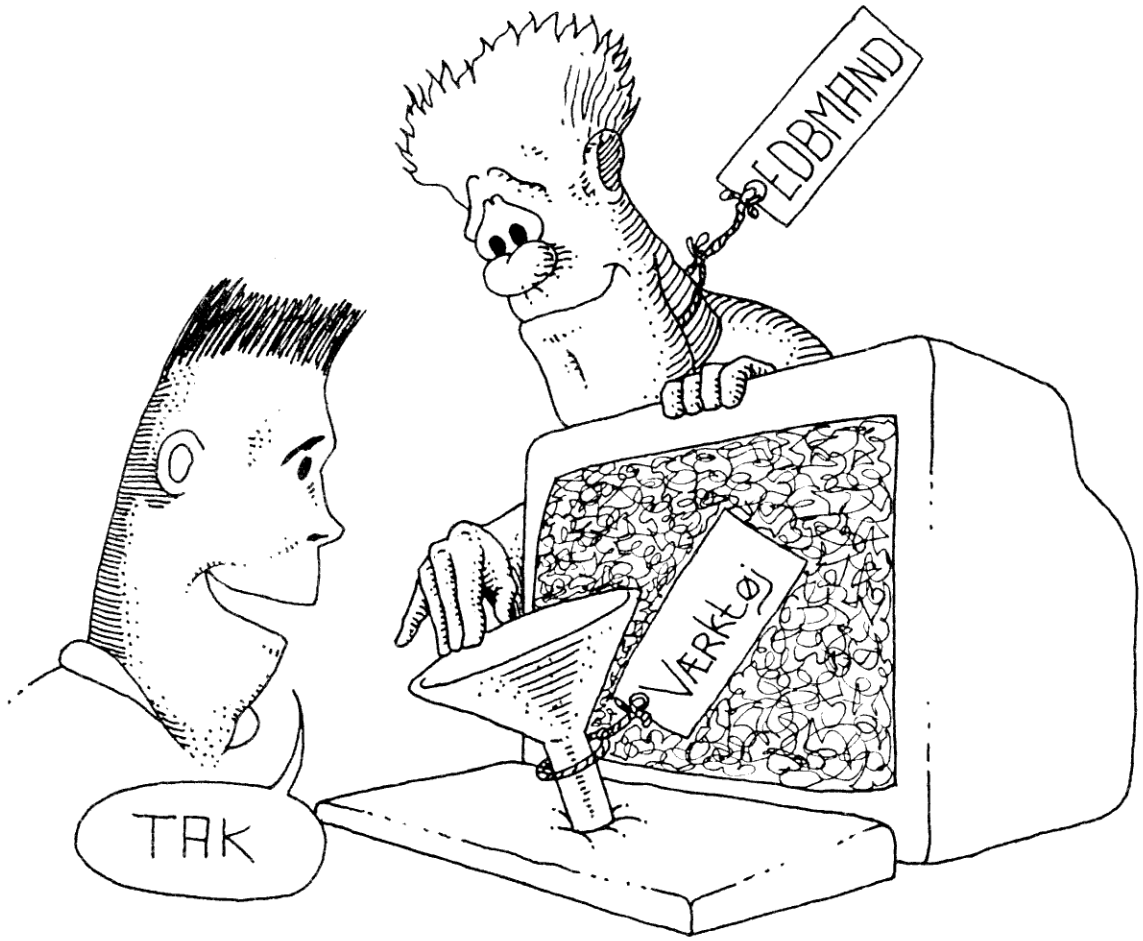
SPECIALEAFHANDLING:

D A T A L O G I S K E V Æ R K T Ø J E R
T I L U D V I K L I N G A F
U N D E R V I S N I N G S P R O G R A M M E R



DATALOGISK INSTITUT, KØBENHAVNS UNIVERSITET, JUNI 1986.
JOHN A. HUNDERUP.

I N D H O L D

Indhold.

FORORD.

Nærværende rapport er en specialeafhandling, der er en del af andendels studiet i datalogisk hovedfag ved Datalogisk Institut, Københavns Universitet.

Rapporten er udarbejdet af John A. Hunderup i foråret 1986 og udgør det afsluttende andendels arbejde. Arbejdet udgør de sidste 300 skriftlige kredittimer svarende til 600 arbejdstimer, ud af 2. dels studiets 800 skriftlige timer. Det faktiske tidsforbrug udgør ca. 900 timer.

I forbindelse med projektet har jeg modtaget råd og vejledning fra flere sider.

Specielt vil jeg takke min vejleder Hugh Tucker, Datalogisk Institut, for inspiration og støtte til projektet.

Endelig takkes Mogens Lyster Knudsen, Danmarks Lærerhøjskole, Torsten Alf Jensen, konsulentfirmaet JJKS, Jens Chr. Hauge, Sparekassernes Data Center, Tom Nymann, Danmarks Sparekasseforening og rektor Kenny Jørgensen, Esbjerg Seminarium for kritik og rettelser samt interesse for projektet.

John A. Hunderup
Juni 1986.

RESUMÉ

Projektets hovedemne er:

- de programmelsystemer, betegnet undervisningssystemer, der anvendes til udvikling af undervisningsprogrammer, dvs. EDB programmer, der i forbindelse med en datamat indgår i en situation, hvor mennesker dygtiggør sig.

Målet for projektet er:

(1) Analyse af eksisterende værktøjer til udvikling af datamatiske undervisningsforløb.

(2) Udvikling af en systematik og nomenklatur for undervisningssystemer.

(3) At give en begrebsmæssig ramme, - der kunne danne baggrund for en metodologi for design og implementation af undervisningssystemer, - opnås ud fra (2)

(4) Opstilling af en systematik, der giver beskrivelsesparametre for programmer, der kan betegnes som datalogiske værktøjer.

(5) At informere om eksisterende datalogiske værktøjer.

(6) At vise hvorledes brugergrænsefladen for et eksisterende undervisningssystem i praksis kan udbygges.

(7) At fremstille ovennævnte punkter, også mere datalogisk prægede (pkt. 2-4), i en let læselig form, tilgængelig for en forholdsvis bred læsergruppe med interesse for undervisningssystemer.

Projektet omfatter fire dele:

(1) Erfaringer med eksisterende undervisningssystemer.

(2) Analyse af arkitekturen for et datalogisk opbygget undervisningssystem, samt udvikling af en teori herfor.

Resumé.

(3) Udvikling af programmel v.h.a. eksisterende datalogisk værktøj.

(4) Konklusion samt anbefalinger for fortsatte undersøgelser.

Del 1:

Der tages udgangspunkt i eksisterende programmuligheder, idet eksisterende værktøjer beskrives dels ud fra egne erfaringer, dels ud fra et konkret projektforløb.

Arbejdet med eksisterende programmer omfatter to forfatterredskaber, nemlig TenCore og COMUS/SOFUS. Danmarks Sparekasseforenings erfaringer fra projekt med indførelse af dataformidlet undervisning fremlægges.

Del 2:

På denne baggrund og ud fra disse erfaringer analyseres arkitekturen for et Datamatisk Interaktivt Integreret Undervisnings System (DIIUS). Der præsenteres en opdeling af et sådant system på baggrund af funktionelle, strukturelle og virtuel-niveau aspekter i eksisterende systemer. Der udvikles en teori, der omfatter en systematik og nomenklatur for et DIIUS. Systematikken giver bl.a. grundlaget for den begrebsmæssige ramme for den konceptuelle forståelse af et sådant system.

Systematikken omfatter blandt andet en strukturel opdeling i "supra-", "intermediær-" og "infra-struktur" programmel.

Anvendelsesmodellen for et undervisningsprogram beskrives i relation til brugergrænseflademodul og lagringsmodul. Interface imellem funktionelle moduler i supra-strukturen beskrives.

Muligheden for en dynamisk omformning af et DIIUS ved hjælp af intermediær-struktur programmellet introduceres.

Resumé.

Der vises en systematik til identifikation af specifikke værktøjer, idet værktøjerne identificeres ud fra funktion og en vurdering af tilhørsforhold til en datalogisk eller en undervisningsmæssig begrebsverden.

Del 3:

Eksisterende datalogiske værktøjer indenfor den opstillede systematik beskrives. Ved hjælp af et eksisterende intermediær-struktur værktøj til opbygning af brugergrænseflade, foretages der design og implementation af programmel, der udbygger et undervisningssystem med en brugergrænseflade, der anvender "mus" og "rullegardinmenuer".

Del 4:

Konklusion på dette projektforløb, samt anbefalinger for undersøgelser i fremtidige projekter.

Del 1-4:

Fremstillingen gives en forholdsvis bred og let tilgængelig form, jævnfør projektets mål punkt (7).

Resumé.

1. INDLEDNING.

Indledningsvis gives et overblik over rapportens struktur, terminologi fastlægges og der gives et indtryk af de faktorer, der har virket inspirerende og idédannende, - og som dermed har dannet baggrund for arbejdet. Endvidere fremlægges en række afklaringer med relation til projektets hovedemne undervisningssystemer.

1.1. Rapportens fire dele.

Rapporten henvender sig til en forholdsvis bred læsergruppe. Det er således også søgt at give den mere datalogisk prægede del 2 en sådan form, at læsere uden en egentlig datalogisk baggrund kan få udbytte af læsningen.

Første del indeholder beskrivelser af eksisterende værktøjer, samt erfaringer med disse. Erfaringerne udgøres af: anvendelse af dataformidlet undervisning i sparekassesektoren, samt arbejde med to konkrete mikrodatamat forfatterredskaber.

Rapportens første del udgør baggrunden for rapportens del 2: "Datalogiske værktøjer", samt en række illustrationer til punkter heri. Del 1 danner endvidere baggrund dels for supplerende undersøgelser i denne rapport, dels for fremtidige undersøgelser.

I anden del foretages der en teoretisk og formel analyse af systematik, struktur og funktion for et generelt programmelsystem til udvikling af datamatiske undervisnings forløb. Samtidig præsenteres et af hovedemnerne for rapporten: et avanceret datalogisk programmelkompleks til opbygning af datamatiske undervisningsprogrammer. Dette programmelsystem betegnes et "Datamatisk Interaktivt Integreret Undervisnings System" (DIIUS). Der lægges her vægt på opbygning af en systematik, samt en nomenklatur til støtte for den konceptuelle opfattelse af et sådant kompliceret programmelsystem.

Det datalogiske værktøjsbegreb introduceres. Rene datalogiske værktøjer, der kan anvendes som byggestene ved opbygning

af undervisningsprogrammer.

Det analyseres og beskrives, hvorledes nødvendige værktøjer kan identificeres og en række mulige værktøjer omtales.

Tredie del omhandler eksisterende datalogiske værktøjer. Et sådant værktøj anvendes til design og implementation af programmel, der udbygger SOFUS *) systemet med en brugergrænseflade, der anvender udpegning v.h.a. mus og rullegardinmenuer. Implementationen foregår således, at der ikke gribes ind i det eksisterende SOFUS system. Den nye brugergrænseflade implementeres med programmel, der let lader sig tilpasse individuelle behov og som er helt uafhængigt af SOFUS programmellet.

*) SOFUS = Skelet til Opbygning af Frie Undervisnings Systemer, et undervisningssystem der beskrives i rapportens del 1.

Fjerde del omfatter konklusion og beskrivelse af nuværende status for udvikling af datamatiske undervisningsforløb ved hjælp af datalogiske værktøjer. En åbning imod fremtidige undersøgelser beskrives.

1.2. Undervisningssystemer og programmer.

Indledningsvis præsenteres vigtige termer i den anvendte terminologi. Der henvises iøvrigt til ordlisten i appendiks B.

Til at betegne den undervisning, der foregår med datamaten og tilhørende undervisningsprogram, vil jeg anvende det generiske begreb datamatisk undervisningsforløb. - Jeg vil undlade at falde for fristelsen til endnu en trebogstav kombination: DUF!

Begrebet undervisningsprogram anvendes i en meget bred forstand. At jeg anvender en meget bred definition, er blot et udtryk for, dels at selve det færdige program ikke er hovedemnet for denne rapport, - dels et udtryk for, at det netop ved anvendelse af generelle værktøjer ikke er særlig

afgørende, hvorledes disse vil blive anvendt, integreret eller sammenhægtet.

Bemærk, at termen undervisningsprogram anvendes som betegnelse for det færdige produkt, der anvendes på datamaten i undervisningssituationen til styring af datamaten, når den anvendes i et datamatisk undervisningsforløb. Ofte anvendes betegnelsen undervisningssystem herfor, - men denne betegnelse vil jeg anvende således:

Undervisningssystem vil betegne det programmel, der anvendes som støtte, værktøj eller datalogisk udviklingsmiljø ved produktion af et undervisningsprogram. Undervisningssystemet anvendes således til at behandle den aktuelle model for et givet undervisningsforløb. Tænk her på parallellen et databasesystem, der er det værktøj, der anvendes til at behandle modellen i den situation, nemlig databasen. Eller et tekstbehandlingssystem, der er det værktøj, der anvendes til at bearbejde modellen i den situation, nemlig tekstfilen.

Med den brede definition jeg anvender på begrebet undervisningsprogram, kan det således dække over systemer, der spænder fra de i dag kendte programmer udviklet ved hjælp af f.eks. forfatterredskaber, - til informationsprogrammer, der kunne anvendes i en enhver formidlingssituation.

Det kunne f.eks. dreje sig om brugermanualer, illustrationer til et foredrag eller lignende. Igen skal begreberne opfattes meget bredt. F.eks. kunne en "brugermanual" udgøres af et interaktivt tekst/video system, og "illustrationerne" til foredraget udgøres af en stor videoskærm, f.eks. på størrelse med en tavle, hvorpå tekst/grafik/billeder/film kunne fremvises.

Det er med andre ord vigtigt, at man i det følgende ikke opfatter begrebet undervisningsprogram for snævert. Groft sagt må vi erkende, at vi befinder os i barndommen for anvendelse af datateknik i forbindelse med undervisning. Når rapportens første del tager sit udgangspunkt i eksisterende systemer og muligheder, bliver der derfor nødvendigvis taget udgangspunkt i snævre og begrænsede systemer.

Indledning.

1.3. Baggrund og idégrundlag.

I dette afsnit præsenteres nogle af de ideer, der har givet mig inspiration til dette arbejde.

Talrige bøger og artikler beskæftiger sig med begreber som f. eks. CAP, CAL, CAI, CAT, CMI etc. eller tilsvarende danske begreber DFU og DSU, - samt en lang række andre begreber, der dækker over en bestemt anvendelse af et datamatsystem i forbindelse med undervisning under en eller anden form.

Der er således tale om beskrivelser/vurderinger af færdigudviklede og evt. kørende systemer. Det er ikke muligt at beskrive og undersøge undervisningsmæssige og EDB-tekniske forhold separat, idet der allerede er sket en tæt sammenknytning af problemer med relation til både datalogi og undervisning.

Mit udgangspunkt for en undersøgelse af problemerne omkring datamatiske undervisningsforløb er en helt anden, - idet jeg vil undersøge mulighederne for en ren datalogisk indfaldsvinkel. Udgangspunktet er her en idé om, at det vil være gavnligt at stræbe imod udvikling og anvendelse af rene datalogiske værktøjer, der i så høj grad som muligt er frigjort for undervisningsmæssige overvejelser.

Målet er ingenlunde at gøre "undervisning til datalogi", - ej heller at gøre "datalogien til undervisning", men derimod at undersøge mulighederne for at frigøre undervisningssystemet fra specifikke datalogiske systemer, og at frigøre datalogien fra specifikke undervisningsmæssige problemer.

Grundideen er, at udvikling og anvendelse af datalogiske værktøjer, der befinder sig på et velvalgt niveau og som udviser "passende" indbyrdes relationer, måske kunne give værktøjer, der i så høj grad som muligt er frigjort fra undervisningsmæssige overvejelser. De kunne således udvikles af EDB fagfolk uden behov for nogen dybere indsigt i undervisning. Måske ville det derefter være muligt at udvikle undervisningsprogrammer, frigjort fra tekniske/datalogiske

bånd. Programmer, der således kunne udvikles af lærere uden større EDB-teknisk indsigt.

