oss av ett konstruerat praktikfall, som dels är tillräckligt komplicerad dels ej alltför tidskrävande att genomföra.

Praktikfallet - IFIP Arbetskonferens

Praktikfallet beskriver ett informationssystem som stödjer arrangerande av IFIP-arbetskonferenser. IFIP (the International Federation for Information Processing) är en internationell organisation för forskning och kunskapsutbyte inom informationsteknologi. Konferenssystemet skall stödja de flesta aktiviteter som förekommer vid anordnande av konferenser. Exempel på dessa är registering konferensbidrag, granskning och acceptering av mottagna bidrag, förberedelse av listor över deltagare etc. Dokumentationen till praktikfall innehåller en detaljerad beskrivning av verksamheten kring konferensförberedelserna och av kraven på informationssystemet.

Granskningsprotokollet

Granskningsprotokollet är en detaljerad lista över egenskaper som vi tycker man bör studera hos CASE-verktyg. Dessutom finns en del bakgrund och motiveringar till många av egenskaperna. För att undvika missförstånd har vi även förklarat innebörden hos vissa begrepp. Syftet med granskningsprotokollet är att studier av olika CASE-verktyg utförs på ett likvärdigt sätt samtidigt som det ger en enhetlig struktur för

rapporter om studier av CASE-verktyg. Egenskaperna har delats in i ett antal huvudområden. En kort översikt av huvudområden och de viktigaste aspekterna:

- Systemutveckling med verktyget
  - Verktygets "spännvidd"
  - Beskrivningstekniker
  - Metodstöd
- Systemutvecklingsdatabasen
  - Datakatalogen
  - Kontroll- och analysmöjligheter
- Användarmiljön
  - Gränssnitt mot användaren och interaktivitet
  - Fleranvändarmiljön
- Egenskaper och faciliteter hos verktyget
  - Portabilitet
  - Kodgenerering
  - Prototypframställning
  - Flexibilitet
- Hur påverkar verktyget systemutvecklingsprocessen
  - Vinster i tid och kvalitet
  - Effekterna på informationssystemets livscykel
  - Slutanvändarnas roll i utvecklingsarbetet
  - Utbildningsbehovet
- Hård- och mjukvarukrav, kostnader och dokumentation

Våra rapporter om de studerade CASE-verktygen Deft och IEW/WS följer denna uppläggning och kan därför utan svårigheter användas som underlag för jämförelser. Vi har dock inte gjort några detaljerade jämförelser utan överlåter detta till den intresserade läsaren.

STUDIE AV DEFT

Deft är ett CASE-verktyg som har utvecklats av det kanadensiska företaget TSB International Inc och finns för Macintosh persondatorer. Deft syftar till att ge stöd för de tidiga faserna i systemutvecklingsprocessen, analys och design, samt skapande och underhållande av systemdokumentation.

Defts uppbyggnad

Deft består av ett antal fristående moduler som dock använder sig av en gemensam datakatalog

Data Flow Diagram Editor, DFD, är ett grafiskt ritverktyg för konstruktion av dataflödesdiagram.

Program Structure Diagram Editor, PSD, är ett hjälpmedel för konstruktion av systemets programkomponenter.

Entity Relationsship Diagram Editor, ERD, är en grafisk editor för datamodellering

Med Form Editor är verktyg för design av skärmbilder och utformning av dialoger mellan systemet och dess användare.

Design Compiler kontrollerar syntax och gör viss konsis-

tensanalys i systemspecifikationer samt generar sk objektfiler för vidaretransport till 4GL-miljöer.

Translator (Gateway) överför objektfiler till Ingres-miljön för uppläggning av databaser och skärmdialoger.

Beskrivningstekniken

De beskrivningstekniker som Deft använder sig av finns i respektive modul. Vid konstruktion av dataflödesdiagram kan representationssätt beskrivna av E. Yourdon eller Gane&Sarson användas. I PSD används en beskrivningteknik som närmast kan betecknas som en blandning av JSP (Jackson Structured Programming) och strukturkartor (structure charts), dvs en JSP utökad med subrutinparametrar. Defts entity-relationship-diagram är kanske i huvudsak ett verktyg för design av relationsdatabaser, även om de kan användas för mera generell datamodellering. Man kan välja mellan presentationssätt enligt Chen/Bachman, Martin eller IRM. För att kontrollera de grundläggande syntax- och designreglerna efterföljs i diagrammen kan Design Compiler användas. Om inget avsteg från reglerna har skett så genereras en objektfil för vidare bearbetning i andra miljöer.

Integration mellan diagram

Integrationsgraden mellan diagrammen i Deft är låg. Det finns ingen koppling mellan objekt i olika diagram. Det enda sambandet är att objektdefinitioner skapade i ett diagram kan senare utnyttjas i samma eller i ett annat diagram.

Metodstöd

Deft är inte bundet till någon specifik systemutveclingsmetod, utan kan användas med de metoder som lämpar sig för de beskrivningstekniker som finns i Deft. För vårt praktikfall tillämpade vi Structured Analys and Structured Design, utan att möta några större problem.

datakatalogen

Datakatalogen är gemensam för de olika modulerna i Deft. Detta innebär att man kan utnyttja datatermer definierade i en annan modul. I datakatalogen lagras information om datatermer och datastrukturer. Information om processer, entiteter och programrutiner lagras i de Macintosh-dokument som respektive modul genererar. Det finns inte möjligheter till några avancerade automatiska analyser på datakatalogens innehåll, förutom vissa syntaktiska kontroller.

Innehållet i datakatalogen och de grafiska presentationerna behöver inte vara konsistenta med varandra, dvs det man ser i ett diagram behöver inte vara det samma som finns i datakatalogen. Detta kan vara ett problem speciellt eftersom datakatalogen saknar viktiga analysfunktioner.

Defts påverkan på systemutvecklingsprocessen

Den mest märkbara effekten som användning av Deft kan medföra är att tyngdpunkten förkjuts från programmering mot analys och design av system, samtidigt som en stor del av dokumentationen skapas automatiskt. En del av programmeringsarbete kvarstår förstås, men kan underlättas genom användning av PSD. En fördel med att använda Deft tillsammans med Ingres (och även andra 4GL-system i framtiden) är att vidareutveckling av system underlättas. De grafiska presentationerna och Defts dokumentation gör det lättare att förstå hur

systemet fungerar, och de nödvändiga modifieringarna kan lätt införas i specifikationer. De modifierade specifikationerna kan sedan på nytt överföras till Ingres. I de fall man behöver ändra programlogiken måste dock koden i Ingres ändras manuellt eftersom PSD-resultat inte kan överföras automatiskt.

De klara grafiska presentationerna gör att även slutanvändaren lättare kan vara med i analys- och designprocessen. Via skärmbilder kan de i tidigt skede se hur systemet kommer att se ut och föreslå eventuella ändringar.

STUDIE AV IEW/WS

IEW/WS, Information Engineering Workbench®/Workstations, är ett CASE-verktyg från KnowledgeWare Inc. och distribueras av Arthur Young AB. IEW/WS finns för IBM PC-kompatibla datorer under MS-DOS operativsystemet och utnyttjar personda-torernas grafiska möjligheter med fönster- och mushantering för bearbetning av systembeskrivningar.

IEW's uppbyggnad

IEW/WS består av ett antal moduler som koncentrerar sig på olika faser inom systemutvecklingsområdet. Modulerna är Planning Workstation, Analysis Workstation och Design Workstation. Förutom dessa finns en produkt som heter Gamma för kodgenerering i IBM-stordatormiljön. Verktygen är oberoende av varandra men integreras via encyclopedien, IEW's datakatalog.

IEW använder sig av Information Engineering konceptet. Detta koncept består av en mängd formella tekniker med vars hjälp verksamhetsmodeller, datamodeller och processmodeller framställs och lagras i en kunskapsbas för att skapa och underhålla system för informationshantering. IEW erbjuder ett stort antal tekniker för att beskriva systemspecifikationer. Dessa beskrivningstekniker använder sig av gemensamma data i encyclopedien och kan ses som olika sätt att betrakta det system man utvecklar och som komplement till varandra.

Beskrivningstekniker i IEW/WS

Planning Workstation

I Planning Workstation finns verktyg för modellering av verksamheten med hjälp av data- och processmodeller. Med Entity Diagrammer skapar man en datamodell (entity-relationship-diagram) över den information som hanteras i verksamheten. För att beskriva verksamheten använder man Decomposition Diagrammer, ett verktyg för hierarkisk nedbrytning av objekt i verksamheten. Exempel på dessa objekt är funktioner, processer, mekanismer och organisatoriska enheter. Förutom dessa två diagramverktyg finns matrisdiagram för beskrivning av objekts egenskaper och associationer mellan objekt.

Analysis Workstation

Analysis Workstation innehåller verktyg för analys och detaljerad beskrivning av delområden i verksamheten. För detta används datamodeller och processmodeller som i Planning Workstation fast mera detaljerat och för varje verksamhetsområde för sig. Dataflödesdiagram används för att ange hur data "flyter" mellan processerna i verksamheten eller mellan grundläggande systemkomponenter. Man kan också i detalj beskriva attributen för dataobjekten och definiera process

logiken för processerna i dekompositionsdiagram och dataflödesdiagram. För beskrivning av processlogik används actiondiagram, och logiken byggs med hjälp av färdiga block. Meningen är man specificerar programlogiken för processerna på lägsta nivån i processhierarkin. För att underlätta analysen och kontrollen av korrektheten och fullständigheten i specifikationerna finns i Analysis Workstation ett antal analys- och rapportfunktioner.

Design Workstation

I Design Workstation finns verktyg för design och realisering av det önskade informationssystemet. Bland dessa kan nämnas strukturkartor, action diagrams och diagram för design av databaser. Man kan med hjälp av dessa verktyg specificera programstrukturer, procedurlogik, skärmbilder samt databas- och filstrukturer. Syftet med Design Workstation är att få med slutanvändarna i utvecklingsprocessen, minska tiden för design och realisering av system och automatisera design, dokumentation och underhåll av system. Dessutom skall designresultaten kunna utnyttjas vid automatisk kodgenerering.

Integration mellan diagrammen

I IEW är alla data gemensamma mellan de olika diagrammen. Detta innebär till exempel att de data som bygger upp dataflöden mellan processerna är en del av den totala datamodellen samtidigt som processerna i dataflödesdiagrammet återfinns i den totala funktions- och processhierarkin. Man behöver endast specificera ett objekt en gång.Man kan också öppna ett diagram inifrån ett annat för att göra en mera detaljerad beskrivning av ett objekt.

Metodstöd

IEW är inte bundet till någon specifik systemutvecklingsmetod även om metoder som följer riktlinjerna i Information Engineering lämpar sig bäst. Typiskt för dessa metoder är att de börjar med att analysera informationsbehovet i verksamheten med hjälp av datamodeller för att sedan övergå i analys och design av procedurer som behövs för att tillfredställa informationsbehovet. IEW kan förmodligen tillämpas på ett effektivt sätt även för andra typer av metoder som IEW's beskrivningstekniker lämpar sig för.

Encyclopedien

Alla specifikationer lagras i den centrala datakatalogen, encyclopedien. Encyclopedien är en semantisk databas över alla objekt, deras egenskaper och associationerna mellan dem. Den möjliggör kontroller av fullständighet och konsistens i systembeskrivningarna. De olika diagrammen reflekterar alltid det aktuella innehållet i encyclopedien. Detta innebär till exempel att en ändring i ett diagram genast återspeglas i andra diagram. De olika grafiska representationerna är således alltid konsistenta med encyclopedien (och med varandra).

Analys och kontroll

IEW har en stor mängd olika analys- och rapportfunktioner. Dessa syftar till att kontrollera konsistensen och fullständigheten i specifikationern samt tjänar som dokumentation över systemspecifikationerna. Exempel på analyser kan nämnas konsistenskontroll över innehållet i dataflöden och datalager i dataflödesdiagram. Man kan också granska objekten i specifikationerna med avseende på vissa egenskaper eller hur de knyts ihop via associationer. En del rapporter kan exporteras och fungerar som bryggor till andra miljöer (t.ex. 4GL-verktyg).

IEW's påverkan på systemutvecklings-processen

Vilken effekt IEW har på företagets systemutveckling beror förstås på den tidigare systemutvecklingsmiljön och de metoder man använt sig av. Generellt kan man dock säga att det blir lättare att analysera alternativa systemlösningar och därmed försäkra sig om att det slutliga systemet bättre uppfyller de uppställda kraven. Det slutliga systemets kvalité kommer sannolikt också att förbättras tack vare IEW's analys- och rapport funktioner. Systemdokumentationen är dock kanske det som får den största kvalitetsökningen. Den blir korrekt och kan genereras automatiskt i form av olika rapporter. Programmeringsarbetet kan reduceras genom att att använda Gamma för generering av kod utifrån IEW's action diagrams. Underhåll av system underlättas eftersom alla systemspecifikationer finns i encyclopedien och kan modifieras i IEW's arbetstationer för att sedan användas för generering av ny kod. Men för att uppnå alla dessa positiva effekter måste man se till att man använder sig av formella och effektiva systemutvecklingsmetoder. Systemutvecklingspersonalen bör också ha en god utbildning i både IEW och den metod man använder.

SLUTSATSER

Trots en del gemensamma beskrivningstekniker så är Deft och IEW ganska olika verktyg. Deft är snabb och smidig men har inte de möjligheter till analyser och kontroller som finns i IEW. IEW täcker också flera moment inom systemutvecklingsprocessen. Å andra sidan gör just den rigorösa on-line kontrollen IEW något långsammare att arbeta med. Största skillnaden mellan Deft och IEW är innehållet och hantering av datakatalogen. IEW's encyclopedi innehåller mycket detaljerad information om systemspecifikationer och hanteras av ett inbyggt expertsystem medan Defts datakatalog är enklare och mera rättfram både vad gäller innehåll och hantering. Deft kan dock visa värdefull särskilt i kombination 4GL-system (Ingres), och är mycket billigare än IEW.

1 CASE

1.1 CASE-begreppet och CASE-verktygen

Ordet CASE är en förkortning av Computer Aided Software Engineering. Namnet syftar till ingengörsmässig programkonstruktion med datorstödda hjälpmedel, men i praktiken har ordet CASE blivit synonymt med datorstödd systemutveckling, och då med tonvikt på informationssystem. De flesta har nog hört talas om 4GL-verktyg, och vet att dessa också är verktyg för systemutveckling. Var kommer då CASE in i bilden? För att besvara denna fråga, måste vi först definiera vad vi menar med 4GL-verktyg, eller 4GL-miljöer om man så vill.

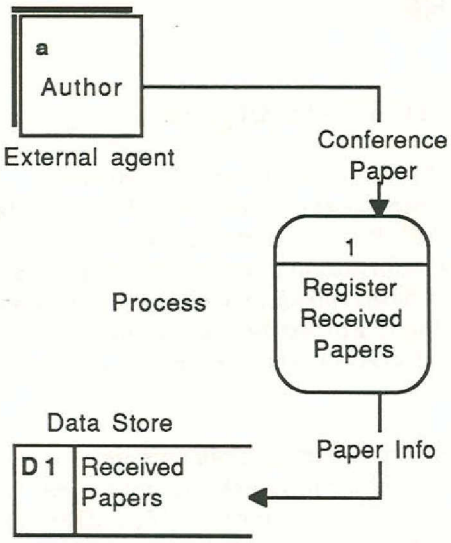
När vi säger 4GL, menar vi inte något programmeringspråk (4:e generationens språk) utan snarare en modern systemutvecklingsmiljö, ett informationshanteringssystem (IH-system). I RDF-rapport nr 16, [RDF83], beskriver man ingående vad som ingår i ett IH-system. Följande citat från denna rapport kan bringa litet klarhet i IH-begreppet:

"De programkomponenter som vi kan förvänta oss att finna i en avancerad ADB-miljö idag är:

- databassystemet med funktioner för gruppering och strukturering av data
- datakommunikation och TP-monitor för hantering av terminaler och kommunikationslinjer
- slutanvändarfunktioner som frågespråk, rapportgenerator och grafiska funktioner för enkel åtkomst av informations för icke ADB-kunniga
- datakatalog-funktioner som möjliggör gemendamma databeskrivningar i alla komponenter av verktyget
- textbehandling och funktioner för kontorsautomation"

Ett IH-system innehåller dessa funktioner integrerade i ett programpaket och har en för alla dessa funktioner gemensam datakatalog, så att alla definitioner och beskrivningar görs endast en gång och samlas på ett ställe, en s.k. datakatalog (data dictionary). Om vi även inkluderar möjligheter för grafisk datamodellering i vår definition av IH-system, har vi en relativt heltäckande miljö för systemutveckling, som vi skulle kunna kalla för en fullt integrerad systemutvecklingsmiljö. I dagens produkter finns sällan alla dessa funktioner integrerade i ett och samma verktyg. I praktiken innehåller 4GL-verktygen funktioner för databasdesign, rapportgenerering, skärm- och rapportlayouter, och något språk för processbeskrivning. De har tyngdpunkten i att konstruera och realisera system. Det som saknas i dessa verktyg är möjligheterna till att analysera och med analysresultaten som stöd grafiskt konstruera informationssystem, samt överföra konstruktionen till fungerande system.

CASE-verktygen har tillkommit för att råda bot på detta förhållande. De innehåller ett grafiskt gränssnitt för olika modelleringstekniker, som används vid analys och design av informationssystem. Dessutom brukar det finnas stöd för en eller flera systemutvecklingsmetoder. Det vanliga är att verktygen erbjuder ett antal grafiska editorer, exempelvis för dataflödesdiagram, Entity-Relationshipdiagram och strukturkartor (Structure Charts). Data som definieras i dessa diagram lagras i en central datakatalog, som binder ihop de olika diagrammen. I figur 1.1 ser vi ett dataflödesdiagram. Dataflöden består av attribut (dataelement) som



Attributes:
Author
Author name
Date_registered
date when the paper arrived to the
Conference Secretary
Paper_Subject
subject that is treated in the pape
Paper_Title

Data Flows:
Conference Paper
Paper_Title
Paper_Subject
Author
Paper Info
Paper_Title
Paper_Subject
Author
Date_registered

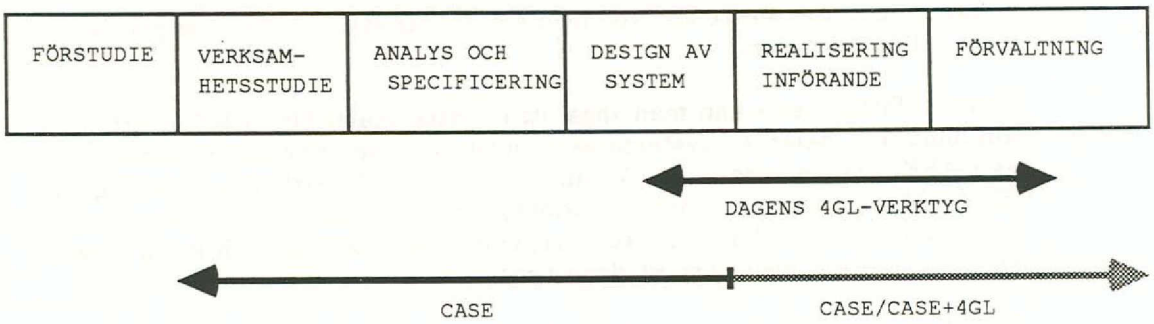
Figur 1.1 Till vänster ett exempel på ett dataflödesdiagram och till höger en del av tillhörande dokumentation ur datakatalogen.

vandrar mellan processer, externa objekt och interna datalager. All information om processer, attribut, dataflöden etc. finns automatiskt dokumenterat i datakatalogen.

Beroende på verktygets ambitionsnivå kan det finnas möjligheter för konsistens- och fullständighetskontroller i datadefinitionerna. I de mera avancerade verktygen finns även stöd för prototypkonstruktion och kodgenerering, normalt utifrån den pseudokod man specificerat procedurer med i designfasen. Andra CASE-verktyg kan mera ses som komplement till 4GL-verktyg såtillvida att analys och design utförs i CASE-verktyget, varefter specifikationerna förs över i 4GL-miljön, där implementationen och kodgeneringen sker. Figur 1.2 ger en fingervisning av orienteringen mellan dagens CASE- och 4GL-produkter.

Eftersom all systemutveckling nästan undantagslöst bedrivs i projektform, har de flesta verktygen någon form av stöd för att tillåta flera personer på ett samordnat sätt arbeta med olika deluppgifter. En del har - eller kan kompletteras med - relativt avancerade projektstyrningsfaciliteter.

Det finns ett stort antal metoder som kan tillämpas vid systemutveckling, och följaktligen finns det en varierande flora av CASE-verktyg. En del av verktygen använder sig av någon särskild systemutvecklingsmetod, medan andra kan anpassas att stödja flera metoder. De vanligaste metoderna som stöds är Structured Analysis and Design, Jackson Structured Design och olika tillämpningar av Information Engineering (IE). Det är kanske på sin plats att nämna att hela CASE-begreppet har stora likheter med IE-konceptet beskrivet av James Martin ([Martin81], [Martin86]). IE-konceptet innefattar i stort sett hela systemutvecklingsprocessen och beskriver ett strukturerat arbetssätt för systemutveckling och hur specifikationer lagras i en central datakatalog.



Figur 1.2 En idealiserad tolkning av hur CASE- och dagens 4GL-verktyg förhåller sig till varandra. Det finns dock mycket stora skillnader mellan olika verktyg. Det finns även CASE-verktyg som kan användas för förstudie.

1.2 Varför CASE

Vad kan man vinna med att använda CASE-verktyg i stället för de traditionella väggkartorna eller papper och penna -metoderna? Den största fördelen är förstås att kunna samla alla specifikationerna på ett ställe och på ett integrerat sätt. Dessa specifikationerna kan när som helst plockas fram och analyseras och enkelt modifieras, utan att behöva göra om stora delar av arbetet.

I traditionell systemutveckling tar dokumentering lång tid blir ofta bristfällig. Statistiken visar att vid utveckling av stora system, tar dokumentationen längre tid än själva kodningen. Då bra CASE-verktyg används skapas dokumentationen automatiskt under arbetets gång och lagras på ett mera fullständigt och strukturerat sätt än vad som är möjligt i traditionell systemutveckling. Diagram som utvecklas på skärmen är just dokumentation. Till dessa diagram kan sedan datakatalog, datamodeller och processmodeller associeras. Tidsvinsterna vid dokumentationen tillsammans med mera korrekt dokumentation kan alltså bli väsentliga.

Slutanvändarna får större möjligheter att delta i systemutvecklingsprocessen, eftersom mycket av arbetet sker i lättfattlig grafisk form. Systemeringsexperterna och användarna har fått en gemensam visuell modell. Detta leder till att slutprodukten bättre tillfredställer användarnas behov och önskemål.

Om verktyget även har möjlighet till kodgenerering eller dess specifikationer kan överföras i 4GL-produkter för automatisk kodgenerering, kan stora tidsvinster uppnås vid implementationsfasen. Prototypsystem kan snabbt utvecklas och modifieras.

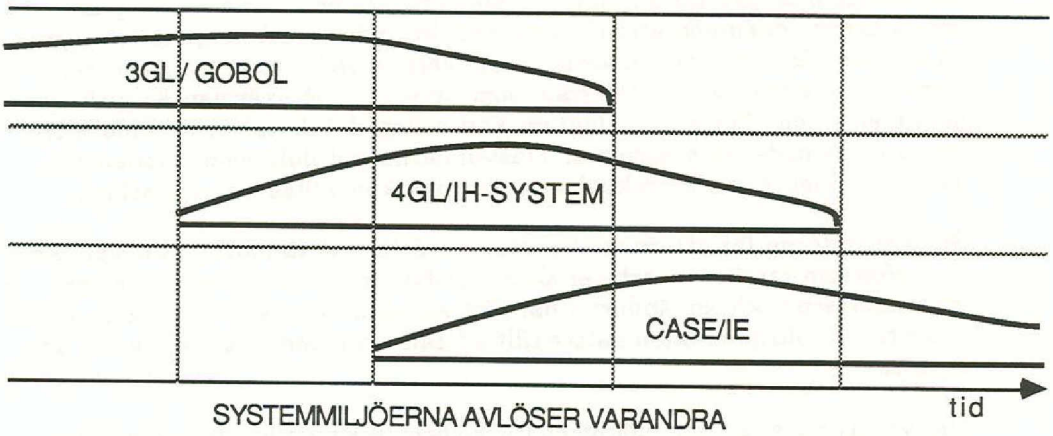
När man har en heltäckande miljö för systemutveckling - ett fullständigt integrerat IH-system - med helt eller delvis automatisk kodgenerering, har man också stora tids- och kostnadsvinster att hämta i förvaltning av system (CASE-verktyg som täcker hela systemutvecklingskedjan och har automatisk kodgenerering och stöd för systemförvaltning brukar betecknas I-CASE, Integrated CASE). Vid ändrade krav går man tillbaka till specifikationerna och modifierar dessa. Ny kod genereras. Om man dessutom utvecklar alla applikationerna i företaget med samma metoder och hjälpmedel, kan man uppnå en enhetlig systemstruktur över hela företaget med hög integrationsgrad mellan applikationerna. Då blir det också lättare att försäkra sig om att informationen från gamla systemet inte går förlorad,

eftersom man har större kontroll över systemens struktur och hur processerna i dem fungerar.

Sammanfattningsvis kan man säga att främsta skälet för CASE är att automatisera delar av systemutvecklingsprocessen. Korrekt användning av CASE-verktyg eventuellt i kombination med 4GL-verktyg ger möjlighet till snabbare och mera användarorienterad systemutveckling, minskade utvecklings- och underhållskostnader, större produktivitet och mera överskådliga system med korrekt dokumentation.

Enligt en undersökning man gjort i Europa och USA har nästan alla företag som helt eller delvis övergått från traditionell Cobol-miljö till en 4GL-miljö gjort väsentliga rationaliseringsvinster. Därför drar man i Riksdataförbunbdets publikation "Moderna Systemutvecklingsverktyg" slutsatsen att "Det kan betraktas som tjänstefel att inte utreda vad en 4GL-miljö kan betyda för den egna organisationen". Eftersom CASE-verktygen kan ses som komplement eller till och med ersättning till några av dagens 4GL-verktyg beroende på verktygen och hur man ser på det hela, finns all anledning att utreda vad CASE betyder för företaget.

För att ge litet tidsperspektiv för relationerna mellan dagens systemmiljöer återger vi här nedan en illustration ur Riksdataförbundets "Moderna Systemutvecklingsverktyg". Bilden ger intryck att CASE-verktygen kommer att ersätta dagens 4GL-verktyg för att sedan avta i framtiden. Kanske kommer CASE-verktygen att ändra form, eller vi kanske får helt nya systemutvecklingsmetoder som ersätter CASE. Men det vet vi ju ännu inget om.



Figur 1.3 En förenklad tolkning av systemmiljöernas förhållande till varandra tidsmässigt. Vi har avstått från att specificera årtal, men man kan dock säga att 3GL/Cobol hade sin storhetstid på 60- och 70-talen, medan 4GL har sin puckel på 80-talet. Det är förmodligen CASE som gäller på 90-talet.

2 UNDERLAG FÖR EN STUDIE AV CASE-VERKTYG

2.1 Varför ett praktikfall och ett granskningsprotokoll

"The only way to learn a new programming language is by writing programs in it", säger Brian Kernighan, en av skaparna till programmeringspråket C. Likadant hävdar vi att att det bästa sättet att lära sig ett CASE-verktyg är att använda det i praktiken. Att läsa en mängd glättade broshyrer om det och därefter bläddra litet i manualer kanske ger en viss bild av verktygets egenskaper. Men är denna bild korrekt? Har man fått en allsidig och rättvis uppfattning av verktygets användbarhet och kvalitéer? Antagligen inte! För att få en exakt uppfattning om vilken nytta verktyget ger för företaget, måste det användas i ett verkligt systemeringsprojekt på det sätt verktyget är ämnat att användas. Detta är förstås sällan möjligt. En månads provinstallation räcker normalt inte, och ofta är företagets metoder inte direkt anpassade till verktygets arbetssätt. Det kan bli aktuellt att ta ställning till om man skall anpassa sina egna arbetsmetoder så att de passar till verktygets, eller om man i stället skall försöka hitta en produkt som stödjer den metod man själv använder. Dessa är svåra frågor, som vi inte tänker behandla här utan hänvisar till en skrift av Janis Bubenko jr, "Selecting a strategy for Computer Aided Software Engineering" (Janis88).

Ett sätt att få en någorlunda objektiv bild av olika verktyg, är att läsa rapporter av andra som använt dessa verktyg. Detta sätt är också det enklaste. Vår uppgift var att tillhandahålla en sådan rapport. Men för att skriva denna rapport och ge en beskrivning av CASE-verktyg och deras egenskaper, måste vi ha en stabil grund att stå på: erfarenhet från en verklig situation.

Att genomföra ett praktikfall som inte är alltför stort och tidskrävande men ändå har en verklighetsbakgrund, kan ge en god bild av ett systemeringshjälpmedel och insikt i de problem man i praktiken kan råka ut för. Och den uppfattning man får av verktyget är avgjort mera konkret än man skulle fått genom broschyrer och manualer. Därför konstruerade vi ett fiktivt praktikfall, som kan genomföras på relativt kort tid, men ändå avspeglar många av de problem och arbetsmoment som återfinns i verkliga systemeringsprojekt. Detta praktikfall, att konstruera ett system för anordning av konferenser, finns beskrivet litet längre fram.

För att våra beskrivningar av verktygen skulle få enhetlig struktur och behandla verktygen på ett likartat sätt, utformade vi ett slags rapportskelett, där vi tar upp de olika aspekterna man bör tänka på vid studie av CASE-verktyg. Där finns även förklaringar och motiveringar till dessa aspekter. När man arbetar med praktikfallet kan man se detta skelett som granskningsprotokoll där man fyller i det man upptäckt och de slutsatser man kommit fram till. När man är klar och skall skriva sin rapport, är det enkelt att ta detta nu uppstoppade skelett med sin färdiga struktur och forma om den till en informativ beskrivning av det behandlade verktyget. Detta förfarande medverkar till att rapporter skrivna av olika personer och som behandlar olika verktyg får en enhetlig struktur och läsaren får lätt att hitta det han/hon vill veta.

2.2 Praktikfallet - IFIP Arbetskonferens

Underlag för en studie av metoder och verktyg för informationssystemutveckling

I följande text kommer vi att beskriva ett konstruerat praktikfall för systemutveckling, konstruktion av ett konferenssystem. Syftet därmed är att kunna analysera och granska metoder och verktyg för informationssystemutveckling. Praktikfallet, som ursprungligen presenterades 1980 av IFIP WG8.1, har ofta använts just för detta syfte. Vi har valt att komplettera den ursprungliga beskrivningen. Motivet är att praktikfallet skall vara fullständigt dokumenterat och specificerat, så att all den information som behövs för dess genomförande framgår ur dokumentationen. Inga egna antaganden skall behöva göras. Om avvikelser eller tillägg ändå måste göras skall anledningen därtill redovisas.

Vi har valt att i görligaste mån beskriva verksamheten kring konferensförberedelserna i textform. Detta för att undvika koppling till de beskrivnings- och modelleringstekniker som kan finnas i verktyg för systemutveckling. Faran med semantiska modeller och strukturerade beskrivningar av verksamheten i ett praktikfall är att konstruktionsarbetet kan övergå i översättning av dessa till ett format som verktygen använder. Detta favoriserar då verktyg vars beskrivningstekniker överenstämmer med verksamhetsbeskrivningen.

Nackdelen med en textuell beskrivning är att den lätt blir svårförstådd och kryddad med tvetydiga formuleringar. Vi har försökt undvika detta genom en överskådlig uppläggning av texten och utförliga beskrivningar. Om oklarheter trots allt uppstår, får man använda sin fantasi och försöka hålla sig till rimliga tolkningar.

2.2.1 Introduktion

IFIP (the International Federation for Information Processing) är en internationell organisation vars syfte är att stimulera forskning och kunskapssutbyte inom området informationshantering. Det tekniska arbetet inom IFIP sker i nio tekniska kommittéer (Technical Committee, TC), som består av representanter för medlemsnationerna. Under varje TC finns ett antal arbetsgrupper (Working Group, WG) som är specialiserade inom någon gren av TC:s intresseområde. Utöver de IFIP kongresser som anordnas var tredje år, organiserar de tekniska kommittéerna konferenser inom sina ämnesområden. Storleken för dessa konferenser varierar från stora internationella händelser till mindre arbetskonferenser. Praktikfallet består av att modellera och specificera ett system som skall stödja en av IFIP:s Working Groups i organiserandet av en arbetskonferens.

En IFIP arbetskonferens är en internationell konferens vars syfte är att samla expertisen från alla IFIP-länderna för att diskutera något ämne som är av speciellt intresse för en eller flera IFIP Working Groups. Det vanliga - och också det som gäller för praktikfallet - är att konferensen inte är för öppen för allmänheten utan deltagarna är speciellt inbjudna. Vid organisering av en sådan konferens är det ofta svårt att försäkra sig om att medlemmarna i aktuella WGs och TCs är inbjudna även om de inte kommer. Vidare är det viktig att se till att tillräckligt många deltar i konferensen så att den går ihop ekonomiskt samtidigt som antalet deltagare inte får bli för stort.

Vid arrangering av en konferens är ansvaret för förberedelsearbetet uppdelat mellan två kommittéer, en programkommitté och en organisationskommitté. Programkommittéen svarar för konferensens tekniska innehåll, som utformning av sessioner utifrån inkomna bidrag från författare och färdigställandet av ett konferensprogram. Organisationskommittén i sin tur svarar för inbjudningar, lokala arrangemang och de ekonomiska aspekterna. Dessa kommittéer har ett nära samarbete med varandra och har behov av gemensam information som skall hållas konsistent och aktuell.

2.2.3 Förklaring av några använda termer

Den engelska terminologin kring konferenser används ofta även i svensk text. Eftersom översättningar dessutom blir mer eller mindre haltande, har vi valt att använda de engelska uttrycken, dock skrivna i kursiv stil för att understryka detta. Här nedan förklarar vi innebörden av vissa begrepp kring arrangerandet av konferenser.

Call for paper
Förfrågan om en skrivelse till en konferens. Innehåller uppgifter om konferensens titel,var och när den äger rum samt vilka ämnen som skall behandlas under konferensen.

Callee
Person som fått förfrågan om en skrivelse

Letter of intent
Meddelande om avsikt att bidra med en skrift med uppgift om formulär som skickats med call for paper.

Paper
Skrivelse om ett specifikt ämne som skall behandlas under konferensen. Ett *paper* kan ha flera författare.

Referee
Person vars uppgift är att utvärdera och betygsätta inlämnade *papers*.

Report
Utvärdering av ett *paper* . En referee betygsätter *paper* enligt givna kriterier.

Sekretariat
Instans där all posthantering sker.

Konferensprogram
Fullständig förteckning över konferensens program med uppgifter om var och när konferensen äger rum, lista över sessioner med uppgifter om tider, ordförande och papers som behandlas samt vilka som presenterar dem.

Session
En konferens består av ett antal sessioner. Sessionen är uppdelad i ett antal lektioner. En lektion består av en presentation av ett *paper* med påföljande diskussion. Sessionen leds av en ordförande.

2.2.4 Verksamhetsbeskrivning

Konferensförberedelserna sätter igång då någon arbetsgrupp (WG) och/eller teknisk kommitté beslutar sig för att anordna en konferens om något aktuellt ämne för utbyte av kunskaper och forskningsresultat. Man väljer kommittémedlemmarna och utser ordföranden och kontaktpersoner för varje kommitté.

För att hålla reda på detaljerna registreras och dokumenteras diverse information. I detta första skede registreras information om kommittéernas medlemmar och konferensen: vilka som är värdar, ämnen som skall behandlas, var och när konferensen äger rum, kontaktpersonerna, deadlines för inlämning av *papers* och *reports* .

Programkommittéens verksamhet börjar med att göra en förteckning över personerna - experter, forskare osv - som kan tänkas kunna bidra med diskussionsmaterial till konferensen. Till dessa personer skall *call for paper* skickas. Tillsammans men *call for paper* skickas också en formulär - *letter of intent* - där uppgifter om det tänkta bidraget och författarna kan anges.

Författare som ämnar bidra med ett paper skickar tillbaka ett ifyllt *letter of intent*. Dessa registreras för att kunna användas vid preliminär planering.

De inlämnade bidragen registreras och kopior av dem tillsammans med en utvärderingsbegäran - *request* - skickas till personer som ställer upp som *referees*. För att kunna följa upp utvärderingsarbetet registreras även uppgifter om *referees* och vilka *papers* dessa skall utvärdera.

Bedömningen av ett *paper* görs på en formulär - *report* - som innehåller olika bedömningskritererier och en betygsskala för dessa.
Programkommittéen samlar in *referee reports*, och sänder vid behov påminnelser till sena referees. Till varje kriterium hör en viktfaktor som anger kriteriets procentuella betydelse vid totalbedömningen av ett *paper* (ex. om värdet för kriteriet 'relevans' har 20% av det totala betyget har det viktfaktorn 0,2). Det sammanvägda betyget från en *referee* fås genom att summera ihop de viktade betygen.

De utvärderade *papers* rangordnas efter de genomsnittliga sammanvägda betygen. Detta genomsnittliga betyg för ett *paper* fås genom att summera ihop de sammanvägda betygen från alla referees och dividera summan med antalet *referees* som utvärderat papret. De bästa *papers* (tex bästa 10%) antas direkt, medan de sämsta (tex sämsta 10%) refuseras. Resten av *papers* antas av programkommittéen efter diskussioner om deras kvalitéer.För varje *paper* registreras resultaten från utvärderingen.

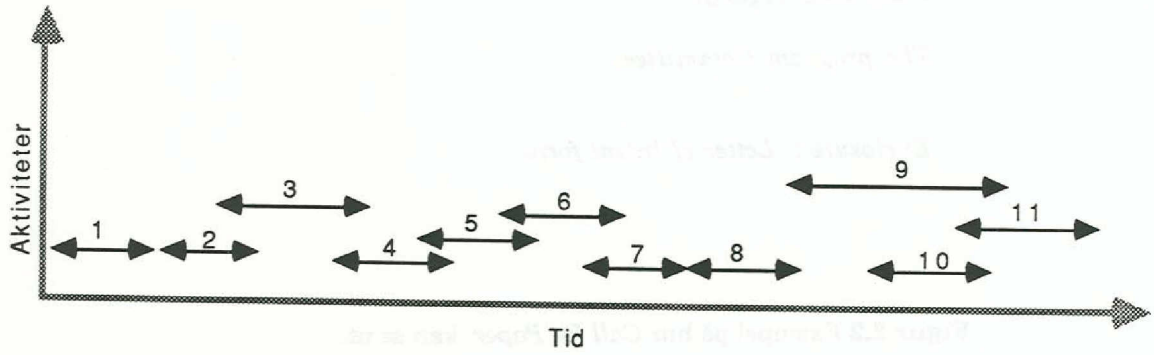
När antagningsprocessen är klar skall programkommittéen utforma det slutliga konferensprogrammet. De utvalda *papers* grupperas i sessioner för presentation och diskussion. För varje session skall man dessutom välja en ordförande och bestämma tid och plats för sessionen samt i vilken ordning papers skall presenteras under sessionen. Uppgifterna om sessioner och presentationer registreras. Det färdiga konferensprogrammet framställs och publiceras. Därmed är programkommittéens arbete slutfört.

När programkommittéen är klar med sin uppgift kan organisationskommittéen börja arbetet med inbjudningar och konferensdeltagandet. Innan dess kan den dock ha arbetat med budgetering, lokala arrangemang och andra aktiviteter kring konferensen.

Organisationskommittéen förbereder en lista över vilka personer som skall delta i konferensen och ser till att alla dessa får en inbjudan. Med inbjudan brukar man ofta bifoga konferensprogrammet. Vissa deltagare, som medlemmar ur aktuella WGs och TCs, är automatiskt berättigade att få en indjudan. Även alla författare som skickat ett *paper* - även om det blivit refuserat - skall få en inbjudan. Accepterade inbjudningar registreras och en slutlig deltagarlista genereras. Organisationskommittéens aktiviteter kring andra uppgifter berörs inte här.

En författare kan när som helst dra tillbaka sitt bidrag eller någon av nyckelpersonerna - t.ex. presentatören för ett *paper* eller en ordförande - kan få förhinder. Detta medför givetvis att man måste modifiera konferensprogramet och uppdatera vissa register.

1- Information om konferensen registreras
2- Val av callees
3- Call for paper skickas iväg
4- Registrering av letter of intent
5- Registrering av ankomna papers
6- Papers skickas i väg till referees för utvärdering
7- Registrering av utvärderingsresultat
8- Utformning av konferensprogrammet
9- Utformning av sessioner
10- Inbjudningar skickas i väg
11- Registrering av deltagarna



Figur 2.1 Tidsförloppet över konferensföberedelserna

2.2.5 Informationsbehovet

Den informationen som behövs att verksamheten skall fungera finns att hämta i olika register och i de dokument som hanteras under konferensförberedelserna. I dessa register finns uppgifter om konferensen, författare, vilka som kan ställa up som referees eller ordföranden eller andra inblandade personer. Dessutom lagras information om dokument som *letter of intent, papers* och *reports* för senare behandling. I figurerna här nedan återfinns utdrag av dessa dokument. Dokumenten är på engelska och innehåller bara de relevanta uppgifterna utan den kosmetika de eventuellt har i verkligheten.

CALL

Mr. Jonas Jonasson
Huvudgatan 6
121 34 Stockholm
Sweden

Dear Colleague

We hereby notify You of TC8's intention to organise an IFIP conference entitled:

"Computer Aided Tools for Software Development"

to be held in London, UK, during April 1989.

Please fill in enclosed form if you intend to submit one or more papers. Your intentions should reach us before Februari 1, 1989.

With kind regards

The program Committee

Enclosure : Letter of Intent form

Figur 2.2 Exempel på hur *Call for Paper* kan se ut.

LETTER OF INTENT

Submitter
Name: _____

Adress: _____
_____
_____

I intend to submit a paper on (mark exactly one):

Artificiell Intelligence and CASE-technology	O
CASE-tools for real-time applications	O
Graphics based specification languages	O

Preliminary Titel: _____
_____

Authors:
1) _____
2) _____
3) _____

Presented by: _____

Please return form to:
Conference "Computer Aided Tools for Software Development"
Hill Street 12,
153 123 1 London, UK

Note: In case you wish to contribute more than one paper, please send us a filled-in copy of this form for each one.

Figur 2.3 Exempel på hur *Letter of Intent* kan se ut.

REQUEST

Mr. Jürgen Schwarz
ApfelStrass 17,
99 09 12 Münich, Germany

Dear Colleague

In view of the IFIP conference entitled:
"Computer Aided Tools for Software Development"
we would like you to evaluate the following paper:
Paper-nr: 17
Subject: Graphics based specification languages
Title: The Formal Specification Language GraphLang

Please send us your opinion before Mars 1, 1989 on the enclosed evaluation report.

With kind regards

The Program Committee

Enclosure: Review Report form

Figur 2.4 Exempel på hur *Request* kan se ut.

REPORT

Referee
Id-nr: *27*
Name: *Anderz Pethers*
Country: *Germany*

Paper
Nr: *17*
Title: *The Formal Specification Language GraphLang*

Judgement

Scale	*poor*	*below average*	*average*	*good*	*very good*
Scientific Quality	O	O	O	O	O
Relevance	O	O	O	O	O
Originality	O	O	O	O	O
Clarity	O	O	O	O	O
Referee's familiarity with the topic	O	O	O	O	O

Recommendation:

strong accept	O
weak accept	O
weak reject	O
strong reject	O

Comment: _____
_____
_____

Please send to: Conference "Computer Aided Tools for Software Development"
Hill Street 12,
153 123 1 London, UK

Figur 2.5 Exempel på hur *Report* kan se ut.

Till varje kriterium i utvärderingsrapporten (relevance,clarity, etc) tillhör som sagt en viktfaktor som bestämmer kriteteriets relativa betydelse vid beräkning av det sammanvägda betyget. Varken kriterierna eller viktfaktorerna är helt fastställda utan kan ändras för varje konferens.

2.2.6 Informationssystemet

Det informationssystem som konstrueras skall kunna stödja de båda kommittéerna i deras arbete med arrangerandet av en IFIP arbetskonferens. Kommittéerna skall kunna använda sig av samma information när så behövs. Systemet skall stödja dem i deras aktiviteter enligt verksamhetsbeskrivningen och tillåta automatisering där det är möjligt. Här nedan återfinns en förteckning över de aktiviter som systemet skall stödja för respektive kommitté samt vilken form av stöd som kommittéerna behöver (jmfr med verksamhetsbeskrivningen). Under varje huvudaktivitet finns några delaktiviteter
eller tillhörande aktiviteter listade.