Dette område er et af datalogiens mange grænseområder, - nemlig grænsen imellem datalogi og pædagogik/undervisning. Men netop i en forståelse herfor og i en erkendelse heraf, ønsker jeg at vise, at udvikling af datalogiske værktøjer, som jeg opfatter dem her, vil kunne lette arbejdet på tværs af denne faggrænse.

Inspirationen til at forfølge disse tanker har jeg fået gennem indsigt i en række af de erfaringer, der hidtil er gjort med anvendelse af datamater i forbindelse med undervisning:

1.3.1. Kravet om undervisningsprogrammer.

Ved udvikling af undervisningsprogrammer har man ofte stået i en situation, hvor man har følt, at der var tale om et valg imellem på den ene side, at lade EDB kyndige udvikle programmer, eller på den anden side at lade lærere uden et væsentligt EDB kendskab udvikle programmer. Denne sidste konstellation har ofte givet anledning til udvikling af programmer, der er af en mindre god datalogisk standard.

"Konflikten" imellem disse to fagdicipliner bliver ikke mindre af, at gode lærere på den ene side tilsynelande bliver dårligere lærere, når de opsluges af facinationen ved at udvikle programmer til datamaten. På den anden side, bliver gode EDB folk tilsyneladende mindre gode EDB kyndige, når de skal koncentrere sig om undervisningsmæssige forhold.

Folkeskole/gymnasie sektoren er netop i disse år konfronteret med det problem, at der er indkøbt store mængder maskinel, for ca. 90% vedkommende Regnecentralens Piccoline, til et samlet beløb, der andrager flere hundrede mill. kr., - mens der på den anden side næsten ikke er afsat resourcer til indkøb af programmel.

Der er derfor et stort og presserende problem med udvikling af et stort antal undervisningsprogrammer, - altså et krav

om volumen. Men naturligvis er der også et krav om teknisk og pædagogisk kvalitet.

Overfor ønsket om volumen står det faktum, at der er begrænsede resourcer til rådighed, - her tænkes både på økonomiske resourcer og på EDB viden hos udviklerne. Heraf affødes et ønske om værktøjer, der er hurtige at anvende, og som er let tilgængelige for de lærere, der skal anvende dem. Det er netop disse værktøjer denne rapport arbejder henimod.

1.4. Datamater og undervisning.

Vi er i øjeblikket i en situation, hvor teknologien konstant tilfører EDB området ny teknik. Problemet, når enmet er EDB baserede undervisningsprogrammer, er det samme som indenfor alle andre EDB områder: der er et konstant behov for udvikling af programmel. Først gennem udvikling af passende mængder programmel bliver maskinellet omdannet til anvendelige brugersystemer, - og først da, er man i stand til at vurdere den egentlige anvendelse af systemerne, maskinerne og undervisningsprogrammerne - nemlig i samspillet imellem EDB-systemet og mennesket!

Dag for dag åbnes der således for nye muligheder for undervisningsmæssige anvendelser af datamater. Identifikation og konstruktion af datalogiske værktøjer kunne bidrage til en hurtigere udnyttelse af disse muligheder. Jo hurtigere der kan udvikles prototypesystemer, jo hurtigere kan der nås til en afklaring af de pædagogiske muligheder mediet evt. indeholder. F.eks. er nye muligheder for grafik og videobilleder, evt. integreret med tekst, langt fra udnyttet i eksisterende systemer.

Generelle datalogiske værktøjer giver mulighed for en hurtigere og mere dynamisk udvikling af undervisnings programmer. En hurtigere udvikling giver et mindre resource forbrug samt bedre muligheder for at udnytte nye teknologiske muligheder.

Læreren vil med sådanne værktøjer få bedre muligheder for at udarbejde datamatiske undervisningsforløb. Læreren vil f.eks. have bedre mulighed for at afprøve nye ideer i prak-

sis, idet værktøjerne giver kortere udviklingstid og stiller læreren mere frit.

Det endelige ideale mål er, som tidligere omtalt, at give mulighed for, at undervisningsprogrammer udvikles af lærere ved hjælp af generelle datalogiske værktøjer, der giver så få bindinger som muligt med hensyn til de undervisningsmæssige muligheder.

En indvending der ofte mødes er, at enhver anvendelse af datamater i forbindelse med undervisningen udgør et valg og dermed et bånd på undervisningen. Det er også korrekt, - vi skal blot være enige om, hvad der skal lægges heri! Anvendelsen af datamaten som undervisningsmiddel i forbindelse med et datamatisk undervisningsforløb adskiller sig ikke principielt fra anvendelse af ethvert andet undervisningsmiddel: tavle, kridt, det talte ord, over-head, film, bøger, video etc.

Lad os derfor fastslå at:

> UDGANGSPUNKTET ER HER EN SITUATION, HVOR MAN ØNSKER AT ANVENDE DATAMATISKE UNDERVISNINGSFØRLØB I EN FORMIDLINGSSITUATION.

Den omtalte indvending er, naturligt nok, affødt af en skræk for manglende muligheder for at kunne styre et meget stærkt medie. Bekymringen er ikke ubegrundet: datamaten er i modsætning til alle hidtil sete undervisningsmidler et autonomt generelt programmerbart værktøj, og man kunne derfor frygte, at datamaten så at sige "fik overtaget" på så store dele af undervisningen, at det var nødvendigt at lærerne siger stop.

Hvornår der skal siges stop, er i folkeskole/gymnasiesammenhæng lærerens valg via hans metodefrihed. Anderledes stiller situationen sig i forbindelse med erhvervsuddannelser. Disse forhold, af politisk karakter, falder imidlertid udenfor denne rapports rammer.

Udgangspunkt er altså en situation, hvor man har valgt og ønsker at anvende datamatiske undervisningsforløb i en formidlingssituation! Derefter er det langsigtede mål, der har

været inspirerende for denne rapport, at arbejde imod en situation, hvor undervisningsprogrammer udvikles af lærere, ved hjælp af værktøjer udviklet af dataloger.

1.5. Hvad er datalogens opgaver?

Som tidligere omtalt omfatter denne rapport undersøgelse af (1) eksisterende værktøjer, (2) datalogiske værktøjer og (3) eksisterende datalogiske værktøjer. Det er nødvendigt at opnå et teoretisk fundament for undervisningssystemer, før end der kan arbejdes videre - denne rapport giver en del af teorien.

Når den teoretiske baggrund er tilstede, kan design og implementation af faktiske værktøjer, der kan indgå i et udervisningssystem, påbegyndes.

Jeg mener at det må være datalogens opgave ud fra et mindre kendskab til undervisning at inspireres til at designe værktøjer til udvikling af datamatiske undervisningsforløb. Datalogens opgave herefter er at præsentere og informere læreren om værktøjerne. Derefter må det forventes, at læreren undersøger om værktøjerne kan bruges, og i givet fald hvordan. Det må forventes at læreren udfører forsøg med værktøjerne, kort sagt prøver dem.

-oOo-

Når man som datalog ønsker at arbejde med området EDB systemer til anvendelse i forbindelse med undervisning, er det tilsyneladende nødvendigt at træde meget varsomt. Man kan af og til få det indtryk, at der, med meget færre bekymringer og uden så megen overvejelse, udvikles og indføres andre EDB systemer, der griber lagt dybere ind i menneskers hverdag. Systemerne indføres og anvendes tillige på en sådan måde, at der ikke eksisterer noget alternativ til dem, brugeren har ikke nogen valgfrihed.

På den ene side finder jeg det vigtigt, at man som "tekni-

ker"/datalog opnår indsigt i den sammenhæng, hvor et eventuelt EDB system skal anvendes. Ligesom jeg finder det afgørende at være i stand til at tage stilling til, hvilke konsekvenser anvendelsen af systemet har for de mennesker, der er brugere af det.

På den anden side må man erkende, at problemområdet datamater og undervisning er et specielt følsomt område. Gennem mange års erfaring fra undervisning i gymnasiskolen har jeg erfaret, at lærere, måske specielt i folkeskole og gymnasie, ofte har et næstent "helligt" forhold til undervisningssituationen, hvor lærer og elever konfronteres, - dette forhold har jeg i høj grad forståelse for. Men jeg har også erfaret, at der findes så store skeptikere, at de vil tage afstand fra anvendelse af ethvert undervisningsmiddel ud over det talte ord og kridtet.

Lad mig først endnu engang slå fast, at jeg ikke ser det som min opgave i denne sammenhæng at tage stilling til, hvorledes datamatiske undervisnings forløb eventuelt kan realiceres. Men jeg ser det som min opgave på længere sigt, som datalog, at præsentere og informere lærere om datalogiske værktøjer.

Derefter føler jeg mig fristet til at anføre et citat af en unavngiven professor ved Harvard Universitet (anvendt af dr. theol. Peter Kemp, Politiken 16. marts 1986):

> "Når nogle får en ny idé her i USA siger vi til dem: 'Try it'. Vi afprøver hellere ideen hurtigst muligt med vilje til at forbedre den eller opgive den, hvis den ikke duer, fremfor at diskutere den ihjel. Men i Europa foretrækker I at diskutere jer til døde fremfor at prøve noget nyt."

1.6. Åbne <--> lukkede systemer.

Undervisningsprogrammer og undervisningssystemer kan karakteriseres ud fra en lang række kriterier. I denne sammenhæng er det af interesse at se på begrebet åbent/lukket, idet begreberne anvendes i en række forskellige betydninger, der

hver for sig er af særlig interesse i denne sammenhæng.

Betegnelsen åbent/lukket vil ofte betegne, i hvor høj grad systemet udviser dynamik. Det åbne system er det system, der udviser høj grad af dynamik, - det lukkede system er det system, der udviser ringe grad af dynamik.

Lukkede dele af virkeligheden er forudsigelige, derfor kan de beskrives fuldstændigt og defineres éntydigt. Når en opgave er lukket lader den sig formalisere og dermed udføre på en datamaskine. Da datamater altid behandler formaliserede opgaver, vil opgaver, hvis totale løsning foretages i datamaten, altid være lukkede opgaver. Hvis mere åbne opgaver søges løst, vil der gå noget tabt.

Den totale opgave er i denne forbindelse afvikling af et undervisningsforløb. Hvis hele dette undervisningsforløb tænkes gjort til et datamatisk undervisningsforløb, - dvs. at den totale løsning af opgaven i denne forbindelse foretages på datamaten, vil det kræve at den totale opgave lader sig formalisere. Dette er næppe muligt. Den totale opgave, i denne forbindelse, lader sig næppe løse på en datamat!

I det følgende er det anvendelsen, der er i fokus. Her kan der tales om, at datamaten anvendes mere eller mindre åbent/lukket.

Vi skal i det følgende se, hvorledes begreberne åbent/lukket kan anvendes i forbindelse med undervisningsforløbet, undervisningsprogrammet, undervisningsmodellen og undervisningssystemet. Der vil i nogen grad være tale om kvalitative og til dels subjektive vurderinger.

1.6.1. Undervisningsforløbet.

Betegnelsen åben/lukket kan anvendes til at betegne, i hvor høj grad et undervisningsprogram er bundet til at gennemløbe et specifikt forløb, hvor eleven således er mere eller mindre bundet til et forud fastlagt forløb (lukket). Eller modsat, i hvor høj grad undervisningsprogrammet kan anvendes som et generelt program til træning af f.eks. en given

intellektuel færdighed, hvor eleven gives mange mulige måder at arbejde med programmet på (åbent).

Det lukkede undervisningsprogram kunne f.eks. være et program af "return-typen". Dvs. et program, hvor elevens eneste synlige adfærd er, at han skiftevis læser på skærmen og trykker "return". Eller måske til sidst blot trykker "return" for at komme hurtigt videre, - deraf navnet!

Grafikdelen af LOGO kan f.eks. betegnes som et åbent system, idet der er tale om et system, der ikke formidler en specifik faglig viden, men hvor der arbejdes med et emne ("skildpaddegrafik"), der træner en mere eller mindre veldefineret intellektuel egenskab.

Eksempler på programmer, der ifølge denne opfattelse kan betegnes som åbne, er de såkaldte dialogprogrammer.*)

*) F.eks. Viggo Sadolin: MYRESNAK, SIMULER, SKOMAL, (ref. 23-25), GEO og TEGN, til anvendelse i den elementære matematik undervisning, - de to sidste under udvikling.

Dialogprogrammerne anvendes som et værktøj i undervisningen, og det er eleven, der afgør i hvilket omfang og til dels på hvilken måde, dette værktøj skal anvendes. Målet med programmerne er ikke blot at benytte dem til løsning af en konkret opgave, de skal også give eleven mulighed for let at kunne eksperimentere og efterprøve forskellige situationer. Desuden skal eleven kunne gøre antagelser og foretage gæt, hvorefter det pågældende dialogprogram skal vise, om antagelserne eller gættene stemmer overens med det faktisk opnåede resultat. Programmet skal også give eleven mulighed for at udarbejde fremgangsmåder og metoder, for derved at få afsløret, hvilke der er brugbare i en given situation. Derved bidrager programmerne til den interaktive proces, hvorved eleven får en udforskende holdning til emnerne.

Dialogprogrammerne giver således ikke eleven et på forhånd tilrettelagt undervisningsforløb, men de fungerer som et hjælpemiddel, der kan benyttes af de enkelte elever på vidt forskellig vis og i vidt forskellig udstrækning.

Imellem på den ene side de meget styrede og dermed lukkede forløb og på den anden side de meget åbne forløb, f.eks. dialogprogrammerne, findes en række mellemstadier. Endvidere kunne ét og samme undervisningsprogram, der f.eks. præsenterer en række tekstinformationer, anvendes enten med mulighed for fri søgning i materialet (åbent), - eller med en styret vej igennem materialet (lukket).

1.6.2. Undervisningsprogrammet.

Åben/lukkethed kan også ses som begreber, der knytter sig til det færdige undervisningsprogram. Det kunne opfattes som et spørgsmål om, hvor vanskeligt det ville være (for læreren) at ændre i et givet undervisningsmateriale, eller i denne forbindelse i et givet undervisningsprogram.

For det første kunne ændringerne omfatte fagligt indhold i undervisningsprogrammet. Ideen om at undervisningsprogrammet er åbent, eller i denne forbindelse snarere dynamisk, er en af ideerne bag SOFUS/COMUS systemet (se senere). Undervisningsmateriale, der er udviklet som traditionelt papirmateriale, udviser som bekendt en række niveauer, f.eks.: en håndskreven note, fotokopierede ark, kompendium ("dem med hæfteklamme i øverste hjørne") og bog. Det samme kontinuum kunne være ønskeligt ved udvikling af undervisningsprogrammer. Dette skulle foregå i en erkendelse af, at det er en proces at udvikle undervisningsmateriale, enten det drejer sig om papirmateriale eller datamatiske undervisningsforløb. Der vil således være behov for programmer, der understøtter en sådan dynamisk udvikling af undervisningsprogrammer.

For det andet kunne dynamiske muligheder i selve undervisningsprogrammet omfatte ændring af brugergrænsefladen. Der vil f.eks. være behov for at ændre brugergrænsefladen, når programmet anvendes af elever med større eller mindre erfaring med det pågældende program, - eller med programmer af lignende type. Individuelle ønsker og krav, der knytter sig til den enkelte elev, kan medføre ønske om ændring af brugergrænseflade. Det kunne f.eks. dreje sig om ønsker med hensyn til farver, menustruktur, kommandoer, mulighed for udpegning f.eks. ved hjælp af mus, snarere end tast-komman-

doer. Endvidere vil anvendelse af programmet til andre formål kunne give anledning til et ønske om ændret brugergrænseflade, f.eks afhængig af om eleven følger et styret forløb eller ej gennem materialet.

Der kunne f.eks. også være ønske om ændring af brugergrænseflade i det tilfælde, hvor et undervisningsprogram skal anvendes af forskellige organisationer, der måske har individuelle krav til brugergrænsefladen, der kræves overholdt.

1.6.3. Undervisningsmodellen.

Den model af et undervisningsforløb, der kan bearbejdes med undervisningssystemet, kan være mere eller mindre lukket, det vil i denne forbindelse sige afgrænset. Der vil dog altid findes visse muligheder indenfor modellens område, om end undervisningsforløbene altid vil være begrænset af den givne model. Arbejdes der med en meget begrænset model for et undervisningsforløb, vil mulighederne for konstruktion af undervisningsforløb være stærkt begrænset af modellens rækkevidde. Der vil f.eks. ofte være tale om en begrænset model, hvis der anvendes et undervisningssystem som f.eks. et forfattersystem eller en forfattermodel (se herom senere).