Programkommittéen:

1. Förbereda en lista över till vilka som *call for paper* skall skickas
   - registrera potentiella författare - *callees*
   - skriva ut en lista över *callees* som registrerats mellan två givna (godtyckliga) tidpunkter
   - skriva ut *calls* färdiga att sändas

2. Registrera inkomna *letter of intent.*
   - lagra uppgifterna i inkomna *letter of intent*
   - ge information om
     - författare och *papers* anmälda mellan två givna datum
     - anmälda *papers* vid en given tidpunkt
     - anmälda *papers* per land
     - anmälda *papers* per ämne

3. Registrera mottagna *papers* vid ankomsten.
   - registrera information om mottagna *papers*
   - ge information om
     - författare och *papers* ankomna mellan två givna datum
     - *papers* ankomna fram till en given tidpunkt
     - *papers* per land
     - *papers* per ämne

4. distribuera *papers* till dem som har accepterat att vara referees.
   - registrera potentiella *referees* och deras ämnesönskemål
   - matcha ihop *papers* och *referees* med största möjliga hänsyn till *referees* önskemål, dock skall ingen *referee* överbelastas. Matchningen skall kunna ändras och kompletteras manuellt
   - registrera uppgifterna om distributionen
   - skriva ut *requests* till *referees*

5. Samla in reports från *referees* och välja ut vilka *papers* som får deltaga i konferensen.
   - registrera uppgifterna i mottagna *reports*
   - skriva ut påminnelser till sena *referees*
   - ge stöd vid antagningsprocessen, denna stöd kan bestå av
     - rangordning av *papers* efter sammanvägda betyg
     - att för varje *paper* som varken är antaget eller refuserat ge en sammantaffning bestående *referees* namn, *referees* betygsättning per kriterium och dennes

egen kunskapsnivå i ämnet
- att vid en godtycklig tidpunkt ge information om antagna *papers* per ämne och per land
- att för varje *referee* ge information om dennes betygsättning av papers
- skriva ut en lista över antagna papers

Anm. Hur programkommittéen använder informationen från systemet vid antagningsprocessen är inte viktigt här. Dock skall ovannämnda information framställas i form av överskådliga rapporter.

6. Gruppera antagna papers i sessioner för presentation och välja en ordförande för varje session.
- ge information om antagna *papers'* ämnesområden
- ge information om papers persession
- registrera gjorda avtal om ordförandeskap
- skriva ut konferensprogrammet (färdig för tryckning)
- skriva ut gjorda avtal

Organisationskommittén:

1. Förbereda en lista över personer som skall inbjudas till konferensen.
- välja ut personer som skall inbjudas. En del av indjudningslistan skall genereras automatiskt av systemet (*referees*, ordföranden, författare till inlämnade *papers*) men den skall även kunna kompletteras av de båda kommittéerna
- lista alla personer som skall inbjudas

2. Undvika att skicka dubbelinjudningar till någon person.
- kontrollera och eliminera dubbelinbjudningar
- skriva ut inbjudningar (klara för tryckning).

Observera att alla inbjudningar inte behöver skickas på en gång utan det kan även ske vartefter arbetet framskrider.

3. Registrera accepterade inbjudningar (konferensanmälan).
- registrera information om deltagare
- när som helst generera förteckning över anmälda deltagare

4. Generera en slutlig lista av deltagare.
- generera en deltagarförteckning.
- skriva ut deltagarförteckningen

Budgeterings och finansiella aspekter i Organisationskommittéens arbete, mötesplan för båda kommittéerna, hotelreservationer och dylikt beaktas inte här, även om de är väl så viktiga i en verklig situation.

Givetvis måste möjligheten till ändringar och uppdateringar i de olika registren finnas under alla arbetsmoment. Vidare bör systemet kunna varna för inkonsekvenser, som att parallella sessioner har samma eller delvis samma deltagare. Villkoren givna under fakta och antaganden här nedan bör uppfyllas.

2.2.7 Fakta och antaganden

Följande fakta angående IFIP och arrangering av IFIP-konfererenser har sammanställts:

1. En IFIP-konferens anordnas av en eller flera Technical Committees (TC)

2. En Working Group (WG) är en del av endast en TC.

3. En TC består av en eller flera WG:s.

4. En person kan bara representera ett land.

5. En person kan representera ett land i flera TC:s.

6. En TC har som mest en representant per land.

7. *Referees*, ordföranden, WG medlemmar, avsändare till *papers* och nationella representanter (TC medlemmar) måste ha en adress.

8. Ämnesområdena för en viss konferens är fördefinierade.

9. Varje konferens har en tidsram.

10. Varje konferens kan ha flera deadlines (datum för *call for paper*, sista inlämningsdag för *papers* , inlämning av *report*).

11. En konferens kan vara utspridd på olika geografiska platser.

12. Registrering av en ny konferens kräver titel, ämnesområden, plats(er), tidsperiod på varje plats och TC som organiserar konferensen.

13. Det finns bara ett sekretariat per konferens som tar emot post.

14. *Letter of intent* accepteras även från personer som inte fått *call for paper*.

15. Ett *paper* kan accepteras även utan föregående *letter of intent*.

16. *Papers* klassas som inlämnade (submitted), accepterade (accepted) eller refuserade (rejected).

17. Ett paper skall behandla ett eller flera av konferensens ämnen.

18. Varje *paper* kan dras tillbaka av avsändaren.

19. Ett *paper* presenteras av en av författarna.

20. Den som skickar in ett *paper* är sedan kontaktpersonen för dess författare.

21. Kontaktpersonen skall vara en av författarna.

22. En person får bidraga med mera än ett paper till en konferens.

23. En person får inte bidraga med flera *papers* med samma titel till en konferens.

24. Det kan finnas flera *papers* med samma titel i en konferens.

25. Personer som är ordföranden eller *referees* till ett *paper* i viss konferens får inte vara författare till ett *paper* i den konferensen.

26. Registrering av ett *paper* kräver uppgift om författare, titel, ämne (eller ämnen) och konferens.

27. Författare till inlämnade *papers* , *referees* och ordföranden är automatiskt kvalificerade att få inbjudan till konferensen

28. Ett *paper* accepteras utifrån de sammanvägda betygen från *referees* eller efter ett beslut från programkommittéen

29. En konferens består av en eller flera sessioner.

30. En session är en obruten följd av presentationer.

31. Flera sessioner kan pågå parallellt.

32. En presentation handlar endast om ett *paper*.

33. En ordförande kan leda flera sessioner per konferens.

34. En person har en unik kombination av förnamn, efternamn och land.

35. Registrering av en person kräver förnamn,efternamn, titel och land.

36. Deadlines skall kunna registreras.

37. Varje session har ett sessionsnummer, som är unikt för konferensen.

38. För varje session skall registreras sessionsnummer, ordförande, tid och plats samt eventuellt datum och tid för publicering av sessionens sammansättning.

39. Registrering av en presentation kräver angivelse över sessionen, dess placering i sessionen, vilka ämnen den behandlar och presentationens längd.

2.3 Granskningsprotokollet

Detta granskningsprotokoll tar upp saker man bör titta på när man granskar ett CASE-verktyg. Under varje aspekt tar vi upp vilka saker man bör redogöra för och vilka frågeställningar som bör behandlas. Givetvis påverkar verktygets egenskaper vilka aspekter man kan eller ens är meningsfullt att behandla, men vi har försökt att välja aspekterna och frågeställningarna så att man kan göra en relativt heltäckande studie på alla typer av verktyg. Men CASE-verktygen utvecklas. De kommer att få nya egenskaper och erbjuda nya möjligheter. Därför avser detta protokoll inte att på något sätt att vara definitivt, utan modifieringar och tillägg kan införas när det känns befogat.

När man studerar ett CASE-verktyg kan det vara lämpligt att ha granskningprotokollet tillgängligt hela tiden och anteckna verktygets egenskaper med protokollet som stöd. Om någon facilitet tycks saknas, kontakta leverantören innan du skriver något definitivt. På detta sätt minskas risken för personligt tyckande och rapporten blir mera objektiv. Leverantören bör också få säga sitt om den färdiga rapporten, och därmed få chansen att rätta till eventuella felaktigheter och kommentera innehållet i rapporten.

GRANSKNINGSASPEKTER

1 Allmänt om Verktyget

- Berätta översiktligt om verktyget : verktygets utvecklare och leverantör, maskiner och systemmiljöer verktyget finns för, hur den är uppbyggd (vilka moduler det består av, modulernas uppgift, etc).

- Är verktyget metodorienterat eller ett generellt CASE-skal? (skall bara nämnas här, detaljerna behandlas senare)

2 Systemutveckling med verktyget

2.1 Verktygets "spännvidd"

- Vilka systemutvecklingsetapper avser verktyget att stödja enligt tillverkaren?

- Illustrera gärna med figur (exempelvis som den på sidan 14).

2.2 Beskrivningstekniker

- Redogör för vilka diagramtyper och andra specifikationssätt som finns i verktyget .

- Vilken typ av syntax eller regler har de olika diagrammen (exempelvis structure diagram enligt JSP-syntax)?

- kontrolleras syntaxen on-line eller på annat sätt?

- Hur arbetar man med dessa diagram?

- Hur uttrycksfulla/fullständiga är beskrivningsteknikerna?
- Går det att definiera egna beskrivningstekniker eller modifiera befintliga?
- Illustrera gärna med exempel på olika beskrivningstekniker!

2.3 Integrationsgrad mellan beskrivningstekniker;

Här skall man behandla integrationen mellan de beskrivningar som lagras i systemet. Följande frågeställningar skall belysas:

- Vilka kopplingar finns mellan diagrammen (eller motsvarande)?
- Är dessa kopplingar dynamiska, dvs medför en ändring i ett diagram automatiskt motsvarande ändring i alla beskrivningar som påverkas?
- Kan man öppna ett diagram inifrån ett annat, exempelvis för mera detaljerade beskrivningar (t.ex. pseudokod för en process)?
- Finns automatiska konverteringsfunktioner mellan olika diagram (t.ex. från entity-relationship-diagram till relationsdatabasdiagram)?
- Redogör även för vilken betydelse kopplingarna mellan diagrammen kan ha, exempelvis att man slipper definiera samma sak flera gånger i olika diagram?
- Finns någon form av automatisk ritning? (exempelvis när man överför specifikationer från ett diagram till ett annat , eller "text in -> bild ut")

2.4 Metodstöd;

- Är verktyget bundet till en systemutvecklingsmetod, eller finns det stöd för flera alternativa metoder? Eller är verktyget rentav ett s.k. CASE-skal där användaren själv har möjligheten att definiera sin metod och tillhörande begrepp och grafteknik?
- Hos en del verktyg kan beskrivningsteknikerna modifieras, och verktyget på så vis bättre anpassas till användarens behov. Beskriv dessa möjligheter om de finns!
- Systemutvecklingsarbetet består av många faser, och under olika faser kan olika metoder användas. Vilka faser eller metodsteg avser verktyget att stödja? Vilka konstruktionsmoment stöds?
- Praktiskt taget alla systemutvecklingsmetoder använder sig av ett antal beskrivningstekniker. Ibland kombineras och utnyttjas dessa tekniker på ett för metoden specifikt sätt. Om verktyget sätter begränsningar på hur och i vilken ordning de olika beskrivningsteknikerna får användas, kan detta hindra användningen av en del metoder, även om beskrivningsteknikerna för metoden finns. Beskriv därför de begränsningar som finns och vilka följder ett olämpligt arbetssätt kan få!
- Rekommenderas någon speciell metod eller metodkedja framför andra? (exempelvis för att möjliggöra smidig kodgenerering)

- Påverkas konsistens och fullständighetskontrollena (om sådana finns) av vilken metod eller metodkedja man väljer, och på vilket sätt i så fall?

- Beskriv hur verktygets grad av metodbundenhet kan påverka arbetet. Som stöd kan man använda följande resonemang:
  Om verktyget är bundet till en enda metod kan detta uppfattas som positivt eller negativt. Risken finns att kreativiteten kvävs eller att man blir låst till ett visst arbetssätt. Men det kan också vara till fördel genom att användaren tvingas till ett strukturerat arbetssätt som minskar risken för bristfälliga specifikationer och inkonsistenser. Om däremot flera metoder stöds, kan det finnas risk för inkonsistenser mellan specifikationer om metoder blandas, om dessa nu överhuvudtaget kan blandas. Men å andra sidan är verktyget mera flexibelt och man har större frihet att använda verktyget på sitt eget sätt.

- När det gäller CASE-skal kan man ta upp vilken metodstöd som kanske redan finns eller kan väljas vid leveransen. Beskriv också hur man bär sig åt om man avser fylla skalet med sin egen metod eller metodkedja. Finns begränsningar som hindrar en från att implementera sina egna metoder fullt ut? Hur strikt formaliserade skall metoderna vara för att kunna implementeras? Vilka typer av kontroller kan byggas in i metoderna?

Figurer som illustrerar metodkedjor och arbetsgången kan göra texten mera lättförståelig och ge bättre insikt i hur verktyget kan användas.

2.5 Begränsningar i och avvikelser från metoden

- Om verktyget stöder en speciell utvecklingsmetod, hur noggrant följs då stegen i metoden? Beskriv avvikelserna!

- Om metoden har svaga punkter, kan verktyget på något sätt överbrygga dessa?

- Det kan också vara tvärtom. Verktyget kanske inte tar vara på metodens alla möjligheter, utan lider av vissa begränsningar gentemot metoden. Om verktyget nu på något sätt avviker från metodens arbetssätt, så beskriv detta och ange om det i så fall är positivt eller negativt (motivera noga)?

3 Systemutvecklingsdatabasen

Man kan säga att systenutvecklingsdatabasen är den viktigaste delen av ett CASE-verktyg. Den består av designdata och regler för hur dessa hanteras och relateras till varandra. Den utgör grunden för alla viktiga operationer: skapande och lagring av designobjekt, modifiering av dessa, analysfunktioner och framtagning av dokumentation. När man talar om systemutvecklingsdatabasen kan man skilja på datakatalogen (den "egentliga" databasen), hur diagrammen relateras till innehållet i datakatalogen samt vilka analyser och rapporter man göra av den lagrade informationen.

3.1 Datakatalogen

- Vilken lagringsteknik används (relationell, hierarkisk eller annat)?

- Beskriv vad som lagras i datakatalogen, och hur data som lagras hänger ihop! Här är det lämpligt att rita ett konceptuellt schema över databasens innehåll, om möjligt.

- Uppdateras datakatalogen automatisk vid editering i diagrammen?

- Finns det direkt åtkomst till datakatalogens innehåll, t.ex. via ett editeringsfönster?

- Kan data importeras från respektive exporteras till verktyg av samma eller annan typ?

- Kan man kombinera datakataloger (delprojekt -> huvudprojekt) ?

3.2 Åtkomstsmöjligheter och konsistens mellan representationer

Här betraktar man datakatalogens förhållande till beskrivningsteknikerna. De punkter vi tar upp här nedan gäller för varje beskrivningsteknik för sig eller mellan beskrivningsteknikerna.

- Går alla ändringar via datakatalogen?

- Kan man editera direkt i datakatalogen utan att använda sig av någon beskrivningsteknik?

- Finns dynamisk koppling mellan verktygets olika delar via datakatalogen? (detta har redan delvis berörts under punkt 2.3)

- Är katalogen alltid konsistent med den grafiska presentationen i diagrammen?

- Finns någon möjlighet att se följderna av en ändring så att rimligheten/riskerna i denna kan kontrolleras?

3.3 Kontroll- och analysmöjligheter

- Vilka kontroller och analyser man kan göra, beror ofta till stor del på verktygets rapportmöjligheter. Det kan därför vara på sin plats att först titta på verktygets rapportfunktioner. Med utgångspunkt från dessa kan man sedan beskriva vilka typer av kontroller och analyser som kan göras för data i datakatalogen samt i och mellan diagrammen. Men det finns givetvis kontroller och analyser som inte ges i rapportform. On-line syntaxkontroll till exempel.

- Exempel på önskvärda kontroller är möjligheterna att hitta namn som inte stämmer överens vid dekomponering av diagram, objekt som inte är sammankopplade med andra, dataflöden eller strukturer som inte är definierade och dylikt.

- Rapporterna är även dokumentation, och därför är det viktigt att de är överskådliga, lättlästa och inte innehåller en massa överflödigt skräp. Det finns därför orsak att titta närmare på rapporternas utseende. Bifoga gärna exempel på rapportutskrifter.

3.4 Säkerhetsaspekter

- Vilka möjligheter finns för att miska risken för förlust av data (vid t.ex. spänningsbortfall)? Exempelvis en "autosave"-funktion!
- Finns möjligheter att återskapa eller korrigera data, s.k. "repair"-funktion?
- Hur hanteras behörighet för åtkomst och modifiering av data?

4 Användarmiljön

4.1 Gränssnitt mot användaren och interaktivitet

- Är det grafiska gränssnittet eget eller enligt något standardpaket (Macintosh, Windows, Gem osv) ?
- Beskriv verktygets gränssnitt mot användaren: fönster, menyer, mus, paletter för snabbval av funktioner.
- Hur går dialogen mellan användaren och systemet till? (t.ex inmatning av data i dialogboxar och/eller genom att direkt rita och skriva på skärmen)
- Kan man arbeta parallellt med flera olika beskrivningar t.ex. genom att ha flera fönster öppna samtidigt?
- Kan man "navigera" mellan olika beskrivningar och gå upp och ner i beskrivningshierarkier?
- Ser användaren direkt följden av sina handlingar? (t.ex) omritning av figurer då beskrivningar ändrats?
- Varnar systemet då användaren försöker göra otillåtna eller farliga operationer?
- Finns ångerfrist vid riskfyllda operationer?

I det här avsnittet är det mycket illustrativt att exemplifiera med ett antal skärmbilder!

4.2 Hjälpfunktioner

- Beskriv vilken hjälp som finns att få under ett arbetspass och hur man når denna hjälp!
- Hur "intelligenta" är hjälpfunktionerna, dvs ger verktyget automatiskt den hjälp användaren för tillfället behöver ? Exempel på sådan hjälp är när ett fel har inträffat och verktyget föreslår alternativ a åtgärder för att avhjälpa felet, eller när verktyget ger besked vad användaren väntas göra härnäst och vilka operationer som han/hon kan välja mellan.

4.3 Fleranvändarmiljön

I de fall där verktyget kan användas i fleranvändarmiljöer, kan man ta upp saker som:

- ändringsskydd
- åtkomstkontroll
- konsistens vid samtidig användning och uppdatering av designdata av olika personer.

5 Egenskaper och faciliteter hos verktyget

5.1 Portabilitet

- Kan de färdiga specifikationerna överföras till andra miljöer och hur lätt/svårt är det?
- Är det färdiga systemet körbart i flera miljöer? (i de fall där möjlighet till kodgenerering finns!)

5.2 Kommunikation

Här tar man upp saker som kan vara intressanta att betrakta i kommunikationssammanhang. Ett par exempel här nedan:

- Vilka andra system (t.ex. databashanterare) kan de färdiga applikationerna kommunicera med?
- Beskriv kommunikationen och nätverksstödet i de fall man har en central datakatalog med flera arbetstationer som kan ha sina egna datakataloger!

5.3 Transformation till och från databasstruktur

- Finns möjligheten att transformera sin modell och specifikationer till en databasstruktur för en databashanterare?
- Finns möjligheten att överföra en databasstruktur tillbaka till specifikationer? (Detta är ett exempel på ett s.k. "reverse engineering")

5.4 Kodgenerering

- Kan verktyget generera kod, och i så fall vilken typ av kod (3GL, 4GL)?
- Hur fullständig är koden? (körbar, programskelett, bara deklarationer)
- Vilka miljöer kan kod genereras för?
- Hur går kodgenerering till?

5.5 Prototypframställning

- Vilken (vilka) form av prototyp kan verktyget framställa? En variant är möjligheten att generera exekverbar kod från kanske ännu ofullständiga specifikationer. En annan variant är simulering av skärmdialoger och rapporter. Det kan också vara så att verktyget inte har någon speciell prototypframställning, men kan ändå användas för detta syfte tack vare enkla kodgenereringsmöjligheter.

- Diskutera också på vilket sätt verktygets prototypmöjlighet kan påverka konstruktionsarbetet och det färdiga systemet! Exempelvis kan slutanvändarna vara mera aktiva under konstruktionsfasen och se till att systemet bättre uppfyller deras behov. Det färdiga systemet kanske får högre kvalité genom att verktyget kan testas på ett tidigt stadium och fel upptäckas och åtgärdas i god tid.

- En egenskap som kan vara värdefull vid konstruktion av komplexa system är möjligheten att konstruera små men fungferande testsystem som sedan successivt kompletteras och utökas till fullständiga system. På så vis kan kan systemets funktioner kontinuerligt valideras under konstruktionsfasen och risken för kostsamma fel eller brister i det färdiga systemet minskar. Det kan därför vara värt att titta på om verktygets prototypingmöjligheter kan användas på detta sätt.

5.6 Återanvändning av specifikationer och kod

- Kan delar av specifikationer och kod exporteras (importeras) och att återanvändas i andra sammanhang?
  Hur enkelt är det anpassa specifikationer för andra sammanhang?
  Kan man söka och hitta de specifikationer som kan återanvändas?

- Konsistens- och fullständighetskontrollerna kan visa sig värdefulla vid anpassning av specifikationer, så här finns möjligheten att se på de tidigare diskuterade kontroll- och analysmöjligheterna i nytt ljus!

5.7 Versionshantering

- Versionskontroll kan betyda olika saker i olika sammanhang. Följande tolkningar kan behandlas:

  1. Olika versioner av systemspecifikationer, vilket kan betyda alternativa versioner eller olika gamla versioner av systemspecifikationer.

  2.Verktyget kan anpassas så att det kan finnas flera olika versioner av själva verktyget. Kan orsaka kompatibilitetsproblem mellan verktygsversionerna?

  3. Olika verktygs-'releases', dvs olika gamla versioner av verktyget. Kan medföra att konvertering av gamla specifikationer måste göras för att de skall kunna användas i nyare verktygsversioner.

5.8 Projektplaneringsstöd

- Vilket stöd för projektplanering ger verktyget?

- Kan fristående projektplaneringsprogram integreras med verktyget?

- Här skall man ta hänsyn till att goda rapportfunktioner kan hjälpa vid projektuppföljning!

5.9 Flexibilitet

När man säger att ett verktyg är flexibelt menar man att det kan anpassas och utformas efter olika behov. Verktyget kan också vara så mångsidigt redan från början att det kan användas på olika sätt utan några ändringar.

Verktyget kan bestå av moduler, så att det kan utformas genom att byta ut eller lägga till moduler med önskade funktioner. Många verktyg kan integreras med andra mer eller mindre frisrående produkter, och på så vis byggas ut med funktioner som det annars inte finns stöd för. Ett verktyg som saknar projektstyrningsfaciliteter kanske kan integreras med en separat produkt med projektstyrning. Ett annat alternativ är att verktyget genererar pseudokod som en annan produkt i sin tur kan ha som indata för kodgenerering eller prototypframställning.

- Beskriv hur verktyget kan anpassas och/eller byggas ut vid behov!

6 Hur påverkar verktyget systemutvecklingsprocessen

6.1 Vinster i tid och kvalitet

Hur stora vinster i tid och kvalitet man kan uppnå genom att använda ett CASE-verktyg vid systemutveckling är förstås svårt att uppskatta. Dessutom beror ju både kvalitets- och tidsvinsterna på vilka verktyg och metoder man använt sig av tidigare, och även hur CASE-verktyget självt används i företaget.

Det kan också vara så att man samtidigt med införandet av CASE-verktyg även har övergått till mera formell och enhetlig arbetssätt vid systemkonstruktion, och då blir hart när omöjligt att säga något säkert om hur mycket av produktivitets- och kvalitetsökningarna härrör från själva CASE-verktyget. En stor del av vinsterna har antagligen sitt ursprung i effektivare arbetssätt.

Det som förmodligen mest påverkas kvalitetsmässigt vid utnytjande av CASE-verktyg är dokumentationen. Vid automatisk dokumentering kan säkert även stora tidsbesparingar göras, men inte heller här kan man göra några exakta uppskattningar. Man kan även uppnå vissa tidsvinster tack vare effektivare projektadministration. Det är lättare att följa upp projektets framskridande eftersom datakatalogen hela tiden innehåller den aktuella beskrivningen av systemet.

Man skall alltså inte fälla några definitiva omdömen, när man granskar verktyget ur dessa synvinklar, utan hellre se på vilka vilka sätt verktyget kan möjligöra ökad effektivitet och kvalitet vid olika arbetsmoment samt systemutvecklingsprocessen som helhet.

6.2 Effekterna på informationssystemets livscykel

Dom flesta CASE-verktyg har sin tyngdpunkt vid modellering av informationssystem med hjälp av olika analys- och designtekniker. Det finns dock stora skillnader mellan verktygen, när det gäller andra systemutvecklingsfaser. Här skall man ta upp hur de olika faserna påverkas genom användning av verktyget. Vi har redan berört detta i avsnitt 2, men skall här betrakta det hela ur vidare perspektiv.

Effektiv användning av ett heltäckande CASE-verktyg kan påverka informationssystemet livscykel tämligen radikalt jämfört med det traditionella arbetssättet. De inledande faserna med analys och planering kommer att få större vikt medan realiseringsdelen med programmering minskar avsevärt. Även förvaltningsdelen kan ändra karaktär i och med att ändrade informationskrav kan återföras till specifikationer med möjlighet till generering av ny kod. Även om kod kan genereras så blir det lättare att hålla specifikationerna aktuella och korrekta vilket underlättar programunderhållet.

De frågeställningar som man berör här är alltså:

- Vilka faser i informationssystemets livscykel påverkas och i vilken grad?
- Ger verktyget möjlighet att snabbt omforma och utbygga redan befintliga system vid exempelvis organisationsförändringar eller ändrade informationskrav?

6.3 Slutanvändarnas roll i utvecklingsarbetet

Här diskuteras på vilka sätt slutanvändarna kan involveras i systemutvecklingsarbetet genom användning av verktyget, samt vilken betydelse detta kan ha för den slutliga produkten.

6.4 Utbildningsbehovet

Hur pass effektivt ett CASE-verktyg kan utnyttjas i företaget beror mycket på kunskaperna om verktyget och dess möjligheter för dem som använder det. Vilka som använder verktyget har också en viss betydelse. Ett extremfall är då bara "experterna" på systemavdelningarna eller hos konsultfirman använder verktyget. I det motsatta fallet tar slutanvändarna över och utvecklar sina egna system. Båda dessa alternativ har sina brister, det senare verkar inte heller realistiskt i dagsläget. Men de belyser i alla fall hur utbildningsbehovet beror på vilka som använder verktyget. Om folk på systemavdelningarna står för användningen av verktyget är det bara ett fåtal som behöver utbildas. I det senare fallet måste även slutanvändarna utbildas.

Hur omfattande utbilning som krävs beror dels på användarnas kunskapsnivå, dels på verktyget. Oftast har systemutvecklingspersonalen tillräckliga teoretiska/praktiska kunskaper. När det gäller slutanvändare beror mycket på deras bakgrund och utbildning.

Om verktyget är komplext och använder sig av komplicerade metoder kan man inte vänta sig att slutanvändarna skall lära sig verktyget, även om de

kan involveras i arbetet. Om verktyget däremot har ett enkelt och lättförståeligt gränssnitt och metoden är lätt att lära sig, kan man effektivare dra nytta av slutanvändarnas kunskaper genom att låta dem aktivt delta i analys- och designarbetet.

Om verktyget är lättanvänt, kan det räcka med att ge ett fåtal en ordentlig utbildning i verktygets användning och dess möjligheter. Dessa personer kan sedan assistera andra och se till att verktyget används på ett korrekt sätt. Man kan å andra sidan tänka sig att alla som använder verktyget behöver en ordentlig utbildning, för att kunna arbeta effektivt.

En sak som lätt glöms bort då man pratar om utbildningsbehovet, är hur ofta utbildningen behöver förnyas. Ett omfattande och mångsidigt verktyg kanske kräver att man då och då fräschar upp sina kunskaper, särskilt om man inte använder verktyget regelbundet. Även nya versioner av verktyget kan ge ett behov av vidareutbildning.

Man skall inte heller låta bara en person stå för all kunskap om verktyget. Det är illa om man blir beroende av en eller två personer på systemavdelningen, eller ännu värre, man blir beroende av externa konsulter. Det är därför viktigt att verktyget är av sådan komplexitet att det lätt kan läras och behärskas av folk med kunskaper i systemutveckling.

De huvudsakliga aspekterna som man skall diskutera under denna punkt kan sammanfattas med följande frågor:

- Vilka förkunskaper bör användarna ha?
- Hur stora utbildninginsatser kräver effektiv användning av verktyget?
- Räcker det med en väl insatt person som assisterar andra, eller behöver alla inblandade utbildas?
- Hur ofta behöver utbildningen förnyas?
- Finns risk för att bli beroende av experter/konsulter?
- Vilken utbildning finns att få för verktyget?
- Vilken utbildning finns att få för metoden?

7 Hård- och mjukvarukrav, kostnader och dokumentation

- Vilka maskiner och operativsystem behövs!
- Investeringarna i hård- och mjukvara som kan bli aktuella vid anskaffning av verktyget. Man kan även ta upp kostnaderna för verktyget.
- Beskriv också vilken dokumentation som följer med verktyget, och hur pass bra denna dokumentation är. Saker som man bör beröra är hur heltäckande dokumentationen är, hur lätt att läsa och hitta i, om det finns bra index och om det finns förklarande exempel.

8 Avslutande översikt

8.1 Sammanfattning av verktyget

- Gör en kort sammanfattning av verktyget och dess disposition i systemutvecklingsprocessen samt vilka verktyget vänder sig till.
- Ta upp de mest uppenbara för- och nackdelarna, men gör det försiktigt och relatera dessa till behoven och vad verktyget är konstruerat för!

8.1 Verktygets framtida utveckling

- Diskutera vilka förbättringar och ändringar som efterlyses, samt motivera dessa önskemål!
- Tala också om vilka förbättringar och ändringar som skall genomföras i kommande versioner, samt vilka ändringar det finns planer på.

8.2 Personliga intryck

Man har givetvis fått personliga intryck av verktyget under granskningsarbetet. I punkterna ovan har man försökt betrakta verktyget objektivt ur alla möjliga synvinklar, men här har man möjligheten att berätta om vad man själv tycker om verktyget, hur det känns att arbeta med det och vilka förbättringar man vill föreslå. Man skall dock betona att det är fråga om subjektiva omdömen.

3 STUDIE AV DEFT

Det kan tyckas att de egenskaper som vi tar upp när vi granskar ett verktyg som Deft är väl ambitiösa. Men det är dock knappast något enskilt verktyg som har alla dessa egenskaper. Meningen med våra granskningaspekter är att ge en informativ beskrivning av ett verktyg ur så många synvinklar som möjligt.

3.1 Allmänt om Deft

Deft är ett CASE-verktyg som utvecklats av det kanadensiska företaget TSB International Inc och marknadsförs i Skandinavien av Infotool AB. Deft körs på Macintosh och utnyttjar dess grafiska användargränssnitt. Designresultatet från Deft kan överföras till 4GL-miljön för vidare bearbetning. Den 4GL-miljö som Deft kan kombineras med under skrivande stund är Ingres på Vax-maskiner, men enligt utsago skall kopplingsmöjligheter även till andra verktyg komma. Deft binder sig inte till någon speciell systemutvecklingsmetod även om den använder sig av beskrivningstekniker som lämpar sig för Structured Analysis och Structured Design.

Deft består av flera moduler som stödjer olika faser i systemutvecklingsprocessen. Modulerna är ihopkopplade via en gemensam datakatalog. De moduler som finns i Deft version 3.0 är:

Data Flow Diagram Editor, DFD:
Grafisk editor för att konstruera dataflödesdiagram med antingen Yordons- eller Gane&Sarsons notationssätt.

Program Structure Diagram Editor, PSD:
Hjälpmedel för konstruktion av systemets programkomponenter. Programstrukturdiagram kan ses som blandning av Jackson Structure Diagram och Structure Chart.

Entity Relationsship Diagram Editor, ERD:
Hjälpmedel för att beskriva dataobjekt (tabeller), vyer (eng. view) och relationer mellan dem.ERD är en grafisk editor som stödjer datamodelleringsprocessen.

Form Editor:
Ett verktyg för grafisk beskrivning av dialogutformning och skärmlayouter.

Kompilator:
Kontrollerar konsistens, syntax etc. i framtaget resultat och generar sk objektfiler.

Translator:
Överför objektfiler som har genererats av kompilatorn till Ingres-miljö där man sedan får tabeller och formulärer automatiskt genererade.

Central Dictionary Reference:
En integrerad datakatalog för och lagring av datadefinitioner och objektbeskrivningar (dataflöden, entiteter, processer osv.).

De grafiska editorerna DFD, PSD, ERD och Form Editor använder sig av samma teknik som andra Macintosh applikationer (paletter, dialogboxar, editeringsfunktioner som cut, paste, select osv)
Den integrerade datakatalogen innebär att alla attribut och alla objekt som skapas i systemet automatiskt kommer att hamna i datakatalogen. Detta gör att alla attribut finns tillgängliga i alla moduler. På så sätt kan man skapa och definiera attribut i en modul och återanvända eller modifiera dessa definitioner i andra moduler.

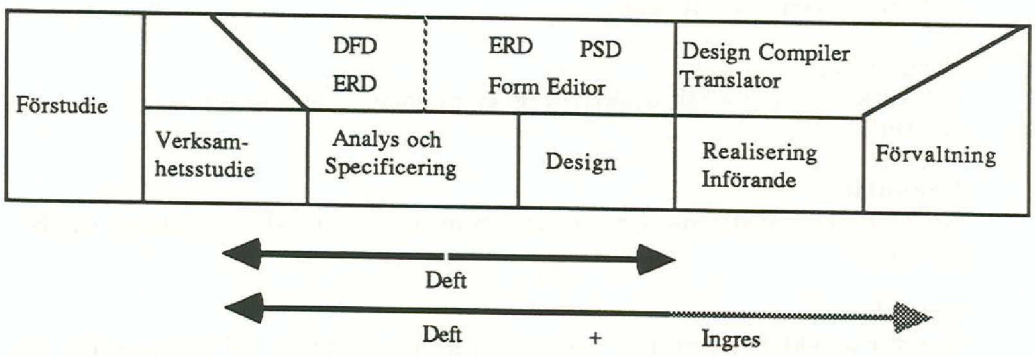
3.2 Systemutveckling med Deft

3.2.1 Defts spännvidd

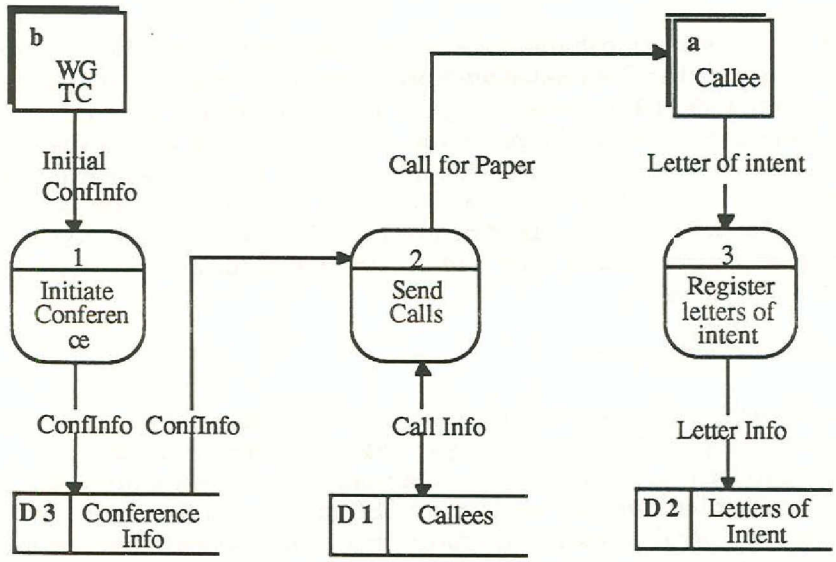
De modulerna som finns i Deft kan ge stöd till de stora och viktiga faserna i systemutvecklingsprocessen, nämligen analys, design och systemsdokumentation. Viss stöd för verksamhetsstudie kan finnas i form av dataflödesdiagram som kan användas för beskrivning och analys av verksamheten. Men om man integrerar Deft med Ingres så kan de tillsammans ge stöd för hela systemutvecklingsprocessen från verksamhetsstudien till implementering. De grafiska presentationerna kan medverka positivt även i systemförvaltningen, eftersom det blir lättare att överskåda ett system och eventuellt införa förbättringar i den.

3.2.2 Beskrivningstekniker

Som vi redan nämnt görs modellering i Deft med tre olika diagramtyper dataflödesdiagram, entity-relationship-diagram och programstrukturdiagram. Dataflödesdiagram kan användas för grafisk beskrivning av verksamheten, men används oftast för att beskriva systemet med hjälp av logiska processer och data som flyter mellan dem. Med entity relationship-diagrammet beskriver man egenskaper hos dataelementen och relationerna mellan dem. Programstrukturdiagram kan användas som designstöd vid programkonstruktion. Vi skall titta litet närmare på dessa beskrivningstekniker.



Figur 3.1 Deft och Ingres i relation till faserna i systemutvecklingsprocessen



Figur 3.2 Ett dataflödesdiagram enligt Gane&Sarson. Här betraktas WG som en datakälla som levererar data till systemet, Callee är både en datakälla och en datadestination.

Dataflödesdiagram:

Ett dataflödesdiagram används för verksamhetsanalys och systemdesign. Diagrammet visar hur olika aktiviteter som sker i systemet är relaterade till varandra och hur data flyter mellan dem. Vid konstruktion av dataflödesdiagram kan representationssätt beskrivna av E. Yourdon eller Gane&Sarson användas. Båda presentationssätten är logiskt identiska och använder sig av hierarkisk orienterade dataflödesdiagram . Skillnaden ligger huvudsakligen i att man presenterar grafiska objekt på något olika sätt. Ett dataflödesdiagram kan konstrueras utifrån följande komponenter:

1 Extern entitet:
De instanser eller enheter utanför systemet som tar emot eller/och levererar data. Man antar att sådana entiteter existerar redan och att man har möjligheter att kommunicera med dem . Varje entitet har ett namn eller en identifierare som är unik i systemet. I Yourdon Diagram representeras varje entitet med det namn som entiteten får av användaren, medan i Gane&Sarson diagram förses varje entitet med en identifierare som skapas automatiskt av systemet.

2 Dataflöde :
Är en pipeline genom vilken ett eller flera informationspaket skickas mellan externa entiteter, processer och datalager. I Deft finns det två typer av dataflöden, reguljära dataflöden som innebär att bara ett informationspaket kan överföras genom den, och sammansatta dataflöden som innebär att flera informationspaket kan överföras genom den.

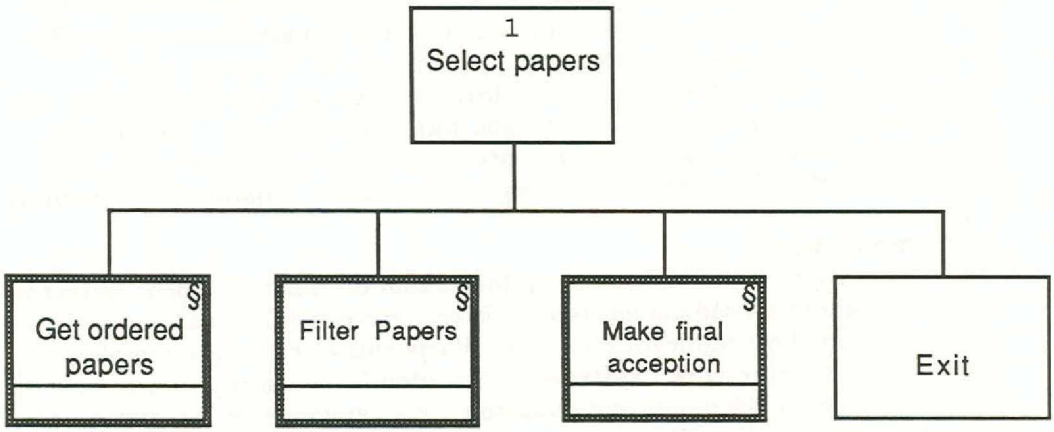
3 Datalager:
Förvaringsplats för data i systemet för vidare användning. I Gane&Sarson notationssätt identifieras ett datalager med ett unikt värde som automatiskt skapas av Deft och det namn den får. I Yourdon notationssätt identifieras varje datalager med det namn det får av användaren.

4- Process:
Innebär någon form av behandling eller modifiering av data. En process representerar en eller flera aktiviteter och identiferas ofta med ett lexikaliskt namn som tyder på processens uppgift. Dessutom förses varje process med en siffra som automatiskt genereras av systemet. Om en process representerar flera aktiviteter så kan man öppna ett nytt dataflödesdiagram på lägre nivå för att närmare beskriva processens delaktiviteter. På samma sätt som i datalager, identifieras varje process i Deft med ett unikt värde (i Gane&Sarson notationssätt) eller med det namn den får av användaren (i Yourdon notationssätt).

Programstrukturdiagram :

Program Structure Diagram Editor (PSD) syftar till att stödja grafisk design av systemets programstruktur och logik. PSD använder sig av en beskrivningsteknik som följer Jackson Structured Programming (JSP). Ett programstrukturdiagram kan ses som ett JSP-diagram utökat med egenskaper från strukturkartor. En fördel med denna kombination är att man kan specificera parameteröverföring mellan programrutiner. Detta saknas i JSP-diagram men finns i strukturkartor.

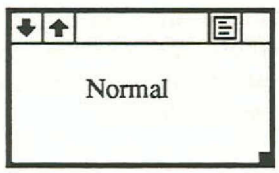
Ett sätt att använda PSD i Deft är att för processer i dataflödesdiagrammet skapa ett programstrukturdiagram, som beskriver processen med hjälp av programrutiner. Ett annat sätt att utnyttja PSD är att designa rutiner kopplade till den kod som ligger bakom ett formulär, eftersom dessa inte genereras automatiskt av Deft, utan måste skrivas i exempelvis Ingres högnivåspråk.



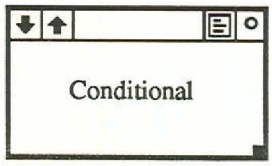
Figur 3.3 Programstrukturdiagram. Processen Select Papers består av tre huvuduppgifter, från vänster till höger. Detaljerad beskrivning för dessa finns i subprogram med samma namn.

Det finns fyra olika typer av rutiner:

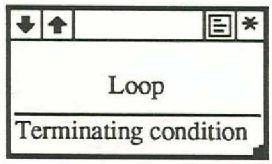
1. Normala rutiner som exekveras sekvensiellt varje gång programkontrollen når dem.



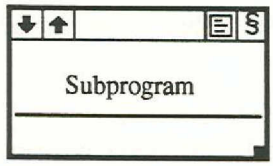
2. Selektioner som exekveras bara när det tillhörande villkoret är uppfyllt. Sådana rutiner existerar i grupp av minst två och som uteslutar varandra.



3. Iterationer som exekveras ett antal gånger tills tillhörande termineringsvillkor blir uppfyllt.



4. Subprogram som är ett anrop på ett annat program som har skrivits separat.



PSD tillåter inte avsteg från JSP-syntaxen så länge man inte kopplar ur denna funktion. Man kan t.ex inte placera ut iterationer och selektioner så att det strider mot JSP-syntaxen. Kopplingen mellan olika programstrukturdiagram sker antingen lokalt eller globalt. Med lokalt menas att om en rutin består av flera subrutiner så kan man "samla ihop" dessa till ett ny tt diagram. Detta nya diagram kan endast nås från "faderrutinen". Med global koppling menas att en rutin av typ subprogram kan referera till ett annat befintligt programstrukturdiagram.

Entity-relationship-diagram:

Ett entity-relationship-diagram beskriver statiska relationer mellan entiteter i systemet. I Deft skiljer man mellan logiska och fysiska datamodeller. Ett sätt att använda den logiska modellen är att beskriva en fullständigt normaliserad datamodell, ett annat sätt är att beskriva objekt och relationerna mellan dem som de ser ut i verkligheten. Den fysiska modellen representerar datamodellen i systemet (man kan göra vissa avsteg från normaliseringsreglerna) som ligger sedan till grund för uppläggning av databaser i t.ex. Ingres.