Det vil oftest anses for ønskeligt, at der er store dynamiske muligheder i undervisningsmodellen, dvs. at den skal være meget åben.

Der vil dog altid være større eller mindre begrænsninger i den model, - der kan bearbejdes af et givet undervisningssystem. En erkendelse heraf kan give anledning til det synspunkt, at underviseren bør have en række modeller med tilhørende undervisningssystemer til sin rådighed. Dette synspunkt deles f.eks. af udviklerne af SOFUS/COMUS systemet.

Ved hjælp af undervisningssystemet konstrueres undervisningsprogrammet, der realiserer en given model. Derfor er de muligheder og begrænsninger, der findes i modellen helt afhængige af de muligheder og begrænsninger, der findes i

undervisningssystemet.

En anden måde at angribe problemet på er derfor at konstruere et undervisningssystem, der er designet til at definere og behandle forskellige modeller.

Denne løsning må anses for en tiltalende, men avanceret, datalogisk løsning, - sådanne systemer findes i praksis ikke i dag. Vi skal senere udvikle en teori for systemer, der omfatter denne mulighed.

1.6.4. Undervisningssystemet.

Begrebet åben/lukkethed kan knyttes sammen med programmerne til udvikling af undervisningsprogrammer, dvs. undervisningssystemerne. Spørgsmålet er her, i hvor høj grad et programmelsystem skal ses som en fastlåst enhed, eller i hvor høj grad det vil være muligt at foretage større eller mindre ændringer eller udvidelser til systemet.

Er systemet lukket, er man i givet fald bundet til udvikling af undervisningsprogrammer med det eksisterende system. Det ville være ønskeligt om programmerne så frit som muligt kunne udbygges, ændres, omstruktureres eller lignende. Dette ønske strider desværre ofte direkte imod kommercielle interesser, - i praksis vil man ofte have "solgt sin sjæl" til leverandøren af det programmel, der anvendes til udviklingen.

Ønsker om ændring af undervisningssystemet kunne f.eks. omfatte ændringer i de operationer, der kan udføres på modellen af undervisningsforløbet, realiseret i undervisningsprogrammet. Eller der kunne f.eks. være et ønske om ændret brugergrænseflade.

Vi skal, som tidligere nævnt, senere se en teori for en begrebsmæssig ramme for et system, der indebærer en stor grad af åbenhed/dynamik.

På figur 1-1 ses en kvalitativ oversigt over begreberne åbent/lukket, som de har været omtalt i forbindelse med

undervisningsforløb, undervisningssystemer, undervisningsmodeller og undervisningsprogrammer:

	ÅBEN	LUKKET
FORLØBET	Frihed for eleven til at anvende programmet efter eget valg. F.eks. dialogprogrammer.	Anvendelsesmåden er fastlåst. F.eks. "return"-programmer.
PROGRAMMET	Simpelt for udvikleren f.eks. læreren at ændre programmet. F.eks ændring af brugergrænseflade eller materialets niveau.	Vanskeligt at ændre i undervisningsprogrammet
MODELLEN	Modellens struktur kan vælges frit. Generelle programmeringssprog eller forfattersprog giver i nogen grad mulighed herfor.	Afgrænset model. Modellens struktur er fastlagt af undervisningssystemet. F.eks. ved anvendelse af forfattersystemer, drivere eller modeller.
SYSTEMET	Undervisningssystemet kan ændres f.eks. med hensyn til brugergrænseflade eller mulige operationer på modellen.	Forfatterredskaber, begrænsede kommercielle systemer.

Figur 1-1: Oversigt: åbne <--> lukkede systemer.

Indledning.

1.7. Undervisningsforløbet som EDB-model?

Når der skal konstrueres en datamatisk realisation af et undervisningsforløb, dvs. undervisningsprogrammer, er situationen efter min mening markant anderledes end den "traditionelle" situation, hvor datamater anvendes til løsning af f.eks. administrative eller beregningsmæssige problemer.

Disse forskelle gør sig gældende indenfor en række områder, der knytter sig til den traditionelle udvikling og anvendelse af datamatiske systemer. Man skal imidlertid ikke tage fejl af, at der naturligvis er en række lighedspunkter imellem udvikling af undervisningsprogrammer og "traditionelle" EDB-systemer. Men jeg vil påstå, at der er en markant forskel i problemernes størrelsesorden.

Dette forhold vil i det følgende antydes gennem en række kvalitative vurderinger. Men det er et område, der burde arbejdes yderligere med, hvis man ønsker at undersøge undervisningsmodeller og undervisningsprogrammer i detaljer. I denne rapport er hovedemnet, som bekendt, undervisningssystemer.

(1) I mange typiske klassiske situationer, hvor EDB modeller udarbejdes, vil situationen være den, at der findes et manuelt system, som man mere eller mindre direkte ønsker at omforme til et datamatisk system. Naturligvis vil der være behov for en række større eller mindre logiske omformninger af systemet, før den endelige logiske model for det datamatisk system nås. Derefter kan implementationen i form af et EDB-system evt. foretages.

Situationen i forbindelse med et undervisningsprogram er imidlertid markant anderledes. Her vil man normalt netop ikke have noget ønske om mere eller mindre direkte at overføre en manuel model til en elektronisk løsning. Når datamater anvendes i undervisningsforløb, vil man netop ofte ønske at udnytte datamatens specielle muligheder i denne forbindelse. Den periode, hvor nogen tør påstå, at der vil være noget som helst ønskværdigt i en undervisningssituation, der

er en mere eller mindre direkte datamatisk udgave af det klassiske undervisningsforløb, er vel nok forbi!

(2) De mulige modeller, der f.eks. kan konstrueres af et undervisningforløb, vil udvise afgørende forskelle, der griber dybt ind i bl.a. struktur og logisk opbygning fra model til model.

For traditionelle systemer findes der naturligvis også flere mulige modeller. Men der vil ikke ses en så markant forskel fra model til model, som det ses i forbindelse med undervisning.

(3) Situationen at den logiske model, der anvendes i den datamatiske løsning, skal omarbejdes mere eller mindre, er velkendt fra traditionelle datamatiske løsninger. Årsagen til en sådan ombearbejdning kunne f.eks. være ændrede brugerkrav og ønsker, eller det kunne f.eks. skyldes en ændring af ydre omstændigheder for et administrativt system.

Kravet om ændring på modellen i forbindelse med undervisning er imidlertid af en række forskellige årsager langt større. F.eks. findes der, som tidligere omtalt, simpelthen en langt større variation af mulige modeller. Der vil også være et ønske om ændring af model ud fra individuelle hensyn til elever eller ud fra pædagogiske overvejelser i almindelighed. Endvidere spiller en række forhold omtalt under åbne <--> lukkede systemer ind.

(4) Ideen om at slutbrugeren medvirker ved udvikling af traditionelle EDB-systemer er velkendt. Denne medvirken anses ofte for gavnlig, idet det medvirker til, at der kan designes en model, der svarer til brugerens vision om et EDB-system.

Igen er forholdet principielt det samme for et undervisningsforløb, men der er dog efter min mening også her tale om en forskel i størrelsesordenen af problemet. Udvikling i tæt samarbejde med brugere, og det vil her sige både forfattere til datamatiske undervisningsforløb og eleverne, synes endnu mere påkrævet.

Der er jo her tale om systemer, der ikke blot skal udføre en konkret afgrænset og veldefinerbar funktion. Der er tale om systemer, der skal indgå i interaktion med mennesker. Endda indgå i en situation, der måske kunne betegnes: kommunikation imellem menneske og undervisningsprogram; - selvom begrebet kommunikation ikke rigtig synes at kunne anvendes i forbindelse med forholdet imellem en maskine og et menneske.

Det synes derfor endnu mere accentueret, at enhver udvikling eller anvendelse af datamatiske undervisningsforløb bør tage udgangspunkt i en analyse og vurdering af situationer, hvor mennesker har arbejdet med systemerne, og hvor der er udført anvendelsesforsøg i større eller mindre udstrækning.

Vi har således set en antydning af, at der findes markante forskelle imellem traditionel systemudvikling og udvikling af datamatiske undervisningsforløb. Vi har omtalt (1) variationen af mulige modeller, (2) vejen fra manuelt system til elektronisk model, (3) mulighed for ændring af modellen samt (4) brugerindflydelse på systemudvikling.

-oOo-

Behovet for velkonstruerede undervisningssystemer er, som vi har set, bl.a. affødt af følgende faktorer:

Krav om volumen med hensyn til undervisningsprogrammer.

Ønsket om at klargøre linien imellem datalog og lærer ved udviklingen.

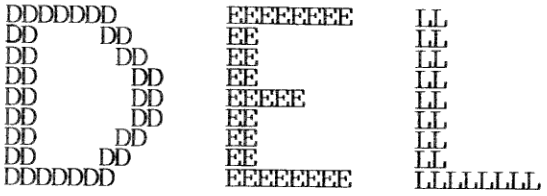
Ønske om hurtighed ved udvikling af programmer, hvorved cyclus for udvikling <-> brugerafprøvning afkortes.

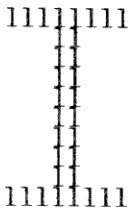
Ønsket om åbenhed med hensyn til undervisningsforløbet, programmet, modellen og systemet.

De særlige forhold der synes at gøre sig gældende i forbindelse med datamatiske undervisningsforløb.

Vi skal i DEL 1 se mulighederne med eksisterende undervisningssystemer.

Indledning.





E K S I S T E R E N D E V Æ R K T Ø J E R

2. FORFATTERREDSKABER

I dag er de eneste udbredte undervisningssystemer, dvs. programmel til støtte ved udvikling af undervisningsprogrammer, de såkaldte forfatterredskaber.

Ved udvikling af undervisningsprogrammer har man således i dag mulighed for at anvende et generelt programmeringssprog eller et forfatterredskab.

Ved forfatterredskaber vil jeg forstå:

1. Forfattersprog
2. Forfattersystemer

I dette kapitel introduceres disse begreber og dansk udviklede forfatterredskaber omtales kort.

2.1. Generelle programmeringssprog.

De generelle programmeringssprog kan med deres generelle egenskaber naturligvis også anvendes til udvikling af undervisningsprogrammer.

Tidligere er omtalt et vigtigt synspunkt i denne rapport, nemlig at undervisningsprogrammer bør udvikles af lærere. Når dette ikke altid bliver tilfældet, er årsagen ofte den, at de ikke har tilstrækkelig baggrund, - idet systemerne til udvikling af undervisningsprogrammer kræver stort EDB-kendskab på systemniveau.

Svagheden ved de generelle programmeringssprog er således, at de kræver, at udvikleren besidder forholdvis megen EDB indsigt på systemniveau, - idet undervisningsprogrammet bygges op helt fra grunden. Der er ikke nogle færdige strukturer, der kan bygges på. Systemerne bliver dermed også forholdsvis langsomme at arbejde med ved udvikling af undervisningsprogrammer. Disse svagheder vil bl.a. fremgå af et projektforløb i Danmarks Sparekasseforening, som senere beskrives.

Deres styrke vil også fremgå af dette projekt: De giver i teorien stor frihed med hensyn til hvilken model af et undervisningsforløb og dermed hvilket undervisningsprogram, der vil kunne implementeres. Men i praksis vil man stå overfor det problem at skulle opbygge undervisningsprogrammet fra bunden hver gang. Anvendelsen af generelle programmeringssprog vil derfor være resourcekrævende på flere områder, idet anvendelsen bl.a. vil medføre: store omkostninger, stort tidsforbrug, krav om højt EDB videns niveau for udvikleren.

Situationen er i dag, at stort set alle de undervisningsprogrammer, der udvikles til anvendelse i folkeskole og gymnasie, er udviklet ved hjælp af et generelt programmeringssprog. Mest udbredt er sproget Comal80 og sjældent f.eks. Pascal eller "C".

Et specielt problem gør sig gældende i denne sammenhæng. Forfatterredskaber, bortset fra de generelle programmeringssprog, kræver at brugeren af undervisningsprogrammerne er i besiddelse af det pågældende undervisningssystem. Dette er ikke tilfældet for de oversatte højniveau sprogs vedkommende, hvor der ikke er royalty på køretidsmodulet, som er den eneste del af systemet, der følger med det oversatte program. I tilfældet et fortolkende sprog, som for det danske markeds vedkommende helt er domineret af Comal80, kræves det at brugeren er i besiddelse af systemet. Men brugerne i skolesektoren vil faktisk altid være i besiddelse af et Comal system, - men derefter melder problemet sig med de forskellige Comal dialekter!

Begrundelsen for anvendelsen af f.eks. Comal80 tager som oftest udgangspunkt i, at det er et sprog, der nu har fået stor udbredelse og dermed sin naturlige plads i undervisningssektoren. Comal80 er derfor det sprog, der beherskes af hovedparten af programudviklerne, nemlig lærerne.

Comal80 er et fortolkende sprog, som Basic, og derfor forholdsvis langsommeligt, uden mulighed for beskyttelse af kildetekst. Comal80 giver et ret simpelt udviklingsmiljø uden de muligheder for smidig anvendelse af f.eks. tekstediting, stykvis opbygning af programmet ved hjælp af link-

Forfatteredskaber

ning etc., som kendes fra andre oversatte højniveausprog. Samtidig savnes en række af de kvaliteter, der er kendt fra sprog som f.eks. Pascal eller "C".

En begrundelse for anvendelsen af et sprog som Comal80 er imidlertid, at det er nemt at lære og forstå, og iøvrigt med manualer på dansk. Sprogets store udbredelse bevirker, at der er et stort udbud af dansksproget litteratur om sproget. Det er nemt at få undervisning i sprogets anvendelser, og det er let at få hjælp, dels fra kollegaer dels fra elever (!). Endvidere er det faktisk muligt for undervisningsinstitutioner at få hjælp fra maskinleverandøren, - hvilket må betegnes som en ekstraordinær situation.

En begrundelse for anvendelse af Comal80 er f.eks., at der umiddelbart er mulighed for at ændre i programmet. En ændring der kan foregå i et programmeringsmiljø, der er velkendt for elever - og lærere. Det kunne f.eks. være af interesse at kunne ændre i en økonomisk model eller lignende.

Ud fra datalogiske vurderinger ville man imidlertid sjældent vælge sproget, - hvis det da ikke var for begrundelsen om, at slutbrugeren skulle kunne ændre i programmet. Denne påstand underbygges af, at programmer udviklet af EDB fagfolk til anvendelse i undervisningssammenhænge sjældent er udviklet i sprog som f.eks. Comal80.

2.2. Forfattersprog og forfattersystemer.

Ved konstruktion af mere omfattende undervisningsprogrammer viser erfaringerne (ref. 16 og 17), at det generelle programmeringssprog ikke slår til. Som det bl.a. er erkendt ved praktiske forsøg, er der en meget lav produktivitet med det generelle programmeringssprog (ref. 16).

Indenfor industrien, hvor der tilsyneladende er et mere kontant krav om hjælpeprogrammer til udvikling af undervisningsprogrammer, med den større produktivitiet af undervisningsprogrammer til følge, er der gjort en række tiltag for anvendelsen af forfatterredskaberne: forfattersprog og for-

fattersystemer.

Senere beskrives i detaljer udviklingen indenfor sparekassesektoren: Danmarks Sparekasseforening, SDS og Bikuben. Indenfor storbankerne er der også en række aktiviteter i-gang, hvor der anvendes forfatterredskaber til udvikling af undervisningsprogrammer.

Problemerne omkring royalty er vel nok mindre i forbindelse med anvendelse i erhvervsvirksomheder, da programmerne ofte vil være udviklet specielt for den pågældende virksomhed under anvendelse af ét system.

I det følgende beskrives begreberne forfattersprog og forfattersystem. Derefter beskrives et par dansk udviklede forfatterredskaber:

2.2.1. Forfattersprog.

Ved et forfattersprog forstås et system, der indeholder et sprog med en række udvidelser set i forhold til et generelt programmeringssprog. Udvidelserne omfatter programordrer på et højere niveau, således at den specielle anvendelse til udvikling af undervisningsprogrammer understøttes.