I ERD-editorn används tre presentationstekniker definierade enligt Chen/Bachman, Martin eller IRM. Vilket man väljer saknar betydelse för datamodelleringsprocessen. Oavsett vilken presentationsteknik som används så kan man välja mellan tre typer av frågespråk: SQL, QUEL eller odefinierad. Vilket av dessa man väljer beror på den miljö i vilken det färdiga systemet skall användas. Man kan t.ex generera SQL-kod från Deft

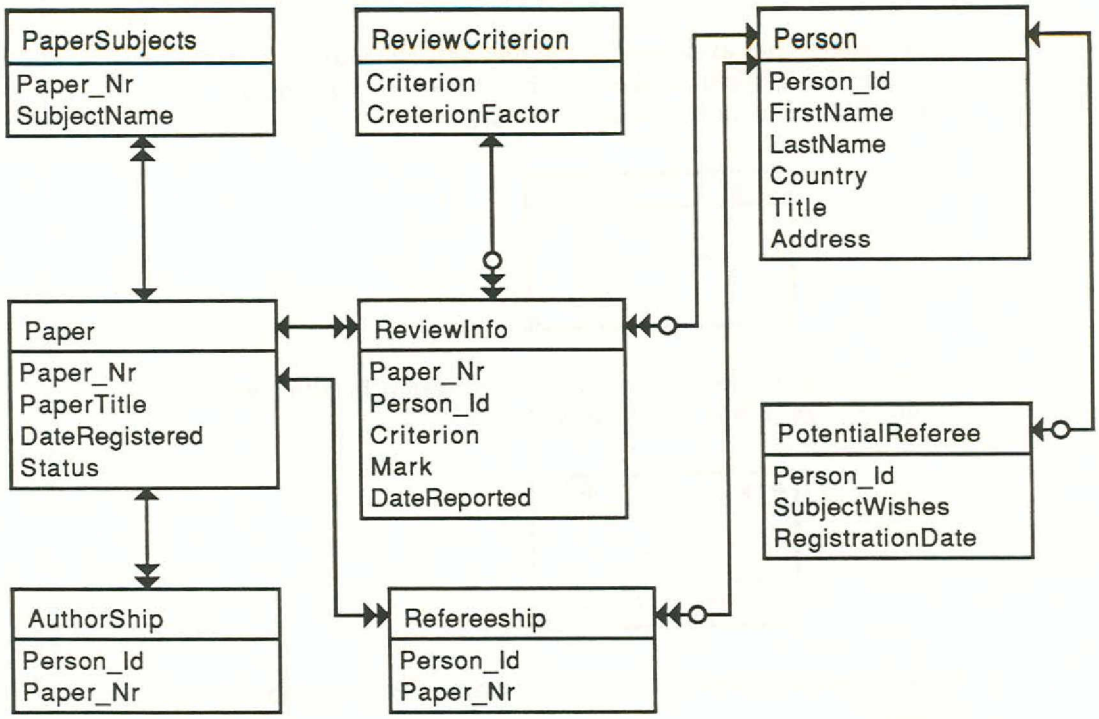


Fig 3.4 Exempel på ett ERD-diagram ritat enligt Chan/Bachmans-notationssätt.

för ett SQL-baserat 4GL-verktyg. Vi kommer att beskriva hur detta går till i senare avsnitt.

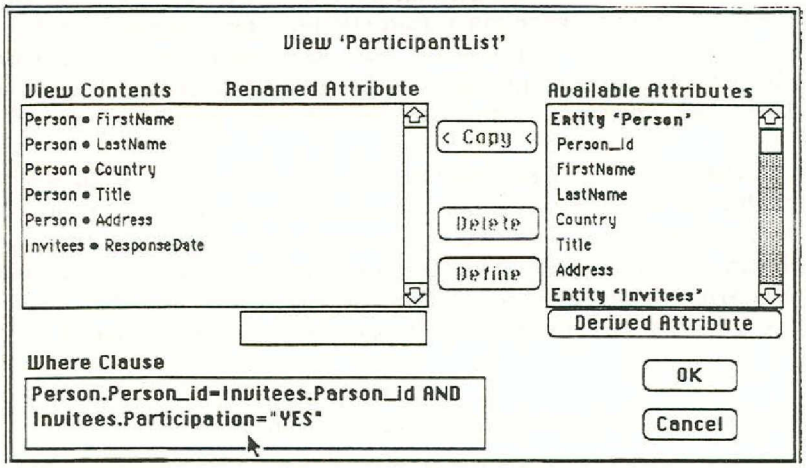
Ett entity-relationship-diagram består av tre komponenter:

1. Entitet:
En entitet i ERD presenterar ett objekt (tabell) och dess attribut. Man antar att det förekommer minst en instans av denna entitet i systemet. Varje objekt förses med attribut som antingen skapas i ERD eller hämtas från datakatalogen

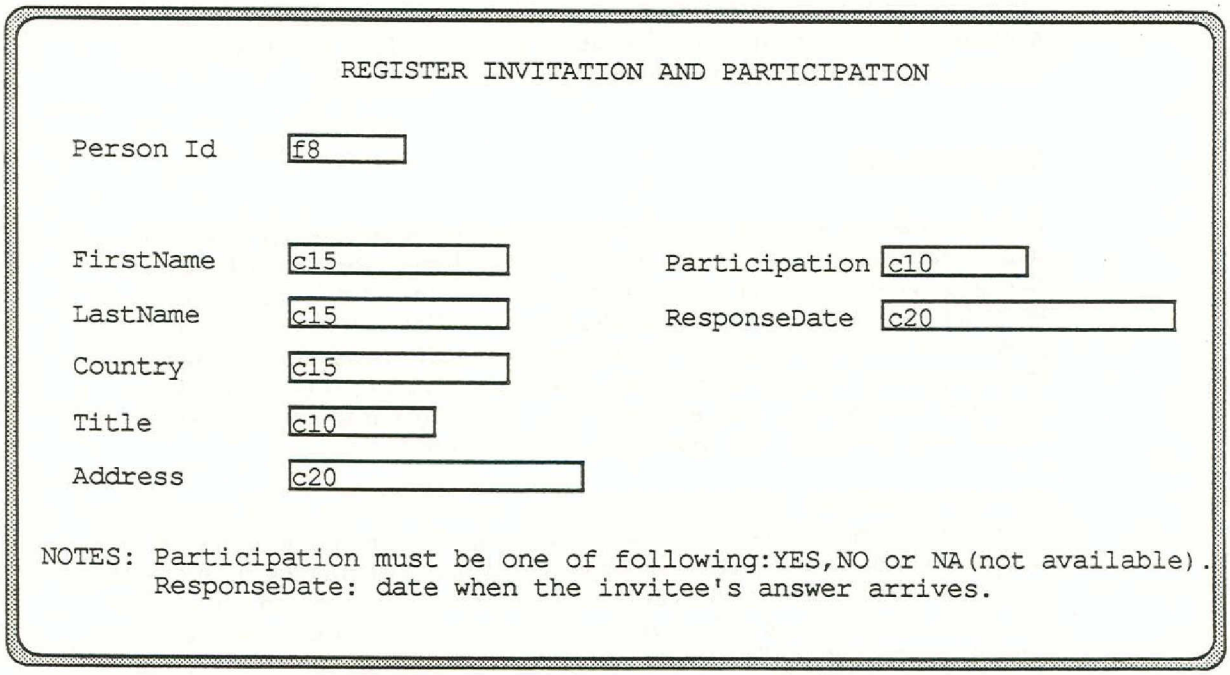
2. Relation:
Relationer i Deft's ERD har en annan innebörd än i traditionella datamodeller. En relation är ett samband mellan två entiteter som ej instansieras (presenterar en tabell i systemet). En relation kan även vara ett samband mellan en entitet och en vy (view).

3. Vy:
En vy är en kombination av attribut från två eller flera tabeller. För att ange vilka attributvärden som är giltiga används ett s.k. matchningsvillkor som motsvarar SQL:s where-klausul. Vyer kan användas för att representera informationskrav, t.ex. rapporter. Vyer innehåller inga "egna" attribut utan de kan alltid härledas utifrån innehållet i tabeller, antingen genom direkt kopiering eller genom härledningsfunktioner.

Ett fullständigt specificerat entity-relationship-diagram kan användas för automatisk generering av ett relationschemat som med hjälp av Design Compiler och Translator kan överföras till en Ingres-miljö.



Figur 3.4 Figuren visar en dialogruta för specificering av en vy.



Figut 3.5 Skärmlayout för registrering av inbjudna och deras deltagande i konferensen.

Formulärer:

Avsikten med Form Editor är att en systemutvecklare på ett enkelt och snabbt sätt skall kunna ta fram skärmbilder, och sedan låta slutanvändaren se hur dessa har utformats och föreslå eventuella ändringar.

Med Form Editor kan man måla skärmbilder för visning och inmatning av data. Varje visnings- och inmatningsfält kan kopplas till ett attribut i en tabell i datamodellen. Kopplingen sker via Datadictionary som länkar ihop fälten i skärmdefinitionen med termer i tabellerna. Man kan specificera valideringar och definiera hur formulärdialogen skall utformas. Förklarande hjälptexter för hur användaren ska använda formulären kan skrivas på skärmen.

När en skärmbild är färdigdefinierat, kan den köras genom Design Compiler. Om inget fel har inträffar så skapas en objektfil som kan överföras till central miljö för uppläggning av förmulärer För varje fält på skärmbilden skapas en variabel som ofta är en post . Elementen i denna post används bla för att definiera fältpositionerna på skärmen och fältets datatyp. Dessa variabler kan senare användas i Ingres 4GL-programspråk som programmeringsstöd.

3.2.3 Integrationsgrad mellan beskrivningstekniker

Integrationsgraden mellan beskrivningstekniker i Deft är låg. Den enda integrationen mellan diagrammen i Deft är att objekt definierade i ett diagram kan senare utnyttjas i andra. Man slipper därmed besväret att definiera samma objekt på flera olika ställen. Det finns t.ex. inga möjligheter att inifrån ett diagram öppna andra typer av diagram för mera detaljerad beskrivning av ett objekt. Till exempel vill man gärna kunna öppna ett PSD för en process i ett dataflödesdiagram för att beskriva dess programlogik. Man kan inte heller relatera en rutin i PSD till en skärmbild eller en datastore i DFD med en eller flera entiteter i ERD.

3.2.4 Metodstöd

Deft kan klassas som ett metodlös CASE-verktyg. Eftersom det inte finns några restriktioner på var eller hur man skall börja arbeta med Deft innebär detta vissa risker för inkonsistens. Man kan tex först börja med datamodellen och avsluta med dataflödesdiagram eller tvärtom. Man kan även presentera ett informationskrav som en vy i entity-relationship-diagram, som en process i dataflödesdiagram eller som en rutin i programstrukturdiagram. Det här kan man göra eftersom Deft följer inte någon speciell metod, som omfattar alla systemutvecklingsfaser.

För vårt praktikfall, IFIP-arbetskonferens, använde vi oss av en metod som liknar Structured Analys And Stuctured Design.En förutsättning för att kunna använda denna metod effektivt är att man har tillgång till datorstöd för administration av det stora antalet datatermer som metoden ger upphov till. I vår lösning finns cirka hundra datatermer som förekommer i ett eller flera diagram. Det skulle kanske inte vara stort problem att hålla reda på dem under kort tid utan datorstöd, men om arbetet pågår under längre tid eller systemet är stort, blir det manuella arbetet tidskrävande. Den snabba on-line accessen till datadefinitioner i Deft ger god stöd för den metod vi använde.

Som stöd för Structured Analys och Stuctured Design kan Defts Design Compiler användas. Med hjälp av den kan man göra en syntaxanalys för att kontrollera om avsteg från Yourdons regler för konstruktion av dataflödesdiagram har gjorts. Ett exempel dessa regler är "lagen om datakonservation", dvs att all data som flyter in i ett datalager måste flyta ut. Deft har tillämpat denna regel genom att låta Design Compiler analysera dataflödesdiagram och generera ett felmeddelande om regeln inte efterföljs.

Möjligheten till nedbrytning av processer i dataflödesdiagram och modularisering av program med programstruktudiagram ger stöd för Structured Analysis och Structured Design

Ett exempel på hur Deft stödjer tekniken för nivåorienterad design är den fullständiga kopplingen mellan nivåerna i ett dataflödesdiagram. Man kan skifta mellan de hierarkiska nivåerna på ett enkelt sätt, och komponenter på den högre nivån kan kopplas till komponenter på den lägre nivån.

Som vi har redan konstaterat, kan Deft klassas som ett metodlöst CASE-verktyg. Frånvaron av en metod kan ibland ses som en fördel, eftersom man inte behöver låsa sig till en viss metod utan kan kreativt använda verktyget och fritt utnyttja dess beskrivningstekniker. Men det kan vara kunskapskrävande att undvika inkonsistens i specifikationerna. Användaren måste arbeta disciplinerat och känna till flera metoder och även likheter och skillnader mellan dem.

Det finns även nackdelar i att en väldefinierad metod saknas. Det blir svårare att samordna ett projekt om de som utvecklar ett system använder sig av olika metoder. Det blir svårare att veta var man skall presentera de olika informationskraven, eller på vilket sätt de skall presenteras. Även kopplingen mellan de olika faserna i systemutvecklingsarbetet kan bli oklar, eller det kan bli en överlappning mellan olika faser.

The Deft Way

I slutet av vårt granskningsarbete fick vi tillgång till ett utkast till en bok som heter "The Deft Way". Boken beskriver en lämplig metod för att använda Deft. Metoden påminner rätt mycket om Structured Analys och Structured Design.

Första steget i metoden är att analysera kraven på det nya systemet och skapa dess funktionella specifikationer i samarbete med slutanvändare. Som stöd för det steget används både DFD och ERD.
Dataflödesdiagrammet används som grund för framtida design av systemet. Med hjälp av dessa diagram analyseras både det gamla och alternativ för det nya önskade systemet . För det nya systemet beskriver man översiktligt systemets uppbyggnad med ett dataflödesdiagram på relativt hög nivå. Med entity-relationship-diagram definierar man vilka entiteter som kommer att finnas och relationerna mellan dem. Observera att man ännu inte behöver ta hänsyn attribut som varje entitet skall ha.

Det andra steget är att uppskatta tidsåtgången och kostnaderna för det nya systemet (ev. flera olika alternativ). Följande arbetsgång föreslås:

- Dela systemet i subsystem och sedan konstruera dataflödesdiagram för varje subsystem.

- Definiera vilka processer som skall finnas i varje dataflödesdiagram och deras uppgift i systemet.

- Programstrukturdiagram används här för att specificera vilka funktioner som processerna skall utföra i form av PSD-rutiner (tasks).

- Uppskatta hur många kodrader man behöver skriva i vare rutin. Om rutinen är stor så kan man bryta ned den i flera subrutiner för att underlätta uppskattningen.

Dessa uppskattningar används för att beräkna tidsåtgången, personalbehovet ungefärliga kostnader för projektet. Om man beslutar att genomföra projektet och bygga ett nytt system enligt några av de möjliga alternativen går man vida till nästa steg.

Det tredje steget är systemdesign som genomgörs i två faser:

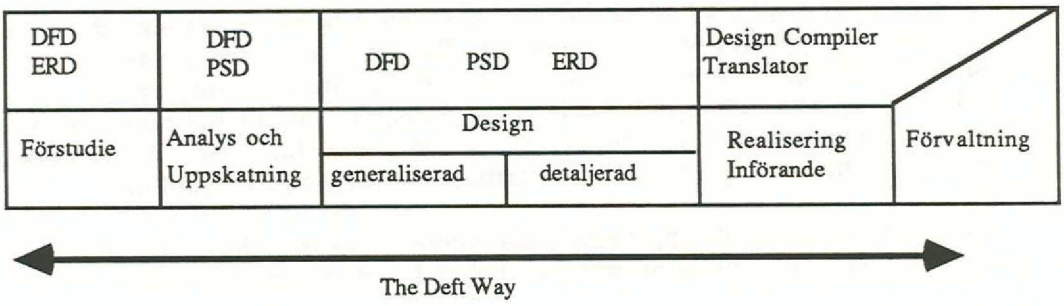
- Översiktligt design: Innebär en generalisering av de tidiga systemsskisserna som tagits fram i första och andra steget. Här börjar man ta hänsyn till hur data skall definieras och bearbetas. Slutanvändaren får vara med redan i detta stadium och föreslå dialogutformning. Slutanvändarens förslag kan sedan användas som underlag i senare arbetsmoment.

- Detaljerad design: Här görs en detaljerad analys av de översiktliga processerna i systemet och systemfunktionerna utformas. Syftet med detta är att eliminera eventuella inkonsistenser och beskriva hur systemfunktionerna skall se ut. DFD och PSD används för att i detalj beskriva systemet på ett hierarkiskt sätt. Processerna på den lägsta nivån i dataflödesdiagrammet beskrivs med hjälp av PSD.

Det fjärde steget handlar om kodgenerering. Deft Way beskriver en teknik för att överföra ett programstrukturdiagram till kod. Deft innehåller ingen automatisk konvertering utan detta måste göras manuellt, men är relativt enkelt och mekaniskt.

I det femte steget testar man systemet. Detta sker med hjälp av andra fristående produkter. Boken föreslår en produkt från Digital Equipment som heter Digital Test Method,DTM.

I det sjätte och sista steget installeras det färdiga systemet och tas i drift. En grafisk sammanfattning av Deft Way kan se ut enligt figur 3.6.



Figur 3.6 The Deft Way i relation till de olika faser i systemutvecklingsprocessen.

Även om det finns en metod som verkar skräddarsydd för Deft, så är det klart att Deft inte på något sätt är bundet till denna metod. Det är snarare så att Deft Way är ett sätt att effektivt utnyttja de hjälpmedel som Deft erbjuder.

3.2.5 Begränsningar i och avvikelser från metoden

Eftersom Deft inte är speciellt hårt knuten till någon systemutvecklingsmetod så kan man inte direkt tala om några begränsningar eller avvikelser med avseende på metoden. Men det finns två typer av möjliga avvikelser i de grafiska modulerna (DFD,PSD, ERD) i de fall de används i samband med vissa metoder:

1. Kontrollerade avvikelser: användaren kopplar bort syntaxkontrollfunktionerna, tex avsteg från JSP syntax är tillåtet i PSD om man kopplar bort denna funktion.

2. Icke kontrollerade avsteg: dvs svag syntaxkontroll i vissa moduler som medför avvikelse från förbestämd syntax (enligt manualen). Det är t.ex tillåtet att direkt överföra data mellan två olika externa entiter i dataflödesdiagrammet och man kan även ge samma namn till två dataflöden och detta är någonting som strider mot både Yourdon och Gane&Sarson presentationssätt.

Ett problem med nivåorienterade metoder som Structured Analys And Stuctured Design är att bestämma när/hur man skall bryta ner en process. Ett annat problem är att veta hur många process skall finnas i varje nivå. Deft ger ingen hjälp för att lösa något av dessa problem. Med Deft blir det dock lättare att redigera sina diagram i och med att det går att flytta objekt och datatermer mellan nivåerna.

3 Systemutvecklingsdatabasen

3.1 Datakatalogen

Defts datakatalog är integrerad till de grafiska editorerna, dvs specifikationer som skapas i editorerna lagras i datakatalogen. Det är dock inte allt som hamnar i datakatalogen. Information om sådana viktiga designobjekt som processer, PSD-rutiner , entiteter och relationer lagras inte i datakatalogen, utan i de Macintosh-dokument som diagrammen sparas i (dokument är filer där Macintosh-applikationerna normalt lagrar det som genererats med dem). Detta innebär givetvis att systemspecifikationerna inte finns samlade på ett ställe, utan är spridda mellan datakatalogen och diagramdokumenten.

I data katalogen lagras följande termer:

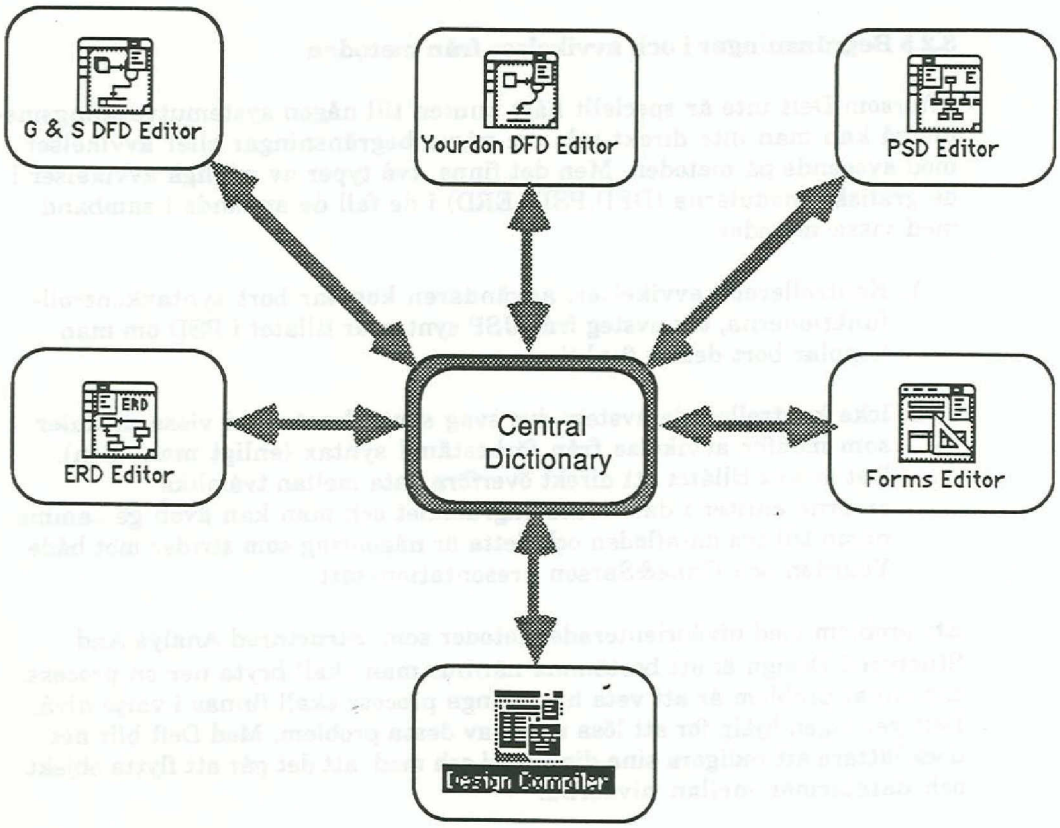
1. Attribut som skapas av de grafiska editorerna.

2. Datalager, enkla och sammansatta dataflöden skapade i DFD.

3. Inflöde och utflöde av data i programstrukturdiagram.

4. Entiteter, vyer och relationer skapade i ERD.

Fysiskt är datakatalogen i version 3.0 en vanlig flat resursfil, med ett begränsat antal termer (ca 26000 st). Datakatalogen är i princip en lista av objekt med deras egenskaper. Inga typer av associationer mellan objekten lagras i datakatalogen.



Figur 3.7 Datakatalogen är hjärtat i Deft. Den ger möjlighet till flexibelt arbetssätt, snabba modifieringar i befintliga objekt och överskådliga sammanställningar av lagrade data.

Kopplingen mellan de grafiska editorerna och den existerande datakatalogen bestäms av användaren. Om det inte finns en datakatalog så kan man skapa en ny från vilken grafisk editor som helst, utom form editor. Man kan modellera i flera separata datakataloger vars innehåll sedan slås ihop med hjälp av Dictionary Merge -modulen. Fördelen med det är att två personer kan jobba mot två eller fler datakataloger, som senare kan slås ihop. Tyvärr kunde vi inte testa Dictionary Merge för att se hur administrationen av termdefinietioner hanteras vid sammanslagning.

Datakatalogen i Deft uppdateras inte automatiskt när man ändrar i grafen. Om man t.ex. tar bort en process i DFD eller en relation i ERD på en skärmbild så påverkas inte innehållet i datakatalogen, och vice versa. Detta faktum kan ibland ge oönskade effekter när man märker att det man ser på grafen inte är detsamma som finns i datakatalogen.

3.2 Åtkomstmöjligheter och konsistens mellan representationer

Modifiering och bortagning av termer i datakatalogerna kan göras från alla grafiska editorerna.Vid bortagning av ett objekt direkt från datakatalogen, varnar Deft för konsekvenserna. Man får då ett meddelande om att andra objekt kommer att påverkas om man beviljar borttagande av ett element. Men även om man väljer att ta bort ett objekt, så uppdateras inte den grafiska bilden. Omvänt gäller att datakatalogen inte uppdateras då objekt

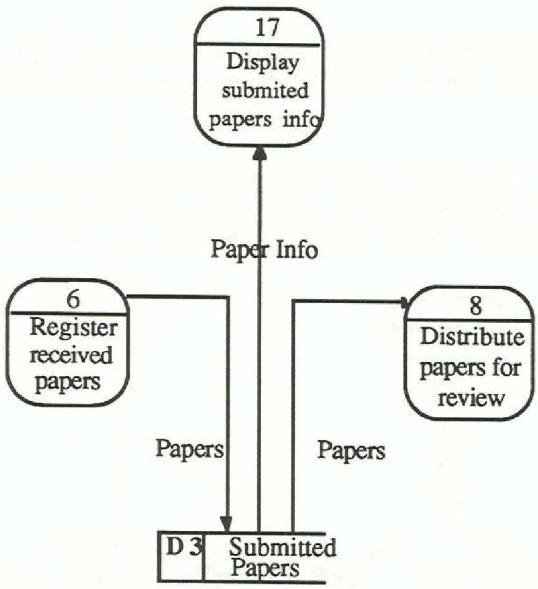
tas bort från grafen. Detta innebär att datakatalogens innehåll inte alltid är konsistent med den grafiska presentationen. Detta kan ses som brist i Deft, eftersom det man ser på grafen inte längre är detsamma som finns specificerat i datakatalogen. Detta beror antagligen delvis på det faktum att i Deft's datakatalog inte innehåller beskrivningar på alla designobjekt och associationer mellan dem. Då kan man inte alltid vid bortagning av ett objekt avgöra vilka andra som blir påverkade. Uppdatering av grafen med bibehållen konsistens blir då en svår uppgift.

Vi tar ett exempel från vårt praktikfall, nämligen bortagning av datalagret Submitted Papers (se figuren nedan). Problemet är att det finns flera komponenter i dataflödesdiagrammet som är starkt beroende av innehållet i Submitted Papers, tex dataflödet Papers och Paper Info .Båda dataflöden är i sin tur kopplade till processerna 6, 8 och 17 och innehåller flera attribut som överförs genom dem. Bortagning av datastore Submited Paper ifrågasätter existensen av både dataflöden och de attribut som de innehåller. Alltså blir det en kedja av konsekvenser som är svåra överblicka.

Bortagning av dataelement från datakatalogen borde även kunna göras grafiskt, och inte bara via explicit manipulering av datakatalogen. Man har ofta bättre och mer överskådlig bild på det element som man vill ta bort och dess relationer med omgivande objekt.

4.3.3 Kontroll- och analysmöjligheter

Det finn två olika sätt att göra kontroller och analyser i Deft, nämligen genom rapportgeneratorn och Design Compiler. Rapportfunktionen är tillgängligt från alla grafiska editorer i Deft. Den genererade rapporten är ett MacWrite dokument som innehåller information om dem objekt som skapats i den aktuella grafiska editorn. Utifrån denna rapport kan en manuell analys och kontroll göras. Det är speciellt viktigt när vi, från avsnittet ovan, vet att datakatalogen och den grafiska presentationen kan vara inkonsistenta.



Figur 3.8 Bortagning av Submitted Paper medför att dataflöden Papers och Paper Info blir hängande i luften och måste tas bort. Detta leder i sin tur till att beskrivningen av processerna måste modifieras.

Den andra möjligheten till kontroll och analys är kopplad till de presentationstekniker som Deft använder sig av. När som helst kan man, dock inte on-line, med hjälp av Design Compiler kontrollera om något avsteg från reglerna för presentationsteknikerna. Design Compiler analyserar diagrammet och genererar en fil som innehåller eventuella felmeddelanden. Som vi har redan sagt är denna kontroll inte helt uttömmande. Man kan göra vissa avsteg utan att kompilatorn upptäcker dem. Exempelvis klagar inte kompilatorn om man ritat dataflöden mellan externa entiteter, fast det strider mot syntaxen i dataflödesdiagram.

Design Compiler upptäcker inkonsistenser som beror på avsteg från "lagen av datakonservation". T.ex. måste de data som flyter in i ett datalager också flyta ut från den.

3.3 Säkerhetsaspekter

Deft är en vanlig Macintosh applikation. Det finns alltid risk att tappa data av någon anledning. Deft tillhandahåller inte några extra åtgärder utöver dem som redan finns i Macintosh när det gäller att skydda dokument.

Deft är ett s.k. "förlåtande system". Man kan ångra den senaste operationen för att på detta sätt minska risken att oavsiktligt förstöra data. Som vi redan sagt finns det varningsmeddelande för bortagning som kan påverka innehållet i datakatalogen.

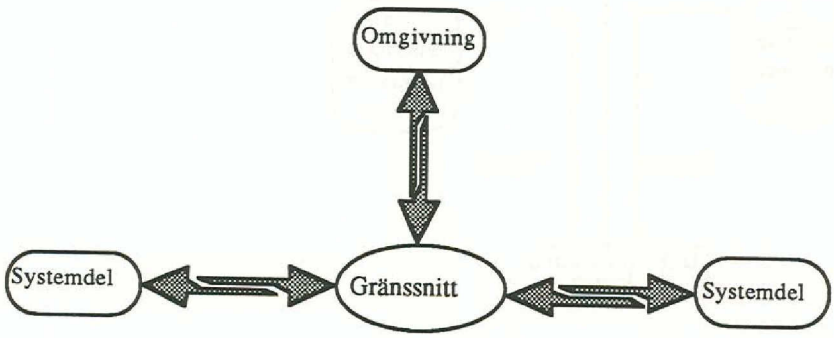
Säkerhetsåtgärder som lösord, automatisk Backup-tagning eller återstart av systemet vid haveri finns inte implementerade i Deft.

3.4 Användarmiljön

3.4.1 Gränssnitt mot användaren och interaktivitet

Gränssnitt kan enkelt definieras som en överenskommelse om hur kommunikationen skall ske mellan två eller flera systemdelar eller över systemgränser.

Hemvisten i Macintosh gör att arbetet med Deft sker i "objektorienterad form" . Objekt som tabeller eller processer presenteras på skärmen som flyttbara ikoner. Man använder musen för bla flyttning och selektering av



Figur 3.9 En illustration av begreppet gränssnitt.

objekt. Kommandon anges via tangentbordet eller med musen och menyerna. Kommandon finns dels i de vanliga Macintosh rullgardinsmenyerna och dels i paletterna som ofta används för ritning och texteditering. Man byter ritnigsmod genom att markera en ikon i paletterna. Varje gång man byter mode så byts Curser-mönster för att vissa i vilken ritnigsmod man befinner sig (ritnig av processer, dataflöden etc).

En av de främsta fördelarna med Deft ligger just i att alla modulerna har samma beteende och utseende mot användaren. Denna enhetlighet och det intuitiva arbetsättet gör Deft till ett lättlärt verktyg.

Deft är ett verktyg som är konstruerat speciellt med tanke på enkel användning och kan lätt användas av icke Macintosh specialister.
Deft använder sig av enkla och lättförståeligakommandon. Vi lärde oss kommandona snabbt och vi slapp söka i manualen.

3.4.2 Hjälpfunktioner

Hjälpfunktioner saknas helt i Deft. Det finns varken on-line hjälp eller inlärningsfunktioner, utan nybörjare får lov att ha manualen bredvid sig. Är man van att arbeta med Mactintosh och känner till begreppen i de beskrivningstekniker som Deft använder behöver man dock sällan mera avancerad hjälp.

3.4.3 Fleranvändarmiljön

Deft är avsedd att användas på enmansarbetstationer. Det finns ingen nätverkstöd eller möjlighet att låta flera användare arbeta parallellt mot en gemensam datakatalog.

3.5 Egenskaper och faciliteter hos Deft

3.5.1 Portabilitet

För dagen kan specifikationerna i Defts datakatalog bara överföras till andra Deftkataloger. Dock kan specifikationerna för entity relationship-diagram och skärmlayouter transformeras och överföras till Ingress 4GL-miljöer, där de kan direkt utnyttjas för framställning av databaser och bildskärmar.

3.5.2 Kommunikation

Deft har precis som andra Macintosh applikationer via klippboken en bra kommunikation med andra Macintosh tillämpningar. Cut, paste och copy funktionerna ger möjlighet till att flytta objekt mellan olika tillämpningar. Ett exempel är de figurer från IFIP som finns i denna rapport. Vi har överfört dem via klippboken.

Den 4GL-miljön som Deft kan kommunicera med är under skrivande stund Ingres på Vax-maskiner.Med hjälp av en speciell modul i Deft som heter Translator (Gate Way) överförs objektfiler, genererade med Design Compi-

ler, till Ingres-miljön där man sedan får tabeller och formulärer automatiskt genererade.

Kommunikationen mellan Deft's moduler är för nuvarande begränsad till datakatalogen och eventuellt klippboken.

3.5.3 Transformation av specifikationerna till databasstruktur

Det intressanta med Deft är att man via Design Compiler och Translator kan transformera sin datamodell till en databasstruktur för uppläggning av databaser i Ingres . Även skärmlayouterna skapade i Deft kan överföras till Ingres. Koden för skärmlayouterna kan sedan utnyttjas vid realisering av systemet med hjälp av Ingres' 4GL-språket. Dessutom kan deklarationer av datastrukturer i språket C genereras automatiskt för att eventuellt inlemmas i program.

I ERD-editorn kan man välja mellan olika sökalgoritmer (hash, btree, isam) för sin databas. Detta innebär att Deft kan generera flera kodversioner beroende på vilka sökalgoritmer man väljer.

3.5.4 Kodgenerering

Deft har ingen automatisk kodgenerering men programstrukturdiagrammen kan lätt användas för manuell kodning. Byggblocken för PSD-rutiner, som loop, normal, conditional och subprogram, kan användas för att beskriva programlogiken i form av ett programeringsspråk eller pseudokod. För varje block kan man dessutom skriva information i fri textformat . Men istället för att mata in information kan man specificera rutinen med program- eller pseudokod för att sedan generera grunden till kod för subrutiner eller programsekvenser med hjälp av datakatalogens rapportfunktion.

Defts krafttfullhet ligger i att man på ett enkelt sätt kan generera kod för databashanterare. Koden genereras utgående från objekt beskrivningarna som defieneras i datamodellen (tabeller, vyer,attribut typer, sökalgoritm etc).Denna kod transformeras till Ingress 4GL-miljön och kan direkt exekveras. Resultatet blir en fungerande databas som innehåller de tabeller och vyer som finns i data modellen.

3.5.5 Prototyping

Det finns ingen prototypfunktion i Deft, men viss bild av användargränssnittet kan ändå skapas i ett tidigt stadium via skärmlayouterna.

Deft tillsammans med Ingres ger dock mera reella protypingmöjligheter. Att beskriva datamodeller, systemstruktur och bildskärmar går snabbt med Deft. Datamodellen och bildskärmarna överförs sedan till Ingres, där de kan testas med mycket litet besvär. Faktum är att man på detta sätt kan få ett fungerande "råsystem": I Ingres skapas databaserna och bildskärmarna inkluderas med små kodsnuttar. Till tabellerna och vyerna i databasen tillhandahåller Ingres automatiskt standardrapporter, och även inmatningsformulär om inga egendefinierade finns. Alla dessa komponenter tillsammans ger ett fungerande råsystem där man kan mata in data i tabellerna i databasen, göra rapporter över dem eller hämta speciell information ur dem med hjälp av vyerna från Deft.

Denna typ av råsystem kan ses som prototyper eller "systemutkast". Är man inte nöjd kan man gå tillbaka till Deft och modifiera sin systembeskrivning för nya test. När man är nöjd med sitt råsystem kan man sätta igång och realisera systemet mera detaljerat med Ingres 4GL-språk. Enklare system som baserar på statisk information kan till och med tas i drift redan innan alla realiseringsfasen är helt genomförd.

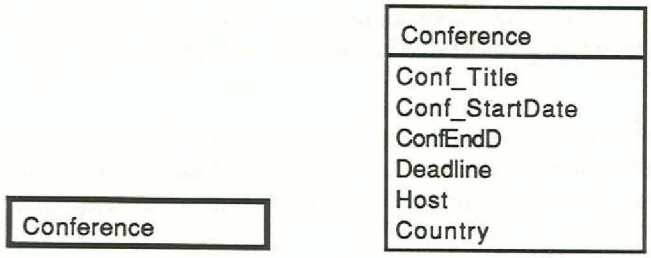
När vi gjorde denna studie av Deft använde vi oss av ett konferenssystem som praktikfall. Vi fick möjligheten att överföra våra systembeskrivningar till Ingres och där i verkligheten provköra delar av konferenssystemet. Överföringsarbetet och "idrifttagandet" tog inte ens en timme.

3.5.9 Flexibilitet

Ett flexibelt system kan definieras som ett system som låter användaren utforma sina objekt på ett fritt och kreativt sätt efter sina behov och sin "smak". Detta innebär att ändringar i objekt i hög grad skall vara oberoende av det övriga systemet. Deft kan klassas som ett flexibelt verktyg. Man behöver inte använda alla verktygsmoduler utan enbart de moduler som man tycker passar för ens behov eller arbetssätt. Däremot kan man inte definiera sina egna symboler eller regler.

Alla objekt i Deft lagras fysiskt oberoende av varandra . Det finns inga som helst begränsninger för hur de skall användas, modifieras eller vad de skall presentera i systemet. När det gäller det grafiska representationssättet så kan storleken på alla objekt bestämmas av användaren utan att det påverkar notationssättet. Även utseende kan i vissa fall bestämmas, men inte alltid.

Som exempel tar vi hur man presenterar tabellerna i ERD.De kan presenteras antingen med en rektangel som innehåller tabellens namn och dess attribut eller bara med en rektangel som innehåller tabellnamnet. Detta kan vara användbart för att lätt kunna överskåda diagrammen om det är för många tabeller i ett och samma diagramm. En annan egenskap som finns i ERD och som har stor betydelse för flexibiliteten är val av den frågespråktyp (SQL, QUEL, eller någon annan) som kommer att användas i 4GL-miljön. Man kan också definiera vilkenn typ av sökalgoritm som skall användas (isam, hash, btree, heap, heapsort).



Figur 3.10 Två olika sätt att presentera tabeller (entiteter) på skärmen, Bilden till vänster visar bara tabellens namn, bilden till höger visar både tabellens namn och attribut.

3.6 Hur påverkar Deft systemutvecklingsprocessen

3.6.1 Vinster i tid och kvalitet

Ett problem i de traditionella systemutvecklingsmiljöerna är att högutbildat folk med höga löner använder en stor del av sin tid på "tråkiga" programeringsuppgifter istället för ett kreativt systemutvecklingsarbete. Användning av ett CASE-verktyg som Deft innebär att tyngdpunkten skjuts från programmeringsarbete till analys och design som i slutändan kan leda till vinst i tid och pengar och ökad kvalité.

Att dokumentera ett system är ett tidskrävande arbete. I Deft generas väsentliga delar av systemdokumentationen automatisk. I alla Deft- moduler kan rapporter av datakatalogens innehåll genereras. Man har också möjligheten att editera rapporterna med något ordbehandlingsprogram.

Det är klart att nyutvecklade system kommer att blir gamla i framtiden. Funktionerna hos ett nytt system blir snabbt föråldrade och eventuellt felaktiga i framtiden. Arbetet att anpassa existerande system till förändrade miljöer kräver stora resurser och är ofta tid- och kostnads-krävande. Man kanske inte ens har någon annan nytta av det gamla systemet än att använda det som systemskiss för det nya. De egenskaper som Deft har, speciellt om man jobbar mot Ingres, underlättar vidareutveckling och underhåll av befintliga system.

3.6.2 Effekterna på informationssystemets livscykel

I de traditionella systemutvecklingsmiljöerna betraktar man ofta den första versionen av systemet som den slutliga. Detta beror förstås på de höga kostnaderna för vidareutveckling av system. Tanken med att kasta bort första versionen av systemet och utveckla den på nytt för att det verkligen ska bli bra är enklare att realisera med Deft. Även efter det att systemet har tagits i drift kan man göra dramatiska ändringar i det utan att behöva lägga ner alltför mycket arbete.

Ett CASE-verktyg som Deft kan ge en bättre koppling mellan systemets beteende och den deklarativa beskrivningen av systemet. (Den deklarativa beskrivningen av systemet är de grafiska modellerna, nämligen ERD, DFD, PSD och formulärerna.). Data och processer presenterade i Deft är lätta att jämföra med olika delar av verksamheten. En tydlig positiv effekt är att en icke insatt person snabbt kan bli produktiv och medverka i det fortsatta arbetet.

Naturligtvis finns det även nackdelar. Det snabba arbetet med systemutvecklingen kan medföra ökad risk för fel. För att eliminera detta krävs goda analysfunktioner. De analyser som Deft tillhandahåller kan inte betraktas som tillräckliga.

Livscykeln hos ett informationssystem sträcker sig från den tidiga kravspecifikationsutformningen till systemimplementationen och systemförvaltningen (Några tycker att även systemsskrotning borde ingå i kedjan). En del av de effekter på systemens livscykel som användning av Deft kan medföra är kvantiferbara medan andra är svårare att kvantifiera. De effekter som vi inte kunde kvantifiera är bland annat det slutliga systemets beteende och naturligtvis systemets livslängd. Livslängden borde dock öka i

och med att bättre dokumentation och överskådliga systembeskrivningar underlättar underhållet och vidareutvecklingen av system.

Andra effekter som vi kunde observera:

1. Kravspecifikationens utformningen är oförändrad.

2. Tyngdpunkten har förskjutits från programmering mot systemanalys och systemdesign.

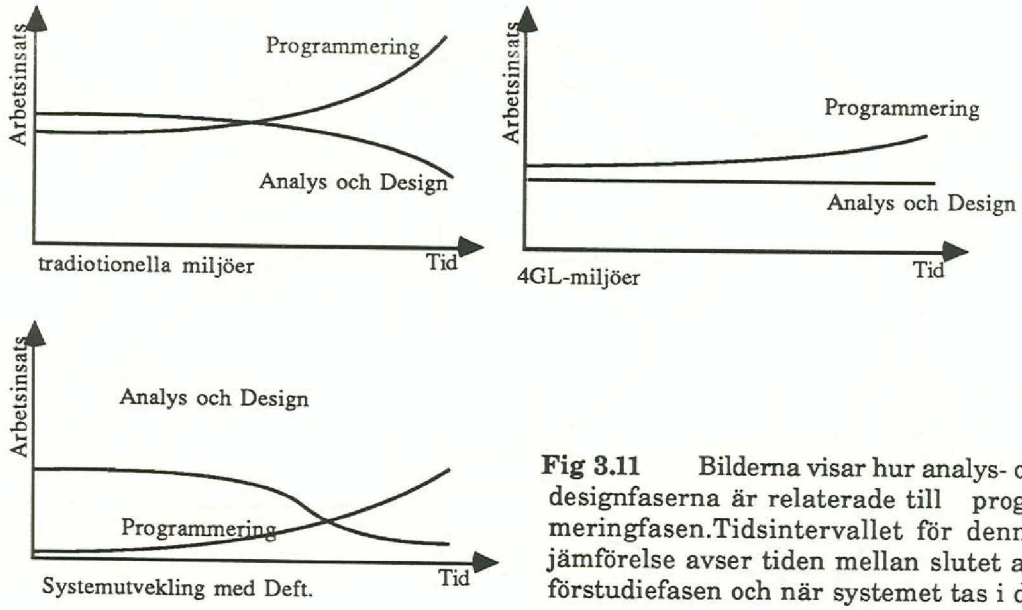


Fig 3.11 Bilderna visar hur analys- och designfaserna är relaterade till programmeringfasen.Tidsintervallet för denna jämförelse avser tiden mellan slutet av förstudiefasen och när systemet tas i drift

3. Systemdokumentationsmomenten har nästan eliminerats. Rapportgeneratorn i Deft producerar den deklarativa beskrivningen av systemet i överskådligt form med viss statistik över systemets komponenter.

4. Den snabba utvecklingsmiljön som Deft erbjuder löper kanske större risk för fel. Men vi tror att en erfaren systemutvecklingspersonal kan minska risken.