Den række højniveauordrer som sproget er udvidet med kunne f.eks. omfatte: understøttelse af opgavekonstruktion, afvikling af et undervisningsforløb, eller særlige grafiske muligheder.

Ofte vil forfattersproget også omfatte muligheder for en strukturering, der direkte afspejler den struktur den endelige præsentation af data udviser. Nøgleordssøgning er også en vigtig facilitet, der f.eks. vil indgå i opgavebedømmelse.

I løbet af 70'erne er der udviklet en lang række forfattersprog, og der findes idag hundreder af forskellige forfattersprog (ref. 21).

Ønsket om at mere ukyndige skulle være i stand til at udvik-

le undervisningsprogrammer, har fra begyndelsen af 80'erne bevirket en udvikling af forfattersystemer:

2.2.2. Forfattersystemer.

Ved et forfattersystem forstås et system, der vil understøtte en mere fastlåst undervisningsmodel. Systemet leder udvikleren igennem forløbet, f.eks. ved hjælp af menuer eller lignende. Det gør systemet lettere tilgængeligt for den uerfarne udvikler.

Andre betegnelser for forfattersystemer er f.eks. "DFU-drivere", "drivere", "forfatter-modeller", "DFU-editorer" etc.

Forfattersystemerne vil give anledning til, at der anvendes en bestemt model af de undervisningsforløb, der kan udvikles med det pågældende system. Systemerne kan derfor betegnes som forholdsvis lukkede. Grænsen imellem forholdsvis lukkede forfattersystemer og forholdsvis åbne forfattersprog er imidlertid mere eller mindre flydende.

Det er vigtigt, at man ikke blot vurderer disse programmer ud fra spørgsmålet om "hvad de kan", - men også ud fra "hvor let er det". Det er netop dette punkt, der giver anledning til et kompromis imellem på den ene side et generelt programmeringssprog, hvor "man kan alt", men hvor "det er vanskeligt", - og på den anden side et forfattersystem, hvor man "ikke kan så meget", men hvor "alt er let".

2.2.3. Danske forfatterredskaber.

Der findes kun ganske få dansk udviklede forfatterredskaber, f.eks.: SOFUS/COMUS, MicroTutor samt DUS/ComCats systemet.*)

Her følger en kort baggrundsbeskrivelse for SOFUS/COMUS, MicroTutor og DUS/ComCats systemerne, - SOFUS/COMUS vil senere findes grundigt beskrevet:

Forfatteredskaber

*) SOFUS/COMUS er udviklet af konsulentfirmaet Jensen & Jørgensen & Knudsen & Sørensen og markedsføres af IBM samt International Computers Limited (ICL) under navnet COMUS. MicroTutor er udviklet af Ivan Boserup og Jørgen Feder (Museum Tusculanums Forlag) og DUS/ComCats af Jydsk Telefon A/S og Århus Tandlægehøjskole.

2.2.3.1. SOFUS/COMUS.

SOFUS/COMUS kan betegnes som et forfattersystem i overensstemmelse med den tidligere definition.

SOFUS/COMUS er et dansk udviklet system, der leveres med alle Comet mikrodatamater, der leveres til offentlige undervisningsinstitutioner i Danmark.

Til IBM-PC kompatible maskiner forhandles systemet under navnet SOFUS.

JJKS der har udviklet SOFUS/COMUS har erfaringer efter arbejde som konsulenter i større projekter, bl. a. i samarbejde med sparekassesektoren og fra en række TUP-projekter (Teknologisk Udviklings Projekter).

2.2.3.2. MicroTutor.

MicroTutor kan betegnes som et forfattersprog, men det har også en række af de styringsfaciliteter, der er karakteristiske for et forfattersystem. Det er under alle omstændigheder forholdsvis let at arbejde med.

MicroTutor har sin oprindelse i et projekt i forbindelse med kurset "Datalogisk Praktik" ved Datalogisk Institut, Københavns Universitet, hvor Jørgen Feder i 1982 skulle udvikle et system til afvikling af øvelser i latin i forbindelse med en ny latinbog af Ivan Boserup. Ud fra en datalogisk tilgang til problemet valgte Jørgen Feder imidlertid den generelle løsning. Der blev således udviklet et generelt system, et forfatterredskab/forfattersystem, der senere

skulle udvikle sig til MicroTutor systemet. Ref. 14 indeholder en beskrivelse af systemet.

2.2.3.3. DUS.

DUS er et forfattersprog, men der kan arbejdes med en såkaldt "konverserende editor", således at der genereres DUS-kode, men ved menu-styring.

DUS står for Dataformidlet Undervisnings System. Udviklingen af systemet og etableringen af DUS-organisationen tager sit udgangspunkt i et samarbejde imellem Århus Tandlægehøjskole og JTAS.

Begge parter startede i begyndelsen af 70'erne forsøg med at overføre en del af undervisningen til datamater. I første omgang den faktabaserede undervisning med enkle svarmuligheder. Senere ønskede begge parter også at kunne skabe mere nuanceret undervisning. Man indgik derfor et samarbejde om udvikling af et forfattersprog. Ideén var samtidig at udvikle et system, der kunne anvendes af udviklere af undervisningsprogrammer, som ikke havde større EDB indsigt på systemniveau.

2.2.3.4. ComCats.

ComCats (The Complete Computer Assisted Training Systems) er en 16-bits videre udvikling af DUS-1, der er implementeret på Regnecentralens Piccolo (8-bit).

De to initiativtagere oprettede DUS-fonden, der bl.a. havde til formål at formidle erfaringerne til den offentlige uddannelsessektor. Aktieselskabet DUS-International stiftedes af DUS-fonden sammen med Danfoss A/S og Kirkbi A/S. Selskabet er i dag ophørt. DUS/ComCats markedsføres i dag af Århus Tandlægehøjskole.

Et senere kapitel indeholder bl.a. en række primærerfaringer med forfatterredskaber, idet forfattersproget TenCore og forfattersystemet SOFUS/COMUS undersøges.

Forfatteredskaber

3. ERFARINGER MED KENDTE VÆRKTØJER.

I dette kapitel præsenteres en række af de primær erfaringer, jeg har gjort med eksisterende programmer til udvikling af undervisningsforløb på datamat.

Primær erfaringerne omfatter erfaringer med de i dag mest udbredte programmer til hjælp ved udvikling af undervisningsprogrammer, nemlig forfatterredskaberne. Jeg har arbejdet med to konkrete systemer, der anvendes på mikrodatamater.

Fårmålet er at give en beskrivelse af de to programmer, - ikke at foretage en egentligt vurdering af dem.

Før end beskrivelsen af de to konkrete systemer skal vi se en redegørelse for et faktisk projektforløb, der omhandler indførelse af datamatformidlet undervisning (DFU) i Danmarks Sparekasseforening.

Denne projektbeskrivelse er valgt som indledning i kapitlet, idet en række af de problemer, der findes omkring programmer til udvikling af undervisningsprogrammer, mødes i dette projekt.

Denne "case", hvor der i praksis er arbejdet med anvendelse af forfatterredskaber, vil således udgøre en del af baggrunden for beskrivelserne af de to undersøgelser af forfatterredskaber.

3.1. Et projektforløb: Datamatformidlet undervisning.

I dette afsnit skal vi se på et konkret projekt: indførelse af datamatformidlet undervisning i de danske sparekasser. Projektet er også beskrevet, ud fra en anden synsvinkel, i ref. 9. Primær materialet findes i ref. 16 og 17.

Oplysningerne stammer bl.a. fra referater af foredrag afholdt af medarbejdere i Danmarks Sparekasseforening, samt fra samtaler med disse medarbejdere.

3.1.1. Projektbaggrund.

Projektet er gennemført som et samarbejde imellem fire grupper: de to største sparekasser, nemlig (1) SDS og (2) Bikuben, (3) Danmarks Sparekasseforening, der omfatter en række mindre sparekasser og endelig (4) Sparekassernes Datacenter (SDC). Endvidere er der til projektet knyttet eksterne konsulenter.

Man havde alt i alt store pædagogiske og organisatoriske forventninger til anvendelse af undervisningsprogrammer. De økonomiske rammer for hele projektet er ca. 3-4 mill. kr.

Der er en række årsager til, at man finder interesse i dataformidlet undervisning, jeg vil ikke gå i detaljer med alle disse aspekter her, men blot nævne nogle af nøglebegreberne: Stigende behov for instruktion og uddannelse i forbindelse med stigende krav om:

1) større og bredere viden hos medarbejderen
2) hurtig tilpasning og fornyelse
3) individualisering
4) fleksibilitet
5) decentralisering

Endvidere spiller økonomiske overvejelser også en rolle. Samtidig er man i en situation, hvor sparekasserne kaster sig over nye virkefelter, hvilket er med til at accentuere ovennævnte forventninger.

Jeg vil her koncentrere mig om de aspekter af forløbet, der omhandler valg af programmer til udvikling af undervisningsforløb.

3.1.2. Det første forsøg.

Man besluttede i efteråret 1981 at gennemføre et forsøg, der bl. a. havde til formål at undersøge forbruget af resourcer i forbindelse med udvikling af undervisningsprogrammer. Det vil sige forbrug af tid, omkostninger og viden i forbindel-

se med udviklingen. Endvidere ønskede man forsøgsvis at udvikle et undervisningsmateriale.

Forsøget kom bl. a. til at omfatte: Uddannelse af medarbejdere til udvikling af undervisningsprogrammer, et anvendelses forsøg med de udviklede programmer. Tekniske undersøgelser af det anvendte udstyr.

I første fase valgte man at anvende mikrodatamater, - muligheden for at anvende det eksisterende terminalsystem viste sig ikke farbar. Dette skulle senere vise sig til fordel for projektet, idet man nu kunne gå den "rigtige" vej og starte med at finde egnet programmel, for derefter at finde egnet maskinel.

Man valgte at anvende generelle programmeringssprog, idet man fokuserede på deres "næsten totale frihed", - som man udtrykte det. Man anvendte Comal80 og Pascal.

Problemet med de generelle programmeringssprog var, at de stillede store krav med hensyn til EDB-viden på systemniveau til de folk, der skulle anvende dem til udvikling af undervisningsprogrammer.

Man måtte erkende afvejningen imellem på den ene side "friheden ved det generelle programmeringssprog", og på den anden side "den forholdsvis ringe produktivitet" dette redskab giver.

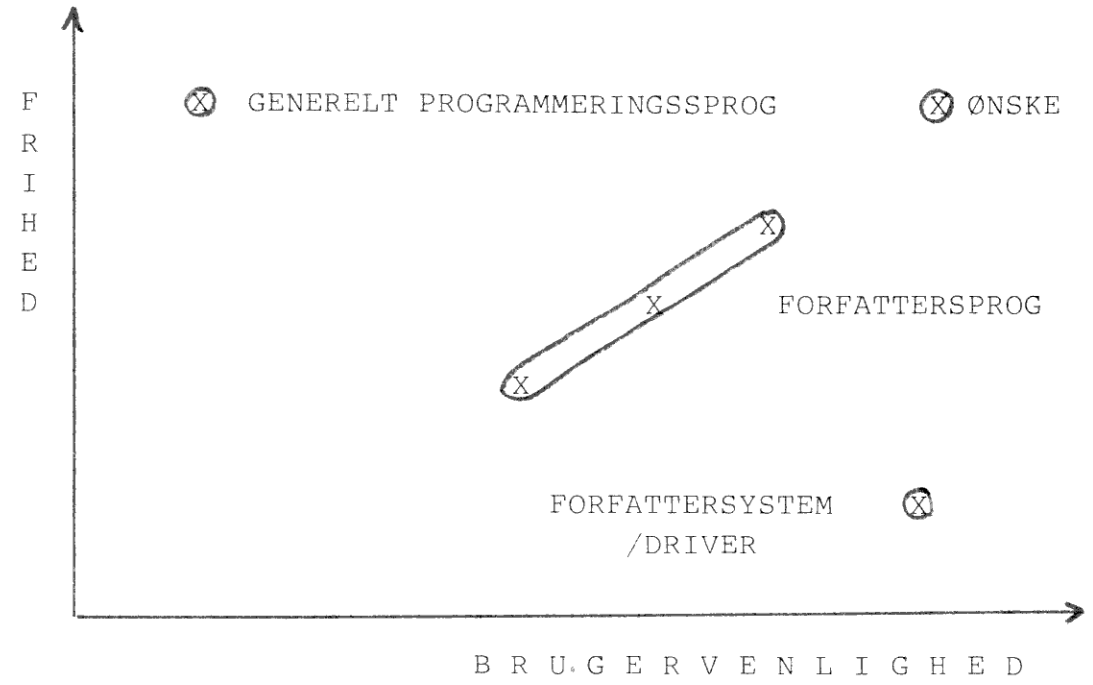
3.1.3. **Første konklusion.**

Man konkluderer, at der bør anvendes programmel, der indeholder en række overbygninger til det generelle programmeringssprog, men som på den anden side er åben for videreudvikling af nye værktøjer!

3.1.4. **Næste skridt.**

Man fandt, at det generelle programmeringssprog gav, hvad man betegnede "pædagogisk frihed". Man så situationen som

"udviklerens dilemma", figur 3-1.



Figur 3-1: Udviklerens dilemma: frihed <---> brugervenlighed.

Brugervenlighed bruges her som målestok for, hvor tilgængeligt et system er for udvikleren.

Man ønskede ikke at give afkald på, hvad man betegnede som "den pædagogiske frihed og fleksibilitet, som det generelle programmeringssprog giver".

Ved det fortsatte arbejde ønskede man at undersøge markedet med henblik på forfatterredskaber eller andre udviklingsprogrammer, samt mulighederne for en eventuel videreudvikling.

Som det fremgår, havde man umiddelbart to krav til et forfatterredskab: det skulle være forholdsvis lettilgængeligt for udvikleren og man ønskede at det skulle give mulighed for, hvad man betegnede som store "pædagogiske frihedsgrader".

Erfaringer med kendte værktøjer

Man stillede endvidere krav til systemet om, at det skulle understøtte grafik (streggrafik), animation, beregninger samt farver.

3.1.5. Konklusion.

En afgørende konklusion var, at "man måtte indgå et kompromis imellem - på den ene side pædagogisk frihed og på den anden side tilgængelighed med hensyn til udvikling", - svarende til et kompromis imellem på den ene side et generelt programmeringssprog og på den anden side et forfattersystem.

3.1.6. Undersøgelse af program muligheder.

For at undersøge eksisterende programmelmuligheder og for at få kendskab til primær erfaringer med DFU, foretog man i efteråret 1983 en studierejse til USA; - fra studieturen drages følgende konklusioner, der stort set er en bekræftelse på de erfaringer, man allerede havde gjort:

1. Erfaringer i USA havde vist, at anvendelse af et generelt programmeringssprog ikke er en farbar vej.

2. Man fik bekræftet konklusionen om, at der er tale om et kompromis imellem på den ene side systemer med store frihedsgrader og på den anden side systemer med stor tilgængelighed for udvikleren.

Valget faldt på TenCore systemet.

Det endelige valg faldt på det redskab, der er betegnet som en kompromisløsning, nemlig TenCore systemet, der er et forfattersprog, med de fordele og ulemper det måtte indebære.

3.1.7. Projektets nuværende status.

De tre projektgrupper: Bikuben, SDS og Danmarks Sparekasseforening (DS) påbegyndte nu den egentlige udvikling af DFU

materiale ved hjælp af TenCore systemet.

Ud fra ønsket om en hurtigere produktion af undervisningsprogrammer, designet indenfor rammerne af en given model, går man herefter to veje. Bikuben og SDS udvikler, ved hjælp af TenCore, en model med hvilken der kan produceres programmer, - indenfor modellens rækkevidde. (ref. 8).

Udvikling af en model er for det første meget tidskrævende, og for det andet er det et spørgsmål, om den udviklede model udgør en sådan model af det datamatiske undervisningsforløb, som man faktisk ønsker! Dette vil først vides, når der er gjort brugererfaringer med undervisningsprogrammer. Under alle omstændigheder medfører anvendelsen af en model, uanset dennes struktur, begrænsninger i de mulige programmer, der kan udvikles med modellen, sammenlignet med de programmer, der kan udvikles med selve TenCore systemet.

Danmarks Sparekasseforening vælger en anden fremgangsmåde: ud fra det første program udviklet i TenCore udtrækkes et "ramme-program", kaldet "Kursus 00" (ref. 26), som indeholder alle kontrolstrukturer, menustrukturer, skærmbilledopbygning etc., - men ikke det rent faglige indhold. Derefter er det muligt på en meget begrænset tid at "fylde" andre faglige informationer ind i rammen. Derved produceres et program under anvendelse af præcis den samme model! Men i praksis på meget kort tid. Det giver mulighed for ikke blot at spare resourcer, men også mulighed for hurtigere at få flere kørende undervisningsprogrammer med henblik på indsamling af flere anvendelses erfaringer.

I løbet af 1985 har man gennemført omfattende anvendelsesforsøg i samarbejde med RECAU (Erik Meistrup og Kurt Nikolajsen), ref. 1. Anvendelsesforsøgene skal ikke nærmere omtales her.

Vi vender os nu imod beskrivelserne af to konkrete forfatterredskaber:

Erfaringer med kendte værktøjer

3.2. Mikrodatamat forfatterredskaber.

Som eksempel på to forfatterredskaber til mikrodatamater, har jeg arbejdet med forfattersproget TenCore og forfattersystemet SOFUS/COMUS.

3.2.1. Forfattersprog: TenCore.

TenCore systemet, der første gang blev markedsført i 1983, er udviklet af Computer Teaching Corporation (CTC) i samarbejde med Courseware Applications, USA. Manden bag systemet er Poul Tenczar, der bl.a. har arbejdet på PLATO projektet, - TenCore har da også store ligheder med PLATO systemet. (TenCore forhandles i Danmark af Courseware Scandinavia.)

Systemet må betegnes som et forfattersprog, idet systemet er bygget op omkring et særligt TenCore sprog. Der er tale om et omfattende system, der viser sig at være forholdsvis kompliceret at anvende ved udvikling af undervisningsprogrammer, og som derfor kræver en del af programmøren. Systemet omfatter således f.eks. ca. 600 sider dokumentation.

Der anvendes en noget uvenlig brugergrænseflade, idet de fleste systemkommandoer styres v.h.a. funktionstaster.

Man kan bl. a. vurdere et forfattersprog på dets åbenhed-/lukkethed. Vurderet på denne måde er TenCore et åbent system (ref. 9), der er en vej ud af systemet. Systemet indeholder nemlig et sprog, der bl. a. omfatter en række, om end simple, ordrer der giver de samme muligheder som et generelt programmeringssprog, i princippet. Problemet er, at man fortsat er bundet til dette simple sprog og dermed til systemet som sådan.

3.2.1.1. TenCore sproget.

Sproget anvender en assemblerlignende syntaks, med et ordreord og dertil hørende parametre, - som det ses af figuren: "Eksempel på TenCore kildetekst". Det savner et moderne programmeringssprogs styrke og elegance, f.eks. en egentlig

strukturering. Det er således forholdsvis tungt at arbejde med.

Eksempel på TenCore kode kan ses i figur 3-2. Linienumrene er ikke en del af TenCore koden, men er tilføjet i figuren, således at der kan knyttes kommentarer dertil i denne forbindelse.

Struktureringen foregår ved at koden kan opdeles i nogle blokke (UNITs), der imidlertid primært er knyttet til den endelige afvikling. Således kan en blok f.eks. definere et enkelt skærmbillede, evt. en række billeder, en udvikling af et skærmbillede (f.eks. med animation) eller en opgave. Figur 3-2 indeholder kode hørende til én unit.

Under afviklingen af programmet kan brugeren ved tryk på en funktionstast styre programafviklingen, således at der hoppes direkte til en anden blok. Sproget understøtter denne brug af funktionstaster, som det ses i linierne 01 - 05, idet f.eks. "next" og "NEXT" refererer til en funktionstast henholdsvis shiftet og ikke shiftet. F. eks. ved tryk på "help" funktionstasten (linie 04) hoppes der til en unit "HELPUNIT", der kunne vise hjælpetekst på skærmen.

```
00        *-----------UNIT DEMO------------
01        BACK      START
02        next      UNIT2
03        NEXT      UNIT2
04        help      HELPUNIT
05        HELPOP    HELPUNIT
06        SCREEN    MEDIUM COLOR
07        COLORB    BLACK+
08        COLOR     BROWN+
09        SIZE      2
10        AT        5:5
11        WRITE     Famous Musicians
12        SIZE      1
13        AT        12:5
14        WRITE     Name one composer of H.M.S. Pinafore!
15        ARROW     14:5
16        OKEXTRA
```

Erfaringer med kendte værktøjer

```
17      OKSPELL
18      PUTLOW
19      ANSWER    (Gilbert,Sullivan) <and,or>
20      .         WRITE     Yes, very good!
21      ANSWER    Gilbert <and> Sullivan
22      .         WRITE     Fantastic! You know them both
23      WRONG     (Rodgers,Hammerstein) <and,or>
24      .         WRITE You are a few decades off
25      NO
26      .         WRITE     Hint: G.....t and S......n
27      ENDARROW
```

Figur 3-2: Eksempel på TenCore kildetekst.

Det der gør, at systemet udmærker sig som et forfattersprog, er en række specielle programordrer, der er specielt anvendelige til udvikling af undervisningsprogrammer. Udvidelserne kan, set i forhold til det generelle programmeringssprog, opfattes som en række højniveau ordrer, der er tilføjet det generelle programmeringssprog.

Udvidelserne omfatter bl.a. ordrer til håndtering af skærmen: der kan skrives med tegn i forskellig størrelse (linie 09 og 12) og på et bestemt sted (f.eks. linie 10), farver kan styres (linie 06 - 08).

Især findes ordrer til håndtering af opgaver og elevsvar, dvs. svarbedømmelse:

3.2.1.2. Svarbedømmelse.

Sproget indeholder en række ordrer til håndtering af den såkaldte svarbedømmelse. Der anvendes en særlig "svar-markør", nemlig en pil. Den er en klarmelding til eleven om, at der skal svares på et spørgsmål. Herefter vurderes elevsvaret således: værdien af den indtastede streng sammenlignes med ordene i et antal "rigtig-lister" (linie 19 og 21), eller med ordene i en "forkert-liste" (linie 23). Listerne kan indeholde en række alternativer (linie 19) og/eller en

række strenge, der skal findes i elevens svar (linie 21).

Programmet kan derefter reagere i overensstemmelse med vurderingen af elevens svar og f.eks. udskrive en tekst (linierne 20, 22 og 24). Det er endvidere muligt at acceptere ekstra ord (linie 16), mindre stavefejl (linie 17), store/små bogstaver (linie 18) eller anden ord sekvens i elevens svar. Det er muligt at acceptere alle ekstra ord (linie 16) eller blot en række specificerede ord (f.eks. linie 19: <and,or>).

Med listen som findes i linie 19 vil f.eks. svaret "Gilbert" eller evt. "Sullivan" blive accepteret som rigtigt. Derudover tillades ordene "and" og/eller "or".

Den anden rigtig liste (linie 21), kræver både ordene "Gilbert" og "Sullivan", og accepterer at ordet "and" findes i svaret.

Eksemplet på figuren indeholder kun én forkert liste (linie 23). Svaret "Rodgers" eller "Hammerstein" ville blive accepteret ifølge forkert-listen, og dermed ville svarteksten "You are a few decades off" blive udskrevet. Accepteres elevens svar ikke i overensstemmelse med nogle af listerne, vil muligheden efter ordren NO blive anvendt. Her ville linie 26 blive udført.

Uanset alle disse faciliteter til svarbedømmelse, er det traditionelt ved denne funktion, at der ligger en række vanskeligheder ved udarbejdelsen af undervisningsprogrammet. Det synes at være mest rimeligt blot at anvende en ret simpel svarevaluering, - hvis man ønsker at anvende denne facilitet.

3.2.1.3. Begrænsninger?

Det er nærliggende at spørge om systemet giver begrænsninger set i forhold til et generelt programmeringssprog. Principielt er svaret nej, - systemet har de samme muligheder som et generelt programmeringssprog, og dermed den samme frihed. De begrænsninger der ligger i de programmer, der kan udvikles

med systemet, falder i to grupper: (1) begrænsninger der generelt ligger i anvendelsen af en programmeret datamat til undervisning og (2) begrænsninger i de mål der viser sig at være vanskelige eller uoverkommelige at nå p.g.a. systemets manglende styrke!

3.2.1.4. Systemstruktur.

TenCore systemet består af tre hovedmoduler: (1) forfatterdel, (2) elevdel og (3) elevadministration. Se figur 3-3.

FORFATTERDEL	ELEVDEL	ELEVADMINISTRATION
TEKSTEDITOR	PROGRAM-	
GRAFIKEDITOR	AFVIKLING	
TEGNSÆTEDITOR		
ON-LINE MANUAL		

Figur 3-3: TenCore systemstruktur.

Forfatterdelen består af tre editorer. En teksteditor, der anvendes til udvikling af kildetekst og til tekstindhold i skærmbillederne i det endelige undervisningsprogram. En grafik editor til tegning af skærmgrafik, samt en tegneditor til konstruktion af egne tegnsæt.

Elevdelen afvikler den mellemkode, der er genereret ud fra TenCore koden i forfatterdelen.

Elevadministrationen giver mulighed for at registrere, hvor langt eleven er nået i materialet, idet det blot registreres om en lektion er helt eller delvis gennemført. Det er muligt at genstarte systemet fra det punkt, hvor eleven nåede ved det foregående arbejde med lektionen. Det er muligt at retablere alle variable fra det foregående arbejde. Disse variable kunne så f.eks. repræsentere point tildelt eleven, eller oplysninger om hvilken vej eleven skal følge gennem undervisningsprogrammet.

Erfaringer med kendte værktøjer

3.2.1.5. Tekst-, grafik- og tegnsæteditor.

Teksteditoren er en simpel linieeditor. Det er kun muligt at arbejde med én unit ad gangen i editoren. Dette er særligt generende, når der udvikles og afprøves interaktivt. Hvis programmet stopper med en udførelsesfejl i én bestemt unit, skal der startes forfra med angivelse af hvilken lektion og hvilken unit, der skal arbejdes med.

Grafikeditoren understøtter streggrafik: linier, kasser, cirkler, halvcirkler, pile og lignende, med variabel stregtykkelse. Derudover understøttes farver og mulighed for udfyldning/farvning.

Der kan i nogen grad arbejdes interaktivt med grafikeditoren, idet de nævnte grafiske elementer tegnes med enkle ordrer. Når en tegnesession afsluttes, kan der genereres TenCore kode, som derefter kan bearbejdes direkte eller via grafikeditoren, - derefter kan der igen genereres kode etc.

Den kode der skrives i, og som genereres af grafik editoren, oversættes til en mellemkode der fortolkes ved afviklingen.

Tegnsæteditoren giver mulighed for konstruktion af egne tegnsæt, f.eks. nationale tegnsæt, græsk, russisk etc., - eller specielle tegn til matematik, musik, tekniske symboler f.eks. elektriske eller mekaniske. Tegnsæteditoren er let at arbejde med og må siges at være en helt nødvendig del af et undervisningssystem, idet der f.eks. må være mulighed for at understøtte nationale specialtegn eller andre symboler.

3.2.2. Forfattersystemet: SOFUS/COMUS.

Det andet forfatterredskab der skal omtales her er SOFUS/COMUS systemet:

SOFUS/COMUS systemet er et forfatterredskab, der kan betegnes som et forfattersystem. Et forfattersystem vil sige et forfatterredskab, der leder/styrer læreren igennem udviklin-

gen af undervisningsmaterialet. Det betyder, at der udvikles et forløb med en, i det væsentligste, på forhånd fastlagt struktur.

SOFUS (Skelet til opbygning af frie undervisnings systemer) er det navn systemet sælges under til anvendelse på IBM-PC og kompatible maskiner, Til ICL/COMET sælges det under navnet COMUS (Comet undervisnings system).

Systemet findes i dag udbygget med grafikeditor samt videoeditor, med mulighed for interaktiv video ved hjælp af videobåndoptager eller CD (Compact Disk = optisk disk).

Yderligere informationer om SOFUS/COMUS kan findes i ref. 9, 12 og 22, ref. 22 specielt anvendelse af interaktiv video. Afsnit 3.2.2 bygger på ref. 10.

3.2.2.1. SOFUS/COMUS "filosofi".

Indledningsvis gives et indtryk af SOFUS/COMUS konceptet som det bl.a. er beskrevet i ref. 22.:

Systemet er udviklet med henblik på at lave et system til udvikling af meget åbne programmer, hvor forløbet i høj grad kan styres af eleven. Forfattersystemet er meget let tilgængeligt.

Der skelnes ikke mellem forfatter og elev på den klassiske måde, hvor forfatteren tilrettelægger et forløb for eleverne. Eleverne kan også optræde som forfattere og f.eks. skrive eller viderebehandle et informationssystem i forbindelse med arbejdet med et emne.

Systemet består af en række editorer til konstruktion af skærmbillede, menustruktur, ordbog og opgaver samt en afviklingsdel, der fortolker de data, som er indlagt med editorerne. I editorerne kan man afprøve visse ting, og man kan hele tiden under udviklingen gå direkte til afvikling og se, hvordan det, der allerede er udviklet, fungerer.

SOFUS/COMUS er i høj grad udviklet på baggrund af en klar

stillingtagen til datamatiske undervisningsforløb. Det følgende er baseret på uddrag af ref. 12.

Først definitionerne på begreberne forfatter/elev og undervisning, som de findes her:

> "....Forfatteren er den, der fremstiller undervisningsmateriale ved hjælp af COMUS, medens eleven er den, der dygtiggør sig gennem at arbejde med dette materiale. I praksis skal man ikke stirre sig blind på denne traditionelle opdeling. Forfatter og elev kan udmærket være én og samme person. En 'forfatter' kan opbygge et COMUS system til eget brug, og eleven kan i arbejdet med COMUS også anvende forfatterdelen.
>
> Ej heller begrebet undervisning bør man tolke traditionelt, når man arbejder med COMUS. Mange eksempler kunne nævnes: men blot brugeren af COMUS holder sig for øje, at systemet er et informatik- og undervisningssystem, vil ideerne til at udnytte COMUS i alle de forbindelser systemets alsidighed og lette anvendelse berettiger, ganske atutomatisk melde sig."

Det understreges at:

> "...begrebet uddannelse i denne forbindelse skal forstås i meget bred betydning. Anvendelse af COMUS til dataformidlet undervisning er også at bibringe elever en viden om et emne ved at eleverne lader sig informere af tekster på en dataskærm, men COMUS er anvendelig til mange andre formål, som har noget at gøre med informationsformidling..."
>
> "....COMUS bygger på en pædagogisk opfattelse af, at indsigt kræver arbejde med reelle helheder og problemer. Når man benytter en datamaskine til at skaffe sig indsigt og færdighed, indgår selve datamaskinen uundgåeligt i den helhed, man beskæftiger sig med. Samtidig med, at man lærer noget om et fagligt emne, lærer man noget om teknologi - herunder især hvorledes teknologi kan benyttes til uddannelse og informationsformidling.

Erfaringer med kendte værktøjer

En nuanceret teknologiforståelse er et væsentligt mål med al undervisning i den udvikling, erhvervsliv og samfund befinder sig i. Derfor er et meget væsentligt argument for at benytte datamaskinen i undervisningen, netop den teknologiforståelse man hermed formidler.

Hvis teknologiforståelse er en del af målet med at benytte DFU, er det imidlertid centralt, at den måde, man anvender datamaskinen på, er lige så bred, som den teknologiopfattelse, man ønsker at formidle.

Megen undervisningsprogrammel er beregnet til at producere programmer, som leder eleverne gennem stoffet ved at stille spørgsmål til eleverne. Elevernes svar på spørgsmålene valideres af maskinen som rigtig eller forkert. Hvis svaret er forkert, gives en ekstra hjælp, og hvis svaret er rigtigt, præsenteres lidt mere information og et nyt spørgsmål. Hvis dette princip er det dominerende indenfor DFU, medfører det, at den teknologiopfattelse, man formidler, bliver meget snæver.