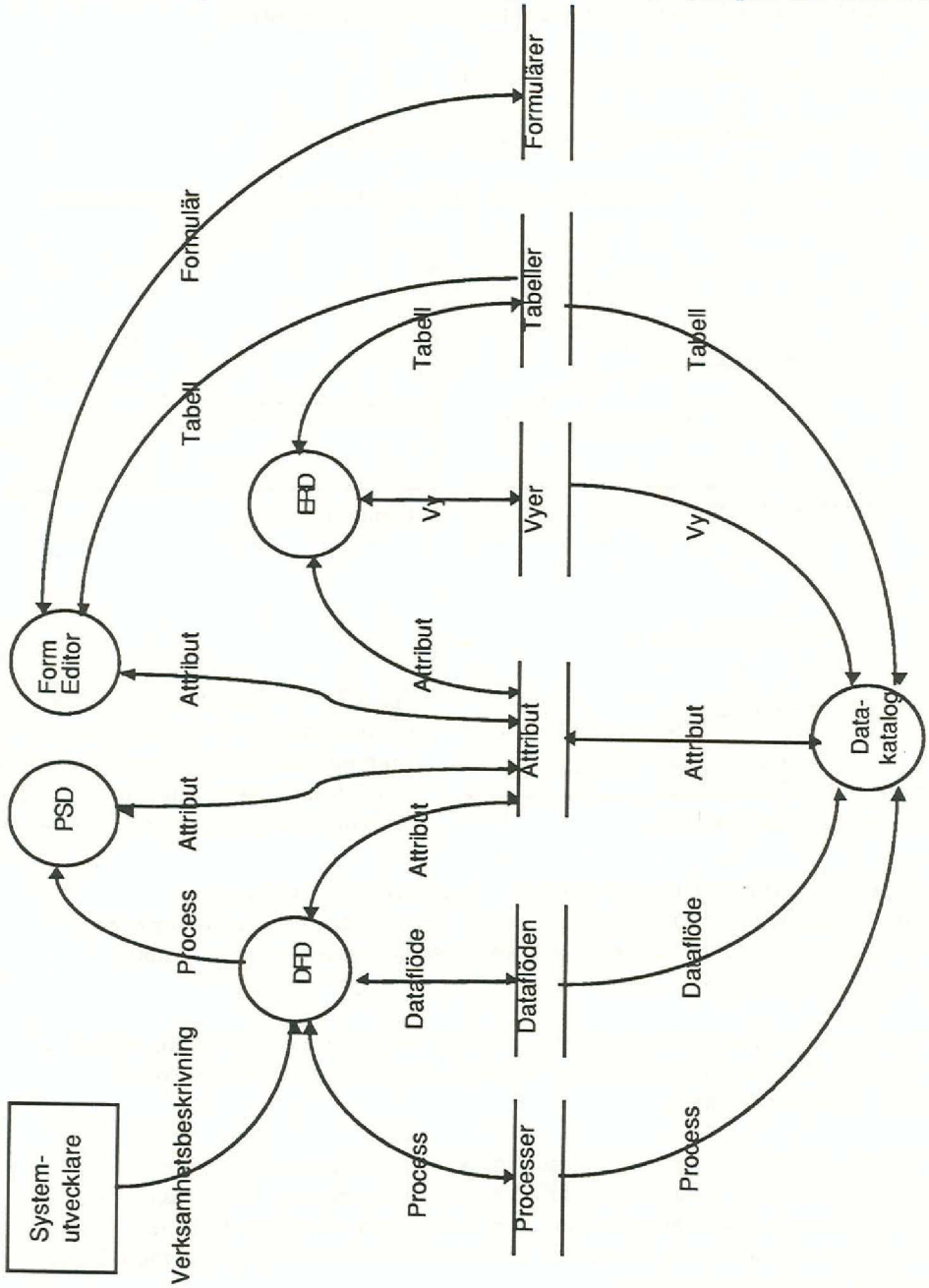
5. Vi har använt oss av Structured Analys And Structured Design för att genomföra vårt praktikfall, IFIP-konferenssystemet. De ansatser Deft vilar på överstämmer med den metod som vi själva fann det lämpligt att använda. Vi kunde även beskriva hur vi arbetar med Deft i Deft (se figuren nedan).

6. Vi kan till slut dra följande slutsats: Tiden då systemet är under utveckling blir kortare, och tiden då systemet är i drift blir längre.

3.6.3 Slutanvändarnas roll i systemutvecklingsarbetet

I Riksdataförbundets rapport nr 16 1983 säger man följande om slutanvändarens roll i systemutvecklingsarbetet: "Systemutveckling kan enkelt beskrivas som en översättning av en beskrivning av vad ett system ska göra till en beskrivning av hur dator och program ska realisera denna beskrivningen. Beskrivningen av vad användaren ska göra kallas användarspecifikationer. I de gamla systemutvecklingsmiljöerna var målet att låta

slutanvändaren medverka i utformningen av användarspecifikationerna.



Figur 3.12 En del av informationssystemutvecklingsarbetet med Deft utryckt mha DFD-editorn.

I 4GL-verktyg är målet att låta slutanvändaren medverka i beskrivning av hur dator och program skall realisera användarsspecifikationerna". Men kopplingen mellan båda delarna är för svag för att slutanvändaren ska kunna se om systemet ska lyckas realisera specifikationerna eller ej.

I Deft har slutanvändaren bättre möjligheter att se hur systemet skall realisera specifikationerna och medverka i flera faser i livscykeln än i de andra traditionella systemutvecklingsmiljöerna. Ett konkret exempel där slutanvändren kan medverka mer är dialogutformningen. Användaren

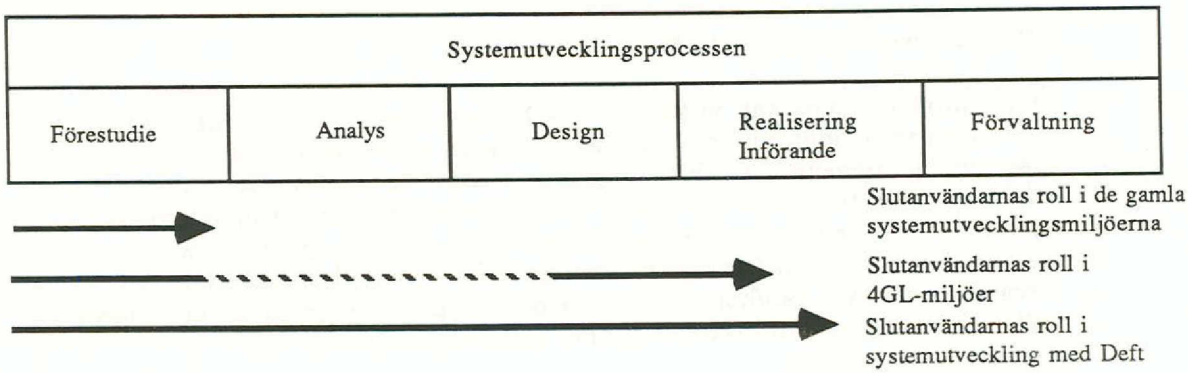


Fig. 3.13 I de gamla systemutvecklingsmiljöerna har slutanvändaren bara medverkat under förestudien. I 4GL-miljöerna kan slutanvändaren även medverka i realisering av användarspecifikationerna men har svårt se om systemet verkligen lyckas realisera dessa.Med Deft kan slutanvändaren lättare se kopplingen mellan användarspecifikationen i förstudien och dess realisering tack vare de grafiska beskrivningarna.

kan i ett tidigt stadium se hur formulären kommer att se ut och föreslå eventuella ändriningar. De grafiska systembeskrivningarna kan göra det lättare för slutanvändaren att förstå hur systemet kommer att fungera.

3.6.4 Utbildningsbehovet

Utbildningsbehovet är helt beroende av vem som skall använda Deft och i vilket syfte. Vi tror att en person med lämplig utbildning eller erfaren ADB-personal kan klara jobbet med Deft utan svårigheter och utan att behöva anlita "Deft-konsulter".

För dem som vill arrangerar leverantören kurser i användning av Deft. Leverantören ordnar också sk användarmöten där Deft-användarna kan utbyta erfarenheter med varandra.

3.7 Hård- och mjukvarukrav och dokumentation

Deft kan köras på alla Macintosh-datorer och kräver ingen speciell hårdvara. Leverantören är för nuvarande INFOTOOL Data AB . Priset är relativt billigt jämfört med andra CASE-verktyg. Den som vill kommunicera med 4GL- språket Ingres måste köpa en extra programmodul som heter Deft Gate way.

Deft`s manualer levereras med och är skrivna på engelska. Manualerna är pedagokist utformade och tillräckligt innehåller allt man behöver veta att kunna använda Deft. De är också lättlästa att och berikade med många förklarande exempel och bilder. För dem som inte har använt Macintosh förr, så finns det utförliga beskrivningar för alla standard-kommandon i Macintosh som Deft använder sig av.

3.8 Avslutande översikt

3.8.2 Sammanfattning av Deft

Sammanfattningsvis kan vi säga att Deft är ett flerfunktionsverktyg för Macintosh persondatorer och avser att stödja en större del av informationssystemutvecklingsarbetet, från tidiga förstudieskisser till prototyping. Verktyget är inte bundet till någon specifik systemutveclingsmetod, men kan med fördel användas för metoder med hierarkiska systembeskrivningar. Användaren behöver inte låsa sig till en speciell metod utan kan använda Deft som han/hon själv anser det lämpligt. De grafiska modulerna tillhandahåller en interaktiv miljö för systemspecifikation och databaskonstruktion.

Tillsammans med Ingres är Deft en naturlig grafisk analys- och designverktyg för framtagning av systembeskrivningar. Men även om man inte kopplar Deft till andra verktyg kan det användas som stöd för analys och design.

I rätt miljö och vid konsekvent användning kan Deft ge betydliga fördelar jämfört mot traditionella metoder.

3.8.2 Deft i framtiden

Deft befinner sig just nu i en dynamisk utveckling; version 3.01 finns redan på marknaden och andra versioner är under utveckling.
Den nya versionen av Deft att ha en datakatalog som bygger på en relationsdatabas. Den kommer att vara snabbare men mer utrymmeskrävande. Antalet termer i datakatalogen kommer inte längre att vara begränsat. I DFD-editorn finns det sk "autopopulate", som innebär att datalager och flöden till/från dem "befolkas" automatiskt när man beskriver endera. Man kan även koppla datalager i DFD till tabeller i ERD och flytta objektbeskrivningar mellan dessa. Deft version 3.01 har även en kraftfull funktion som kallas "revers ingineering", som innebär att man kan importera befintliga tabeller och skärmbilder från Ingres till Defts datakatalog. Detta steg betyder att Deft 3.01 kan användas för vidareutveckling av gamla system .

Deft version 4.0, som kommer senare, har en dictionary som kallas "repository" eller encyclopedi och innehåller data om diagrammen. Deft 4.0 kommer att tillåta flera användare i Macintoshsnätverk att arbeta parallellt. Kopplingen mellan de olika diagrammen kommer att förbättras och man kommer att kunna förflytta sig fritt mellan de olika editorerna under en session. Detta innebär att man kan jobba med många fönster samtidigt och lätt växla mellan olika editorer med bibehållen koppling. Editeringsmöjligheterna blir bättre i version 4.0. Man kan editera även när diagrammen visas i mindre skala (dvs i "reduce to fit"-moden).

3.8.1 Personliga intryck

Vårt personliga intryck är att Deft 3.0 är ett bra verktyg för att utveckla databas-tillämpningar. Vi tycker att urvalet av beskrivningstekniker är bra för många systemutvecklingsmetoder. De är lätta och smidiga att arbeta med, och kan mycket väl integreras med varandra, men tyvärr har Deft ännu inte gjort det.

Med hjälp av de grafiska editorerna går det både lättare och snabbare att skapa de tillämpningar man vill ha. Ritstödet reducerade det besvärliga manuella arbetet och gav oss bättre syn på hela konceptet. Avståndet mellan den egentliga verksamheten och det fysiska systemet har minskat jämfört med traditionell systemutveckling. Vi tycker också att man får bättre överblick över systemstrukturen och kan införa förändringar utan att onödigt extra arbete. Vi tror att en helhjärtad satsning på Deft integrerat med Ingres kan leda till snabbare systemutveckling och minskade kostnader. Men även om man inte har Ingres så kan Deft användas som datorstöd för design och analys av system.

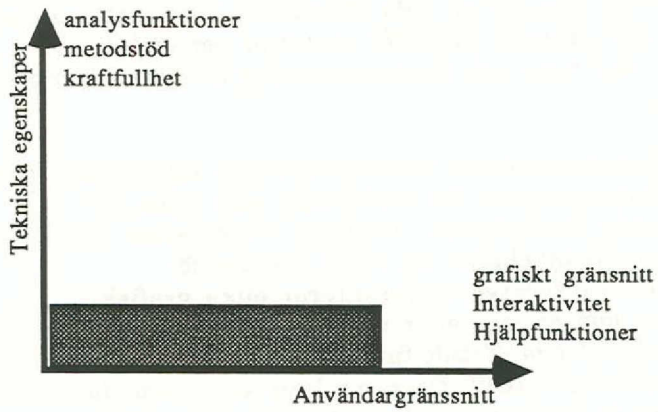


Fig 3.14 Deft har ett avancerat användargränssnitt medan de tekniska egenskaperna inte är fullt så utvecklade.

Även om vi inte hade någon praktisk erfarenhet i systemutveckling kunde vi med Deft på relativt kort tid ta fram systembeskrivningar för vårt praktikfall och överföra dessa till Ingres och få ett fungerande system, klar för tester och provkörning.

Vad vi saknade mest i Deft var möjligheterna till automatisk analys och konsistenskontroller i datakatalogen. Även bättre integration mellan diagramtyperna hade säkert underlättat arbetet, då särskilt integrationen mellan DFD och ERD.

4 STUDIE AV IEW/WS

4.1 Allmänt om IEW

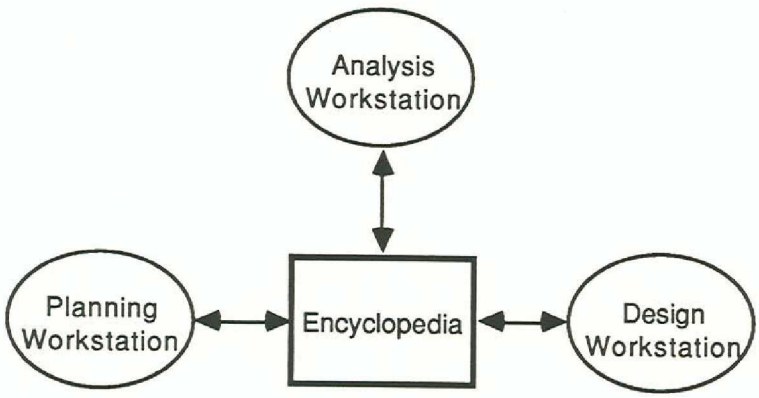
Information Engineering Workbench® (IEW) är ett verktyg för datorstödd systemutveckling och är utvecklad av det amerikanska företaget Knowledgeware, Inc. IEW marknadsförs av Arthur Young International, i Sverige Arthur Young AB. IEW finns i både persondator- och stordator-version. I stordatorversionen finns även funktioner som hantering av en gemensam datakatalog. I den här rapporten koncentrerar vi oss på den persondatorbaserade versionen, Information Engineering Workbench® / Workstations (IEW/WS), som går på IBM PC-AT och kompatibla datorer under MS-DOS operativsystemet. IEW/WS utnyttjar persondatorers grafiska möjligheter med mus- och fönsterhantering för bearbetning av systembeskrivningar.

IEW/WS består av ett antal moduler som koncentrerar sig på olika faser inom systemutvecklingsprocessen. De modulerna som finns för närvarande är Planning Workstation, Analysis Workstation och Design Workstation. Varje modul innehåller ett antal verktyg för systembeskrivning. Dessa verktyg representerar olika grafiska beskrivningstekniker, som är mer eller mindre karakteristiska för systemutvecklingsfasen i fråga. Totalt finns det elva olika verktyg, eller Diagrammers som de kallas i IEW. Dessa verktyg kan användas oberoende av varandra, även om de är integrerade via Encyclopedia, IEW's datakatalog.

IEW är inte bundet till någon specifik systemutvecklingsmetod, eller metodkedja. De olika beskrivningsteknikerna kan kombineras på ett önskat sätt för att anpassa IEW till en specifik metod eller till ett visst arbetssätt, även om viss formalism påbjuds.

4. 2 Systemutveckling med IEW

Det kanske blir lättare att förstå IEW's relation till systemutveckling om vi först gör en översiktlig genomgång av det formella synsätt på utveckling av informationssystem som IEW baserar på.



Figur 4.1 IEW's tre arbetsstationer delar på information via en gemensam encyclopedi.

IEW har sin grund i Information Engineering konceptet. Enligt James Martin, [Martin86], är Information Engineering en mängd formella tekniker med vars hjälp verksamhetsmodeller, datamodeller och processmodeller framställs. Dessa modeller lagras i en omfattande kunskapsbas och används för att skapa och underhålla system för informationshantering. Vidare skall dessa tekniker stödjas av höggradigt automatiserade verktyg så att slutanvändarna skall kunna aktivt delta i utvecklingsarbetet . Resulterande system skall lösa lokala problem men integreras horisontellt så att de samverkar med varandra och samtidigt vara integrerade vertikalt så att de är förankrade i de mål man satt upp för verksamheten.

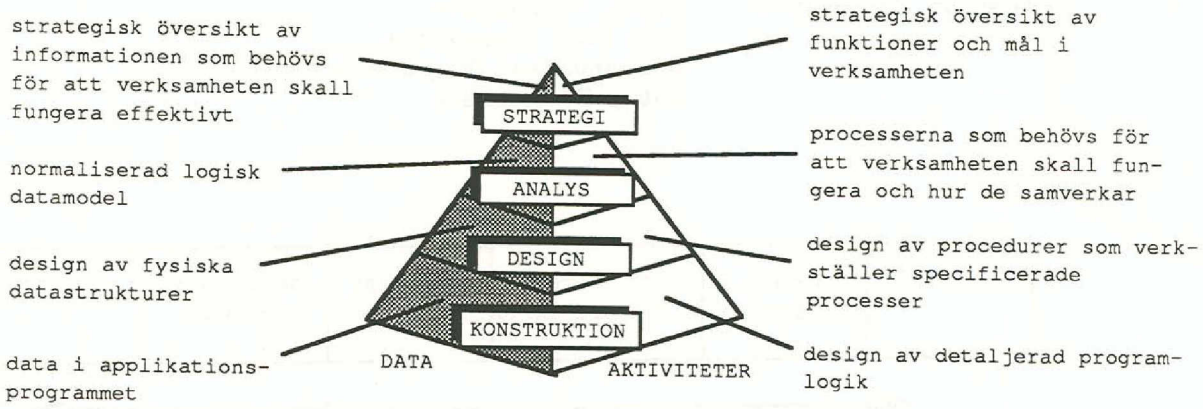
Enligt Information Engineering kan aktiviteterna kring systemutveckling grupperas i fyra steg. Dessa brukar illustreras med hjälp av en pyramid med tre sidor enligt figuren. Ur den framgår att systemutveckling delar sig i fyra huvudsteg: strategisk planering, analys, design och konstruktion. Under dessa steg övergår information till logisk datamodell, fysisk datastruktur och till sist applikationsprogrammens syn på data.

Information Engineering är inte någon strikt systemutvecklingsmetod utan kan snarare betraktas som en referensmodell för formella metoder. Många av de tekniker och verktyg som Information Engineering beskriver som stöd för systemutveckling finns i IEW. Encyclopedien är den kunskapsbas som verktygen skall samverka med. Även förklaringen till IEW's modulbaserade uppbyggnad blir mera uppenbar om man ser på hur Information Engineering ser på aktiviteterna under systemutvecklingsprocessen.

IEW definierar inget formellt arbetssätt, utan låter användaren avgöra hur han eller hon skall arbeta. Genom sin stegorienterade uppbyggnad och starka integration mellan beskrivningsteknikerna uppmuntrar dock IEW till en viss grad av formalism i arbetssättet.

4.2.1 IEW's "spännvidd"

Om man har både IEW/WS och stordatorversionen av IEW till sitt förfogande, så har man ett relativt heltäckande verktyg för systemutveckling. Det som saknas i skrivande stund är generering av färdig kod - för det används



Figur 4.2 Hur Information Engineering ser på systemutvecklingen. Pyramidens baksida representerar den teknologi som det färdiga systemet kommer att anpassas för, och som man måste analysera och planera för.

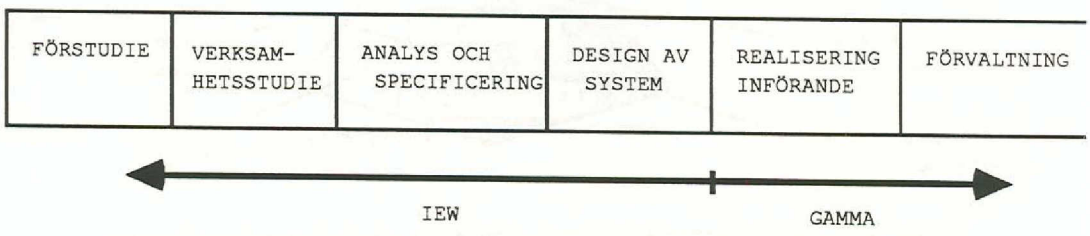
en separat produkt , Gamma, från samma företag. Man har möjligheten att genomföra planering, analys och design i en enhetlig och integrerad miljö. All data man arbetar med lagras i encyclopedien och finns tillgänglig för analys och vidare bearbetning. Designresultatet kan överföras till stordatormiljön, där kodgenerering med Gamma sker. Beroende på hur fullständigt man specificerat procedurerna i Design Workstation har man då fått testfärdigt system.

Ett sätt beskriva systemutvecklingsprocessen på är att dela upp den i ett antal faser. Ofta använder man en uppdelning i sex faser: Förstudie, verksamhetsstudie,analys och specificering,design, realisering och införande, samt förvaltning. Med viss reservation för förstudien kan IEW utnyttjas i alla dessa faser. När det gäller förstudien, reserverar vi oss därför att IEW innehåller funktioner som väl kan tänkas vara till hjälp vid arbetet under förstudien, men dock saknar stöd för många viktiga delar i förstudien (t.ex. projektplanering , budgetering och detaljerade kravspecifikationer). Däremot är IEW välutrustad vad gäller verksamhetsstudien, analys- och designfaserna. I verksamhetsstudien kan även strategisk planering inkluderas i enlighet med Information Engineering konceptet. Realisering och införande av det specificerade systemet kan också genomföras med IEW kompletterat med Gamma. I och med att IEW stöder de tidigare faserna underlättas även förvaltningen av system.
Detaljerna om IEW's stöd för de olika faserna behandlar vi senare i rapporten. Här nöjer vi oss att illustrera IEW's "spännvidd" med en överskådlig figur.

4.2.2 Beskrivningstekniker

Som vi redan tidigare nämnt så består IEW av tre moduler med var sin uppsättning av integrerade grafiska verktyg . Många av beskrivningsteknikerna är allmänt utnyttjade av diverse metoder, medan några kanske är mera okända. Vi börjar med lista upp dessa verktyg och går sedan genom dem modul för modul.

Planning Workstation	Matrix Diagrammer Decomposition Diagrammer Entity Diagrammer
Analysis Workstation	Action Diagrammer Data Flow Diagrammer Decomposition Diagrammer Entity Diagrammer



Figur 4.3 IEW's relation till faserna systemutvecklingsprocessen.

Design Workstation	Action Diagrammer
	Structure Chart Diagrammer
	Presentation Diagrammer
	Data Structure Diagrammer
	Flat File Database Diagrammer
	Relational Database Diagrammer
	Hierarchical Database Diagrammer

Planning Workstation

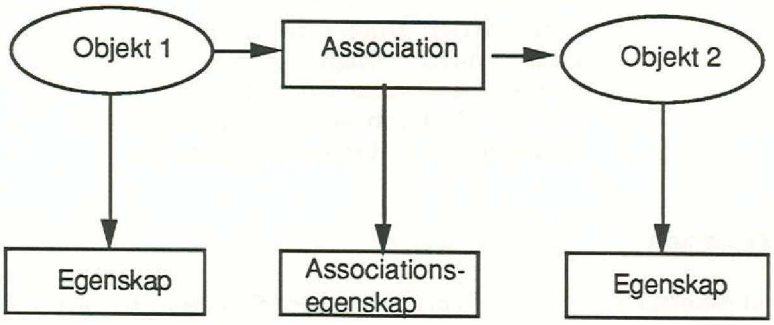
Beskrivningsteknikerna i Planning Worksation (PWS) syftar till att stödja igenkänning och utvärdering av verksamhetsmål och de faktorer som påverkar verksamhetens framgång. En översikt av av företaget eller verksamheten kan skapas och dess funktioner, data och informationsbehov skall kunna kartläggas med hjälp av de verktyg som Planning Workstation erbjuder. Vi börjar med en kort sammanfattning av de aktiviteter som PWS stödjer:

- Utveckla en modell över verksamheten i form av dess mål, funktioner och framgångsfaktorer
- Identifiera den information som är basen för verksamhetens mål, funktioner och framgångsfaktorer
- Faställa en gemensam ram för utveckling av informationsystem i verksamheten
- Avgöra vilken prioritet utvecklingen av informationssystem har i verksamheten
- Utveckla system som har grunden i den strategiska planeringen i verksamheten

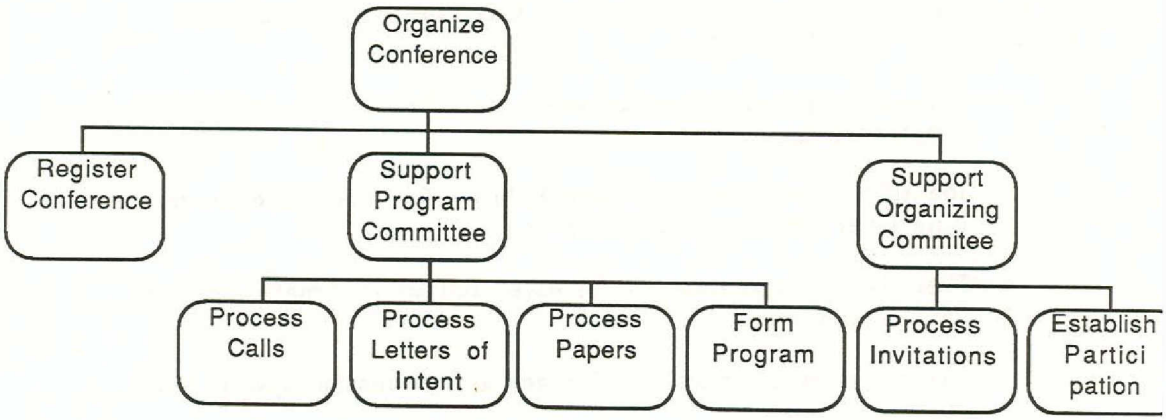
Syftet med planering är att frågor beträffande informationshantering förs in i företagsledningen. Data betraktas som strategisk resurs och man inför strategisk planering av informationssystem för att öka konkurrenskraften och skapa nya affärsmöjligheter. Vi skall nu titta på hur diagramverktygen i PWS kan användas för detta syfte.

Diagramverktygen kan användas oberoende av varandra, men de objekt som skapats i ett diagram blir omedelbart tillgängliga i andra. Objekten kan sedan analyseras och bearbetas vidare. På detta sätt kan man iterativt öka sin objektmängd och bearbeta den från olika synvinklar i de olika diagrammen. Ett diagram är bundet till ett fönster . Bearbetning av diagram kan även ske parallellt genom att ha flera fönster öppna samtidigt.

De objekt man skapar i Planning Workstation har fördefinierade typer. Varje objekt har en typ, och till varje objekttyp kan ett antal egenskaper associeras. Objekt kan också associeras till varandra via fördefinierade associationstyper. Även vissa associationer har egenskaper. Vi kan exempelvis skapa ett objekt *Person Info* att vara av typen informationsbehov. Om vi har ett objekt av typen *process* och som heter *Get Person Info*, kan vi definiera en association: *Information Need Person Info is supported by process Get Person Info*. Denna association har egenskapen Degree som kan ha värdet "direct" eller "indirect" eller ett tal mellan 0 och 1 för att tala om på vilket sätt eller i vilken grad processen *Get Person Info* tillfredställer informationsbehovet *Person Info*.



Figur 4.4 Objekt, associationer och egenskaper i IEW

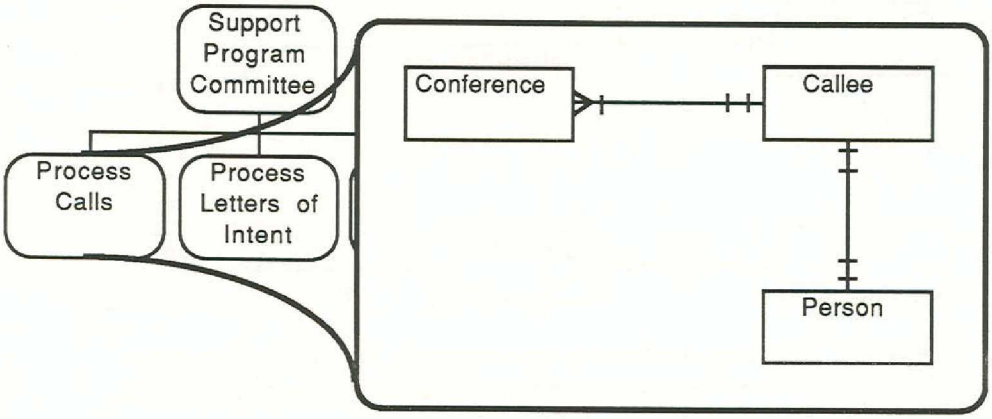


Figur 4.5 Exempel på ett decomposition diagram. Man kan göra dekompositioner av verksamhetsmål och -funktioner, processer, organisationens uppbyggnad, verksamhets- och ämnesområden, etc.

Decomposition Diagrammer används för att successivt dela upp information eller objekt i mindre beståndsdelar. Exempelvis kan en övergripande funktion i verksamheten delas upp i ett antal processer som i sin tur bryts ned i delprocesser. Likadant kan organisatoriska enheter delas upp i en hierarki av avdelningar och underavdelningar.

Planning Workbench har olika typer av dekompositionsdiagram för olika typer av objekt. Alla dekompositionsdiagram fungerar dock likadant och har ungefär samma operationer tillgängliga. Ett dekompositionsdiagram kan representera objekttyper som datagrupperingar , funktioner, verksamhetsmål, platser, mekanismer, organisatoriska enheter, problem, processer och ämnes-/verksamhetsområden. Gemensamt för alla dessa är att de kan delas upp i någon form av hierarki.

Editering av diagram sker med hjälp av musen och tangenterna. Objekt kan läggas till genom att rita ut dem med musen eller via menyval och definieras genom att fylla uppgifter i dialogrutor. De kopplas till varandra genom att man ritar ett streck mellan dem, och flyttas genom att först markera dem och därefter släpa dem med musen. Utritning av objekthierarkin sker automatiskt vartefter man definierar och kopplar ihop sina objekt. Övriga operationer på objekten sker genom att först markera dem och sedan välja ett



Figur 4.6 Till en process kan en datamodell över associerade data öppnas och editeras.

önskat menyalternativ. Andra operationer som man kan utföra är borttagning av objekt ur aktuella diagrammet, borttagning av objekt ur encyclopedien, ifyllning och editering av detaljer i objektdefinitionerna i ett speciellt detaljfönster (detta gäller förresten för alla typer av diagram i IEW/WS). Man kan också öppna ett annat diagram för ett objekt. Detta diagram kan vara ett nytt dekompositionsdiagram för vidare nedbrytning. För ett objekt av typen ämnesområde eller process kan man öppna ett entitetsdiagram för att beskriva de data objektet associeras med. För alla objekttyper kan man öppna ett egenskapsmatris för att undersöka och editera objektens egenskaper.

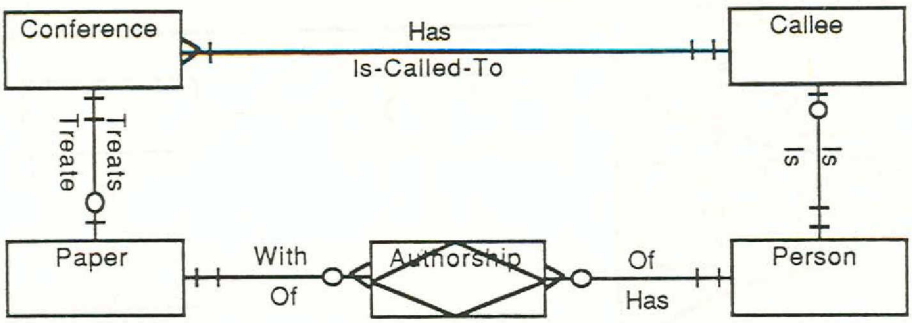
Entity Diagrammer är ett verktyg för att beskriva relationer och samband mellan data i verksamheten. Ett entitetsdiagram (eller Entity-Relationship Diagram) i Planning Workstation används för att identifiera viktiga entiteter i verksamheten. Dessa entiteter kan vara olika typer av personer, platser och andra saker som på något sätt är viktiga för verksamheten.

Genom att skapa ett entitetsdiagram över objekten i verksamheten får man en överblick av vilka typer av data som hanteras och hur dessa hänger ihop. Denna datamodell kan sedan successivt förfinas och kompletteras. I och med att man kan koppla entitetsdiagram till processer och verksamhetsområden kan man få en sammanhängande bild av verksamheten i form av processer och data samt hur dessa relateras till varandra.

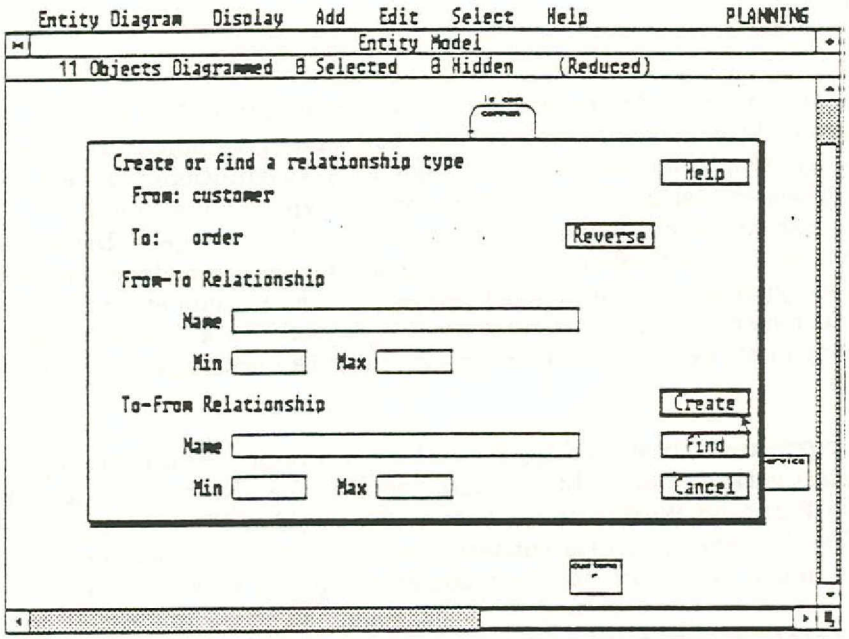
Det finns två slags objekt i ett entitetsdiagram, entiteter och relationer. Relationerna anger vilka samband det finns mellan entiteterna, och hur dessa samband ser ut. I figuren nedan framgår hur entiteterna och relationerna ser ut.

Editering av ett entitetsdiagram går till på motsvarande sätt som editering av dekompositionsdiagram. Entitet antingen ritas med musen eller skapas via menyval. Relationer enklast genom att dra ett streck mellan två entiteter och därefter specificera relationen i en dialogruta som dyker upp på skärmen. Flyttning och bortagning av objekt fungerar likadant som i dekompositionsdiagram.

Man kan låta bli att specificera kardinaliteter för en relation när den skapas. Då kommer de att betraktas som okända och betecknas med frå



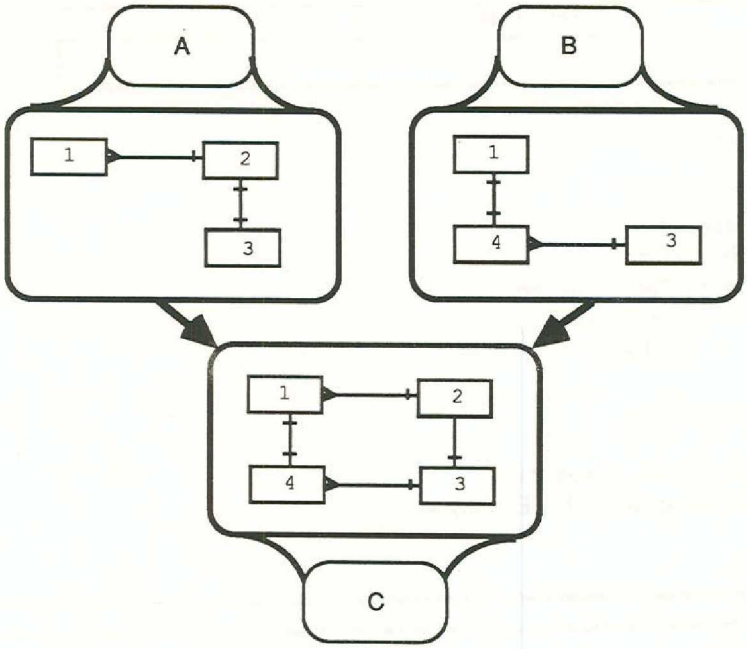
Figur 4.7 Entiteten Authorship är associativ, dvs den beskriver en association eller ett samband mellan entiteterna Paper och Person.



Figur 4.8 En dialogruta för inmatning av namn och kardinaliteter för en relation mellan två entitetstyper - här mellan customer och order.

getecken i diagrammen. Alla entiteter och relationer måste dock ha ett namn. För en relation gäller dessutom att även dess invers måste namnges.

Det som är intressant när det gäller entitetsdiagram, är att de kan kopplas till ett visst objekt (av typen process eller ämnesområde). På detta sätt kan man exempelvis definiera hur en process ser på data som hanteras i verksamheten. Det är faktiskt så att en process datamodell är en delmängd , eller vy (view), av hela verksamhetens datamodell. De objekt man lägger till i en process datamodell kommer automatiskt med i verksamhetens datamodell. När ett objekt tas bort, måste man specificera om det skall tas bort ur hela datamodellen eller endast ur det aktuella diagrammet. Allt detta ger intressanta möjligheter när man skapar sina datamodeller i entitetsdiagram. Man kan först skapa en översiklig model för hela verksamheten som sedan succesivt utökas och kanske modifieras när man bearbetar den i mera detalj via processer och verksamhetsområden. Det kan



Figur 4.9 Vyerna för processerna A och B kombineras till en vy för process C.

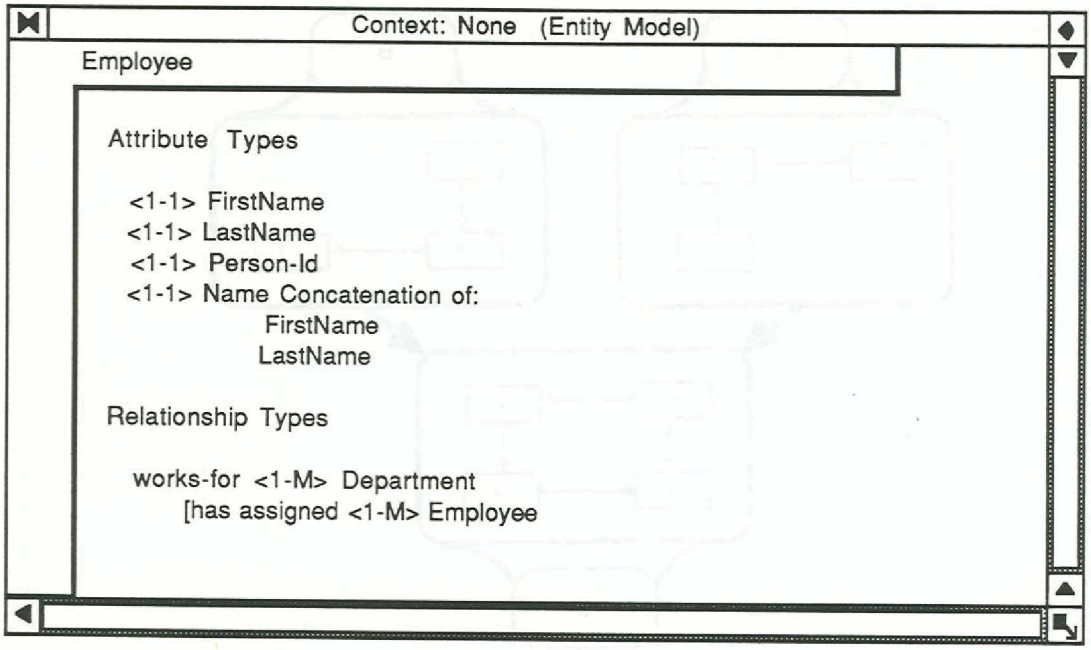
också vara så att när man beskriver en process' syn på verksamhets data endast behöver plocka in befintliga objekt från verksamhetens datamodell. För att göra detta arbetssätt ännu lättare har man i IEW möjligheten att skapa en vy för en process genom att kombinera ihop befintliga vyer via kommandot Combined Views.

Från ett entitetsdiagram kan man inte öppna nya diagram. Däremot kan man öppna ett fönster för närmare beskrivning av en entitet - entity type description. I entity type description- fönstret kan man definiera egenskaper (attribut) för en entitet och se vilka relationer som är med. Varje attribut har en kardinalitet som talar om hur många förekomster egenskapen kan ha och om attributet skall ha ett unikt värde bland alla entitetsinstanser, dvs om det kan användas som unik identifierare för en instans av en entitetstyp. Attribut kan kombineras för att skapa sammansatta attribut. Exempelvis kan entitetstypen *Person* ha attributet *namn* som är en sammansättning av attributen *förnamn* och *efternamn*.

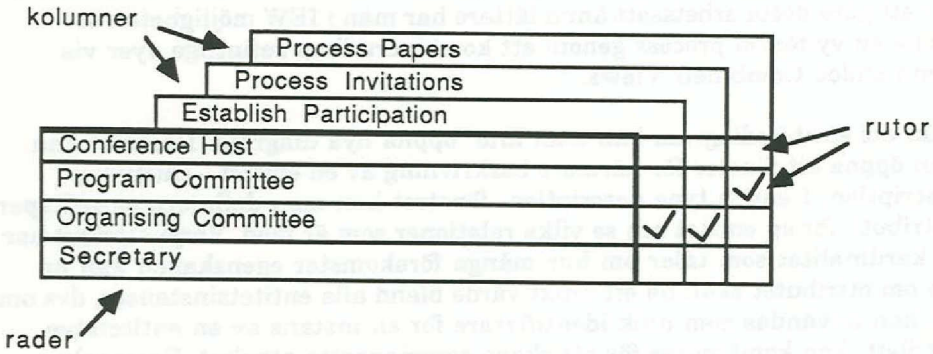
Med **Matrix Diagrammer** kan skapa två typer av matrisdiagram, associationsmatriser och egenskapsmatriser.

Med associationsmatrisdiagram kan men se och editera associationer mellan objektpar. Raderna består av objekt av en typ, kolumnerna av objekt av annan typ. I skärningen mellan rader och kolumner kan man sätta ett kontrollmärke för att indikera en association mellan objekten. En del av associationerna kan även ges egenskaper. Om man har valt att visa egenskaperna och sådana finns specificerade visas det med en asterisk.

När man öppnar en associationsmatris blir man tillfrågad vilka objektyper raderna och kolumnerna skall bestå av (de kan vara lika). Därefter visas matrisen och kan editeras. Editering är enkelt. En association läggs till eller tas bort genom att klicka med musen i rutan som utgör skärningen mellan de avsedda objekten. Man kan ocksa skriva in asso-ciationegenskaperna direkt i rutan. Även nya objekt kan skapas i



Figur 4.10 Entity type description -fönster för entiteten Employee. Atributen och relationerna till entiteten visas.

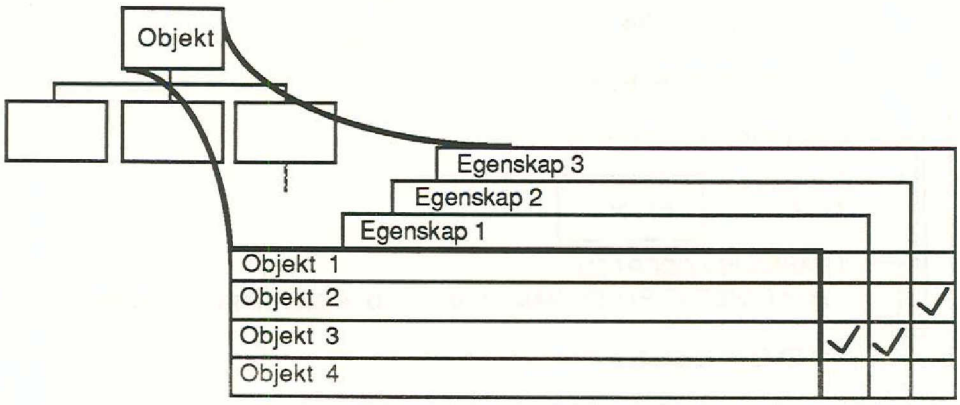


Figur 4.11 En associationsmatris. Här är raderna organisationsenheter och kolumnerna är processer. Associationstypen är: "enhet är ansvarig för process". Matrisen visar t.ex. att organisationskommittén svarar för hanteringen av inbjudningar.

associationmatrisen om det behövs, och de skapade objekten blir omedelbart tillgängliga i alla andra diagram.