Med COMUS kan man benytte sig af ovennævnte princip, og der findes ganske givet områder, hvor princippet kan retfærdiggøres.

Principperne og mulighederne i COMUS lægger imidlertid op til, at man baserer DFU materialet på, at det er eleverne, der stiller spørgsmålene og finder svarene ved at søge frem til netop det sted i materialet, hvor den for den givne situation relevante oplysning findes....."

3.2.2.2. SOFUS/COMUS strukturen.

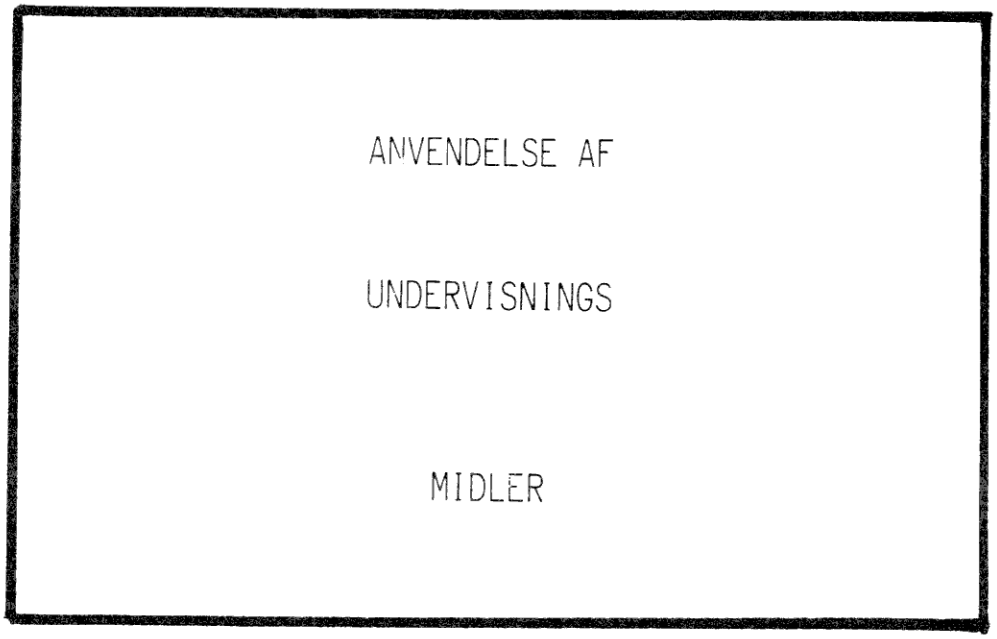
SOFUS/COMUS kan betegnes som et struktureret informationssøgnings system, hvor eleven arbejder med et emne gennem kommunikation med dette informationssøgnings system. Eleven er således den, der styrer undervisningsforløbet. Det er eleven, der selv kan formulere spørgsmål og søge at finde mulige svar i materialet.

Erfaringer med kendte værktøjer

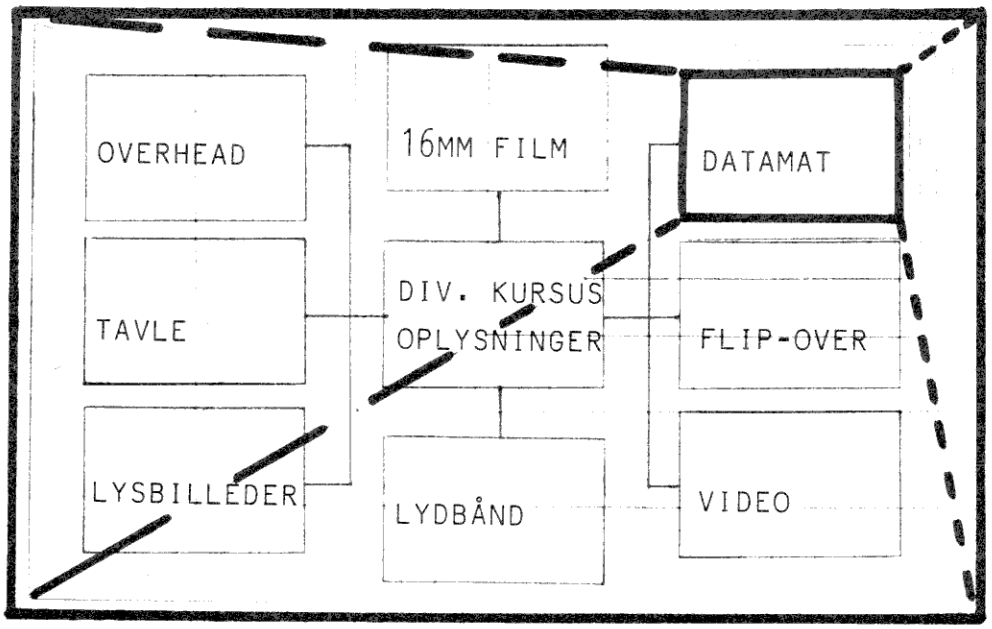
SOFUS/COMUS systemet er indrettet til at behandle tekstinformation. Informationerne i et SOFUS/COMUS undervisningsprogram er struktureret i en hierakisk træstruktur. Ved roden befinder der sig et såkaldt forsidebillede, der bruges til at identificere det pågældende undervisningsprogram. Ved alle knudepunkter i træet befinder der sig menuer. Bladene i træet består af informationsbilleder. Ved et informationsbillede forstås et skærmbillede med tekst.

Valg af undermenu eller informationsbillede, sker ved en indzoomning på h.h.v. undermenubillede eller informationsbillede, - hvis træets blade er nået.

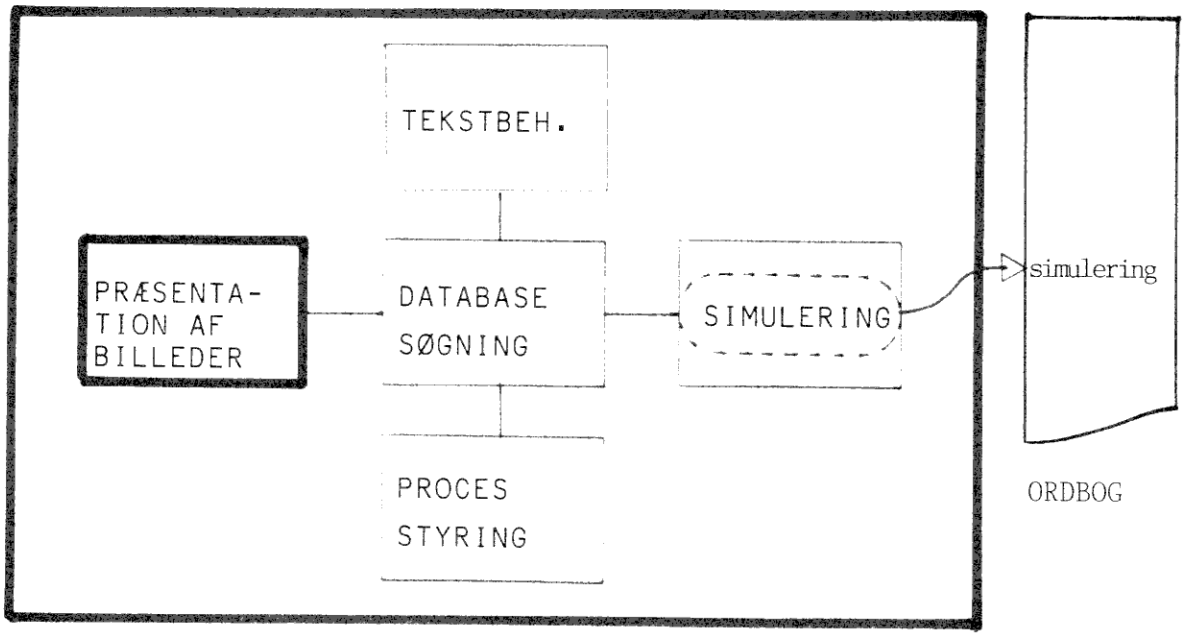
Figur 3-4 til 3-7 viser eksempler på de skærmbilleder, der kunne forekomme, når der zoomes fra forsidebillede til menubillede til undermenubillede og endelig til informationsbillede. Det er samtidig illustreret, hvorledes der er adgang til en ordbog.



Figur 3-4: A - Forsidebillede.

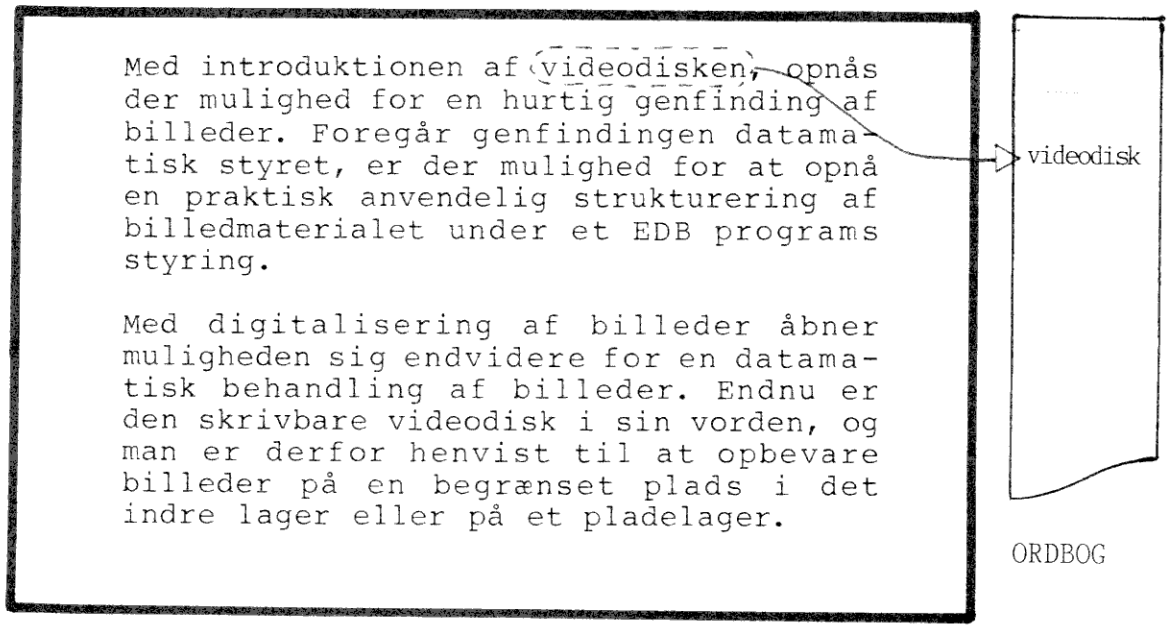


Figur 3-5: B - Menubillede 1. niveau.



Figur 3-6: C - Menubillede 2. niveau: "DATAMAT".

Erfaringer med kendte værktøjer



Figur 3-7: D - Informationsbillede: "Præsentation af billeder".

Informationerne i systemet befinder sig således i de enkelte informationsbilleder, der altid findes som blade i træet. Det bevirker, at man i arbejdet med SOFUS/COMUS systemet hele tiden har en sikker fornemmelse for, "hvor man befinder sig" i informationsmaterialet. Det er hele tiden muligt i ét trin, at vende tilbage til den menu, hvorfra man har valgt det aktuelle informationsbillede. På den måde er det umiddelbart muligt, at få overblik over i hvilken struktur informationsbilledet indgår.

3.2.2.3. Redskaber.

Ved arbejdet med et SOFUS/COMUS undervisningsprogram, har man, uanset hvor "man befinder sig" i den hierakiske informations struktur, mulighed for at arbejde med supplerende redskaber.

Man har således mulighed for at få en oversigt over de opgaver, der findes som en del af systemet, mulighed for at slå op i en ordbog, for at anvende en regnemaskine, og en-

delig for at anvende skærmen som et stykke elektronisk kladdepapir. Der er helt fri mulighed for, på et vilkårligt tidspunkt, at skrive på skærmen og gemme disse informationer.

Ordbogen anvendes på den måde, at man udpeger det ord, hvorom der ønskes forklaring. Der kan peges på et ord i en vilkårlig sammenhæng, f.eks. et ord i et informationsbillede eller i et menubillede. Et vindue på skærmen åbnes til ordforklaringen, sålænge den anvendes.

På ethvert tidspunkt er der mulighed for at få en papirkopi af det aktuelle skærmbillede.

3.2.2.4. Anvendelse af et undervisningsprogram.

Eleven har en lang række muligheder for at arbejde med tekstmaterialet. Han kan primært bevæge sig rundt i den hierakisk strukturerede information.

Eleven kan endvidere indtaste tekst på skærmen, eventuelt rette i den eksisterende tekst. Derefter er der mulighed for at gemme notatet, hvis dette ønskes. Eleven har løbende adgang til en oversigt over de eksisterende notater. I forbindelse med notater er det muligt at blankstille skærmen, således at et nyt "stykke kladdepapir" fremkommer på skærmen. Ordbog og regnemaskine kan frit anvendes på et vilkårligt tidspunkt.

3.2.2.5. Konklusion.

Der er alt i alt tale om et forholdsvis simpelt og let tilgængeligt system. Systemets styrke er måske netop, at det er simpelt; indenfor praktiske grænser sætter systemets simpelhed ikke grænser for, hvor omfattende SOFUS/COMUS informations strukturer, der kan konstrueres med systemet!

Personligt finder jeg systemet, med dets store mulighed for elevstyring, meget tiltalende. Dets styrke fornemmes først, når man har haft lejlighed til at arbejde med et større

undervisningsprogram!

3.3. En teori for undervisningssystemer?

Der er tilsyneladende et stigende behov for, og ønske om, anvendelse af forfatterredskaber. Behovet er naturligvis affødt af en stigende interesse for anvendelse af datamatiske undervisningsforløb. Det må forventes, at der løbende vil opstå krav om en udvikling med hensyn til bl.a. kompleksitet og formåen, både for forfatterredskabernes og undervisningsprogrammernes vedkommende.

Det eksisterende programmel savner imidlertid tilstrækkelig fleksibilitet, dynamik og styrke,- og det vil dermed have vanskeligt ved at tilpasse sig såvel en teknologisk, som en pædagogisk udvikling.

F.eks. indenfor områderne oversættere, databasesystemer, datakommunikation, og senest også indenfor grafiske kerner, f.eks. GKS, GSX og VDI*), herunder CAD/CAM systemer samt ekspertsystemer, er der foregået en mere eller mindre intens udvikling af en teoretisk baggrund for systemerne. Udvikling af teoretisk baggrund er vel nok mest gennemført indenfor et rent datalogisk område, nemlig oversættere. Denne udvikling af teori har bl.a. medført at det faktiske arbejde med udvikling af oversættere er blevet væsentligt begrænset i omfang.

*) GKS = Graphics Kernel System, GSX = Graphics System Extension, VDI = Virtual Device Interface.

I sammenligning med den udvikling, der er set indenfor disse komplicerede programmelsystemer, synes programmelsystemer til udvikling af undervisningsprogrammer at befinde sig på et meget lavt datalogisk stade.

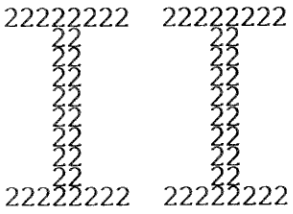
Denne mangel på en teoretisk baggrund er i større eller mindre grad gældende for alle integrerede interaktive systemer!

Der savnes bl.a. en teori omfattende en egentlig systematik.

Erfaringer med kendte værktøjer

Udvikling af en metodologi for disse programmelsystemer er en nødvendig og vigtig faktor for at opnå en udvikling af systemerne. En væsentlig baggund herfor er opbygning af en nomenklatur og systematik til beskrivelse af systemerne; desuden er det nødvendigt med en grundig forståelse af funktion og struktur.

I næste kapitel undersøges den konceptuelle ramme for et avanceret programmelsystem til støtte for udvikling af datamatbaserde undervisningsprogrammer, idet der præsenteres en nomenklatur samt en systematik.



D A T A L O G I S K E V Æ R K T Ø J E R

4. BEGREBSMÆSSIG RAMME FOR ET UNDERVISNINGSSYSTEM

Udviklingen af interaktive systemer har hidtil savnet en metodologi*) for design og konstruktion. Fundamentet herfor er en nomenklatur og systematik. Dette kapitel indeholder et forslag til en sådan begrebsmæssig ramme omfattende nomenklatur og systematik. Forslaget er baseret på kendskab til og analyse af eksisterende programmelsystemer.

I dette kapitel vil en systematik blive beskrevet for et avanceret programmelsystem til udvikling af undervisningsprogrammer. Dette tænkte system vil i det følgende blive refereret til som et "Datamatisk Interaktivt Integreret Undervisnings System", i det følgende forkortet med DIIUS. Formålet er at opbygge en begrebsmæssig ramme bestående af nomenklatur og systematik, på hvilken en metodologi for design og konstruktion af DIIUS kan bygges.

Fremstillingen i dette kapitel bygger på overvejelserne i ref. 15.

Først præsenteres et overblik over systematikken. Derefter udvikles systematikken med flere detaljer. Nomenklaturen fastlægges løbende.

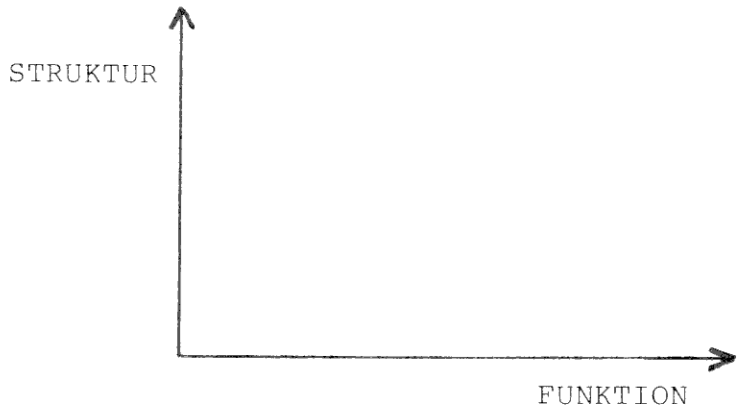
*) Nudansk Ordbog: "Metodologi (af græsk mèthodos..) fremstilling af de metoder, der anvendes inden for en videnskab...."

4.1. Systematik

Formålet med dette afsnit er blot at lette tilegnelsen af det følgende ved at give et rids af den struktur, der senere vil blive behandlet i detaljer. Samtidig vil væsentlige termer blot blive nævnt, - men også senere forklaret detaljeret.

Et DIIUS kan først og fremmest opdeles ved hjælp af en analyse af struktur og funktion. Den strukturelle opdeling kan tænkes at foregå ad en lodret akse og den funktionelle

ad en vandret akse, som det ses af figur 4-1.



Figur 4-1: Opdeling af et DIIUS.

En grovere opdeling kan i første omgang anvendes i forbindelse med et DIIUS, opdelingen kan meget vel udvides. Det er jo ofte den situation der opstår, når der arbejdes yderligere med den konceptuelle forståelse af et datamatisk system.

Vi skal således senere forfine denne opdeling. I første omgang præsenteres hovedtrækkene i systematikken:

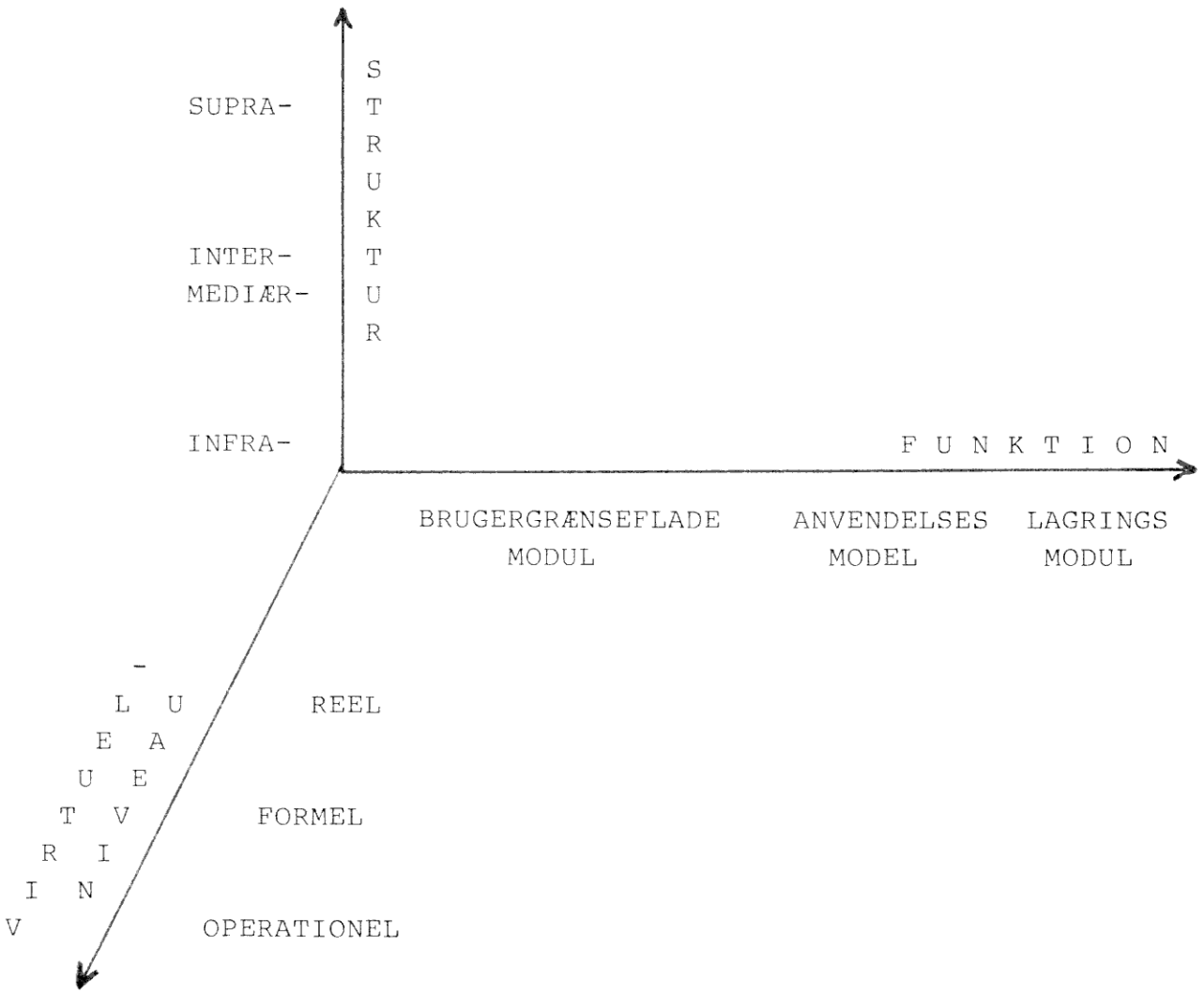
En strukturel opdeling, der kunne tænkes at foregå ad en lodret akse, anvender tre lag, det øvre synlige lag, som med betegnelsen fra ref. 15, betegnes supra-strukturen, og det nedre underliggende lag infra-strukturen, herimellem findes intermediær-strukturen.

En funktionel opdeling, der kunne tænkes at foregå ad en vandret akse, anvender tre moduler, som vi senere skal se en yderlige opsplitning af, nemlig brugergrænsefladem odulet, modulet for anvendelsesmodellen og lagringsmodulet.

Endvidere vil vi anvende en virtuel-niveau opdeling, der kunne tænkes at foregå ad en tredie akse, og som opdeler systemet i tre virtuelle niveauer, nemlig et operationelt niveau, et formelt niveau og et reelt niveau.

Begrebsmæssig ramme for et undervisningssystem.

Vi skal således ialt anvende opsplitninger af et DIIUS ad tre akser, - som det ses af figur 4-2. De dele der opstår ved disse opdelinger, vil blive betegnet moduler, uanset at der evt. senere foretages yderligere opdelinger, eller at et "del"modul evt. sammen med andre "del"moduler kunne udgøre et "hoved"modul. Betegnelsen "del-" eller "hoved-"modul anvendes således ikke, - blot modul. Et modul kan godt senere vise sig at kunne opdeles i andre moduler under anvendelse af yderligere opdelings kriterier. Termen modul anvendes således generelt og bredt.



Figur 4-2: Strukturel, funktionel og virtuel-niveau opdeling.

4.1.1. Strukturel opdeling.

Et datamatisk system opdeles ofte i lag, der successivt "bygger på hinanden". Begrebet er f.eks. kendt fra datamaters strukturelle opbygning, hvor en underliggende, virtuel, maskine siges at udføre ordrer for og understøtte en øvre liggende maskine. Denne idé om den strukturelle niveaudeling med underliggende virtuelle funktioner, også kendt fra struktureret programudvikling, - må siges at være af grundlæggende natur for forståelse af komplicerede datamatiske systemer og vil ikke blive yderligere uddybet her.

Infra-strukturen omfatter programmel til opbygning og understøttelse af intermediær- og supra-struktur programmellet. Intermediær-strukturen omfatter programmel til justering af brugerparametre i supra-struktur programmellet. F. eks. som det ses i figur 4-3:

SUPRA-STRUKTUR	Brugergrænseflade Informationspræsentation: Tekst/grafik Afvikling af UV-forløb Elevadministration
INTERMEDIÆR-STRUKTUR	Bruger justeringer af f.eks. brugergrænseflade og model med aksiomer
INFRA-STRUKTUR	Programmeringssprog Editorer Lænke og lade programmer Database systemer Grafiske kerner

Figur 4-3: Overordnede strukturel opdeling af DIIUS.

4.1.2. Funktionel opdeling

Den funktionelle opdeling deler DIIUS i moduler, der kan afgrænses ved en naturlig funktionel grænse, således at der opstår en logisk samling af funktioner i hvert modul og således at grænsefladerne lader sig definere præcist.

I denne første funktionelle opdeling står anvendelses modellen centralt, - undersøgelser af eksisterende interaktive systemer viser, at der herefter yderligere kan identificeres to moduler nemlig brugergrænsefladem odul og lagringmodul.

Den funktionelle opdeling opsplitter således DIIUS programmellet i tre moduler, nemlig brugergrænsefladem odulet, modulet for anvendelsesmodellen med tilhørende aksiomer og lagringsmodulet.

Ved aksiomer forstås de primitive eller grundlæggende ændringer eller operationer, der kan udføres på anvendelsesmodellen.

Aksiomerne betragtes som en del af anvendelsesmodellen. Når der derfor i det følgende blot refererers til anvendelsesmodellen, vil der altid menes modellen og aksiomerne.

Af figur 4-4 fremgår den første funktionelle opdeling:

BRUGERGRÆNSEFLADE MODUL	Brugerinterface. F.eks. præsentation af model Interface til operationer på model.
ANVENDELSESMODEL	Modellen af det datamatiske undervisningsforløb. Aksiomer hørende til modellen.
LAGRINGSMODUL	Lagring og genfinding af data på baggrunds lager.

Figur 4-4: Overordnede funktionel opdeling af DIIUS.

Vi har hermed anvendt to kriterier for opdelingen af DIIUS, nemlig funktion og struktur. Yderligere eksempler på hvad disse moduler kan omfatte ses af figur 4-5:

BRUGERGRÆNSE-FLADE MODUL	ANVENDELSES MODEL OG AKSIOMER	LAGRINGS MODULET	
virtuel input menu styring	den faktiske model operationer herpå	database	SUPRA-STRUKTUR NIVEAU
justering af brugergr. flade	oprettelse af nye aksiomer	ændring af struktur	INTERMEDIÆR-STRUKTUR NIVEAU
UIMS kommando pro- sessorer grafiske kerner	prog. sprog linkere spoolere	DBMS	INFRA-STRUKTUR NIVEAU

UIMS = User Interface Management System
DBMS = Data Base Management System

Figur 4-5: Overordnede funktionel/strukturel opdeling af DIIUS.

4.1.3. Virtuel-niveau opdeling

Med henblik på den kommunikation, der foregår imellem de funktionelle moduler i supra-strukturen, foretages en opdeling i tre virtuelle niveauer.

Som vi senere skal se, foregår kommunikationen kun imellem funktionelle moduler på supra-struktur niveau, - derfor anvendes den virtuelle-niveau opdeling kun for supra-strukturen.

Opdelingen er virtuel, fordi de moduler der opstår ved denne opdeling ikke kan identificeres som separate programmelmoduler. Opdelingen afspejler udelukkende en konceptuel opfattelse af kommunikationen. De moduler vi tidligere har opdelt DIIUS i udgøres heller ikke nødvendigvis af separate pro-

grammelmoduler, - men de kunne godt udgøres af separate programmelmoduler, der kunne identificeres.

Der anvendes tre virtuelle niveauer, nemlig et operationelt niveau, et formelt niveau og et reelt niveau. Se figur 4-6.

OPERATIONELT NIVEAU	FORMELT NIVEAU	REELT NIVEAU
kommandoer/ operationer	formel data beskrivelse	faktiske data

Figur 4-6: Overordnede virtuel-niveau opdeling af DIIUS.

Hovedstrukturen for et DIIUS er nu præsenteret. Vi skal herefter se en yderligere strukturering og funktionsopdeling, hvor analysen koncentreres om supra-struktur programmellet:

4.2. Supra-struktur programmellet

Efter at den overordnede systematik er fastlagt, skal vi nu se på en yderligere forfining af systematikken, her indenfor supra-struktur programmellet.

Programmel modulerne, der udgør supra-struktur programmellet, er bl.a. adskilt ud fra den funktionelle opdelings kriterier. På supra-struktur niveauet er det moduler, der aktivt behandler og flytter data. Hvert modul udfører en specifik funktion med hensyn til behandling af data.

Vi har tidligere set den strukturelle opdeling i supra-, intermediær- og infra-struktur. Vi skal i dette afsnit se en findeling med hensyn til funktion og virtuel-niveau indenfor supra-strukturen. Den virtuelle-niveau opdeling anvendes kun for supra-strukturen, - hvorimod den funktionelle opdeling anvendes for infra-, intermediær- og supra-struktur.

Først vender vi os imod den funktionelle findeling:

Begrebsmæssig ramme for et undervisningssystem.

4.2.1. Funktionel findeling

Der er, som vi har set tidligere, identificeret tre hovedmoduler:

(1) Brugergrænsefladem odul
(2) Anvendelsesmodel
(3) Lagringsmodul

Anvendelses modellen indeholder en model af undervisningsforløbet eller undervisningsprogrammet. Endvidere omfatter anvendelsesmodellen, som tidlige omtalt, aksiomer, dvs. de basale operationer, der kan anvendes til ændring af modellen.

Det vil være af interesse at kunne identificere delmoduler i anvendelsesmodellen, og dermed foretage en funktionel findeling heraf.

Hvordan en funktionel findeling foretages vil altid i større eller mindre grad bero på et skøn. Det er den situation, der er kendt fra programudvikling, hvor adskillelsen af funktioner i underprogrammer ligeledes beror på en individuel vurdering og skøn.

Det vil være ønskeligt at kunne identificere funktionelle moduler, der er almengyldige, således at denne systematik, som de indgår i, bliver generel og dermed gyldig for alle undervisningssystemer.

Bl.a for at opnå en større forståelse for strukturen af et DIIUS og for at få en yderligere fornemmelse for hvad anvendelsesmodellen omfatter, vil det være af interesse at identificere delmoduler.

En yderligere definition og præcisering af delmoduler kan først foretages, når der er indhøstet flere praktiske erfaringer i arbejdet med begreberne i praksis. Jeg vil derfor tillade mig at fremsætte et forslag til en mulig opdeling, således at denne opdeling kan danne udgangspunkt for supple-

rende undersøgelser. Forslaget er baseret på en analyse af eksisterende undervisningssystemer og undervisningsprogrammer.