Egenskapsmatrisdiagrammen fungerar på samma sätt och visar vilka egenskaper objekten av ett objekttyp har.

Från associationsmatrisdiagram kan inga andra diagram öppnas. Från egenskapsmatriser kan man däremot öppna ett entity type description-fönster för objekt av typen entitet. Dessutom kan man för varje nedbrytbar objekt öppna ett dekompositionsdiagram och ett entitetsdiagram. En egenskapsmatris kan öppnas från många olika diagram.



Figur 4.12 Egenskapsmatris för ett objekt i ett dekompositionsdiagram.

Analysis Workstation

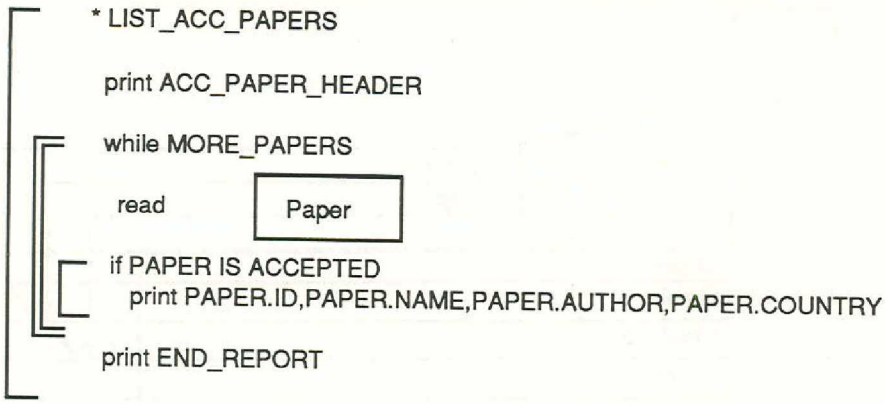
Analysis Workstation (AWS) erbjuder ett antal beskrivningstekniker karakteristiska för aktiviteterna i den tidigare beskrivna pyramidens andra del. Syftet med analysen är att förstå vad som händer i verksamhetens olika delområden och hur information varieras, skapas och modifieras av processerna i verksamheten. För att uppnå detta måste man analysera vilka data och processer som är nödvändiga för verksamhetens funktion samt hur dessa data och processer relateras till varandra. Detta arbete har inletts i planeringssteget och meningen är att resultaten från arbetet med Planning Workstation skall utgöra en grund för analysfasen, som dock snarare koncentrerar sig på verksamhetens delområden än verksamheten som helhet.

Analysis Workstation tillhandahåller beskrivningstekniker av vilka några ofta används i strukturerad analys (structured analysis). De uppgifter man bör kunna utföra med hjälp av Analysis Workstation är:

- utveckla en detaljerad datamodel för ett verksamhetsområde
- utveckla en detaljerad processmodell för ett verksamhetsområde
- identifiera områden för design av informationssystem
- avgöra prioriteten för design av ett informationssystem

De verktyg som Analysis Workstation tillhandahåller är Action Diagrammer, Data Flow Diagrammer, Decomposition Diagrammer och Entity Diagrammer. De två senaste är samma som i Planning Workstation, förutom att Entitetsdiagram i Analysis Workstation tillåter även beskrivning av datatyper för attribut (dvs lagringsformat, längd osv). I detta avsnitt kommer vi att begränsa oss till Action Diagrammer och Data Flow Diagrammer.

Med **Action Diagrammer** konstruerar man action diagrams, som är en teknik för att på ett enkelt sätt beskriva programlogiken för processer. Ett action diagram baserar på ett hierarkiskt system av paranteser som utgör byggnadsblocken i action diagrams. Dessa paranteser beskriver normalt ett antal sekventiella operationer , men kan kapslas på flera nivåer . De kan dessutom innehålla anrop på andra processer och databasoperationer. Meningen är att man med Action Diagrammer skall beskriva programlogiken för processer på lägsta nivån i processhierarkin.



Figur 4.13 Action diagram för processen List accepted papers.

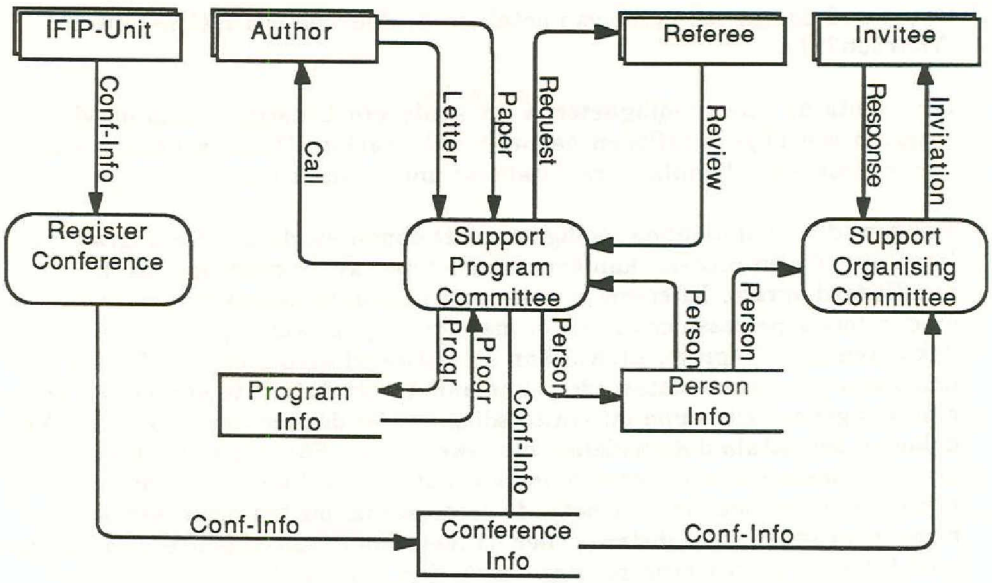
Komponenterna i AWS's actiondiagrams är actions, dataflöden, databasoperationer, exitsats, sekventiella processer, samt ett antal repetitions- och selektionssblock. En Action beskriver en operation, ett program, en subrutin eller en kodrad i textformat. Dataflöden är indata eller utdata definierade för processen i dataflödesdiagram. Databasoperationerna är grundläggande aktioner för hantering av lagrade data, och finns i fyra typer: Create, Read, Update och Delete. En exitsats medför att programexekvering omedelbart avbryts. De sekventiella processerna är processer som man redan har specificerat genom action diagrams, eller kommer att göra det. Det finns tre olika slags repetitionsblock: Do...Until-, Do While-, och For Each-block. Utöver dessa finns det tre olika typer av If-block: vanlig if-sats, If Else-block och Multiway-block som är en if-sats med flera else-if efterföljare.

När man konstruerar action diagrams lägger man till nya block och actions genom att placera markören på ett önskat ställe och antingen skriva in enkel operation eller genom att plocka in repetitions- eller selektionsblock, sekventiell process eller databasoperation via ett menyval. Man kan också skapa nya processer i ett actiondiagram. Editering av ett action diagram sker i huvudsak genom att klippa och klistra aktioner och block eller via menyval.

Från ett action diagram kan man öppna ett entitetsdiagram för att undersöka och editera datamodellen för en process. Och eftersom en process i ett actionsdiagram representerar ett annat actionsdiagram kan man välja en process och öppna actionsdiagram för den.

Data Flow Diagrammer är verktyg för att skapa dataflödesdiagram. Ett dataflödesdiagram används huvudsakligen inom verksamhets- och systemanalys för beskrivning och analys av de grundläggande procedurella systemkomponenter och data som flyter mellan dem. Dataflödesdiagram används ofta som det första steget i verksamhetsmodellering och strukturerad analys. De visar hur data flyter genom en verksamhet eller ett datasystem och vilka transaktioner dessa dataflöden resulterar i.

Komponenterna i ett dataflödesdiagram är processer, dataflöden, datalager, externa agenter och knutpunkter. I knutpunkterna kan flera dataflöden förenas till ett sammansatt dataflöde eller ett sammansatt flöde kan falla sönder till mindre dataflöden. Ett dataflöde skall börja eller sluta i en process, extern agent , datalager eller knutpunkt. Ett dataflödesdiagram kan ha flera nivåer. Exempelvis kan en process bestå av flera andra processer på lägre nivå .



Figur 4.14 Exempel på ett dataflödesdiagram.

Editering av dataflödesdiagram går till på samma sätt som editering av dekompositions- och entitetsdiagram, och vi kommer inte att behandla det närmare. I stället koncentrerar vi oss på det som kanske skiljer IEW's dataflödesdiagram från motsvarande i andra CASE-verktyg.

När man man skall använda tidigare definierade processer i ett dataflödesdiagram - dessa må vara definierade i dekompositions- eller actionsdiagram - behöver man inte definiera om dem utan de kommer automatiskt att ritas ut på rätt nivå. Om t.ex. processen A enligt dekompositionshierarkin består av processerna B och C, så kommer B och C ritas ut automatiskt när man öppnar ett dataflödesdiagram för att i detalj beskriva processen A. Även dataflöden som man redan har definierat, kan återanvändas genom att peka ut dem i en meny. Ett dataflöde kan således förekomma på flera platser i ett diagram, och kommer då alltid att ha samma definition.

Två viktiga principer framhålls i IEW när det gäller att skapa dataflödesdiagram. Det första är att man inte bör ha flera instanser av ett datalager mellan två processer. Den andra principen gäller externa agenter. För dessa gäller att man inte bör ha flera instanser av en extern agent i en process eller dess underprocesser på någon nivå. Inte heller skall man ha externa agenter på andra nivåer än rotnivån, dvs högsta nivån i processhierarkin. Visserligen är dessa operationer tillåtna, men om någon av dem utförs kommer IEW att varna om att det finns risk för inskonsistenta data och processmodeller. Om man ändå bryter mot dessa regler kan man råka ut för att få åtgärda inkonsistenta modeller i ett senare skede.

En intressant egenskap hos IEW's dataflödesdiagram är möjligheten att kunna identifiera och deklarera processer som central transforms och transaction centers. Visserligen utförs denna analys inte automatiskt men manualen beskriver ingående hur man manuellt identifierar central transforms och transaction centers. Denna teknik används för att konvertera dataflödesdiagram till strukturkartor. Den är inte på något sätt specifik för Information Engineering, utan används främst inom tillämpningarna av

structured design och beskrivs i detalj av E. Yourdon och L.Constantine, [Yourdon79].

Inressanta är också möjligheterna att analysera konsistens i dataflödesdiagram och följa dataflöden genom nivåhierarkin. Dessa aspekter kommer vi dock att behandla mera detaljerat under punkt 4.3.4

Förutom den ovannämnda möjligheten att öppna ett dataflödesdiagram på lägre nivå för en process, kan även andra typer av diagram öppnas från ett dataflödesdiagram. Eftersom processerna i ett dataflödesdiagram är en del av den totala processhierarkin kan man för varje process öppna ett dekompositionsdiagram. Likaså kan ett entitetsdiagram öppnas för att analysera processens datamodell. Innehållet i ett dataflöde kan enkelt specificeras genom att öppna ett entitetsdiagram för det och där beskriva vilka delar av den totala datamodellen som skall ingå. För löven, dvs processerna på lägsta nivån i processhierarkin kan action diagram öppnas. Det är ju till sist löven som är de arbetande processerna, medan processerna på högre nivå snarare är abstraktioner på mera komplexa uppgifter som utförs med hjälp av processerna på lägre nivå. Den programlogik som specificerats för lövprocesserna kan senare utnyttjas i designfasen då det är dags att börja designa procedurer i informationssystemet.

Datatyper och flödesbeskrivngar:
Förutom de ovannämnda beskrivningsteknikerna finns det några andra hjälpmedel i Analysis Workstation för specificering av data och dataflöden. Från ett entity type description- fönster i entitetsdiagram eller encyclopediens objektlista kan man öppna ett fönster för beskrivning av datatyper för attribut. Här kan man namnge datatyper och ge deras semantiska innebörd, som helt enkelt motsvarar datatypbeskrivningarna i COBOL's datadivision.

Flödesbeskrivningar (flow expressions) är det andra sättet att specificera innehållet i dataflöden. Det första var ju att definiera dess datamodell med ett entitetsdiagram. Flödesbeskrivningar specificerar komponenterna och deras sammansättning i ett dataflöde med en speciell syntax.

```
Person.Id | (Person.Firstname + Person.LastName)
```

Figur 4.15 En flödesbeskrivning för ett dataflöde. Dataflödet innehåller antingen attributet Person.Id eller attributen Person.FirstName och Person.LastName.

Detta sätt att ange innehållet i ett dataflöde är kanske krångligare än att göra det via entitetsdiagram. Men å andra sidan kan man specificera dataflöden mera exakt (man kan t.ex. ange alternativa attribut som i figur 4.15). Och är man van så kanske det går lika snabbt eller snabbare än via entitetsdiagram.

När man analyserar och specificerar system är det viktigt att det till beskrivningsteknikerna man använder sig av finns konsistens- och fullständighetskontroller. I Analysis Workstation finns ett antal rapport- och analysfunktioner för att kunna göra dessa typer av kontroller. Eftersom dessa funktioners kapacitet beror på innehållet i och hantering av datakatalogen, kommer de att beskrivas under punkt 4.3.4 i samband med granskningen av IEW's encyclopedi.

Design Workstation
Design Workstations uppgift är att tillhandahålla verktyg för design av informationssystem, pyramidens tredje nivå. Vid design flyttar man tyngdpunkten från den logiska beskrivningen av *vad* systemet skall göra till den fysiska specifikationen *hur* systemets funktioner skall realiseras. Processerna och data skall nu överföras till procedurer och datastrukturer. Även procedurernas gränssnitt mot användaren skall definieras i denna fas. Genom att använda resultaten från planering och analys vill man kunna konstruera system som har sin grund i verksamhetens strategiska mål och tillfredställer informationsbehoven i verksamheten.

Med verktygen i Design Workstation kan man specificera programstrukturer, procedurlogik, skärmbilder samt databas- och filstrukturer. Syftet med Design Workstation är att:

- få med slutanvändarna i designprocessen
- minska tiden för design och realisering av system
- automatisera design, dokumentation och underhåll av system
- koppla automatisering av design till påföljande kodgenerering

För att uppfylla dessa mål är Design Workstation utrustad med en mängd verktyg: Structure Chart Diagrammer, Action Diagrammer, Presentation Diagrammer, Data structure Diagrammer, Flat File Database Diagrammer, Relational Database Diagrammer samt Hierarchical Database Diagrammer.

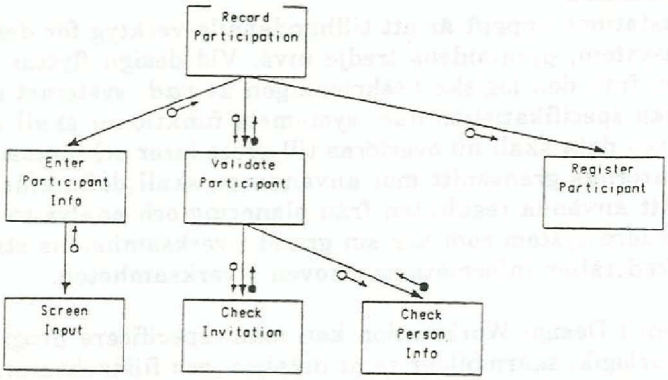
Structure Chart Diagrammer används i strukturerad design för att beskriva systemets uppbyggnad med hjälp av moduler. Structure charts beskriver anropstrukturen mellan modulerna och visar hur modulerna kommunicerar med varandra via parametrar.I Structure charts koncentrerar man sig på *vad* modulerna gör. *Hur* de arbetar, dvs deras programlogik beskrivs i Design Workstation med hjälp av action diagrams.

Komponenterna i structure charts är moduler, anrop och parametrar. Det finns olika typer av moduler: vanliga moduler - dvs en mängd programsatser som läser, bearbetar och matar ut data, ospecificerade moduler vars syfte ännu inte bestämts, 'include routines' som är delar av andra moduler samt biblioteksmoduler som placerats i ett bibliotek. Anropen kan vara synkrona, asynkrona eller rekursiva. Parametrarna är antingen data eller kontroller (flags) och kan vara anropsparametrar eller formella parametrar.

En modul kan anropas av flera andra moduler. För att indikera detta grafiskt kan alla anropspilarna dras till denna modul. Normalt har man dock en instans av modulen för varje anrop, för att inte klottra ned diagrammet. Om en modul kan betraktas som en standardrutin eller den anropas av andra program, läggs den vanligen i något programbibliotek där den är allmänt tillgänglig. Biblioteksrutinerna har en speciell symbol i structure chart diagram.

Man skapar och editerar structure charts genom att rita med musen, skriva in definitioner via tangentbordet och utföra diverse operationer via menyerna.

För att definiera programlogiken för en modul kan man markera den och öppna ett action diagram. Alla anrop och anrops- respektive formella para



Figur 4.16 Structure chart . De små pilarna är anropsparametrar. Ihåliga är data, ifyllda kontroller. Parametrarna visas mera detaljerat då ett action diagram öppnas för en modul.

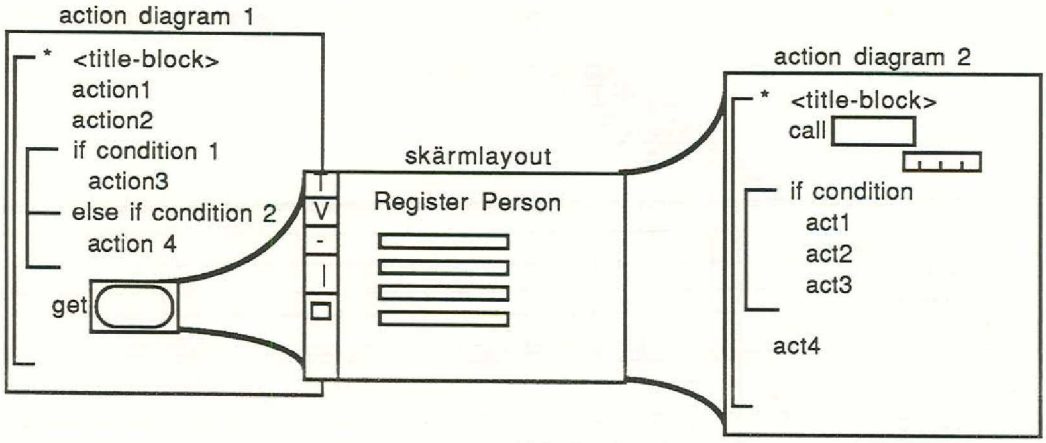
metrar som redan definierats för modulen visas då automatiskt. På detta sätt kan man definiera systemstrukturen med structure charts och sedan definiera logiken för de enskilda delarna i systemet med actiondiagrams.

Action Diagrammer i Design Workstation skiljer sig något från motsvarande i Analysis Workstation. Action diagrams i Design Workstation beskriver procedurer i form av moduler som exekveras och kan anropas av andra moduler. Dessa procedurer används sedan vid realisering av informationsystemet. Modulernas programlogik definieras på samma sätt som processernas programlogik i Analysis Workstation. Meningen är att man utnyttjar processbeskrivningarna från analysfasen. De hämtas in i Design Workstation och används som utgångsmaterial vid design av systemprocedurer.

Komponenterna i DWS's action diagrams är actions, data actions, exitsats, moduler, parametrar, datastrukturer, parametrar, skärmbilder samt repetitions- och selektionblock. Arbetssättet är det samma som i action diagrams i Analysis Workstation.

Om man har specificerat systemstrukturen med hjälp av structure charts innan man börjar beskriva modulernas innehåll, sparar man mycket arbete, eftersom alla anrop och parametrar redan finns i modulerna. Om man utnyttjar principerna i strukturerad design - vilket innebär att modulerna på högre hierarkisk nivå främst används till att styra beteendet hos de underliggande modulerna medan modulerna på lägsta niva får utföra det egentliga arbetet - så behöver man normalt "bara" definiera styrlogiken för modulerna på högre nivåer och beskriva databasoperationerna och databearbetningen samt in- och utmatning i modulerna på lägsta nivå. Men det finns givetvis andra sätt och IEW sätter inga begränsningar här.

I ett action diagram kan man öppna ett nytt actiondiagram för en anropad modul. För en anropad modul kan man också öppna ett structure chart för att se hur den är uppbyggd av andra moduler. Även data bearbetas i action diagram och därför finns möjligheten att direkt härifrån specificera datastrukturer via data structure diagram (behandlas senare i detta avsnitt). Vi nämnde tidigare att man kan designa skärmbilder i Design Workstation. Har man gjort det kan man referera till skärmbilderna i actiondiagram. Då är det också möjligt att öppna ett fönster med skärmbeskrivning direkt från action diagram.



Figur 4.17 Ett action diagram där en skärm anropas. Till denna skärm tillhör ett annat action diagram som definierar operationer för hantering av skärmvariabler.

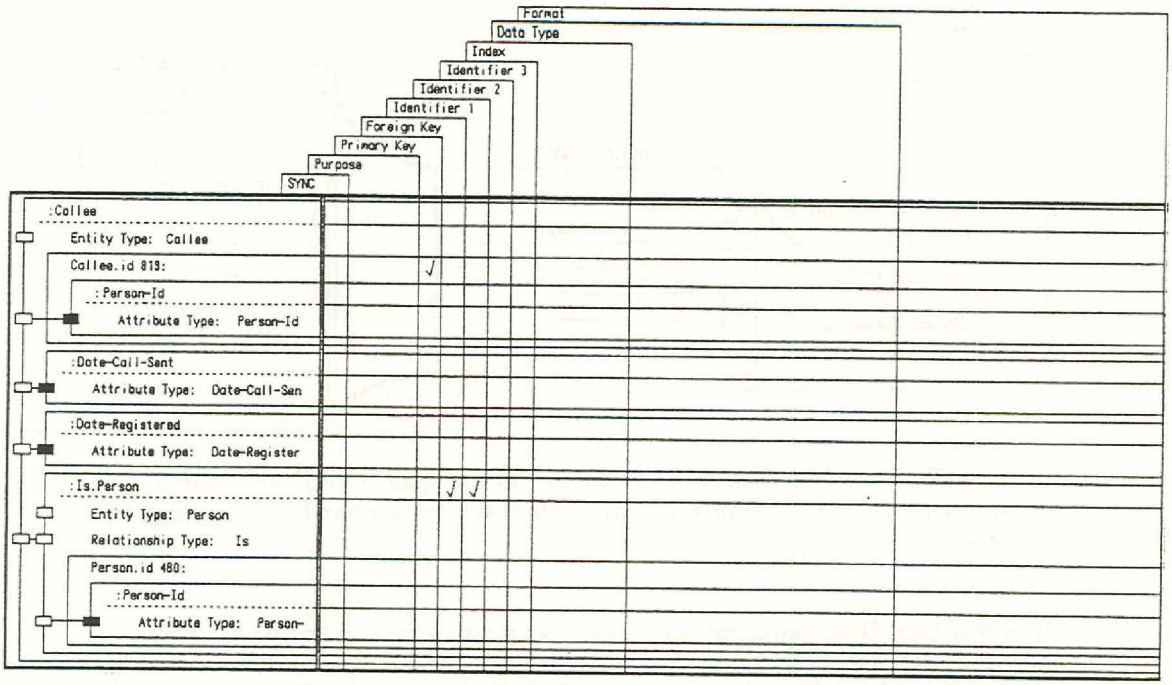
Presentation Diagrammer är ett verktyg för att skapa skärmbilder för monitorer under IMS eller CICS. Text och grafiska tecken för IBM 3270 kan definieras och editeras med musen. Variabelfält för visning eller in- och utmatning av data skapas genom att rita med musen fylla i dialogrutor. Variablerna som skärmen hanterar är lokala för skärmen eller globala, dvs finns definierade även utanför skärmen. Variabler kan skapas och deras datastruktur specificeras via data structure diagram direkt inifrån skärmbeskrivningen. För text-, variabel- och grafiska fält kan attribut specificeras för att indikera fältets visningsformat och huruvida det är skyddat eller inte.

Operationer som hanterar enskilda variabler eller hela skärmlayouter kan definieras med action diagrams inifrån skärmbeskrivning och kopplas till variabler eller skärmar. Dessa operationer kan användas för att t.ex. initiera variabelvärden, kontrollera och bearbeta inmatade värden eller visa meddelanden på skärmen. Observera att dessa operationer inte har något att göra med programlogiken i den anropande systemmodulen, utan är lokala för skärmen.

En färdigdefinierad skärm kan imiteras i Design Workstation för att skapa en bild hur det kommer att se ut i ett fungerande system. För att inga missöden skall uppstå då skärmbilderna tas i verkligt bruk, finns möjligheten att kontrollera att skärmdefinitionen verkligen är fullständig och att inga konflikter mellan objekt eller variabler föreligger.

Den information man skapat i skärmbeskrivningarna är tillgänglig i action diagrams och i data structure diagrams. I action diagrams har skärmbilderna en egen modulsymbol som kan öppnas för att visa skärmdefinitionen. En data structure diagram kan öppnas för visning och specificering av variabeldefinitioner för en skärm.

Data Structure Diagrammer är ett verktyg för data structure diagrams. Dessa används för design av detaljerade datastrukturer för flera typer av data: skärmbilder, relationer, poster i databaser av flata filer, segment i hierarkiska databaser samt parametrar i modulanrop. Man också skapa mera generella datastrukturer för användning i flera sammanhang.



Figur 4.18 Data structure diagram. Raderna utgör en hierarkisk beskrivning av strukturer och kolumnerna består av strukturers egenskaper.

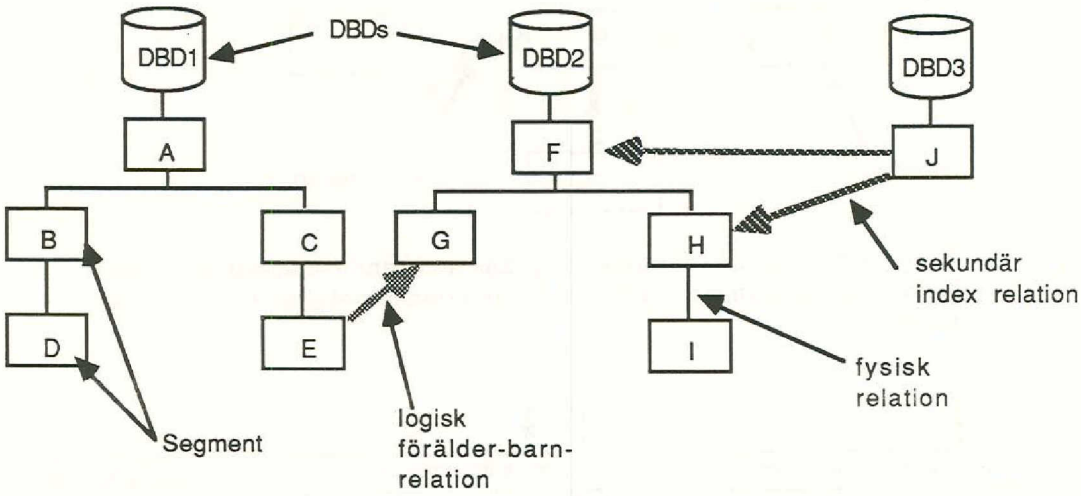
Komponenterna i ett data structure diagram är struktur, repetitionsblock, "option group", entitetsbindning, attributbindning, relationsbindning samt kolumner som anger egenskaper hos strukturer. Datastrukturerna kan vara hierarkiska, dvs en struktur kan innehålla andra. De kan också innehålla flera förekomster av en struktur eller till och med alternativa förekomster mellan olika strukturer (option groups). En struktur kan definieras som entitet med entitetsbindning. En struktur som ingår i en annan kan definieras som attribut till denna med attributbindning, eller man kan ange en relation mellan dem med en relationsbindning.

Oftasbehöver man inte definiera sina datastrukturer med datastruktur--diagram. Man kan göra det, åtminstone initialt , i ett annat diagram, t.ex. i något av databasdiagrammen, och sedan komplettera eller modifiera med datastrukturdiagram. Det är också så att beskrivningar i datastrukturdiagram är mera detaljerade i andra typer av diagram.

Från datastrukturdiagram kan man inte öppna andra diagram eftersom de är databeskrivningar på lägsta nivå. Däremot kan datastrukturdiagram öppnas från skärmbeskrivningsdiagram, actiondiagram (för parametrar) och från alla databasbeskrivningsdiagram.

Hierarchical Database Diagrammer är egentligen en samling av tekniker för att beskriva hierarkiska dabaser på olika sätt och på olika detaljeringsnivåer. Här ingår Database Description Diagram, DBD, Database Descrition Diagram i textformat, DBD Text , och Program Specifikation Block, PSB.

Database Description Diagam, DBD används för design av databaser med hjälp av en hierarki av segment och relationer mellan dessa. Komponenter i DBDs är segment (dataposter), fysiska och logiska relationer mellan segment samt sekundära indexrelationer (secondary index relationships).



Figur 4.19 Database description diagram med olika typer av relationer mellan segmenten.

Man kan också ange vilken typ av åtkomstmekanism som skall tillämpas för en DBD. Man kan välja mellan DEBD, HISAM, HIDAM, HDAM, INDEX eller LOGICAL.

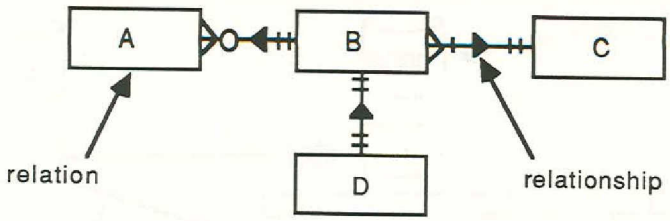
Från ett DBD diagram kan man öppna ett data structure diagram för detaljerad beskrivning av datstrukturer för segment. Man också visa beskrivningen av ett DBD i textformat. Denna textuella beskrivning av DBDs är mera detaljerad än den grafiska; för segment visas även fält och deras värden.

Program Specification Block, PSB, är en listning av program control blocks, PCBs, som är en PSB-version av DBDs. Precis som DBD-textbeskrivning, är PSBs främst ett verktyg för design av IMS-databaser och kräver att man behärskar DL/I-språket . Vi kommer inte att beskriva dessa verktyg närmare.

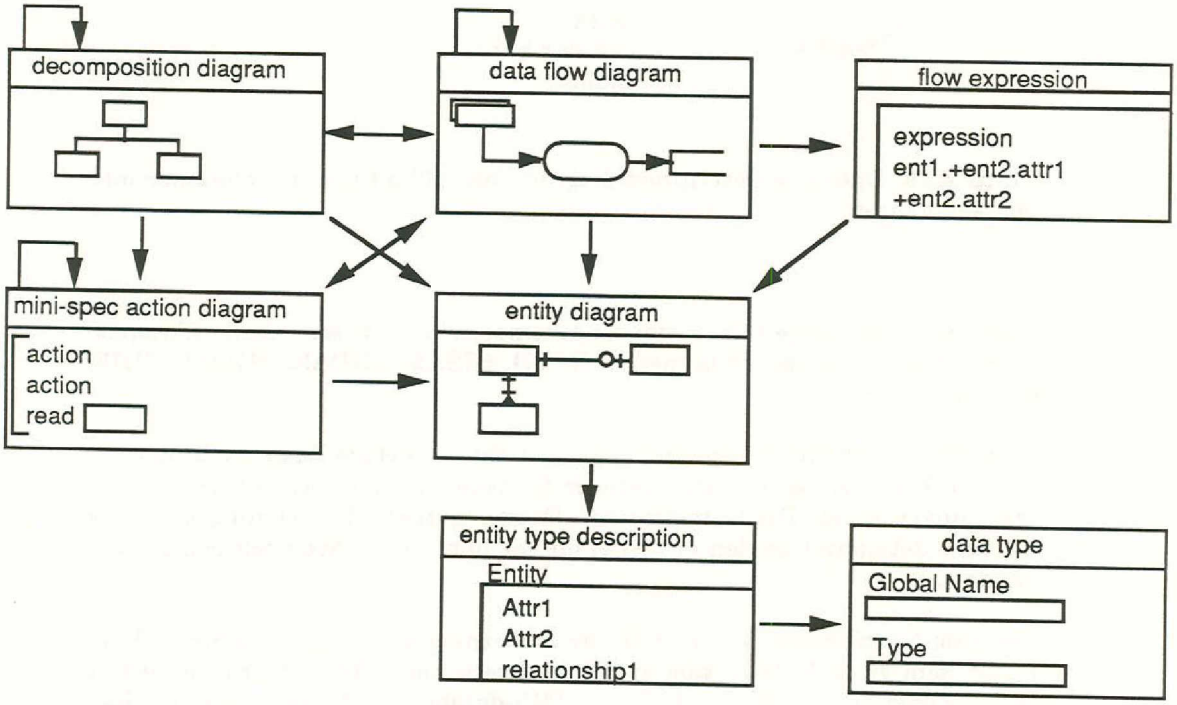
Flat File Database Diagrammer är ett verktyg för beskrivning av poststrukturen i en databas för flata filer. Man definierar poster med bestämd längd och samband mellan posterna. Detta verktyg påminner starkt om entitetsdiagram, bortsett från att relationerna inte har något namn.

Relational Database Diagrammer är motsvarande verktyg för relationsdatabaser. Här har dock samband benämningen relationships och poster (tabeller) kallas för relationer. Detta verktyg har en särskild fördel. En datamodell beskriven i ett entitetsdiagram kan automatiskt transformeras till modell för relationsdatabaser. I övrigt är verktyget identiskt med flat file database diagrammer.

För båda dessa databasverktyg gäller att man inifrån dem kan öppna ett data structure diagram för visning eller editering av poster respektive relationer.



Figur 4.20 Flat file database diagram. samband (relationships) har bestämda riktiningar. Diagram för relationsdatabaser har samma utseende.



Figur 4.21 Exempel på samverkan mellan diagrammen i Analysis Workstation.

4.2.3 Integrationsgrad mellan beskrivningstekniker;

Det har säkert redan framgått att integrationen mellan de olika beskrivningsteknikerna är mycket hög i IEW. Oberoende av var man definierar ett objekt, kommer den att finnas tillgänglig i alla berörda diagram. I de fall det är relevant, ritas det ut automatiskt. Man kan arbeta med flera diagram samtidigt och dra nytta av specifikationerna oberoende var de skapats. Det som mest framhäver den höga integrationsgraden är dock möjligheten att öppna ett diagram inifrån ett annat för mera detaljerade beskrivningar av ett objekt eller för att se på det ur en annan synvinkel. Som exempel kan vi ta möjligheten att för en process i ett dekompositionsdiagram öppna ett dataflödesdiagram för att se processerna på lägre niva och hur data flyter mellan dem.

När det gäller diagram i olika moduler kan de inte integreras på samma sätt eftersom modulerna är separata program. Det finns dock en del

sammankopplingar. Alla diagram delar på (relevanta) data oberoende av vilken modul de definierats i. Därmed har man brygga mellan de olika modulerna via gemensamma data och vissa beskrivningstekniker. Till exempel är entitetsdiagram och dekompositionsdiagram gemensamma mellan Planning Workstation och Analysis Workstation. Likaså kan action diagrams i Design Workstation utnyttja resultaten från action diagrams i Analysis Workstation, även om de används för något olika syften.

4.2.4 Metodstöd;

Vi har redan konstaterat att IEW inte är bundet till någon specifik systemutvecklingsmetod, även om viss orientering mot Information Engineering - liknande metoder finns. IEW sätter inga begränsningar på vilka beskrivningstekniker som skall användas eller i vilken ordning de skall användas. Användare kan alltså utnyttja de tillgängliga beskrivningsteknikerna fritt och kombinera användningen av dem på ett sätt som passar dem. Man kan dock inte välja sin metod helt fritt. Även om IEW har ett rikt sortiment av beskrivningtekniker och verktyg så saknas stöd för ett antal kända metoder. Typiskt för dessa metoder är att de har sina egna speciella bekrivningstekniker. För att ta några exempel kan vi nämna JSD-metoden, [Jackson83], och Structured Requirements Definition, [Orr81]. Man kan dock med god fog säga att IEW ger stöd för ett stort antal tänkbara metoder och metodkedjor. Om en metod har egenskaper som gör att IEW inte kan användas utan vidare, kan man ta en annan ställning och fråga sig om inte metoden kan modifieras så att den passar för IEW i stället. Det kan vara värt att utreda om man kan kosta på sig en sådan modifiering av metoden. Särskilt om IEW innehåller beskrivningstekniker som är likvärdiga med motsvarande tekniker i metoden.

Naturligt nog är IEW speciellt lämpat för metoder som följer riktlinjerna i Information Engineering. Typiskt för dessa metoder att man börjar med att analysera informationsbehovet och skapa datamodeller över verksamheten för att sedan övergå till analys och design av procedurer som behövs för att tillfredställa informationsbehovet. De vanligaste beskrivningsteknikerna för dessa ändamål är entity-relationship-diagram, dataflödesdiagram och dekompositionsdiagram. IEW har alla dessa och dessutom verktyg för beskrivning av programlogiken för procedurerna och överföra datamodellerna till databasstrukturer. En del av aktiviteterna i den här typen av metoder kan också delvis automatiseras i IEW. Det som för dagen saknas i IEW är möjligheten att utifrån en modell över verksamhetens processer skapa en systemstruktur i form av procedurer som verkställer dessa processer. Att automatisera detta helt förefaller svårt, men det finns exempel på algoritmer som ger väsentlig hjälp för detta arbete.

Det stöd som finns för transformation av dataflödesdiagram till strukturkartor i IEW, är tämligen obetydligt. För att illustrera detta kan vi betrakta hur IEW förhåller sig till en annan vida tillämpad och i varierande former förekommande systemutvecklingsmetod, Structured Analysis and Structured Design (SASD). I Structured Analysis analyseras verksamheten och det befintliga systemet med hjälp av dataflödesdiagram, som också används för att beskriva det nya systemet. En normaliserad datamodell över informationen i dataflöden skapas och processlogiken för processerna i dataflödesdiagrammet specificeras. Structured Design går ut på att överföra dataflödesdiagrammet från analysfasen till strukturkartor som beskriver den nya systemstrukturen i form av en modulhierarki. Det finns två huvudsakliga strategier att göra denna transformation. Det ena är att i ett dataflödesdiagram identifiera s.k. central transforms, som består

av en eller flera processer. Med hjälp av dessa kan en initial strukturkarta tas fram för att beskriva det nya systemets struktur med i form av en modulhierarki, som sedan kompletteras och förfinas. Ett annat sätt att ur dataflödesdiagram konstruera strukturkartor är att hitta s.k. transaction centers bland processerna, som sedan används för att komma fram till en modulhierarki. I IEW kan en process deklareras som transaction center och/eller central transform, men man får inget stöd för att hitta dessa. Inte heller kan man utnyttja dessa deklarationer för annat än manuell analys, för vilken de är tämligen värdelösa eftersom man inte kan definiera vilka central transforms/transaction centers processerna ingår i. Så även om beskrivningsteknikerna för SASD finns så måste överföring av dataflödesdiagram till strukturkartor göras för hand, vilket kan vara nog så betungande för stora och komplexa system.

Metodlösa verktyg brukar ofta mötas av två typer av argument. Det ena är att det finns risk för inkonsistenta och ofullständiga specifikationer om alltför vidlyftiga och informella arbetssätt kan tillämpas. Det andra är att om verktyget har en väl utstakad metod eller metodkedja, så får man användarna att arbeta på ett enhetligt och formellt sätt med väldefinierade arbetsuppgifter i varje moment. Det första argumentet klarar IEW lätt. Med den kontrollapparat som IEW tillhandahåller i form av rapporter och analyser minimeras risken till brister i specifikationerna till samma eller lägre nivå som för de flesta metodknutna CASE-verktyg. Det andra argumentet är förstås riktigt vad gäller saknaden av väldefinierade arbetsuppgifter för varje systemutvecklingsfas. Denna saknad kan dock kompenseras genom att använda en väldefinierad metod som IEW lämpar sig för. Man har också större frihet vid val av en sådan metod.

Efter att granskat IEW's egenskaper ur metodsynpunkt, kan vi dra slutsatsen att om de beskrivningstekniker som IEW erbjuder lämpar sig för den metod man vill använda, kan IEW förmodligen utnyttjas på effektivt sätt. Det är dock önskvärt att man har egna effektiva sätt att manuellt transformera logiska systembeskrivningar till konkreta systemstrukturer. Har man väl klarat av det så kan programlogiken för modulerna i systemstrukturen relativt enkelt beskrivas med action diagrams. Förutsatt att dessa actiondiagrams har beskrivts på ett lämpligt sätt kan även automatisk kodgenerering företas.

4.2.4 Begränsningar i och avvikelser från metoden

När det gäller CASE-verktyg i allmänhet kan man tänka sig att de metoder de är konstruerade för lider av vissa brister. Dessa brister kan medföra begränsningar för vad man kan göra med verktyget. Man kan också tänka sig att även om en viss metod har en del brister så kan verktyget överbrygga dessa genom att göra lämpliga avsteg från metoden.

Vad IEW beträffar så kan man notera ett par egenskaper som delvis beror på vilken metod som används och delvis på det faktum att IEW baserar på beskrivningstekniker som är typiska för Information Engineering.

Om man tittar på metoden Structured Analysis and Structured Design så kan man konstatera att den datamodell som man kommer fram till är en biprodukt som extraheras ur innehållet i dataflöden i dataflödesdiagrammet. Om processerna eller dataflödena i verksamheten ändras - vilket de ofta gör, kan också datamodellen behöva modifieras. Detta kan orsaka problem, särskilt för stora system med komplicerad fil- och datastruktur. Denna effekt kan dock delvis elimineras genom att använda inslag från

Information Engineering, som i större utsträckning baserar systemdesignen på en statisk datamodell som är mera oberoende av förändringarna i verksamheten.

Men just användning av Information Engineering beskrivningteknikerna för specificering av händelser och processer i verksamheten leder till att tidskritiska system där händelserna har starkt tidsberoende blir svåra att specificera. I Information Engineering är ju den statiska datamodellen i förgrunden medan i tidskritiska system är datamodeller ofta av sekundär betydelse. IEW kan inte riktigt kompensera detta. Man kan dock framhålla att informationssystem normalt inte är tidskritiska utan snarare dataorienterade.

I övrigt är det svårt att tala om begränsningar i de metoder - eller avvikelser från dem - som kan komma i fråga i samband med IEW, eftersom IEW ger så stor frihet vid metodval. Vårt intryck är dock att IEW innehåller tillräckligt med faciliteter för att eliminera en del av begränsningarna i en metod, om lämpliga modifieringar i metoden genomförs.

4.3 Systemutvecklingsdatabasen

4.3.1 Datakatalogen

IEW's centrala datakatalog heter Encyclopedia, dvs encyklopedi eller uppslagsverk. Namnet syftar till att det inte bara är frågan om att lagra data om enskilda objekt, utan snarare om att lagra och hantera kunskap om systemet under utveckling. Encyclopedien är en semantisk databas över objekt, deras egenskaper och associationer mellan dem . Den möjliggör kontroller av fullständighet och konsistens i systembeskrivningarna.

Encyclopedien lagrar huvudsakligen två former av information om ett system, en datamodell och en processmodell. Datamodellen är den statiska datamodellen om verksamhetens eller systemets informationsbehov . Processmodellen beskriver funktioner och processer i verksamheten eller systemet . Även kopplingarna mellan dessa modeller lagras: vilka processer hanterar vilka data.

När man ritar diagram och skapar och specificerar objekt, lagras informationen om dem i encyclopedien. Dessa objektdefinitioner är inte specifika för diagrammet, utan lagras centralt och endast en gång. Dessa definitioner kan nås och modifieras från andra diagram eller direkt i encyclopedien. För att möjliggöra en effektiv hantering av informationen i encyclopedien använder IEW ett inbyggt expertsystem, Knowledge Coordinator. Knowledge Coordinator håller reda på objeken och deras betydelse i den totala modellen. De olika diagrammen ses som projektioner över den totala modellen och verktyg för bearbetning av denna modell. Knowledge Coordinator realiserar detta synsätt genom att automatiskt uppdatera encyclopedien vid editering av diagram. Knowledge Coordinator håller även i den grafiska representationen av encyclopediens innehåll.

En användare kan skapa flera encyclopedier och växla mellan dem utan att gå ur IEW. Det finns funktioner för att skapa, ta bort, kopiera hela eller delar av en encyclopedi. Man kan ta BackUp-kopior av en encyclopedi och återställa dem på samma eller en annan PC. Man kan också "reparera" en encyclopedi, vilket innebär att inkonsistent information ändras eller tas bort . Encyclopedier kan blandas och kombineras fritt.

4.3.2 Åtkomstmöjligheter och konsistens mellan representationer

Specifikationer kan ändras via diagram eller andra typer av fönster. Eftersom det bara finns en definition av objekt finns det ingen risk för inkonsistens mellan definitionerna.

Ändringar i diagram sparas omedelbart i encyclopedien. Diagram kan ses som projektioner över encyclopedien ur olika synvinklar. Det innebär att den grafiska representationerna alltid reflekterar det aktuella innehållet i encyclopedien. Det finns alltså ingen risk att det man ser på bilden inte är detsamma som finns lagrat i encyclopedien, vilket är fallet hos en del CASE-verktyg.

Denna strikta koppling mellan encyclopedien och diagram ger intressanta effekter när det gäller parallellt arbete med flera diagram. En ändring i ett diagram kommer omedelbart att orsaka en motsvarande ändring i andra diagram via encyclopedien. Kopplingen mellan diagrammen är alltså dynamisk.

När man editerar i ett diagram och vill ta bort ett objekt finns två sätt att gå till väga. Man kan välja att ta bort objektet ur det aktuella sammanhanget men låta dess definition och existens i andra diagram bli opåverkad. Man kan också välja att ta bort objektet ur encyclopedien och därmed från alla diagram. Att ta bort ett objekt kan ge bieffekter som att även andra objekt kommer tas bort. Om man exempelvis tar bort ett datalager i ett dataflödesdiagram blir alla dataflöden till och från det meningslösa och tas automatiskt bort. Vid sådana situationer visar IEW ett meddelande om vilka andra objekt som kommer att påverkas om operationen genomförs.

4.3.3 Kontroll- och analysmöjligheter

IEW har en stor mängd möjliga rapporter och analyser. Två av rapporttyperna är tillgängliga från alla moduler och diagram medan resten är modulspecifika. Vi redogör för de gemensamma först och går sedan genom rapport- och analystyperna för varje modul.

De för alla modulerna gemensamma rapporttyperna är **encyclopedia summary report** och **object summary report**. Encyclopedia summary report är en lista som visar antalet object för varje objektyp i encyclopedien. Object summary report är en mera omfattande rapport. Den visar en förteckning av objekt och deras egenskaper och associationer. Man kan välja vilka objekttyper rapporten skall gälla för. Vissa möjligheter att ange vilka egenskaper och associationer som skall tas med finns. De egenskaperna man kan ange är objektens definition, kommentar, skapare och tidpunkt för skapandet. Utöver detta kan man bara ange om resten av egenskaperna och associationerna tas med eller inte. Man kan alltså inte exakt specificera vilka andra egenskaper och associationer rapporten skall gälla för. Objekt summary report ger uttömmande information om objekten med begränsade möjligheter att skala bort det som man inte vill se. Det är dock just dessa begränsningar som minskar rapporten användbarhet. Eftersom man inte kan exakt specificera vad man vill se, blir rapporterna ofta fyllda med oönskad information som gör det svårt att hitta det man letar efter. Man vill ofta även ange vilket värde en egenskap skall ha eller vilket objekt en association skall vara bunden med för att tas med i rapporten. Om man t.ex. bara vill se vilka processer som bearbetar personuppgifter, får man i object summary report med alla processer som hanterar diverse data. För

komplexa modeller blir det då svårt att snabbt hitta denna information i rapporten. I kommande versioner av IEW skall detta problem dock delvis avhjälpas med en genomtänkt uppläggning av informationen i object summary reports.

Planning Workstation
I denna modul finns följande rapporter och analyser:

affinity analysis
exception analysis
hierarchy report
project action analysis

Affinity analysis grupperar samman objekt av en typ enligt deras "släktskap" . Denna gruppering baserar på hur många associationer av en typ objekten har gemensamt jämfört mot hur många de skulle kunna ha. Användaren kan själv ange kriterier hur stark "släktskapen" skall vara för att bilda en grupp. Denna typ av analyser kan användas för att se samband mellan objekten i verksamheten och för att skapa underlag för projekt.

Exception analysis går genom objektens egenskaper och associationer för att avgöra om valda egenskaper till objekten har tilldelats värden och om objekten har rätt antal associationer till andra objekt. Användaren anger själv vilka egenskaper och associationer som skall granskas. Rapporten består av en förteckning över objektinstanserna som inte uppfyller de givna kriterierna samt hur de avviker från dessa. Denna rapport kan användas för att analysera fullständighet i olika modeller.

Hierarchy report visar hur objekten länkas samman via deras associationer. Användaren definierar vilka associationer kedjan skall bestå av. Syftet med denna rapport är främst att visa kopplingar mellan olika typer av objekt. Om man t.ex. vill se aktiviteter som finns på olika platser i verksamheten och vilka avdelningar som svarar för dessa aktiviteter, kan man göra en följande rapportstruktur:

1. Function is responsibility of Org.unit
2. Organizational Unit is based at Location

som representerar kedjan:
Aktivitet är ansvarsområde för Enhet placerad på Plats

Rapporten kommer att bestå av alla förekomster av denna kedja i verksamhetsbeskrivningen (vi påminner att specificering av associationer mellan objekt görs med associationsmatriser som beskrevs under pkt 4.2.2)

Project action analysis används för att knyta projekt till data och processer i sina modeller. Denna analys används oftast efter det man skapat projekt med hjälp av verktyget för affinity analysis. Rapporten består av en förteckning över projekt samt vilka data- och processobjekt de innehåller.

Analysis Workstation

Här har man följande rapport- och analysmöjligheter till sitt förfogande:

flow expression report
connectivity analysis
data conservation analysis
data flow course analysis
view content analysis
subject area content analysis
parent analysis

Flow expression report är en förteckning över dataflöden. Dataflöden kan visas med källa, destination, innehåll och i vilka aktiviteter de förekommer. Man har friheten att välja en godtycklig kombination av vilka av dessa man vill se.

Connectivity analysis ger möjligheten att kontrollera att varje dataflöde är korrekt kopplad till en källa och en destination. På detta sätt kan en del av konsistensen i dataflödesdiagram analyseras. Man kan välja vilka dataflöden som skall analyseras. Rapporten består av dataflöden med den kompletta vägen från källan till destinationen genom alla nivåerna i diagrammet.

Data conservation analysis används i dataflödesdiagram för att upptäcka ifall data läcker in i eller tappas ur systemet. I praktiken betyder detta att försäkra sig om att:

data som kommer in i ett datalager är samma som finns där och som går ut från det

data som kommer in i en knutpunkt är samma som går ut från den

data som kommer in i eller lämnar en extern agent är en del av dess datamodell

en process' datamodel är den samma som summan av datamodellerna i inflöden, utflöden och flöden mellan dess delprocesser

Rapporten är en förteckning av processer, datalager och knutpunkter och dataflöden som är kopplade till dem samt om data i dessa är korrekt lagrade.

Data flow course analysis beskriver vägen för ett dataflöde genom nivåhierarkin i ett dataflödesdiagram. Detaljeringsgraden av rapporten kan väljas av användaren. Man kan även följa upp hur innehållet i dataflödet ändras under färden.

Parent analysis kan utföras för alla nedbrytbara objekt (aktiviteter, ämnesområden, och organisationsenheter). Rapporten är en lista över objekten och deras "föräldrar" i dekompotionsdiagrammet.

Subject area content analysis skapar en lista över innehållet i ämnesområden. Detaljeringsgraden i innehållet kan specificeras till en godtycklig kombination av valda entiteter, attribut och relationer.

View content analysis skapar en förteckning av de vyer som innehåller givna entiteter, attribut eller relationer.

Design Worksstation

Här finns möjlighet till följande rapporter och analyser:

call analysis
DBD content analysis
PSB content analysis
screen map report
unattached segment report
COBOL data structures
DBD externals
PSB externals
action diagram text

Call analysis ger möjligheten att snabbt kontrollera om de formella parametrarna i structure charts stämmer överens med anropsparametrarna.

DBD content analysis skapar en lista över de fysiska DBDs som är källor till segment i logiska DBDs (se beskrivningen av Database Description Diagrammer).

PSB content analysis visar DBDs och de segment som är del av ett givet PSB samt hur dessa segment refereras.

Screen map report används för att analysera skärmbilder och för att generera screen maps. Resultatet är en screen map och/eller fellista om skärmdefinitionen inte var korrekt.

Unattached segment report skapar en förteckning över alla segment som inte tillhör någon databas.

COBOL structures kan genereras för generella datastrukturer och alla COBOL data-area's datastrukturer som skärmar, poster , moduler etc.

DBD externals skapar en rapport och en informationsmängd som kan exporteras till andra system- exempelvis till stordatormiljön. DBD external-informationen är den samma som i DBD textfönster (se beskrivningen av DBDs i pkt 4.2.2).

PSB externals, dito för PSBs.

Action diagram text skapar en textversion av action diagrams, inklusive alla grafiska objekt. Informationen i action diagram report kan överföras till andra system.

Det har säkerligen framgått att IEW har en rik flora av diverse rapporter och analyser. De viktigaste av analysfunktionerna är kanske de som har att göra med konsistensen och fullständigheten i specifikationerna. Men även de som används för att analysera verksamhet och system och samband mellan objekten i dessa kan säkert utnyttjas för diverse ändamål.

Vad gäller rapporter så är ju poängen med många av dem att de utgör en stor del av dokumentationen om verksamheten och systemet. Därför är det viktigt att de är tydliga och lättlästa. Alla IEWs rapporter uppfyller inte detta krav, utan kan i vissa fall vara svåra att hitta i. Orsaken till detta är att

man inte alltid kan sålla ut den information som man är ute efter på grund av bristande möjligheter att själv påverka rapportutformningen.

En del av rapporterna skapar bryggor till andra miljöer genom att framställa information som kan exporteras utanför systemet. Vi har tyvärr inte kunnat granska deras användbarhet och låter bli att kommentera dem.

4.3.4 Säkerhetsaspekter

Det är väl onödigt att påpeka att systemspecifikationerna är mycket kostsamma att återskapa efter förlust. Ändå finns det exempel på CASE-verktyg som inte ger nämnvärt stöd för säkerhetsåtgärder.

IEW har en del egenskaper som hjälper att skydda och säkra specifikationer mot förlust eller otillbörliga ändringar. Den viktigaste av dessa är förmodligen lösenord-funktionen. Varje användare kan ges ett personligt lösenord. Dessutom kan man för varje registrerad användare ange vilka rättigheter han/hon har i systemet. Med rättigheter menas att en användare får registrera nya användare, ändra symbolers utseende eller ändra sitt eget lösenord. Denna funktion ger möjligheten att kontollera vilka som har tillgång till specifikationerna. Man kan till exempel låta den systemansvarige vara den ende som får registrera nya användare och ändra lösenord. Om någon användare slutar (eller missköter sig) kan den systemansvarige avregistrera honom/henne. Likaså om man misstänker att obehöriga fått tillgång till lösenord, kan man låta alla användarna kvittera ut nya lösenord. Det som saknas är möjligheten att ange vilka användare som har behörighet till vilka encyclopdias. Detta innebär dock inte att man i IEW inte kan kontrollera vilka som modifierar specifikationerna. För varje objekt registreras automatiskt när och av vem det senast har ändrats.

En möjlighet att skydda sig mot förlust av data är att regelbundet göra säkerhetskopior av encyclopdias. IEW har ett menykommando för BackUp-tagning. Man kan också råka ut för plötsliga spänningsbortfall under ett arbetspass. Internminnet raderas och all information som inte fanns sparat är borta. Nu är det lyckligtvis så att IEW updaterar encyclopedien på hårddisken vid varje operation som ändrar objektdefinitioner. Det gör IEW förstås litet långsammare, men en avgjord fördel är att risken att förlora större informationsmängder drastiskt minskar.

4.4 Användarmiljön

4.4.1 Gränssnitt mot användaren och interaktivitet

IEW använder sig av GEM, som är en produkt för att skapa ett grafiskt gränssnitt mot MS-DOS-operativsystemet på persondatorer. För att använda IEW/WS behöver man inte ha GEM, eftersom ett litet runtime-system av GEM finns inbyggt i IEW. Den miljö som IEW/WS tack vare GEM erbjuder är enkel och lätt att hantera (påminner starkt om Macintosh- och MS-Windowsmiljöer). Arbetsmiljön är interaktiv och baseras på mus och fönster. Musen används för att rita och editera i diagram och göra kommandon från menyer som finns längst upp på skärmen. Eftersom man aldrig behöver (men kan) ge kommandon från tangentbordet behöver man inte heller lära sig några sådana. All inmatning av data sker genom att fylla i dialogrutor som IEW visar på skärmen när det behövs. Allt detta gör att man snabbt lär sig att arbeta med IEW.

Den här typen av interaktiva system gör att man alltid ser följden av sina handlingar omedelbart. Om en operation kan ha oanade konsekvenser varnar IEW och ger möjlighet att återkalla operationen. Likaså ser man alltid verktygets aktuella tillstånd på skärmen. Diagrammen i fönstren återger specifikationernas aktuella utseende och ger möjligheten editera dessa interaktivt.

Man kan ha flera fönster öppna samtidigt vilket innebär att man skåda och editera flera diagram parallellt.

4.4.3 Hjälpfunktioner

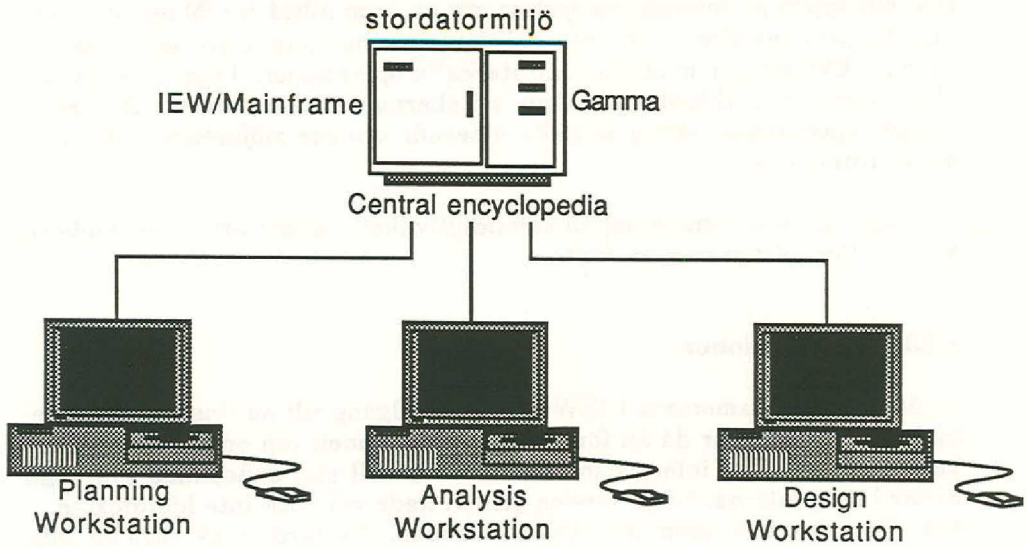
Under alla arbetsmoment i IEW har man tillgång till on-line hjälp via en hjälp-meny. Man får då en förteckning över ämnen och operationer man kan få hjälp i. Den information som finns är till stor densamma som man hittar i manualerna. I den version som vi hade var dock inte hjälptexten ännu inte inkriven inom alla ämnesområden. Det torde dock vara en tidsfråga innan hjälp finns att få om snart sagt allt.

Hjälpfunktionen är i viss mån "intelligent". Med det menar vi att när man begär hjälp, så får man ibland automatiskt den hjälp som man för tillfället behöver. Ett exempel på sådan hjälp är när verktyget automatiskt visar de operationer som man för tillfället kan utföra, eller vad användaren väntas göra härnäst. I IEW kan man finna denna typ av hjälp i dialogrutorna för inmatning av specifikationer.

4.4.4 Fleranvändarmiljön

IEW/WS är ett verktyg för enmansarbetsstationer. I varje arbetsstation sker arbetet med en lokal encyclopedi. Här finns alltså inga problem som är kopplade till fleranvändarmiljöer, där flera personer samtidigt kan komma åt samma data. Det är dock inte meningen att IEW-arbetsstationerna skall vara helt isolerade öar. Normalt går arbetet till så att man har en central encyclopedi i stordatormiljön till vilken arbetsstationerna är kopplade via nätverk.

I den centrala encyclopedien finns alla specifikationerna samlade och dessa kan uppdateras och hämtas av arbetsstationerna. Dessa operationer görs genom att kopiera hela eller delar av den centrala encyclopedien till arbetsstationerna och kombinera data från de lokala encyclopedierna med data i den centrala encyclopedien. Att se till att specifikationerna i den centrala encyclopedien är aktuella och att inget av misstag skrivs över vid kombinering är en uppgift för projektledningen och de systemansvariga.



Figur 4.22 En illustration om hur arbetet med IEW/WS och IEW/Mainframe kan organiseras. Arbetstationerna delar på data i den centrala encyclopedien.

4.5 Egenskaper och faciliteter hos IEW

4.5.1 Portabilitet

Specifikationer i IEW kan överföras till andra IEW-miljöer samt till Gamma, som är en separat produkt för kodgenering och simulering av skärmbilder och rapporter . Men det finns bryggor till andra miljöer ocksa. En del rapporter som IEW tillhandahåller är ju beskrivningar av specifikationer i textform. Genom att transformera dessa specifikationer till lämpligt format kan de utnyttjas även av andra produkter. Det finns redan nu ett stort antal databashanterare och 4GL-miljöer till vilka det finns överföringsmöjligheter.

4.5.2 Kommunikation

Kommunikation kan betraktas ur flera olika synvinklar. Vi nöjer oss med en kort översikt av kommunikationen mellan IEW arbetsstationer och den centrala miljön samt vilka databaser den färdiga applikationen kan kommunicera med.

För att kunna arbeta effektivt med IEW måste arbetsstationerna vara uppkopplade till en central dator via någon form av kommunikationsnät. IEW har inga speciella kommunikationsfunktioner för överföring av encyclopedier. För att föra över delar eller hela encyclopedier till den centrala miljön måste de först kopieras till en flat fil. Denna fil används som tillfälligt mellanlager som överförs via nätet för att senare laddas in i en annan encyclopedi.

De färdiga applikationerna måste oftast kunna kommunicera med någon databashanterare som sköter administrationen kring lagring och åtkomst av data. De program som IEW - eller rättare sagt Gamma - genererar kan

för tillfället göra databasanrop till ISAM-databaser (flata filer) , DL1 (hierarkiska databaser) eller till DB2 (relationsdatabas).

4.5.3 Transformation av specifikationer till databasstruktur

Under analysfasen brukar man skapa en normaliserad logisk datamodell för systemet. I IEW görs detta med Entity Diagrammer. Med det kan man skapa en fullt normaliserad datamodell. Även datatyperna för attributen kan specificeras via Data Type Window. I designfasen övergår man från den logiska datamodellen till en databasbeskrivning. Om man tänker använda sig av en relationsdatabas i den färdiga applikationen, kan man i IEW transformera datamodellen i Analysis Workstation till ett relationsdatabasschema i Design Workstation. Man kan ange om relationerna (relationships) i schemat skall implementeras med 'foreign keys' eller associativa relationer (relations).

Om man vill använda sig av flata filer eller hierarkiska databaser kan ingen automatisk konvertering göras för närvarande utan verktygen för dessa databaser i Design Workstsation måste användas.

4.5.4 Kodgenerering

IEW har ingen egen kodgenerering, utan för det måste Gamma användas. Kodgenering utifrån IEW's action diagrams sker i IBM-stordatormiljön under operativsystemet MVS. Kod kan genereras för tre olika COBOL-dialekter: ANSI-68, ANSI-74 och COBOL 2. Från koden kan anrop till databashanterare göras, för tillfället ISAM, DL1, IDMS och DB2. Den kod som genereras är fullständig, om action diagrammen varit fullständiga och korrekt specificerade.

Kopplingen mellan action diagrams och exekverbar kod är något lös. Action diagrams har för tillfället inget stöd för notationssätt för olika språk. För att kunna utnyttja kodgenereringsmöjligheterna måste ett slags COBOL notation användas. Dessvärre finns för närvarande ingen möjlighet att göra syntaxkontroll för denna notation direkt i IEW, utan det måste göras i Gamma.

För att transformera databasscheman till kod för databashanterare finns det olika sätt att tillgå. Kod för hierarkiska databaser fås genom DBD-text och PSB-text -rapporter. Dessa beskriver i detalj innehållet i databaserna i text-format. Eventuellt kan denna kod behöva översättas till den kod som den aktuella databashanteraren använder. För relationsdatabaser kan SQL-kod genereras. För hantering av flata filer kan COBOL datastrukturer genereras och inlemmas i program.

4.5.5 Prototypframställning

IEW har inga egna simulerings eller prototypingfunktioner som simulering av sessioner med inmatningsformulär och rapporter eller interpretation av specifikationer. Dock öppnar sig vissa prototypmöjligheter om IEW kompletteras med Gamma i den centrala miljön. Det man då kan göra är simulering av skärmdialoger och rapporter. Action diagrams från IEW transformeras och exporteras till Gamma där de interpreteras. Det finns även tillbakakoppling från Gamma till IEW: om man ändrar koden i

Gamma kan dessa ändringar överföras tillbaka till IEWs action diagrams och modulbeskrivningar.

Man kan också se möjligheter för mera påtaglig prototypframställning med hjälp av Gamma. Genom att med IEW's action diagrams först konstruera mindre och enklare programbeskrivningar och sedan generera kod utifrån dessa i Gamma kan man skapa små testsystem. Dessa testsystem kan sedan valideras och successivt kompletteras. På så vis får man fungerande och korrekta versioner att utgå ifrån då stora och komplexa system konstrueras. Denna process underlättas också av tillbakakopplingen från Gamma.

4.5.6 Återanvändning av specifikationer

Vi har redan gått genom hur encyclopedier kan kombineras (under pkt 4.3.1 och delvis under 4.4.4). Genom denna teknik har man också möjligheten att på ett enkelt sätt plocka ut delar av specifikationer ur en encyclopedi och klistra in dem i en annan. För att undersöka konsistensen mellan de nya och befintliga specifikationerna kan analysfunktionerna användas. Man kan också utnyttja "repair"-funktionen för encyclopedier som "lagar" inkonsistenta data och tar bort det som inte kan "lagas". Det är alltså relativt lätt att återanvända gamla specifikationer.

Det finns ingen möjlighet att söka och kopiera de specifikationer som uppfyller givna kriterier. I viss mån kan dock object summary report användas för att hitta objekt med givna egenskaper.

4.5.7 Versionshantering

När man pratar om versionshantering i olika sammanhang, kan det uppstå litet förvirring. Man kan koncentrera sig på hur specifikationer skapade med olika versioner (releases) av verktyget hanteras. Man kan också mena hantering av olika versioner av systemspecifikationer. Den tredje tolkningen går ut på att titta på hur specifikationer som har skapats av verktyg anpassade för olika metoder hanteras vid överföring.
Den sista tolkningen är inte aktuellt för IEW eftersom man inte kan skräddarsy någon individuell version av IEW. Anpassning av IEW sker inom ramen för användningen av de befintliga beskrivningsteknikerna i IEW. Vi tar en närmare titt på de två första tolkningarna.

Encyclopedier skapade med olika versioner (releases) av IEW är nöd vändigtvis inte kompatibla. Ibland måste konvertering av encyclopedierföretas. Tumregeln är då följande: Olika versioner betecknas med X.Y där X är versionsnumret för själva encyclopedien. För att två IEW-moduler skall kunna fungera med varandra måste båda ha samma X. Exempelvis fungerar Analysis Workstation 3.5 inte tillsammans med Design Workstation 4.0. Däremot fungerar Design Workstation 4.0 med t.ex. Analysis Workstation 4.1. Om man alltså har olika versionsnummer för encyclopedier så måste konvertering företas vid övergång till nyare versioner.

IEW har ingen inbyggd versionskontroll för systemspecifikationer. Det betyder dock inte att man inte kan arbeta med olika versioner av specifikationer på ett smidigt sätt. Funktionerna för manipulering av encyclopedier gör att man enkelt kan kopiera och kombinera encyclopedier. Det vanligaste fallet då olika versioner av specifikationer behövs är då man vill göra "vad-om"-analyser. Detta tacklas i IEW genom att skapa flera exemplar av en encyclopdia. Specifikationerna i dessa kopior kan sedan ändras och

konsekvenserna av ändringar analyseras. Detta är i och för sig inte något unikt, men om man betänker möjligheten att låta flera personer göra analyser av olika aspekter samtidigt blir det genast mera intressant. Låt oss anta att man vill göra en "vad-om"-analys för ett antal funktioner i ett system. För dess skull vill man dock inte stoppa arbetet med de delar av systemet som inte påverkas av dessa analyser. Det man då efterlyser är möjligheten att låta utvecklingen av oberörda delar fortsätta och efteråt lägga till de delar som man analyserat och utvecklat separat. Detta kan uppnås i IEW genom att först kopiera den gemensamma utgångsmodellen. Några av dessa genomgår analyser och påföljande modifierigar. Andra kopior kanske utvecklas vidare i de delar som inte berörs av analysresultaten. När man till sist fått klart för sig hur systemet skall se ut med tanke på analysresultaten kan man ta dessa separata encyclopedier och kombinera ihop dem på ett lämpligt sätt: analysresultaten klipps in i de vidareutvecklade versionerna som i sin tur kan läggas ihop för att få den slutliga versionen.

Man kan också skapa kopior av encyclopedier för att ha tillgång till specifikationerna av ett system under olika utvecklingsfaser. Om man sedan upptäcker att man är på fel spår, så kan man plocka tillbaka de gamla och korrekta specifikationerna och göra ett nytt försök.

4.5.8 Projektplaneringsstöd

Någon egentlig projektplaneringsfunktion finns inte i IEW. Däremot finns viss stöd för att analysera områden för verksamhet för att skapa projekt samt analysera innehållet för dessa projekt (se affinity analysis och project action analysis för Planning Workstation under pkt 4.3.4).

Viss hjälp vid projektplanering och -uppföljning får man med de olika rapporterna och diagrammen i IEW. Om allt finns lagrat i en central encyclopedi så kan man snabbt skaffa sig överblick över projektets status och följa upp hur väl projektetapperna uppnåtts.

Lösenordsfunktionen kan också användas av projektledaren för att kontrollera vad projektmedlemmarna har rätt att göra.

4.5.9 Flexibilitet

När man säger att ett verktyg är flexibelt brukar man mena att det kan anpassas och utformas efter olika behov. IEW kan inte skräddarsys enligt önskemål från användare. Detta behöver inte vara någon nackdel, eftersom denna typ av anpassning lätt leder till kompatibilitetsproblem mellan verktygsversionerna. Man kan ändå säga att IEW är flexibelt. Det finns inga allvarliga begränsningar vad gäller valet av systemutvecklingsmetoder och arbetssätt. Var och en kan välja de lämpliga kombinationerna av de beskrivningstekniker som IEW tillhandahåller.

Vad gäller utseendet hos de grafiska symbolerna och färgerna i diagrammen har användarna ett antal alternativ att välja mellan. På så vis kan man ofta behålla de notationssätt man är van vid sedan tidigare. Varje användare kan definiera den grafiska miljö han/hon föredrar. Några nya symboler kan inte skapas utan man får hålla sig till de tillgängliga alternativen.

Även möjligheten att använda den typen av IEW-moduler som man anser sig ha behov av främjar flexibiliteteten. Det är inte alls säkert att ett företag

vill eller har behov av att använda Planning Workstation. Det kanske räcker med Analysis och Design Workstations. Man kan också tänka sig att företaget vill använda Planning Workstation separat för att analysera och modellera sin verksamhet och ha den som stöd för omstrukturering av verksamhetsområden och för långsiktig planering. Med detta vill vi illustrera att IEW's moduler inte är beroende av varandra utan kan användas ensam eller tillsammans eller till och med för olika syften i olika sammanhang. Oberoende hur man använder IEW har man dock alltid tillgång till dess dokumentation och dess analysmöjligheter.

4.6 Hur IEW kan påverka systemutvecklingsprocessen

4.6.1 Vinster i tid och kvalitet

Vilken effekt utnyttjande av IEW ger på kvaliteten hos de utvecklade informationssystemen och produktiviteten hos systemavdelningarna, kan omöjligen kvantifieras. Att positiva effekter finns är klart, men hur stora dessa är beror mest på hur systemutvecklingen i företaget bedrivits tidigare. Vi kan dock se ett antal tänkbara effekter. En del av dessa är tydliga och omedelbara medan andra är mera indirekta.

Korrekt användning av IEW kan ge ett antal positiva effekter på de slutliga systemens kvalitet. Det är lättare att göra analys av alternativa systemlösningar tack vare möjligheten att kombinera encyclopedier på ett enkelt sätt. Även under designfasen kan man analysera konsistensen och fullständigheten i specifikationerna för att upptäcka brister och felaktigheter. Datoriserade verktyg för diagramritning och den automatiska dokumentationen ger systemutvecklarna mera tid över till kommunikation med slutanvändare. Systemen kan således få höjd kvalité i form av effektivare och välanalyserade lösningar och en bättre slutanvändaranpassning. Men många av fördelarna är förstås inte specifika för IEW, utan gäller CASE-verktyg med goda analys- och designfunktioner i allmänhet. I likhet med många andra CASE-verktyg ger även IEW en bra och automatisk dokumentation, som antagligen har bättre kvalitet än den man haft tidigare.

Det är sannolikt att effektiv användning av IEW enligt Information Engineeringkonceptet påverkar de slutliga systemens kvalité även på ett indirekt sätt. Planering och analys av verksamheten kan göras på ett sätt där även informationssystemen tas med som del av verksamheten . Det kan leda till att informationssystemen bättre anpassas till verksamhetens mål och aktiviteter.

När det gäller tidsvinster, kan man genast fastslå några tydliga effekter. Den mest markanta förbättringen finns att hämta i realiseringsfasen. IEW tillsammans med Gamma kan nästan helt eliminera det manuella programmeringsarbetet. En del av det förflyttas naturligtvis till designfasen i form av konstruktion av action diagrams, men den totala kodningstiden blir ändå mycket mindre än med manuella metoder. Om man dessutom tar hänsyn till de simulerings - och prototypingsmöjligheterna som användning av IEW tillsammans med Gamma ger, blir effekten ännu större. Den automatiska dokumentationen ger säkerligen tydliga tidsbesparingar, men storleken av dessa beror på hur stor vikt man tidigare lagt på en noggrann dokumentation.

Vi kan dock inte ge några uppskattningar över hur stora de totala tidseffekterna blir. Tidsbesparingen vid realisering kan delvis komma att

uppätas av ökade insatser för analys och design. Om även planering räknas med blir tidseffekterna ännu svårare att uppskatta. Men här är det vanskligt att stirra på tidsvinsterna. Även om den totala tidsbesparingen inte blir markant, så måste den ändå relateras till faktorer som ökat kvalitet, bättre dokumentation och bättre verksamhetsanpassning hos systemen.

4.6.2 Effekterna på informationssystemets livscykel

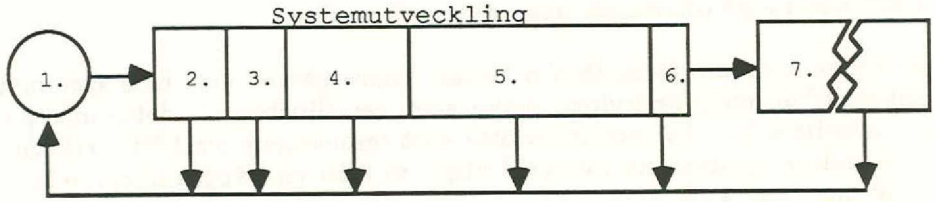
Vi har redan konstatetat att IEW praktiskt taget spänner över hela systemutvecklingskedjan med förstudien undantaget, och illustrerade detta med en figur i avsnitt 4.2.1. För att underlätta vårt resonemang om IEW's relation till informationssystemens livscykel utgår vi från en något annorlunda illustration, figur 4.23.

I den traditionella systemutvecklingen går en stor del av tiden att programmera och testa systemen. Analysfasen har ofta varit bristfällig på grund av svårigheterna att hantera modeller av komplexa system. Dokumenationen i design- och realiseringsfaserna har sällan gjorts ordentligt. Den sista fasen - utveckling och underhåll - har därmed blivit svårare. När sedan systemet blivit gammalmodigt och nya behov har uppstått, börjar nästa varv, såvida man inte helt skrotar systemet. Nu kommer den intressanta upptäckten: Eftersom man inte har grundliga analysresultat och korrekta modeller över det gamla systemet och dokumentationen är bristfällig, måste stora del av analys och designarbetet göras om för att man skall få en uppfattning om exakt vad det är som behöver ändras. Arbetet är nu minst lika svårt som när det gamla systemet utvecklades, varför risken är stor att man inte gör grundligt arbete nu heller. Den onda cirkeln är sluten.

Nu när vi har problematiken klar för oss kan vi börja resonera om hur IEW kan ändra på denna situation. En av grundideérna med CASE-verktyg är att de skall ge stöd för analys- och designfaserna i systemutvecklingsprocessen. Detta gäller i allra högsta grad för IEW. Systemutvecklarna uppmuntras - och även tvingas - till att göra djupare analyser och designa systemen på strukturerat och formellt sätt. En stor del av arbetet läggs i dessa faser. När modellen över systemet är konsistent och systemstrukturen och procedurbeskrivningarna fullständiga kan exekverbar kod genereras och testas. Efter ett antal justeringar kan systemet tas i drift. Om vi nu tittar på tidsfördelningen mella faserna så ser vi att det är analys- och designfaserna som är dominerande. Programmeringsarbetet har krympt till en bråkdel om systemets design gjorts tillräckligt noggrannt. En rik och korrekt dokumentation har skapats automatiskt under arbetet. Den korrekta dokumentationen och grafiska modellerna över systemet torde göra underhållet och vidareutvecklingen lättare och mindre tidskrävande.

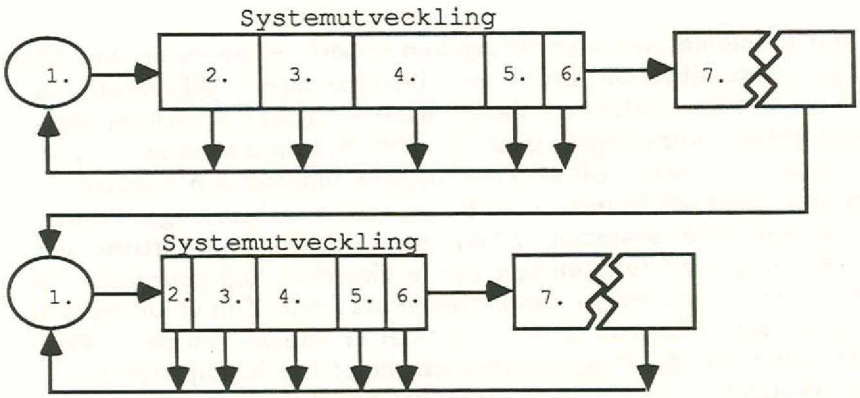
När man sedan beslutar sig för ett nytt system, och börjar utveckla det kommer den stora skillnaden mot förr. Nu har man exakta modeller och ordentlig dokumentation i form av IEW's encyclopedi. Det gamla systemet är färdiganalyserat och redo att vidareutvecklas. Om man skrotar det gamla systemet kan ändå stora delar av det återanvändas genom att plocka ut dessa ur encyclopedien. I detta andra varv av systemets livscykel kommer analys- och design fasen kräva mindre tid och besvär än i det första varvet. Nu har alltså dessa faser krympt i förhållande till den totala utvecklingstiden, som även den har reducerats. Figur 4.24 illustrerar detta nya förhållande.

1. Nya problem och/eller möjligheter
2. Analys och dokumentation av nuvarande system
3. Analys av kraven för nya system
4. Design av nya system
5. Realisering,tester och validering av nya system
6. Införande
7. Underhåll och vidareutveckling



Figur 4.23 Informationsystemens livscykel. Under varje moment kan nya problem eller möjligheter uppstå, vilket ger hela processen en iterativ karaktär. Längden på boxarna är proportionell med den tid fasen upptar av den totala utvecklingstiden.

1. Nya problem och/eller möjligheter
2. Analys och dokumentation av nuvarande system
3. Analys av kraven för nya system
4. Design av nya system
5. Realisering,tester och validering av nya system
6. Införande
7. Underhåll och vidareutveckling



Figur 4.24 Informationsystems livscykel med effektiv användning av IEW tillsammans med Gamma. Längden på boxarna är proportionell med den tid fasen upptar av den totala utvecklingstiden i varje cykel.

För att resonemanget skall vara korrekt krävs att IEW används på ett effektivt sätt och att all vidareutveckling och underhåll sker med hjälp av IEW, så att systemspecifikationerna alltid är aktuella och korrekta. Detta resonemang gäller även andra CASE-verktyg med motsvarande egenskaper.

4.6.3 Slutanvändarnas roll i utvecklingsarbetet

En av ideérna bakom Information Engineering är att involvera slutanvändarna i design-processen. De skall kunna skapa sina egna skärmdialoger

och rapportlayouter. De skall även vara med i utvecklingen av procedurer med lättanvända grafiska hjälpmedel. IEW's enkla grafiska gränssnitt gör att slutanvändare utan större problem kan delta i design-processen. Detta förutsätter dock att de känner till principerna bakom de grafiska modelleringsteknikerna. En del av diagrammen kan slutanvändarna lätt förstå och arbeta med medan andra kräver så stora teoretiska kunskaper att det inte är lämpligt att låta alla arbeta med dem. Exempelvis kan action diagrams tyckas vara enkla hjälpmedel för att definiera programstruktur och -logik, men de kräver ändå en hel del kunskaper om programmering. Med relativt begränsade utbildningsinsatser och stöd av mera erfarna IEW-användare kan slutanvändarna nog själva skapa sina skärmdialoger och fungera som aktiva deltagare vid design av procedurer.

Även under analysfasen kan slutanvändarna tänkas vara med och validera data- och processmodellerna. Det är ju ändå de som ofta har kunskaperna om vilka data som finns och hur de hanteras i verksamheten. Detta kräver förstås viss utbildning om data- och processmodeller.

Men även om dessa grafiska hjälpmedel för modellering och design borde möjliggöra större slutanvändarinsatser, skall man inte överskatta detta. Enligt vad vi har erfarit har slutanvändarna inte alls så lätt att ta till sig systembeskrivningar. Inte ens mycket enkla grafiska verktyg behöver betyda att de kan ta över en del aktiviteter. Antagligen beror detta på att de som förespråkar dessa fantastiska möjligheter för slutanvändare själva har utbildning inom området och lång erfarenhet bakom sig. Man kan inte förvänta sig att personer som plötsligt ställs inför nya begrepp och nya tekniker med kort introduktionstid skall kunna framställa och analysera komplicerade strukturer och samband med hjälp av dessa begrepp och tekniker. Att lära sig den höga abstraktionsnivån som det är frågan om i dessa sammanhang tar sin tid.

4.6.4 Utbildningsbehovet

Diskussionen om slutanvändarnas roll leder oss till utbildningsbehovet. Vilka skall använda verktyget och vilka utbildningsinsatser krävs?

Som vanligt kan man inte ge något entydigt svar. När det gäller personer som skall använda verktyget måste man ställa frågan: vilka har de nödvändiga kunskaperna som behövs? Alla systemutvecklare och programmerare kan givetvis antas ha tillräckliga kunskaper. Vad gäller slutanvändare så beror det mycket på vilken yrkeskategori de tillhör. Det är förvisso skillnad i förutsättningarna hos banktjänstemän och ingenjörer som är vana att arbeta med CAD-verktyg.

IEW är ganska omfattande och det tar sin tid att helt och hållet behärska verktyget och känna till dess möjligheter och begränsningar. Om man har klara metoder att tillämpa vid systemutveckling kan systemutvecklare och programmerare som är vana att arbeta med grafiska beskrivningstekniker komma igång med IEW snabbt utan större utbildningsinsatser. Det är dock lämpligt att ge vissa nyckelpersoner en djupare utbildning i verktyget för att sedan låta dem assistera andra när det behövs. Risken att bli beroende av dessa nyckelpersoners kunskap är inte särskilt stor eftersom det inte är att använda IEW som är det svåra. Det svåraste och mest viktiga är att skaffa sig metoder och arbetssätt som gör att man kan utnyttja IEW effektivt. Har man väl fungerande metoder och stabiliserad arbetsordning så kan förlust av dessa nyckelpersoner snabbt kompenseras.

Om slutanvändare aktiveras i utvecklingsarbetet så tror vi att de snabbt kan bli produktiva under assistans av erfarna IEW-användare. De utbildningsinsatser som då främst behövs är inte användning av IEW utan snarare av teoretisk karaktär: metoder och beskrivningstekniker.

Vi har framhållit att lära sig använda IEW inte kräver någon kostsam utbildning. Men detta är inte hela sanningen. För att kunna utnyttja IEW effektivt måste man kunna de viktigaste grafiska beskrivningsteknikerna. Dessa är entitetsdiagram, dekompositionsdiagram och actiondiagrams. Entitetsdiagram ger grunden för upprättande av databaser. Dekompositionsdiagram och action diagram behövs för att kunna beskriva programlogik och generera kod. För kodgenerering är korrekta action diagram nödvändiga. Det är således viktigt att de som designar system har ordentliga kunskaper i hur man konstruerar action diagrams och dessutom har kunskaper i COBOL. Viss utbildning om dessa diagramtekniker kan komma att behövas.

Även om IEW är ganska omfattande så är det inte sannolikt att man glömmer viktiga saker och behöver ny utbildning. Det kan givetvis hända för slutanvändare som inte använder verkyget ofta, men risken är närmast obefintlig för systemutvecklare och programmerare som använder verktyget regelbundet.

Den utbildning man kan få till IEW verkar ganska heltäckande. Förutom full utbildning i IEW's arbetsstationer finns även mera metodorienterad utbildning. Den metodorienterade utbildningen kommer enligt uppgift från leverantören att utvidgas i den närmaste framtiden.

4.7 Hård- och mjukvarukrav samt dokumentation

För att kunna använda IEW/WS måste man ha IBM PC-AT eller kompatibel . Även IBM Personal system/2 -datorerna går att använda, dock ej modell 30. Operativsystemet är MS-DOS /PC-DOS version 3.1 eller senare. För PS/2 krävs dock version 3.3 . För att få full nytta av IEW behöver vissa utrustningskrav uppfyllas. Man behöver ha tillräckligt med minne, grafisk skärm, mus och skrivarutrustning. Minst 2.6 megabytes internminne behövs för att alla arbetsstationerna skall fungera ordentligt. Vad gäller grafiken så krävs minst CGA 640x200 färggrafik. Andra grafikkort som stöds är EGA, VGA och XGA för IBM 3270 AT. De skrivare som går att använda är för tillfället IBM och Epson radskrivare, IBM Color Jetprinter, Hewlett-Packard Laserjet och Diablo Model C150 Color Ink.

Dokumentationen till IEW/WS är ganska omfattande. Man får noggranna installationsbeskrivningar och en handbok för varje arbetsstation. Handböckerna är välskrivna och har bra innehållsförteckning vilket gör att de är lätta att hitta i. Varje diagramtyp och fönster har fått ett eget kapitel där man noggrannt beskriver syftet med diagrammet och hur man arbetar med det. I slutet av varje handbok finns en genomgång av de rapporter och analyser som finns tillgängliga i arbetsstationen. De få oklarheter som vi råkade ut för kunde snabbt klaras ut genom telefonkontakt med leverantören.

4.8 Avslutande översikt

4.8.1 Sammanfattning av IEW

IEW är ett hjälpmedel för systemutveckling med ett stort antal grafiska verktyg för systembeskrivning. IEW är inte bundet till någon specifik systemutvecklingsmetod utan ger stöd för flera olika metoder och metodkedjor. Dock är man begränsad till de beskrivningstekniker som ingår i Information Engineering. Alla definitioner och beskrivningar lagras i en datakatalog kallad encyclopedi, och finns därefter tillgängliga för analys,modifiering eller detaljbeskrivning. De olika diagrammen är integrerade och alltid konsistenta med varandra, eftersom de bara ger en grafisk illustration av det som finns i encyclopedien. Diagrammen kan betraktas som olika projektioner över encyclopediens innehåll.

IEW erbjuder ett stort antal olika rapporter och analysmöjligheter. Analyserna hjälper till att kontrollera konsistensen och fullständigheten i specifikationerna. Rapporterna ger möjligheten att få detaljerad systemdokumentation i textform.

IEW's encyclopedi är en omfattande semantisk databas över informationen om systemet och verksamheten. Funktionerna för godtycklig kombinering av encyclopedier tillsammans med analysmöjligheterna ger god stöd för uppdelning av arbetet i delprojekt.

Då IEW används tillsammans med produkten Gamma för simulering och generering av kod utifrån specifikationerna från Design Workstation, kan det betecknas som ett integrerat CASE-verktyg (I-CASE), som täcker hela systemutvecklingskedjan.

4.8.2 IEW's framtida utveckling

Det finns en del kompletteringar och förbättringar som kommer att genomföras i nya versioner av IEW. Bland annat kommer den tidigare nämnda produkten Gamma att helt integreras med IEW. I samband med detta kommer möjligheten för syntaxkontroller för COBOL att kunna finnas redan i action diagrams. Rapportlayouter kommer att kunna utformas på liknande sätt som man nu definierar skärmdialoger. Likaså kommer Gammas möjligheter för simulering av skärmdialoger och rapporter samt kodgenering kunna utföras i den miljö som IEW/WS och IEW/Mainframe erbjuder.

Enligt leverantören kommer även analys- och rapporteringsfunktionerna att utökas. Bland annat nämns utökat undantagsrapportering, som ger bättre möjligheter att hitta avvikelser från vissa logiska regler.

Förmodligen kommer även versioner för operativsystemet OS/2 släppas ut inom överskådlig framtid. Med OS/2 får man också nya möjligheter vad gäller användning av IEW i nätverk. Hur IEW kommer att utnyttja de nya möjligheterna som OS/2 och kraftfullare persondatorer ger, vill leverantören ännu inte kommentera..

4.8.1 Personliga intryck och åsikter

Vi upplevde arbetet med IEW som relativt enkelt och smidigt. Dock orsakade IEW's omfattning och det stora antalet olika beskrivningstekniker tillsammans med "bristen" på en klart definierad metod en viss förvirring i början. Den grafiska miljön känns tilltalande och ger goda möjligheter till överblick och detaljstudier över systembeskrivningarna.

Trots det stora antalet olika rapporter var det dock just rapportfunktionerna som vi upplevde som mest stela. Ofta önskar man en utskrift av objekt med just de egenskaper och associationer man är intresserad av. Valfriheten att själv i detalj specificera sina rapporter saknas och ledde till att rapporterna ofta blev fyllda med överflödig information som man inte var intresserad av. Det hade ofta negativa konsekvenser för läsbarheten hos rapporterna. Det skall dock sägas att rapporterna ger all den information som man normalt behöver.

Om vi får ge tips om hur IEW kan förbättras så finns ett par saker vi vill ta upp. För det första kunde användaren ges en möjlighet att själv i detalj specificera sina rapporter. Detta tror vi skulle kunna leda till snabbare och smidigare felsökning och ge ännu bättre kontroll över korrektheten i specifikationerna. Knowledgeware planerar utökad undantagsrapportering men frågan är om det bara är avvikelser av konsistensregler man är intresserad av. De logiska felaktigheterna behöver inte leda till inkonsistens i beskrivningarna, utan enbart leda till en felaktig funktion hos det färdiga systemet. Att spåra dessa fel tillbaka till specifikationer kan avgjort underlättas om man exakt kan ange vilken information man är ute efter.

Den andra förbättring vi kan tänka oss är att komplettera IEW med någon algoritm för analys och gruppering av processer i dataflödesdiagram till 'central transforms' eller 'transaction centers'. Vi är medvetna att det inte är någon trivial uppgift, men viss stöd stöd den här typen av transformationer borde man kunna åstadkomma. Genom dessa transformationer skulle man avsevärt underlätta och effektivisera konstruktionen av strukturkartor utifrån dataflödesdiagram.

I övrigt tycker vi att IEW/WS är ett avancerat verktyg, som ger stora möjligheter för bättre analys och effektivare konstruktion av informationssystem.

5 SLUTSATSER;

5.1 Deft och IEW - skillnader och likheter

Till att börja med så kan vi konstatera att IEW och Deft är ganska olika verktyg. IEW går på PC- och IBM-stordatorer, och har ett stort antal olika beskrivningstekniker med diverse rapport- och analysfunktioner. IEW kan lätt uppfattas som ett komplext system. Deft, som går på Macintosh-datorerna, har bara ett fåtal verktyg för systembeskrivning och är lätt att lära sig. Några speciella analysfunktioner finns inte, men Design Compiler utför vissa konsistens och fullständighetskontroller.

Den största skillnaden mellan Deft och IEW ligger förmodligen i innehållet och hanteringen av datakataloger. IEW's datakatalog är en stor databas som hanteras av ett inbyggt expertsystem. Detta möjliggör diverse analyser och effektiv hantering av den information som finns lagrat i datakatalogen. Deft's datakatalog är en Macintosh resursfil och innehåller bara en del av objekten och deras uppbyggnad. Följaktligen har inte Deft några avancerade rapport och analysfunktioner. Även kopplingen mellan beskrivningsteknikerna i Deft blir lidande av dett faktum.

Deft är dock smidigare att arbeta med. I och med att inga kontroller mot datakatalogen görs under ritarbetet, går det också snabbt att skapa enklare systembeskrivningar. IEW's rigorösa kontroller vid varje operation gör att det ibland går litet trögt, men å andra sidan har IEW större trygghet med tanke på konsistens och korrekthet.

När det gäller att konstruera relationsdataser för Ingres, är Deft enligt vårt tycke ett bra hjälpmedel. Även om motsvarande möjligheter finns i IEW, så är Defts ERD-editor mycket smidigare att arbeta med.

Varken Deft eller IEW har några egna kodgenereringsfaciliteter, men genom sin nära koppling till Gamma är IEW ett mera komplett CASE-verktyg. Nu är det kanske så att Deft inte heller har den ambitionen, utan siktar mer på att vara ett komplement till 4GL-verktyg.

Sammanfattningsvis kan man konstatera att IEW är ett omfattande och avancerat verktyg för IBM-miljöer. IEW är dock dyrt och kräver större insatser för att kunna användas effektivt. Deft i sin tur är ett enklare enklare och billigare verktyg för snabb modellering och med nära koppling till Vax-miljön.

5.2 Bakåtblick och lärdomar

När vi började vårt arbete, tyckte vi att vi att vi visste ganska bra vad vi skulle göra och hur vi skulle göra det. Men som det ofta gör visade det sig att uppgiften inte alls var så väl avgränsad och väldefinierad som vi först antagit. Efter att ha satt oss in i situationen kom vi på att det fanns ett antal obesvarade frågor hängande i luften. Vi skulle granska CASE-verktyg. Men exakt vad är ett CASE-verktyg? `Hur lyder definitionen? Vilka krav måste ett systemutvecklingshjälpmedel uppfylla för att kunna kallas ett CASE-verktyg? Vilka egenskaper bör man titta på när man studerar ett CASE-verktyg? Hur bör en seriös studie av sådana verktyg utformas? Vilka krav måste vårt praktikfall uppfylla? Hur skulle vi utforma vår slutliga

rapport? Hur skulle vi överhudtaget lägga upp vårt arbete och disponera vår tid?

Vad CASE-begreppet stod för fick vi klart för oss under de inledande diskussionerna med våra konktakpersoner på SISU. Det var inte så svårt. Men svårare blev det när gällde att definiera begreppet CASE-verktyg och de krav ett sådant verktyg måste uppfylla. Det visade sig att det fanns olika och delvis motstridiga definitioner. Lyckligtvis fanns det kompetenta personer och tillgång till litteratur om ämnet på SISU, så att vi kunde bilda en uppfattning om vad CASE-verktyg är och vilka krav de bör uppfylla.

Nästa steg var att börja fundera på hur vi skulle lägga upp vårt arbete och göra någon form av tidsplan. Detta behövdes för att vi överhuvudtaget skulle få någon struktur i vårt arbetete och och för att vi och våra kontaktpersoner skulle kunna följa hur vi klarade oss. Första uppgiften var att konstruera praktikfallet. Efter - och delvis parallellt - med detta skulle vi utreda vilka egenskaper man bör undersöka hos ett CASE-verktyg och utforma ett granskningsprotokoll för att vara säkra på att vi undersökte verktygen noggrannt och på ett likvärdigt sätt. Arbetet med verktygen skulle ske med granskningsprotokollet som stöd för anteckningar om verktygens egenskaper. Slutligen skulle vi utforma och skriva en rapport baserad på våra anteckningar och erfarenheter. Tidsåtgången uppskattade vi med hjälp av de erfarenheter vi fått under det inledande arbetet och den tid vi hade på oss.

Vår arbetstsplan visade sig vara bra och tidsplanen realistisk (med rapportskrivningen undantaget). Vi klarade etapperna utan allvarligare förseningar. Vi hade inte satt någon dead-line när vår slutliga rapport skulle vara klar, och det var också realistiskt eftersom vi då hade riskerat att spränga dead-linen. Att skriva en rapport med bra struktur och välmotiverade slutsatser var något svårare än vi först trodde.

Vilka svårigheter stötte vi då på under arbetet, och vilka lärdomar fick vi? Vid konstruktionen av vårt praktikfall visade det sig svårt att få dokumentationen tillräckligt detaljerad och entydig. Att uppskatta hur lång tid implementationen av praktikfallet skulle ta i verkligheten var inte heller lätt, men det berodde nog på att vi saknade erfarenhet på området. Senare upptäckte vi också att ett praktikfall sällan täcker alla viktiga aspekterna i verklig systemutveckling. För det behövs olika typer av praktikfall. Exempelvis är kraven olika för ett statiskt registerbaserat informationssystem och ett realtidsystem med hårda tidskrav. Vårt praktikfall är inte särskilt lämpligt för att testa CASE-verktyg för design av realtidssystem.

Vad gäller granskningsprotokollet gäller motsvarande sak även där. Vårt granskningsprotokoll tar inte tillräcklig hänsyn till CASE-verktyg för realtidssystem. Men för att förbättra vårt granskningsprotokoll är det lätt att utöka det med motsvarande aspekter och definitioner. Man kan också tänka sig olika typer av granskningsprotokoll för olika typer av CASE-verktyg. Eller ett "modulariserat" protokoll med alternativa huvudområden innehållande aspekter för granskning av olika typer av verktyg. Då skulle man ha ett gemensamt protokoll för att studera verktyg som en del gemensamma egenskaper men som skiljer sig i sin inriktning eller tillämpningsområde. Vi fick lägga ned en hel del arbete för att skapa ett protokoll som skulle täcka de flesta viktiga egenskaperna och synvvinklarna att betrakta ett CASE-verktyg. Trots det är det inte troligt att vårt val av granskningsegenskaper och vårt sätt att dela upp dem är det bästa möjliga, eller ens korrekt. För att få ett bra och välstrukturerat protokoll

krävs nog en del experimenterande. Man får så att säga iterera fram ett bra protokoll. (Det samma gäller även för praktikfall).

Att granska verktygen praktiskt är ganska tidskrävande om man vill göra en noggrann analys och verktyget är omfattande. Att genomföra ett praktikfall visade sig vara ett mycket bra sätt att få en realistisk uppfattning i verktygens kvalitéer. Erfarna granskare skulle säkert kunna framställa en bra och mångsidig rapport om ett CASE-verktyg även utan ett praktikfall, men risken är stor att man missar många betydelsefulla och i det praktiska arbetet förekommande problem och finesser. En sak som också talar för ett praktikfall är att den systemutvecklingsmetod man tillämpar till verktyget är mycket viktigare än man tror. Fel vald metod, eller att inte välja någon metod alls, kan dramatiskt påverka ens uppfattningen om verktygets värde. Men att kunna välja lämplig metod och att bedöma verktygets lämplighet för flera olika metoder kräver kunskap. För oss innebar det extra självstudier och täta kontakter med verktygens utvecklare/leverantörer. Kontakter med utvecklare och leverantörer behövs också för att man skall kunna försäkra sig om att man inte förbisett några möjligheter eller fått något om bakfoten.

Det svåraste och mest tidskrävande var att bedöma betydelsen och effekterna av det man observerat under granskningen. För att rapporten skall bli objektiv måste man ge en rätt betoning på de granskade aspekterna och deras betydelse. Men hur man än gör är risken stor att beskrivningen blir subjektiv. Den metod man tillämpat påverkar förstås hur man lyckas med praktikfallet, vilket i sin tur påverkar ens uppfattning om verktygets brister och fördelar. Detta faktum understryker metodens och metodkunskapernas betydelse.

Man bör även analysera hur verktyget tar sig då det används i olika miljöer och på olika sätt (eller till och med för olika syften). Man måste då dels använda sin fantasi och dels inhämta kunskap från dem som använt verktyget (eller liknande verktyg) i sitt arbete.

Vi försöker sammanfatta våra erfarenheter och nyvunna kunskaper om detta slags studier i form av praktiska råd:

- Förbered och planera arbetet ordentligt innan du börjar
- Praktikfallet skall vara inom verktygets "målområde"
- Ett väldokumenterat praktikfall sparar tid och besvär
- Ta reda på de metoder som är (kan vara) lämpliga för verktyget och skaffa kunskap om dem. Egen erfarenhet eller hjälp av erfarna systemutvecklare är värdefullt
- Studera och diskutera granskingsprotokollet och gör anteckningar under granskningsarbetet
- Jämför med andra verktyg (under arbetet, ej nödvändigtvis rapporten) för att vidga vyerna, gärna verktyg med delvis olika inriktingar och kapaciteter.
- Analysera noggrannt hur de observerade egenskaperna påverkar systemutvecklingen, här är hjälp från "gamla rävar" mycket värdefullt.
- Diskutera betoningen av granskningsaspekter och verktygets egenskaper med flera personer för att få en någorlunda balanserad bild. Gärna med både forskare och erfarna systemerare och programmerare.
- Dubbelkolla allt oklart med utvecklaren eller leverantören.
- Låt leverantören och utvecklaren kommentera rapporten

REFERENSER

DeMarco79 Tom DeMarco, Structured Analysis and System Specifikation, Prentice-Hall, Inc., 1979

Jackson83 Michael A. Jackson, System Development, Prentice-Hall, Inc., 1983

Janis88 Janis Bubenko jr, Selecting A Strategy For Computer Aided Software Engineering, SYSLAB, 1988

Martin81 James Martin and Clive Finkelstein, Information Engineering, Savant Research Studies, 1981

Martin86 James Martin , Information Engineering, Savant Research Studies, 1986

Orr81 Kenneth Orr, Structured Requirements Definition, Ken Orr and Associates, Inc., 1981

RDF83 Sophus Lie-Nielsen och Lars Colliander, Principer för en ny generation av systemutvecklingsverktyg, Riksdataförbundets rapportserie nr 16 1983

RDF88 Peter Söderström och Dag Ohlstedt, Moderna systemutvecklingsverkyg, Riksdataförbundet 1988

Weth88 James C. Wetherbe, Systems Analysis and Design, West Publishing Company 1988.

Yourdon79 Edward Yourdon and Larry l.Constantine, Structured Design, Prentice-Hall, Inc., 1979

APPENDIX

FÖRKLARINGAR TILL APPENDIXET

I detta appendix återfinns delar av den dokumentation som vi kunde skapa med hjälp Deft respektive IEW. Observera att endast en bråkdel av all dokumentation visas på grund av utrymmesskäl. Syftet är inte att redovisa vårt praktikfall, utan att ge läsaren en fingervisning hur dokumentationen från respektive verktyg ser ut och vad den innehåller.

DEFT

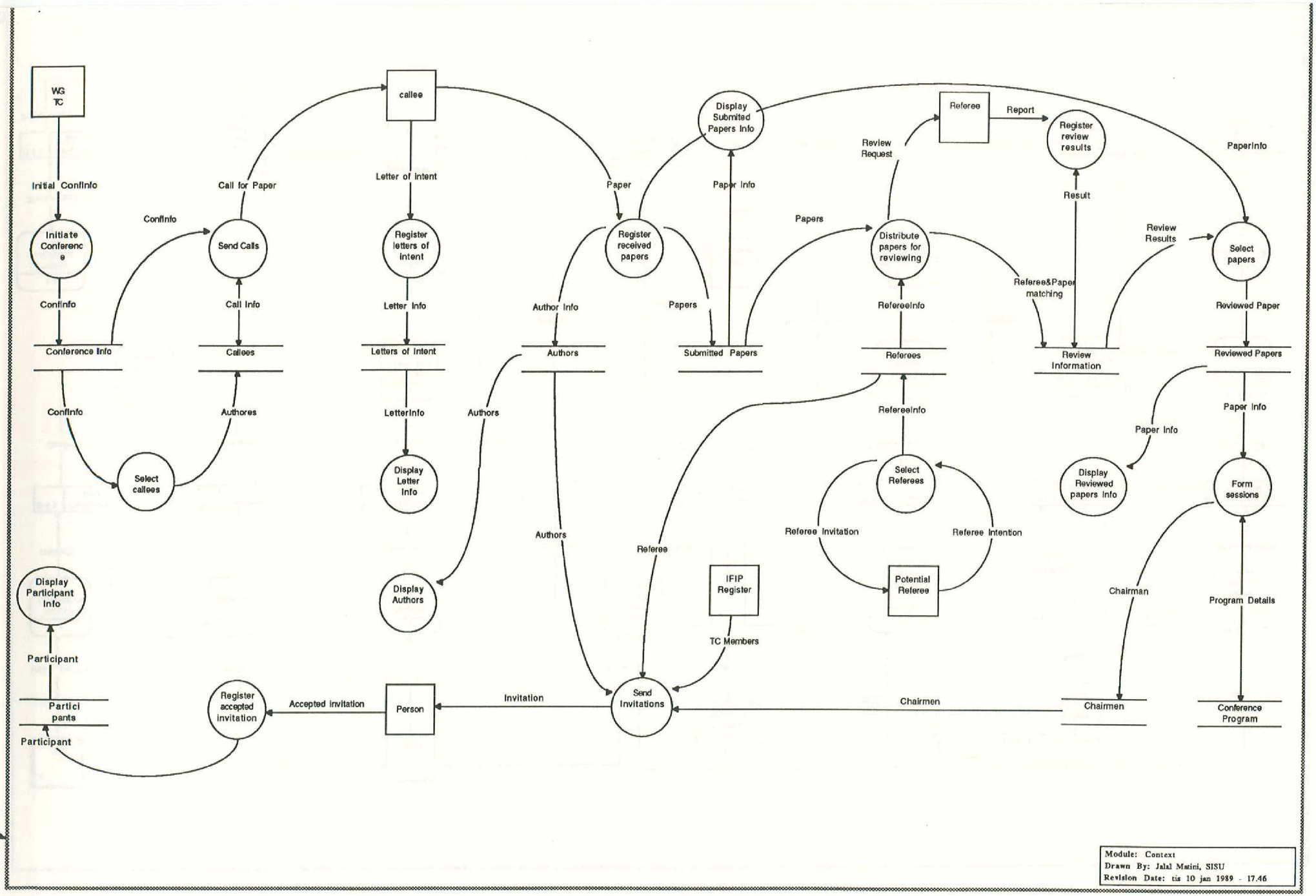
Dokumentationen till DEFT visar exempel på diagramutskrifterna och till diagrammen relaterade rapportutskrifter. Vi har tagit med exempel på alla typer av dokumentation som kan genereras med DEFT:

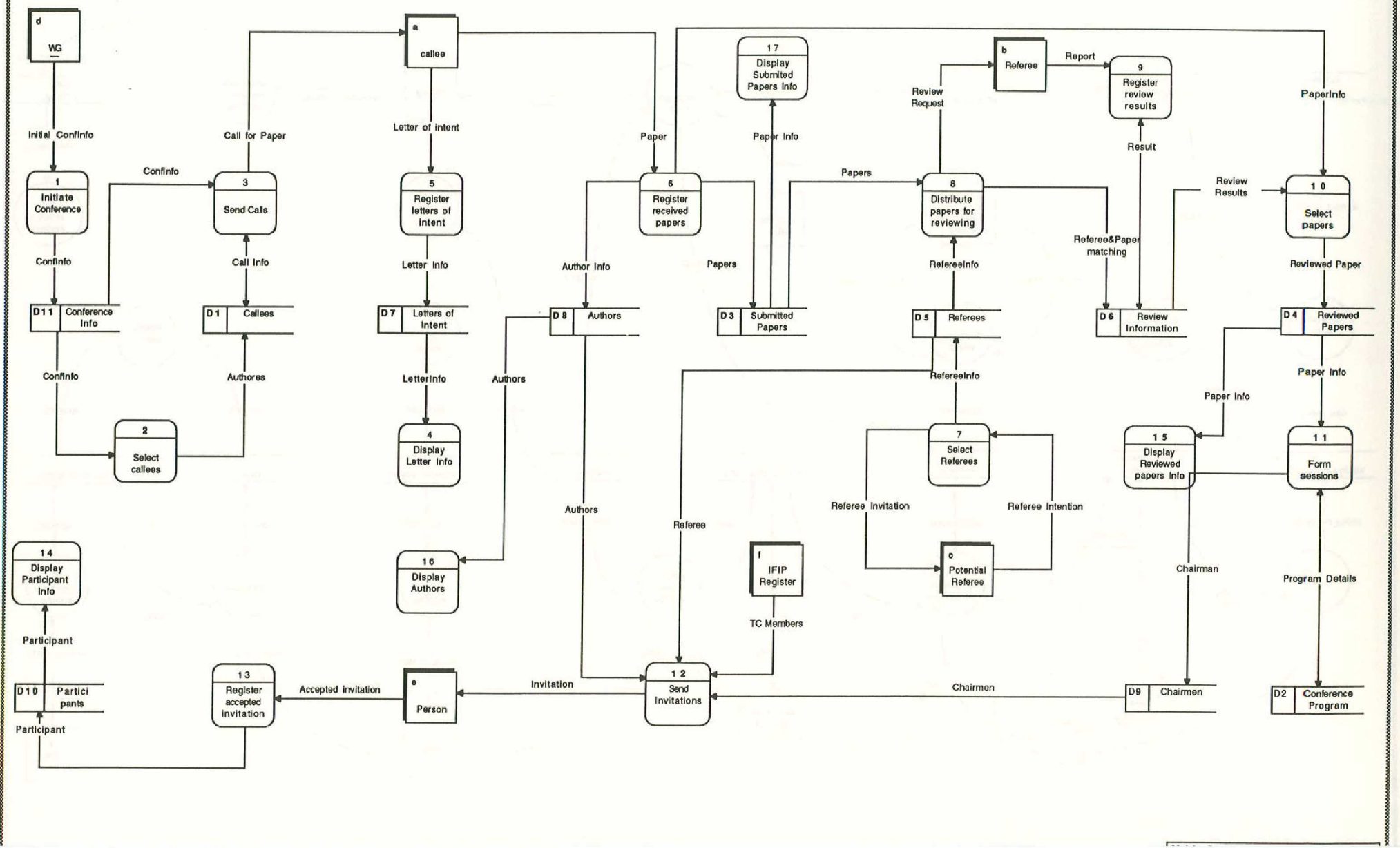
s. **1:** Dataflödesdiagram över konferenssystemet, högsta nivå, Yourdon.
2: Dataflödesdiagram över konferenssystemet, Gane&Sarson.
3: Dataflödesdiagram lägre nivå för beskrivning av processer på högsta nivån, här processerna Select Papers och Form Sessions.
5: Delar av en rapport från Dataflödesdiagram, innehåller information om attribut, dataflöden, datalager, sammansatta flöden och processer.
13: Entity-Relationship-diagram över dataobjekten i konf.systemet.
14: Rapport för ovanstående ERD.
20: Programstrukturdiagram för processen Select Papers. Blocken med tjocka kanter och märket '§' i över högra hörnet är subprogram som finns på andra filer.
21: Rapport för ovanstående programstrukturdiagram.
22: Prograstrukturdiagram för en av subprogrammen i ovanstående i PSD.
23: Motsvarande rapport.
24: Exempel en skärmlayout för registrering av uppgifter.
25: Motsvarande rapport.

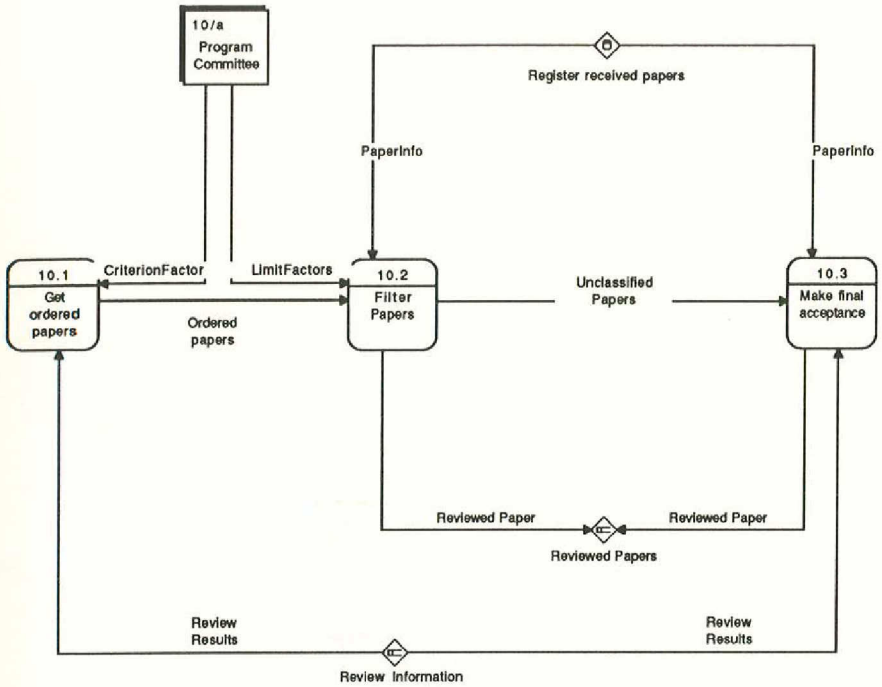
IEW

IEW har ett stort antal olika diagram, analyser och rapporter, varför vi bara visar exempel på några få av dessa. Diagrammen har skrivits ut med en vanlig matriskrivare, varför deras kvalité inte är den bästa möjliga.

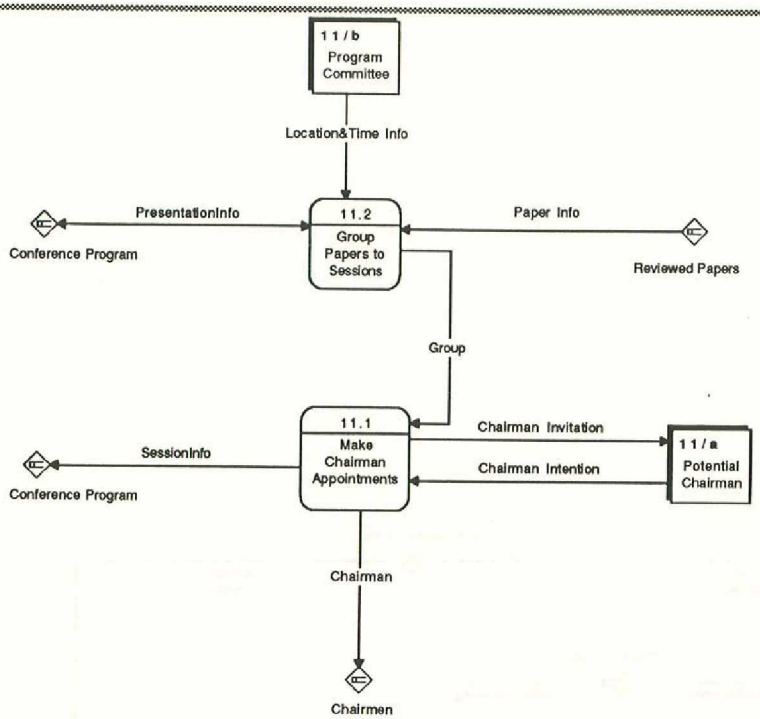
s. **26:** Dekompositionsdiagram över organisationen kring konferensen.
27: Den totala datamodellen över konferenssystemet i form av Entity-relationship-diagram.
28: Dekompositionsdiagram över verksamheten kring konferensanordnande. ('...' under ett block betyder att underliggande elementen inte visas).
29: Dekompositionsdiagram för processen Select Papers, som inte visades i sin helhet i föregående diagrammet.
30: Dataflödesdiagram på högsta nivå.
31: Dataflödesdiagram för processen Process Papers. Observera de streckade blocken som är import/export-kanaler till överliggande nivån.
32: Exempel på ett actiondiagram. Dessa används för att beskriva programlogiken för processer eller procedurer. Här beskrivs logiken för proceduren List_acc_papers (List accepted papers).
33: Encyclopedia Summary Report. Statistik över objekten i encyclopedien.
35: Object List, dvs en förteckning över alla objekt i systemspecifiklationerna, även objekttypen visas.


WG TC
Initial ConfInfo
Initiate Conferenc e
ConfInfo
Conference Info
ConfInfo
Select callees
Authores
Call Info
Callees
Call for Paper
callee
ConfInfo
Send Calls
Letter of Intent
Register letters of Intent
Letter Info
Letters of Intent
LetterInfo
Display Letter Info
Author Info
Authors
Authors
Display Authors
Paper
Register received papers
Papers
Submitted Papers
Paper Info
Display Submited Papers Info
Papers
Distribute papers for reviewing
RefereeInfo
Referees
RefereeInfo
Select Referees
Referee Invitation
Referee Intention
Potential Referee
Referee
Review Request
Referee
Report
Register review results
Result
Review Information
Referee&Paper matching
Review Results
PaperInfo
Select papers
Reviewed Paper
Reviewed Papers
Paper Info
Form sessions
Program Details
Conference Program
Paper Info
Display Reviewed papers Info
Chairman
Chairmen
Chairmen
Authors
Send Invitations
Invitation
Person
Accepted invitation
Register accepted invitation
Participant
Partici pants
Participant
Display Participant Info
IFIP Register
TC Members
Module: Context
Drawn By: Jalal Matiri, SISU
Revision Date: tis 10 jan 1989 - 17.46


d
WG
Initial ConfInfo
1
Initiate Conference
ConfInfo
D11
Conference Info
ConfInfo
2
Select callees
ConfInfo
3
Send Calls
Call for Paper
Call Info
D1
Callees
Authores
a
callee
Letter of intent
5
Register letters of intent
Letter Info
D7
Letters of Intent
LetterInfo
4
Display Letter Info
Author Info
D8
Authors
Authors
Authors
16
Display Authors
Paper
6
Register received papers
Papers
D3
Submitted Papers
Paper Info
17
Display Submited Papers Info
Papers
8
Distribute papers for reviewing
RefereeInfo
D5
Referees
RefereeInfo
7
Select Referees
Referee Invitation
Referee Intention
o
Potential Referee
Referee
f
IFIP Register
TC Members
12
Send Invitations
Invitation
e
Person
Accepted invitation
13
Register accepted invitation
Participant
D10
Partici pants
Participant
14
Display Participant Info
b
Referee
Report
9
Register review results
Result
Review Request
Referee&Paper matching
D6
Review Information
Review Results
10
Select papers
Reviewed Paper
D4
Reviewed Papers
Paper Info
11
Form sessions
Program Details
D2
Conference Program
PaperInfo
Paper Info
15
Display Reviewed papers Info
Chairman
D9
Chairmen
Chairmen



Module: Select papers
Drawn By: Jalal Maini, SISU
Revision Date: mån 9 jan 1989 - 18.18


11/b
Program Committee
Location&Time Info
PresentationInfo
Conference Program
11.2
Group Papers to Sessions
Paper Info
Reviewed Papers
Group
11.1
Make Chairman Appointments
Chairman Invitation
Chairman Intention
11/a
Potential Chairman
SessionInfo
Conference Program
Chairman
Chairmen

Report on Deft document 'IFIP-DFD_G&S'. Printed on tisdag 10 januari 1989.

Dictionary: IFIP-konferens

Attributes:

Address

AverageMark
an average mark of a paper for an evaluation criterion

CardNr
Cardinal Number

Conf_EndDate
When the conference should end.

Conf_StartDate
When the conference should begin.

Conf_Title
The title of the conference.

Country
The country where the person is living

Criterion
Has some of values : Relevance, technical Quality, style, clarity, or Referee Knowledge

(Resten av attributen har borttagits från rapporten av författaren)

Data Flows:

Accepted invitation
- FirstName
- LastName
- Address
- Country

AcceptedPapers
- Person_Id
- Paper_Nr
- Subjects
- DateRegistered

Author Info
- FirstName
- LastName
- Address
- Country
- Person_Id
- Title

Authors

FirstName
LastName
Country
Address
Title

Authors
FirstName
LastName
Country
Address
Person_Id
Title

Call Info
FirstName
LastName
Country
Address
Title

Call for Paper
Conf_Title
Conf_StartDate
Conf_EndDate
Location
Host

(Resten av dataflöden av borttagits av författaren)

Data Stores:

Authors
FirstName
LastName
Country
Address
Person_Id
Title

Callees
FirstName
LastName
Country
Address
Title

Conference Info
Conf_Title
Conf_StartDate
Conf_EndDate
Location
Subjects
Deadline
Host

Invitees
FirstName
LastName

Address

(Resten av datalagren har borttagits av författaren)

Composite Flows:
Program Details
PresentationInfo
SessionInfo

Process: Context

LOCAL FLOWS:

[3-a] Call for Paper
From: [3] Send Calls
To: [a] callee

[D1-3] Call Info
From: [D1] Callees
To: [3] Send Calls

[a-6] Paper
From: [a] callee
To: [6] Register received papers

[6-D3] Paper Info
From: [6] Register received papers
To: [D3] Submitted Papers

[8-b] Review Request
From: [8] Distribute papers for reviewing
To: [b] Referee

[b-9] Report
From: [b] Referee
To: [9] Register review results

(Resten av de lokala flödena (dvs. i det diagram som rapporten genererades från) har borttagits av författaren)

Total of 40 Data Flows defined.

PROCESSES

---

[1]: Initiate Conference

Establish and register the initiate conference information.

[2]: Select callees

Prepare a list of persons (authors) to whom calls for paper are to be sent. We are assuming that
the list of potential authors may be generated from other registers or is entered manually.

[3]: Send Calls

Get Callees from Callees register and print calls and forms for letters of intent to be sent (letters
of intent are supposed to be fill-in forms which are completed with information about receivers).

[4]: Display Letter Info

[5]: Register letters of intent

Register the information about letter of intent and the date it was received.

[6]: Register received papers

Register the information on a paper and the date when it arrived. Give every paper a uniqe paper_Id

(Resten av processinformationen har borttagits av författaren)

Total of 17 PROCESSES defined.

EXTERNAL ENTITIES

---

[a]: callee

A person who is expected to submit a contribution to the conference.

[b]: Referee

A person who reviews papers and fills in review forms (reports) to be send to the Program Committee
for the paper selection process.

[c]: Potential Referee

jhd ikyud

[d]: WG TC

Extern entity which presents a working group which demands that the conference be held.

[e]: Person

People who take part in the conference.

[f]: IFIP Register

Total of 6 EXTERNAL ENTITIES defined.

DATA STORES

[D1]: Callees

This is information about potential authors which is stored in order to make it possible for other
processes to select information about callees. This should also make it possible to fetch information
about authors .

[D2]: Conference Program

This contains all the information about the conference program, e.g. when the sessions should take
place, the chairmen etc.

[D3]: Submitted Papers

Information about papers and the dates they were received.

[D4]: Reviewed Papers

The papers which have been reviewed, and information about the authors.

(Resten av de lokala datalagren har borttagits av författaren)

Total of 11 DATA STORES defined.

(Här nedan ett exempel på informationen om processerna i ett dataflödesdiagram. Motsvarande information ges för varje process i diagrammet, författarens anm.)

Process: Send Invitations [12]

IN FLOWS:

[D9-12] Chairmen
From: [D9] Chairmen
To: [12] Send Invitations

[D8-12] Authors
From: [D8] Authors
To: [12] Send Invitations

[f-12] TC Members
From: [f] IFIP Register
To: [12] Send Invitations

[D5-12] Referee
From: [D5] Referees
To: [12] Send Invitations

OUT FLOWS:

[12-e] Invitation
From: [12] Send Invitations
To: [e] Person

LOCAL FLOWS:

[12.2-12.3] Other Invitees
From: [12.2] Get Other Persons to Invite
To: [12.3] Generate Invitee List

[12.3-12/D1] Other Invitees
From: [12.3] Generate Invitee List
To: [12/D1] Invitees

[12/D1-12.1] Invitee List
From: [12/D1] Invitees
To: [12.1] Generete Invitations

Total of 3 Data Flows defined.

PROCESSES

[12.1]: Generete Invitations

Gets the list of invitees from Invitee register and prints invitations to be send.

[12.2]: Get Other Persons to Invite

Person that are not automatically invited (like authors,referees etc.) can be entered in this process.

[12.3]: Generate Invitee List

Gets list of potential invitees and eliminates redundance.
Generates a register (Invitee) of persons to invite.

Total of 3 PROCESSES defined.

EXTERNAL ENTITIES

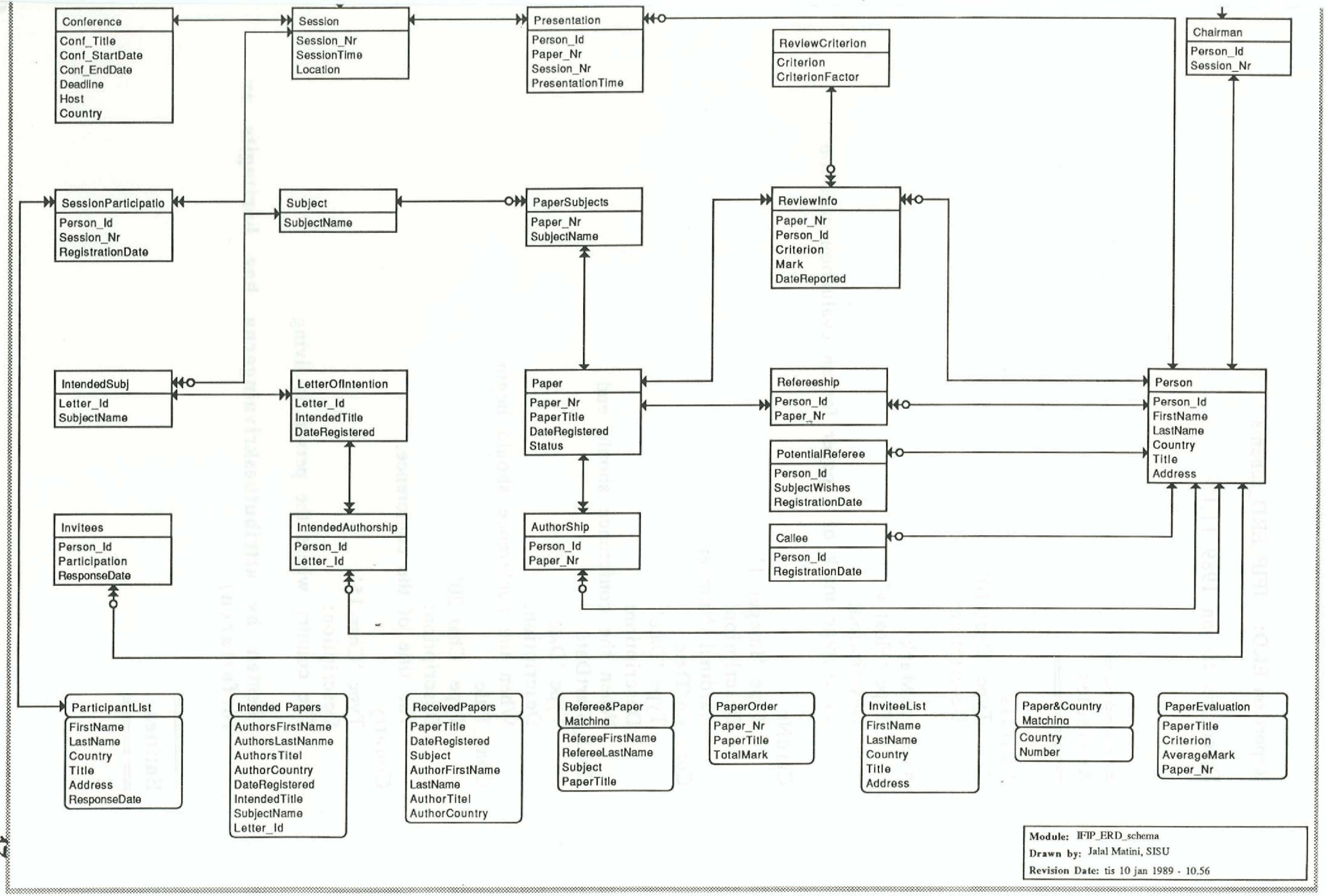
No EXTERNAL ENTITIES defined.

DATA STORES

[12/D1]: Invitees

One Data Store defined.

(Informationen om de resterande processerna har borttagits av författaren)


Conference
Conf_Title
Conf_StartDate
Conf_EndDate
Deadline
Host
Country
Session
Session_Nr
SessionTime
Location
Presentation
Person_Id
Paper_Nr
Session_Nr
PresentationTime
ReviewCriterion
Criterion
CriterionFactor
Chairman
Person_Id
Session_Nr
SessionParticipatio
Person_Id
Session_Nr
RegistrationDate
Subject
SubjectName
PaperSubjects
Paper_Nr
SubjectName
ReviewInfo
Paper_Nr
Person_Id
Criterion
Mark
DateReported
IntendedSubj
Letter_Id
SubjectName
LetterOfIntention
Letter_Id
IntendedTitle
DateRegistered
Paper
Paper_Nr
PaperTitle
DateRegistered
Status
Refereeship
Person_Id
Paper_Nr
Person
Person_Id
FirstName
LastName
Country
Title
Address
PotentialReferee
Person_Id
SubjectWishes
RegistrationDate
Invitees
Person_Id
Participation
ResponseDate
IntendedAuthorship
Person_Id
Letter_Id
AuthorShip
Person_Id
Paper_Nr
Callee
Person_Id
RegistrationDate
ParticipantList
FirstName
LastName
Country
Title
Address
ResponseDate
Intended Papers
AuthorsFirstName
AuthorsLastNanme
AuthorsTitel
AuthorCountry
DateRegistered
IntendedTitle
SubjectName
Letter_Id
ReceivedPapers
PaperTitle
DateRegistered
Subject
AuthorFirstName
LastName
AuthorTitel
AuthorCountry
Referee&Paper Matching
RefereeFirstName
RefereeLastName
Subject
PaperTitle
PaperOrder
Paper_Nr
PaperTitle
TotalMark
InviteeList
FirstName
LastName
Country
Title
Address
Paper&Country Matching
Country
Number
PaperEvaluation
PaperTitle
Criterion
AverageMark
Paper_Nr
Module: IFIP_ERD_schema
Drawn by: Jalal Matini, SISU
Revision Date: tis 10 jan 1989 - 10.56

Report on ERD: IFIP_ERD_schema

Dated: tis 10 jan 1989 11.17

==========
Attributes
==========

Address
Type 'Char 10'
Description:

AverageMark
Type 'Float 4'
Description:
an average mark of a paper for an evaluation criterion

CardNr
Type 'Integer 1'
Description:
Cardinal Number

Conf_EndDate
Type 'Date'
Description:
When the conference should end.

Conf_StartDate
Type 'Date'
Description:
When the conference should begin.

Conf_Title
Type 'Char 20'
Description:
The title of the conference.

Country
Type 'Char 15'
Description:
The country where the person is living

(Resten av attributbeskrivningarna har borttagits av författaren)

========
Entities
========

---

[1]: AuthorShip

-----Get Info-----

A table to match the authors and the papers which they have written. This is a result of the normalising process.

-----Defined Keys-----

Primary Key:

Storage Type: Unique BTree

Person_Id
Paper_Nr

-----Related Objects-----

Entity 'Person' (one)
Entity 'Paper' (one)

-----Attributes-----

Person_Id
Paper_Nr

---

[2]: Callee

-----Get Info-----

This is a table of all the potential callees. The key is the person-Id, and the date when a person has become a callee in the system.

-----Defined Keys-----

Primary Key:

Storage Type: BTree

Person_Id

-----Related Objects-----

Entity 'Person' (one)

-----Attributes-----

Person_Id
RegistrationDate

---

[3]: Chairman

-----Get Info-----

A table containing information about who will be chairman for the different sessions. This is a result
of the normalising process to eliminate many to many relations.

-----Defined Keys-----

Primary Key:

Storage Type: Unique BTree

Person_Id

-----Related Objects-----

Entity 'Person' (one)
Entity 'Session' (many)

-----Attributes-----

Person_Id
Session_Nr

(Resten av entitetsbeskrivningarna har borttagits av författaren)...

==========

Views

==========

---

[1]: PaperEvaluation

-----Auto Related Objects-----

Entity 'Paper'
Entity 'ReviewInfo'

-----Related Objects-----

-----Attributes-----

Where Clause:
Paper.Paper_Nr = ReviewInfo.Paper_Nr AND
ReviewInfo.Criterion = ReviewCriterion.Criterion

Type	Object	Attribute	Rename
Entity	Paper	PaperTitle	PaperTitle
Entity	ReviewInfo	Criterion	Criterion
Derived		AverageMark	

Expression: average(Mark) where ReviewInfo.Criterion = ReviewCriterion.Criterion

Entity	Paper	Paper_Nr	

---

[2]: PaperOrder

-----Auto Related Objects-----

View 'PaperEvaluation'

-----Related Objects-----

-----Attributes-----

Where Clause:
ReviewInfo.Criterion=ReviewCriterion.Criterion GROUP BY ReviewInfo.Paper_Nr

Type	Object	Attribute	Rename
View	PaperEvaluation	Paper_Nr	

View PaperEvaluation PaperTitle

Derived TotalMark
Expression: SUM(ReviewCreterion.Factor *
PaperEvaluation.AvaregeMark)

---

[3]: InviteeList

-----Auto Related Objects-----

Entity 'Person'

-----Related Objects-----

-----Attributes-----

Where Clause:
Person.Person_id=Invitees.Person_Id

Type	Object	Attribute	Rename
Entity	Person	FirstName	
Entity	Person	LastName	
Entity	Person	Country	
Entity	Person	Title	
Entity	Person	Address	

(Resten vybeskr. har borttagits av författaren)...

8 views defined.

==========

Relationships

==========

---

[1]: AuthorShip +Person

-----Entities/Views-----

Entity 'Person' (one)
Entity 'AuthorShip ' (zero or many)

-----Attributes-----

Entity 'AuthorShip ' Entity 'Person'
Person_Id Person_Id

---

[2]: Chairman+Person

-----Entities/Views-----

Entity 'Person' (one)
Entity 'Chairman' (one)

-----Attributes-----

Entity 'Chairman' Entity 'Person'
Person_Id Person_Id

---

[3]: Chairman+Session

-----Entities/Views-----

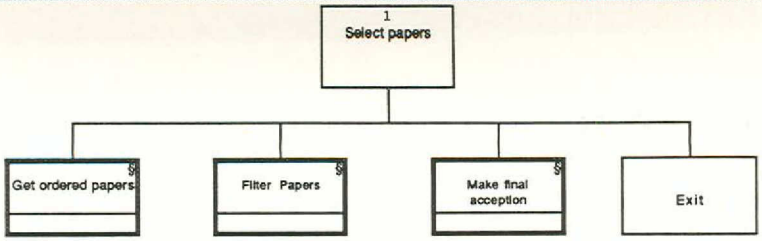
Entity 'Session' (many)
Entity 'Chairman' (one)

-----Attributes-----

Entity 'Chairman' Entity 'Session'
Session_Nr Session_Nr

(Resten av relationsbeskr. har borttagits av författaren)...

23 relationships defined.



Module: Select papers
Drawn by: Matinik:Kinnula, SISU
Revision Date: mån 2 jan 1989 - 17.02
Page: 1 XRefPage: 1

Report on PSD document
IFIP_PSD_SelectPapers

Get ordered papers

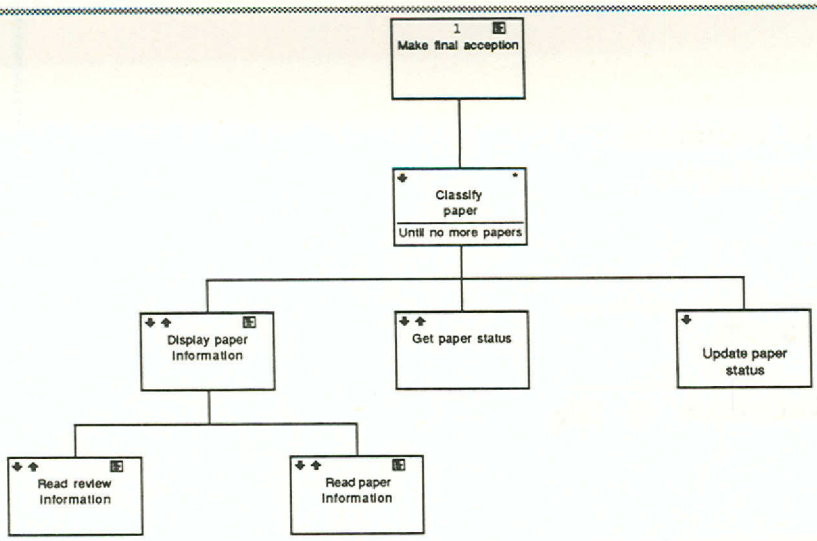
Subprogram Reference to file
IFIP_PSD1:

Filter Papers

Subprogram Reference to file
IFIP_PSD2:

Make final acception

Subprogram Reference to file
IFIP_PSD3:
The following subprogram calls
had no file reference.



Submodule: Make final acception
Drawn by: Matinik Kinnula, SISU
Revision Date: mån 9 jan 1989 - 20.14
Page: 1 XRefPage: 1

Report on PSD document IFIP_PSD_MakeFinalAccept

Make final acception [1]

Get Info:
Make the final decision for papers that have not been assigned status yet. The final paper status is entered manually.

Classify paper

Display paper information

Get Info:
Display information about a paper (authors, subject, review results) on screen

---

Read review information

Get Info:
Read review information from file or database.

Read paper information

Get Info:
Read information about the paper (title, author, subject, country) from file or database

---

Get paper status

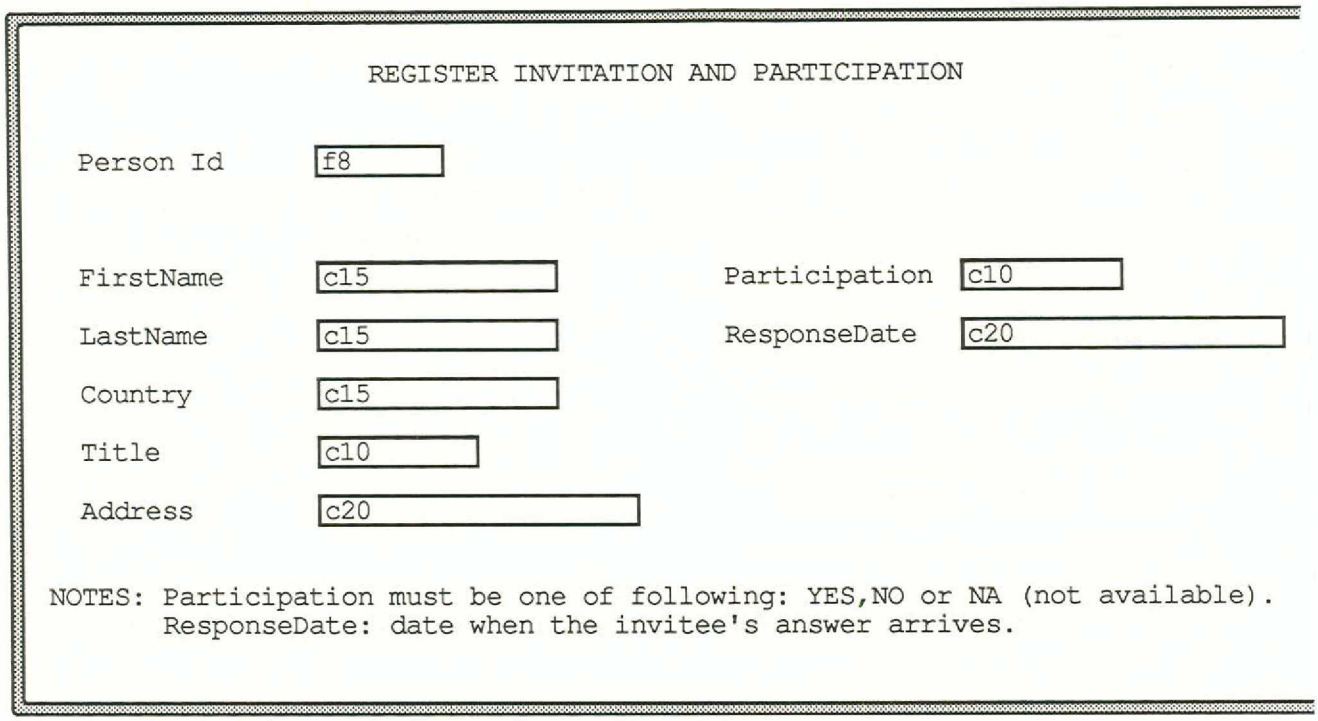
Update paper status

The following subprogram calls had no file reference.

Form Name: Register Invitees

Form Owner: SISU

Designed by: Jalal Matini



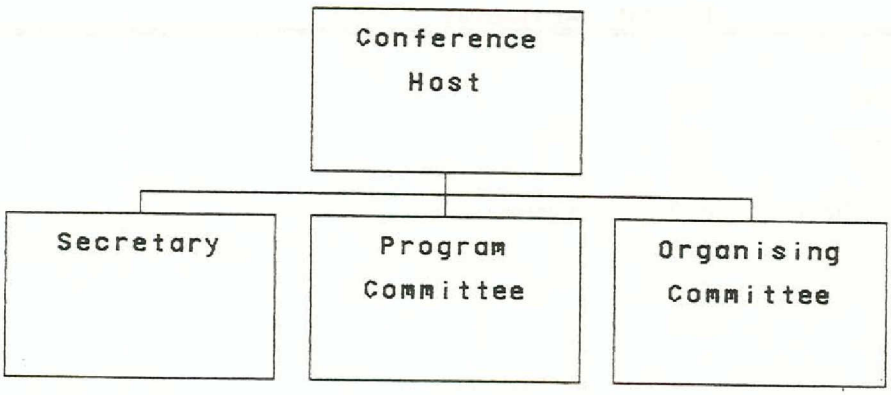
Purpose: to register persons to be invited to the conference, and to register whether they intend to participate

Form Name: Register Invitees

Form Owner: SISU

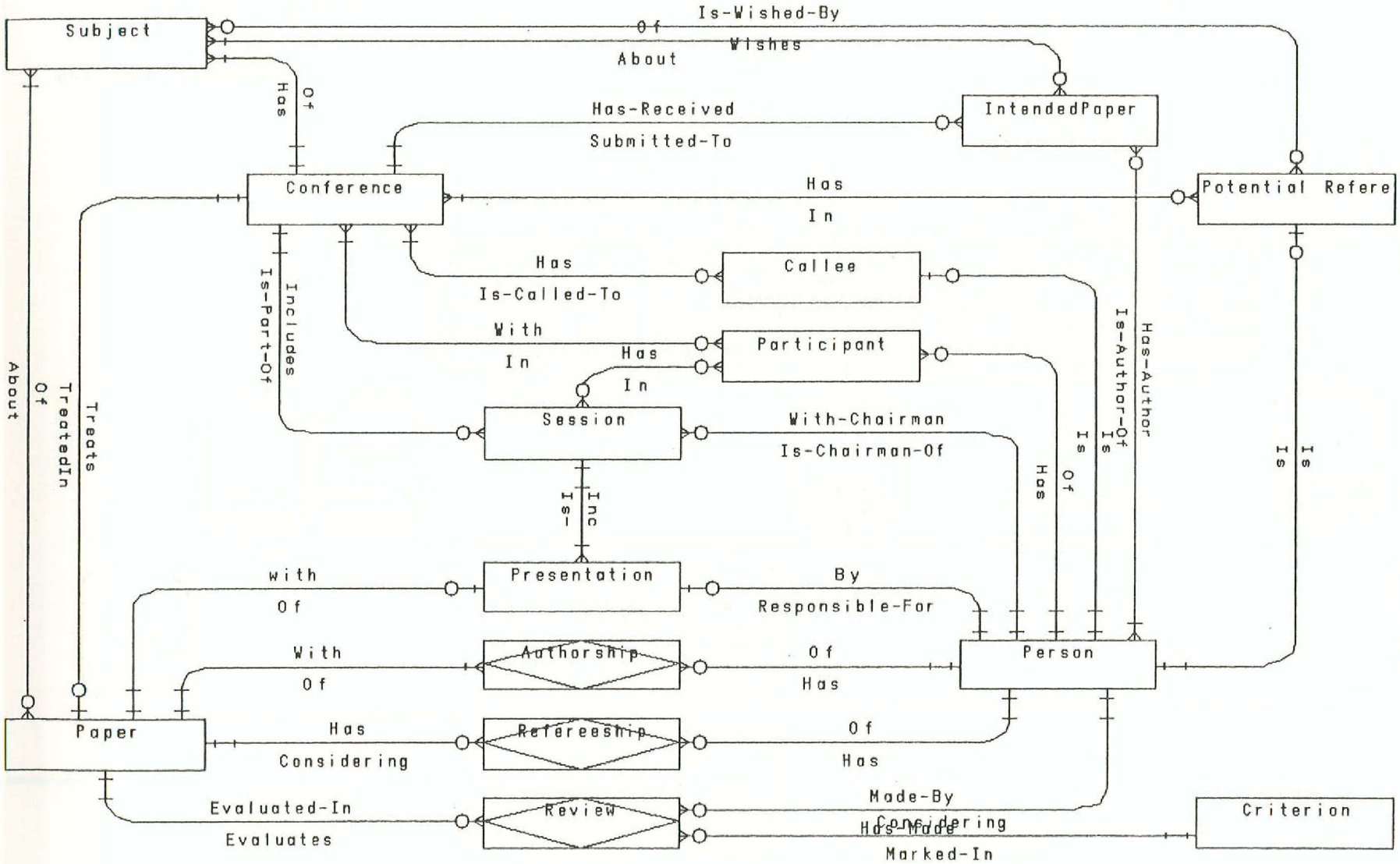
Designed by: Jalal Matini

Field	Title	Format	Attributes
Person_Id	Person Id	f8	
FirstName	FirstName	c15	
Participatio	Participation	c10	Blink
LastName	LastName	c15	
ResponseDate	ResponseDate	c20	Autotab
Country	Country	c15	
Title	Title	c10	
Address	Address	c20	



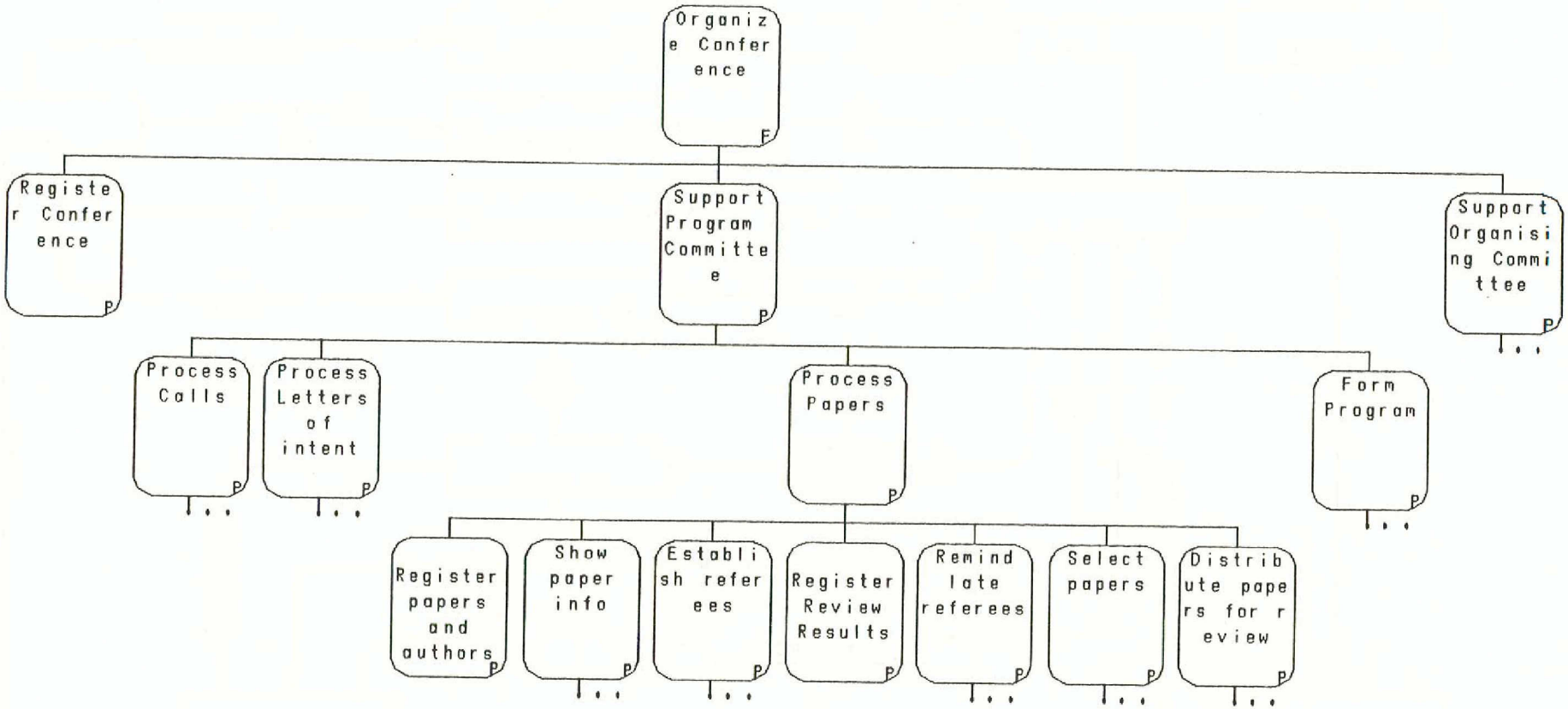
Conference Host

November 1, 1988 9:56:50



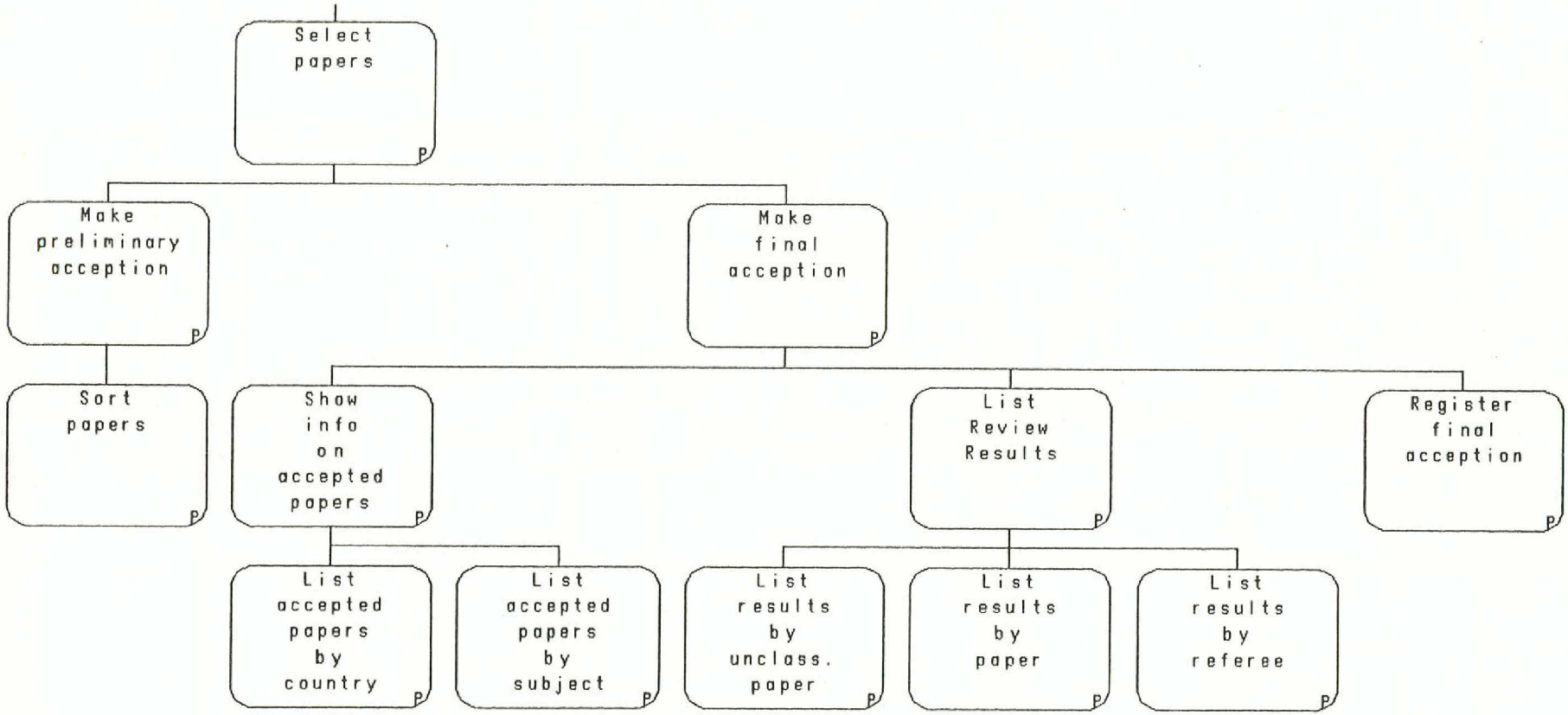
27

ntity Model



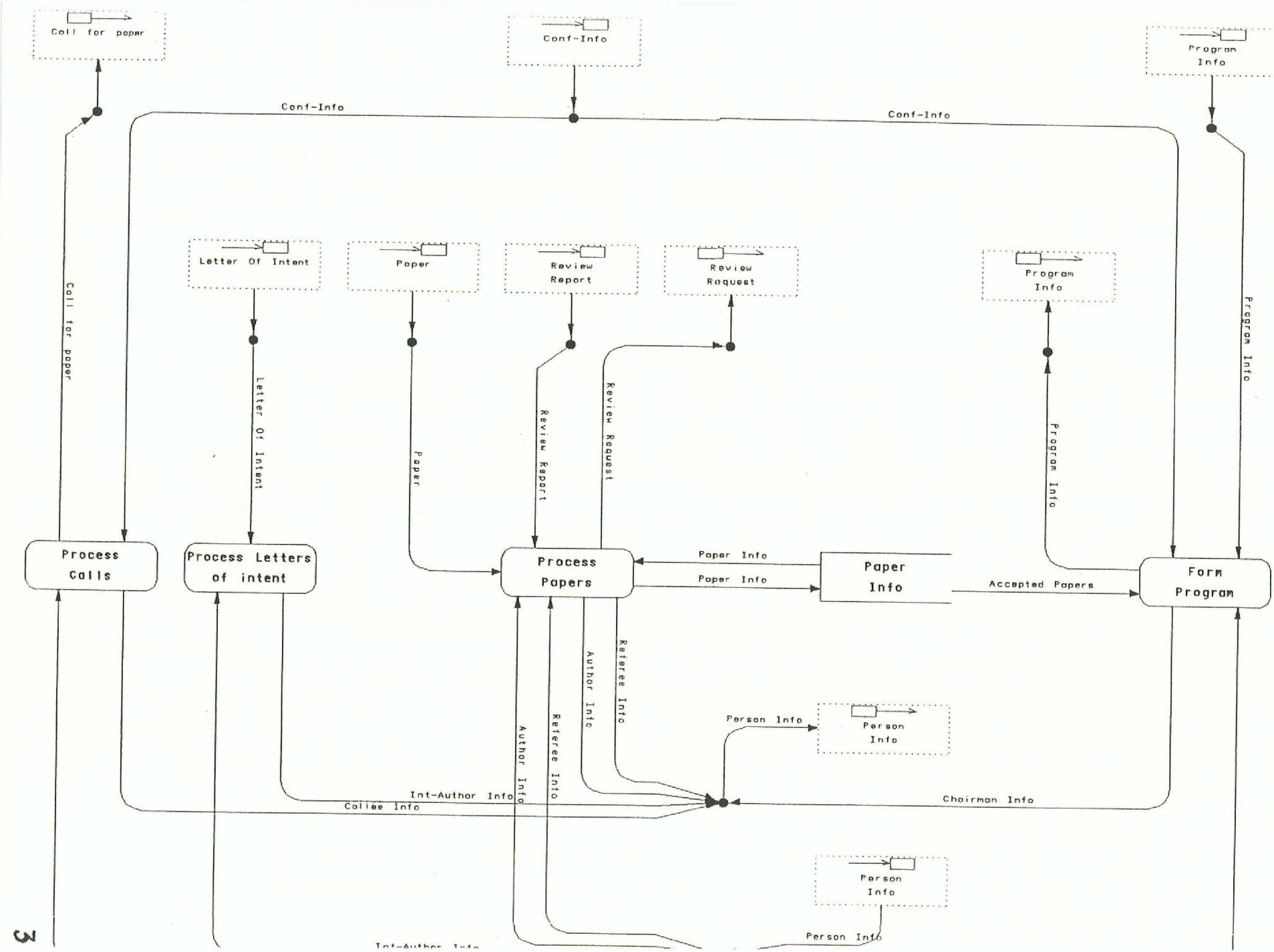
Organize Conference

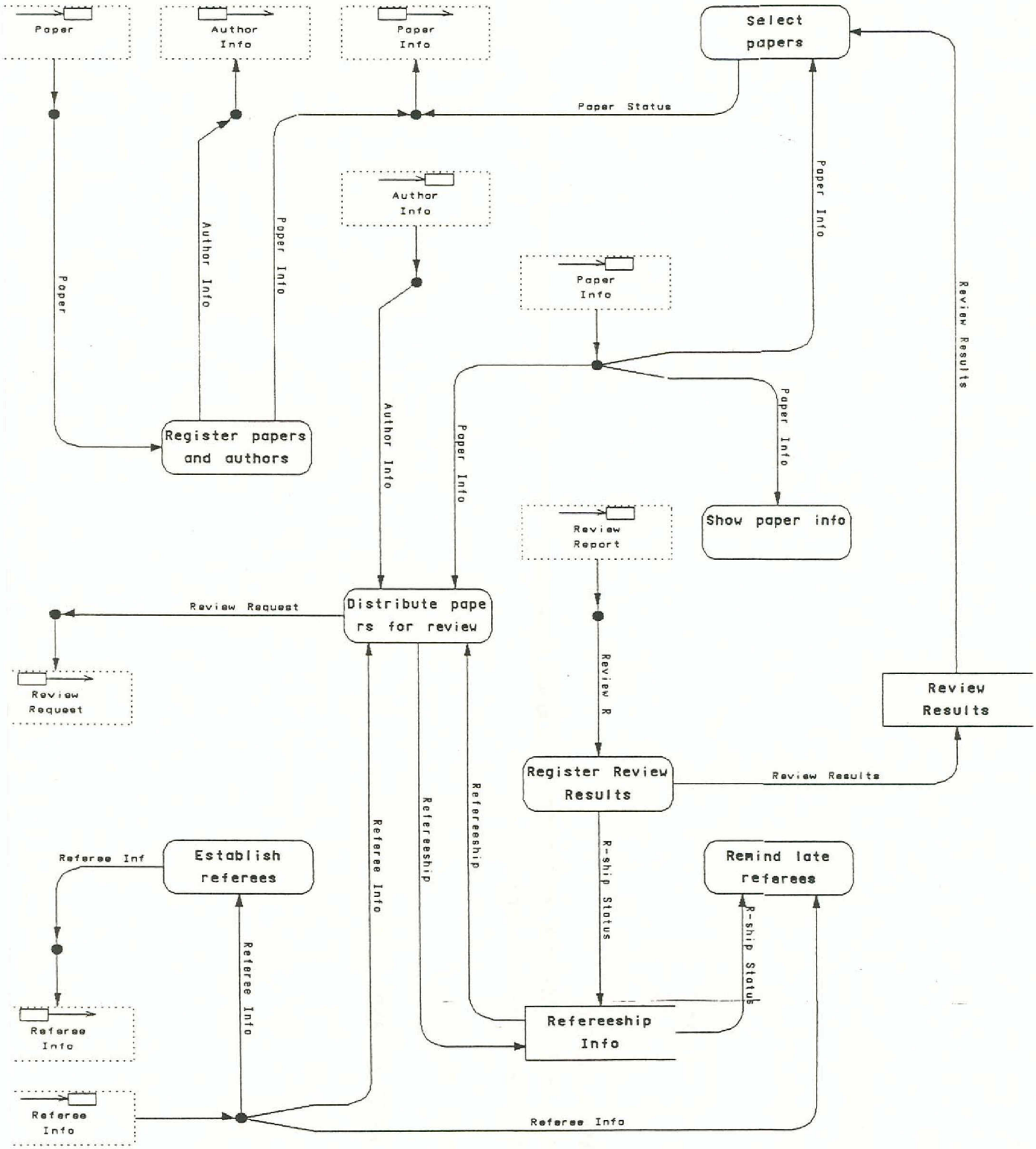
November 1, 1988 9:51:40

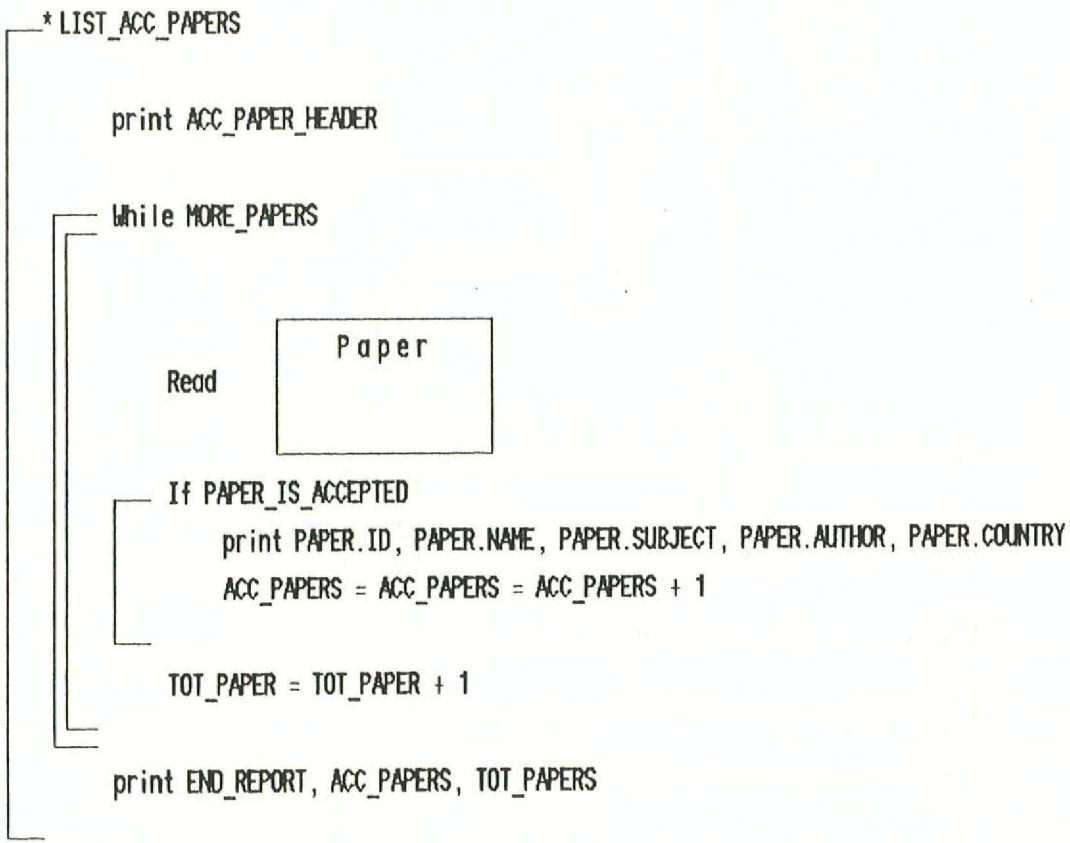


Select papers

November 1, 1988 9:52:50


Call for paper
Conf-Info
Program Info
Conf-Info
Conf-Info
Letter Of Intent
Paper
Review Report
Review Request
Program Info
Call for paper
Letter Of Intent
Paper
Review Report
Review Request
Program Info
Program Info
Process Calls
Process Letters of intent
Process Papers
Paper Info
Paper Info
Paper Info
Accepted Papers
Form Program
Author Info
Referee Info
Author Info
Referee Info
Person Info
Person Info
Int-Author Info
Callee Info
Chairman Info
Person Info
Person Info
Int-Author Info


Paper
Author Info
Paper Info
Select papers
Paper Status
Author Info
Paper Info
Paper
Author Info
Paper Info
Paper Info
Review Results
Register papers and authors
Author Info
Paper Info
Paper Info
Review Report
Show paper info
Review Request
Distribute papers for review
Review Request
Review R
Review Results
Register Review Results
Review Results
Referee Inf
Establish referees
Refereeship
Refereeship
Referee Info
R-ship Status
Remind late referees
Referee Info
R-ship Status
Referee Info
Refereeship Info
Referee Info
Referee Info
Papers
ir 1, 1988 13:33:04



List accepted papers

November 1, 1988 10:16:05

Encyclopedia Summary Report

Object counts:

57	attribute types
3	sequential processes
0	critical assumptions
0	critical success factors
0	data collections
30	data flows
8	data stores
45	data structure templates
1	data types
0	DBDs
14	entity types
4	external agents
0	flat file databases
1	functions
0	goals
29	information needs
71	junctions
0	libraries
0	locations
0	mechanisms
0	modeling sources
1	modules
4	organizational units
0	problems
64	processes
0	projects
0	PSBs
0	records
23	relations
1	relational databases
26	relationship types
0	screens
0	segments

0	subject areas

Planning association counts:

3	Function consists of Process
59	Process consists of Process
313	Process involves Entity Type
3	Organizational Unit manages Organizational Unit
128	Process supports Information Need

Object List

Authorship.Authorship	Attribute Type
Authorship.Paper-Nr	Attribute Type
Authorship.Person-Id	Attribute Type
Callee.Date-Call-Sent	Attribute Type
Callee.Date-Registered	Attribute Type
Callee.Person-Id	Attribute Type
Conference.Address	Attribute Type
Conference.Call-Response-DeadLine	Attribute Type
Conference.Conf-EndDate	Attribute Type
Conference.Conf-StartDate	Attribute Type
Conference.Conf-Title	Attribute Type
Conference.Country	Attribute Type
Conference.Host	Attribute Type
Conference.Mail-Address	Attribute Type
Conference.Participation-Response-Deadline	Attribute Type
Conference.Review-Deadline	Attribute Type
Criterion.Crit-Factor	Attribute Type
Criterion.Criterion	Attribute Type
IntendedPaper.DateRegistered	Attribute Type
IntendedPaper.Intended-Title	Attribute Type
IntendedPaper.Letter-Nr	Attribute Type
Paper.Country	Attribute Type
Paper.DateRegistered	Attribute Type
Paper.Paper-Nr	Attribute Type
Paper.Paper-Title	Attribute Type
Paper.Ranking	Attribute Type
Paper.Status	Attribute Type
Participant.Invitation-Registered	Attribute Type
Participant.Invitation-Sent	Attribute Type
Participant.Person-Id	Attribute Type
Participant.Response	Attribute Type
Participant.Response-Date	Attribute Type
Person.Address	Attribute Type
Person.Country	Attribute Type
Person.First-Name	Attribute Type
Person.Last-Name	Attribute Type
Person.Person-Id	Attribute Type
Person.Title	Attribute Type
Potential Referee.Date-Registerd	Attribute Type
Potential Referee.Person-Id	Attribute Type
Presentation.Paper-Nr	Attribute Type
Presentation.Person-Id	Attribute Type
Presentation.Presentation-Time	Attribute Type
Presentation.Session-Nr	Attribute Type
Refereeship.Paper-Nr	Attribute Type
Refereeship.Person-Id	Attribute Type
Refereeship.Refereeship	Attribute Type
Refereeship.Terminated	Attribute Type
Review.Criterion	Attribute Type
Review.Mark	Attribute Type
Review.Paper-Nr	Attribute Type
Review.Person-Id	Attribute Type
Review.Review	Attribute Type
Session.Place	Attribute Type
Session.Session-Nr	Attribute Type
Session.Session-Time	Attribute Type
Subject.Subject	Attribute Type
Accepted Papers	Data Flow
Author Info	Data Flow
Callee Info	Data Flow
Call Form	Data Flow
Call for paper	Data Flow
Chairman Info	Data Flow
Conf-Info	Data Flow
cut	Data Flow
Int-Author Info	Data Flow
Invitation	Data Flow
Invitation Form	Data Flow
Invitee Info	Data Flow
Late Referees	Data Flow

Letter Info	Data Flow
Letter Of Intent	Data Flow
Paper	Data Flow
Paper Info	Data Flow
Paper Status	Data Flow
Participant Info	Data Flow
Participation	Data Flow
Person Info	Data Flow
Program Info	Data Flow
R-ship Status	Data Flow
Refereeship	Data Flow
Referee Info	Data Flow
Response	Data Flow
Review Report	Data Flow
Review Request	Data Flow
Review Results	Data Flow
Sorted Papers	Data Flow
Conference Info	Data Store
Intended Paper Info	Data Store
Paper Info	Data Store
Participation	Data Store
Person Info	Data Store
Program Info	Data Store
Refereeship Info	Data Store
Review Results	Data Store
Entity Token	Data Store
Authorship	Data Type
Callee	Entity Type
Conference	Entity Type
Criterion	Entity Type
IntendedPaper	Entity Type
Paper	Entity Type
Participant	Entity Type
Person	Entity Type
Potential Referee	Entity Type
Presentation	Entity Type
Refereeship	Entity Type
Review	Entity Type
Session	Entity Type
Subject	Entity Type
Author	Entity Type
IFIP-Unit	External Agent
Invitee	External Agent
Referee	External Agent
Organize Conference	External Agent
Accepted papers	Function
Accepted papers per country	Information Need
Accepted papers per subject	Information Need
Authors	Information Need
Callees	Information Need
Chairman appointments	Information Need
conference program	Information Need
Contracted referees	Information Need
Intended authors	Information Need
Intended authors and papers	Information Need
Intended papers per country	Information Need
Intended papers per subject	Information Need
Invitees	Information Need
Late referees	Information Need
Papers per session	Information Need
papers received by a given date	Information Need
Paper distribution per referee	Information Need
Participants	Information Need
Referees per paper	Information Need
Referee subject wishes	Information Need
Review res. per unclass. paper	Information Need
Review results per paper	Information Need
Review results per referee	Information Need
Session information	Information Need
Session participation	Information Need
Submitted papers and authors	Information Need
Submitted papers per country	Information Need

Submitted papers per subject	Information Need
Willing referees	Information Need
select papers	Module
Conference Host	Organizational Unit
Organising Committee	Organizational Unit
Program Committee	Organizational Unit
Secretary	Organizational Unit
Decide final distribution	Process
Distribute papers for review	Process
Establish Participation	Process
Establish referees	Process
Form Call	Process
Form Invitation	Process
Form Program	Process
Get late referees	Process
Group papers into sessions	Process
list/print appointments	Process
List/print program	Process
List accepted papers by country	Process
List accepted papers by subject	Process
List Callees	Process
List distribution info	Process
List info on intended papers	Process
List intended papers and authors	Process
List intended papers by country	Process
List intended papers by subject	Process
List Invitees	Process
List papers by country	Process
List papers by referee	Process
List papers by subject	Process
List paper distribution	Process
List participants by session	Process
List particpants	Process
List referees	Process
List referees by paper	Process
List results by paper	Process
List results by referee	Process
List results by unclass. paper	Process
List Review Results	Process
List submitted papers	Process
Make final acception	Process
Make preliminary acception	Process
Match papers and referees	Process
Organize-Conference	Process
Print Calls	Process
Print Invitations	Process
Print reminders for late referes	Process
Print review requests	Process
Process Calls	Process
Process Invitations	Process
Process Letters of intent	Process
Process Papers	Process
Register Callees	Process
Register chairman appointments	Process
Register Conference	Process
Register distribution	Process
Register final acception	Process
Register info on person	Process
Register Invitees	Process
Register Letters of intent	Process
Register papers and authors	Process
Register participation	Process
Register referees	Process
Register Review Results	Process
Remind late referees	Process
Select papers	Process
Show info on accepted papers	Process
Show paper info	Process
Sort papers	Process
Support Organising Committee	Process
Support Program Committee	Process
Authorship	Relation

Callee	Relation
Conference	Relation
Conference.Has.Callee	Relation
Conference.Host	Relation
Conference.With.Participant	Relation
Criterion	Relation
IntendedPaper	Relation
IntendedPaper.About.Subject	Relation
Paper	Relation
Paper.About.Subject	Relation
Participant	Relation
Person	Relation
Person.Is-Author-Of.IntendedPape	Relation
Potential Referee	Relation
Potential Referee.In.Conference	Relation
Potential Referee.Wishes.Subject	Relation
Presentation	Relation
Refereeship	Relation
Review	Relation
Session	Relation
Session.Has.Participant	Relation
Subject	Relation
IFIP-DBASE	Relational Database
Authorship.Of.Paper	Relationship Type
Authorship.Of.Person	Relationship Type
Callee.Is-Called-To.Conference	Relationship Type
Callee.Is.Person	Relationship Type
Conference.Has-Received.IntendedPaper	Relationship Type
Conference.Has.Callee	Relationship Type
Conference.Has.Potential Referee	Relationship Type
Conference.Has.Subject	Relationship Type
Conference.Includes.Session	Relationship Type
Conference.Treats.Paper	Relationship Type
Conference.With.Participant	Relationship Type
Criterion.Marked-In.Review	Relationship Type
IntendedPaper.About.Subject	Relationship Type
IntendedPaper.Has-Author.Person	Relationship Type
IntendedPaper.Submitted-To.Conference	Relationship Type
Paper.About.Subject	Relationship Type
Paper.Evaluated-In.Review	Relationship Type
Paper.Has.Refereeship	Relationship Type
Paper.TreatedIn.Conference	Relationship Type
Paper.With.Authorship	Relationship Type
Paper.with.Presentation	Relationship Type
Participant.In.Conference	Relationship Type
Participant.In.Session	Relationship Type
Participant.Of.Person	Relationship Type
Person.Has-Made.Review	Relationship Type
Person.Has.Authorship	Relationship Type
Person.Has.Participant	Relationship Type
Person.Has.Refereeship	Relationship Type
Person.Is-Author-Of.IntendedPaper	Relationship Type
Person.Is-Chairman-Of.Session	Relationship Type
Person.Is.Callee	Relationship Type
Person.Is.Potential Referee	Relationship Type
Person.Responsible-For.Presentation	Relationship Type
Potential Referee.In.Conference	Relationship Type
Potential Referee.Is.Person	Relationship Type
Potential Referee.Wishes.Subject	Relationship Type
Presentation.By.Person	Relationship Type
Presentation.Is-Part-Of.Session	Relationship Type
Presentation.Of.Paper	Relationship Type
Refereeship.Considering.Paper	Relationship Type
Refereeship.Of.Person	Relationship Type
Review.Considering.Criterion	Relationship Type
Review.Evaluates.Paper	Relationship Type
Review.Made-By.Person	Relationship Type
Session.Has.Participant	Relationship Type
Session.Includes.Presentation	Relationship Type
Session.Is-Part-Of.Conference	Relationship Type
Session.With-Chairman.Person	Relationship Type
Subject.Is-Wished-By.Potential Referee	Relationship Type

Subject.Of.Conference	Relationship Type
Subject.Of.IntendedPaper	Relationship Type
Subject.Of.Paper	Relationship Type
List accepted papers	Sequential Process
Register info on conference	Sequential Process
Register prel acception	Sequential Process
Authorship.id	Template Data Structure
Authorship.id 911	Template Data Structure
Callee.id	Template Data Structure
Callee.id 819	Template Data Structure
Conference.id	Template Data Structure
Conference.id 666	Template Data Structure
Conference.id 717	Template Data Structure
Conference.id 959	Template Data Structure
Criterion.id	Template Data Structure
Criterion.id 283	Template Data Structure
Criterion.id 667	Template Data Structure
IntendedPaper.id	Template Data Structure
IntendedPaper.id 215	Template Data Structure
IntendedPaper.id 246	Template Data Structure
IntendedPaper.id 94	Template Data Structure
Paper.id	Template Data Structure
Paper.id 191	Template Data Structure
Paper.id 747	Template Data Structure
Paper.id 808	Template Data Structure
Participant.id	Template Data Structure
Participant.id 443	Template Data Structure
Person.id	Template Data Structure
Person.id 278	Template Data Structure
Person.id 480	Template Data Structure
Person.id 561	Template Data Structure
Potential Referee.id	Template Data Structure
Potential Referee.id 259	Template Data Structure
presentation.id	Template Data Structure
presentation.id 329	Template Data Structure
Presentation.id 370	Template Data Structure
Presentation.id 97	Template Data Structure
Refereeship.id	Template Data Structure
Refereeship.id 314	Template Data Structure
Review.id	Template Data Structure
Review.id 334	Template Data Structure
Review.id 596	Template Data Structure
Review.id 728	Template Data Structure
Session.id	Template Data Structure
Session.id 736	Template Data Structure
Session.id 817	Template Data Structure
Session.id 9	Template Data Structure
Subject.id	Template Data Structure
Subject.id 642	Template Data Structure
Subject.id 672	Template Data Structure
Subject.id 692	Template Data Structure