På baggrund af denne analyse er anvendelses modellen foreløbig opdelt i følgende seks moduler/delmodeller:

(1) Forløbs afvikling
(2) Informations håndtering
(3) Kommunikation med eksterne processer
(4) Feedback og opgaver
(5) Redskaber
(6) Elevadministration

En række eksempler, i forbindelse med den følgende beskrivelse, vil belyse hvad de enkelte delmoduler f.eks. kunne tænkes at omfatte:

(2) Informations håndteringen omfatter præsentation af information, f.eks.: grafik, digitale billeder eller tekst på skærm, skriver eller plotter.

(3) Kommunikation med eksterne processer omfatter f.eks. kommunikation med eksterne databaser, styring af AV-medier, f.eks. dias, video-bånd eller video-disk, process regulering og/eller måling.

(4) Feedback og opgaver omfatter elevfeedback, det kunne f.eks. være elevens styring af selve undervisningsforløbet, eller det kunne være elevens svar på en opgave.

(5) Redskaber der f.eks. kunne omfatte: ordbogsfunktion, regnemaskine, mulighed for notater på skærm eller papir, kalender og ur etc. Redkaberne vil ofte være tilstede, uanset hvor eleven befinder sig i materialet, og hvad han arbejder med.

(6) Elevadministration omfatter f.eks. registrering af en identifikation for enkelte elever eller grupper/hold med hensyn til bl.a. elevens profil, dvs. vidensområder, niveau, point tildeling og gennemførte lektioner.

Begrebsmæssig ramme for et undervisningssystem.

Registrering af, hvorledes materialet tidligere har været anvendt i forbindelse med én elev eller en gruppe elever.

(1) Forløbsafvikling sammenknytter de andre fem moduler til en samlet enhed ved afvikling af en lektion eller en præsentation. Forløbsafviklingen aktiverer således de andre modeller i et forløb, der kunne være fastlagt af en lærer, - eller en elev. Forløbet kunne fastlægges før, under eller efter afviklingen af en lektion el.l.

Anvendelses modellen foreslås således opdelt i følgende seks delfunktioner (figur 4-7):

ABSTRAKT NIVEAU	EKSEMPLER
FORLØBS AFVIKLING	Afvikling af lektion eller præsentation. Sammmenknytning af andre funk. moduler.
INFORMATIONS HÅNDTERING	Præsentation af information: f.eks. grafik, tekst, billeder.
KOMMUNIKATION MED EKSTERNE PROCESSER	Eksterne databaser, styring af AV-medier, process regulering.
FEEDBACK OPGAVER	Elevens feedback, opgaver.
REDSKABER	Regnemaskine, ordbog, notater etc.
ELEV ADMINI-STRATION	Elevprofil, point tildeling. Hidtidige arbejde med materialet.

Figur 4-7: Funktionel opdeling af anvendelses-modellen.

I et DIIUS skal disse funktionelt adskilte moduler sammenhægtes eller integreres til ét system.

4.2.2. Virtuel-niveau opdeling af funktionelle moduler

Imellem de enkelte funktionelt adskilte moduler skal flyde en datastrøm, der på laveste niveau udgør den faktiske kommunikation imellem modulerne.

En betingelse for transport af informationsindhold er en overensstemmelse med hensyn til datatype, som derefter muliggør den egentlige overførsel af data.

De funktionelle moduler kan i overensstemmelse hermed beskrives på tre niveauer:

(1) det operationelle niveau
(2) det formelle niveau
(3) det reelle niveau

(1) Beskrivelser på det operationelle niveau omfatter beskrivelser af operationer, der kan udføres på anvendelsesmodellen. Beskrivelserne på dette niveau omfatter således beskrivelser af kommandoer på et abstrakt niveau. Der er knyttet operationelle beskrivelser til hvert funktionelt modul.

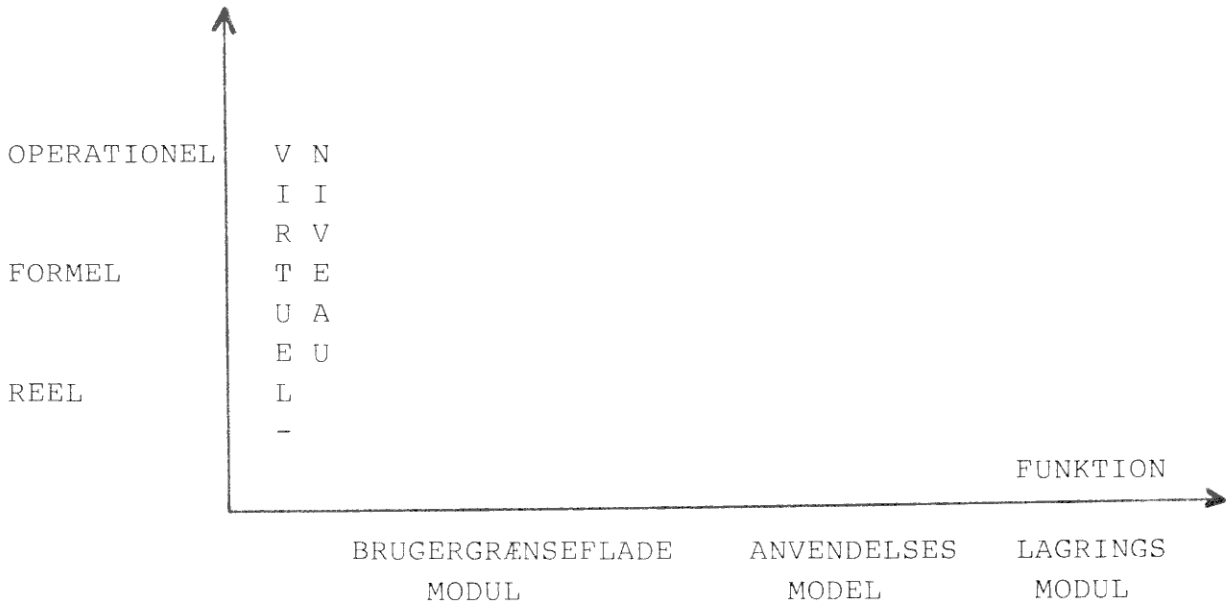
Bemærk: anvendelsesmodellens aksiomer består af den mængde primitive ændringer, der kan foretages på anvendelsesmodellen, og som er tilknyttet anvendelsesmodellen

(2) Beskrivelser på det formelle niveau omfatter beskrivelser af datatyper, som de f.eks. er kendt fra Pascals typer.

(3) Beskrivelser på det reelle niveau omfatter de faktiske data, som de f.eks. er kendt fra Pascal variabel erklæringer med tilhørende tildelinger.

Som vi senere skal se, skal der foregå en kommunikation

imellem de funktionelle moduler, den virtuel-niveau opdeling er af betydning i denne kommunikation. Figur 4-8 viser opdelingen med hensyn til virtuel-niveau og funktion:



Figur 4-8: Virtuel-niveau/funktionel opdeling.

Til forståelse af denne opdeling vises beskrivelser på de tre niveauer, i det følgende eksempel:

4.2.2.1. Eksempel: Virtuel-niveau.

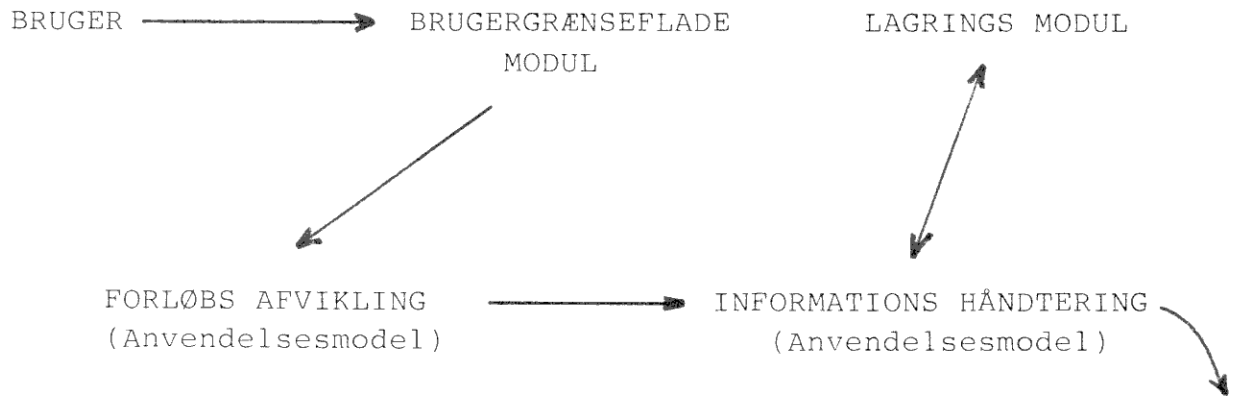
Lad os forestille os følgende situation:

> Video billede med frame nummer 1001 ønskes vist på 7/8 af skærmen. På 1/8 af skærmen ønskes samtidig vist en tilhørende tekst, identificeret ved post nummer 86 i en skærmbillede fil.

Operationen vil være kendt i såvel brugergrænseflade modulet som i anvendelsesmodellen, - beskrivelsen tilhører det operationelle niveau.

Beskrivelsen af operationen kunne tænkes sendt fra bruger-

grænsefladen til anvendelsesmodellens forløbsafvikling, der derefter ville foretage yderligere kommunikation med modulet til informations håndtering og lagrings modul, - som det ses af figur 4-9.



Figur 4-9: Virtuel-niveau: Eksempel.

Infra-struktur programmellet, der indeholder moduler, f.eks. en grafisk kerne, der foretager selve den fysiske præsentation af materialet på skærm eller skriver, vil derefter blive aktiveret; - infra-strukturen er ikke emnet her.

Lad os se hvorledes denne operation kunne beskrives på de tre niveauer:

(1) Operationel beskrivelse:

Operationen kunne på det operationelle niveau symboliseres ved følgende beskrivelse:

```
VIS  FRAME = 1001.7 ; TEXT = 86.1
```

(2) Formel beskrivelse:

For at få overført data, der repræsenterer denne operation, kunne anvendes følgende Pascal typedefinition:

```
TYPE
   FRAME_NO_TYPE     = 1..MAX_FRAME_NO;
   TEXT_NO_TYPE      = 1..MAX_TEXT_FRAME_NO;
   FRACTION          = 1..8;

   SCREEN_DEF = RECORD
                   FRAME_NO      : FRAME_NO_TYPE;
                   FRAME_FRAC    : FRACTION;
                   TEXT_NO       : TEXT_NO_TYPE;
                   TEXT_FRAC     : FRACTION;
                END;
```

Typebeskrivelsen udgør i dette eksempel en beskrivelse på det formelle niveau. Typebeskrivelsen er knyttet til en operation, - i eksemplet af typen "VIS".

(3) Reel beskrivelse:

De faktiske data ville kunne overføres imellem de to moduler ved data tildelt følgende Pascal variabel:

```
VAR
   PICTURE : SCREEN_DEF;
```

Følgende tildelinger kunne anvendes:

```
WITH PICTURE DO BEGIN
   FRAME_NO   :=1001;
   FRAME_FRAC :=7;
   TEXT_NO    :=86;
   TEXT_FRAC  :=1;
END;
```

Data, der nu er indeholdt i variablen PICTURE, udgør en beskrivelse på det reelle niveau, - de faktiske data. Figur 4-10 viser en oversigt over beskrivelser på de tre niveauer:

```
OPERATIONEL        VIS  FRAME = <n>.<f> ; TEXT = <n>.<f>

FORMEL             TYPE
                        SCREEN_DEF = .......

REEL               VAR
                        PICTURE : SCREEN_DEF;

                   WITH PICTURE DO BEGIN
                        ....   :=   .....
                        ....   :=   .....
```

Figur 4-10: Eksempel - Beskrivelser på tre virtuelle niveauer.

Efter at have set, hvad den virtuelle niveaudeling af et struktureret DIIUS omfatter, skal vi se, hvorledes man kan opfatte det som om, de enkelte funktionelle moduler kummunikerer via de tre virtuel-niveauer.

4.3. Kommunikation i et virtuel-niveaudelt system.

For at opnå et smidigt flow af data gennem vort nu både funktionelt og virtuel-niveau opdelte DIIUS, må vi løse problemet med kommunikationen imellem de enkelte moduler.

For at få en forståelse af kommunikationen imellem funktionelle moduler med en struktur, der omfatter en virtuel niveaudeling, vil det være en hjælp at vende sig mod det datalogiske område, hvorfra vi i denne sammenhæng har lånt begrebet "kommunikation", nemlig fra datakommunikation.

Der findes indenfor dette område en forholdsvis veludviklet og til dels også anvendt og implementeret teori for sammenkobling af systemer, der udviser en vis grad af forskelligartedhed.

Der er udviklet en begrebsmæssig ramme, indenfor hvilken de

kommunikerende systemer bør befinde sig. Det er ISO's (International Standards Organization) OSI model (Open Systems Interconnection). Det faktum at OSI er fastlagt af ISO, der er en standardiseringsorganisation, skal ikke lede en til den udbredte misforståelse, at OSI-modellen skulle være en standard for datakommunikation.

OSI-modellen udgør en begrebsmæssig ramme for udvikling af datakommunikations programmel; den endelige realisation af kommunikationsprogrammellet defineres af en række protokol standarder.

For forståelse af kommunikationen i et system, der er trelagsdelt i et operationelt, formelt og reelt niveau, kan man meget vel have den begrebsmæssige beskrivelse af datakommunikation, som den ses beskrevet i OSI 7-lags modellen, i tankerne.

Da vi i denne sammenhæng kan drage nytte af den parallel, der kan drages til OSI-modellen, - følger en kort beskrivelse af OSI-modellen, eller 7-lags modellen. Se endvidere ref. 19 og 20.

4.3.1. ISO OSI-modellen

OSI modellen anvender en lagdeling, hvor et øvre lag tilsyneladende arbejder på en underliggende virtuel "maskine". Denne virtuelle maskine betegnes ofte som "protokolmaskinen".

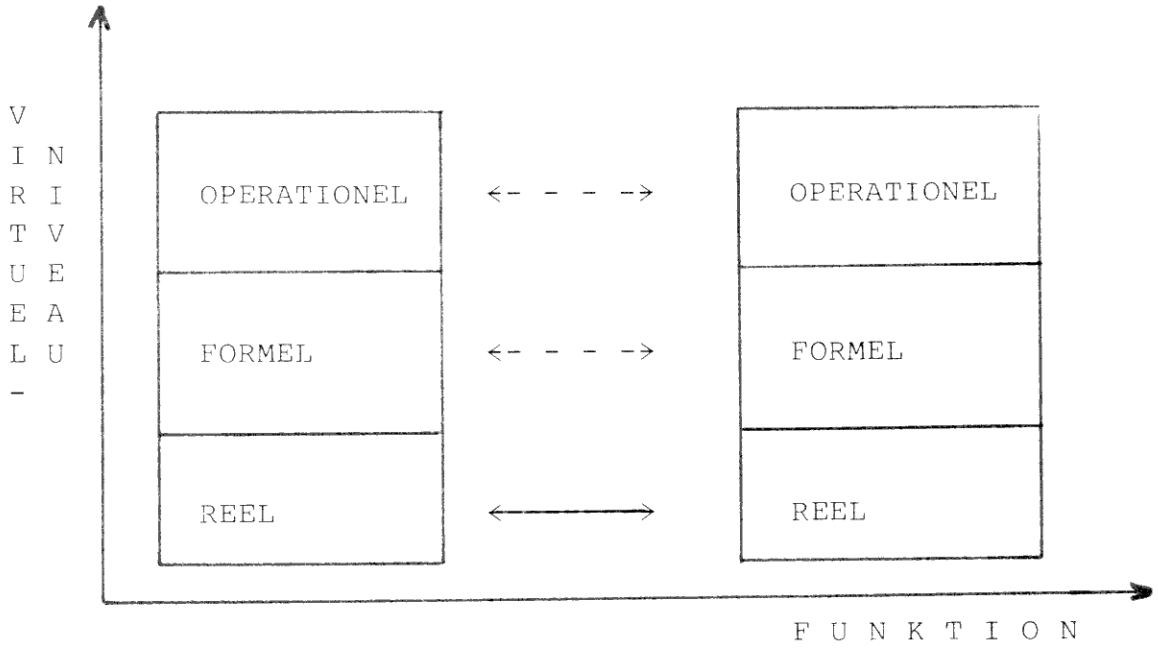
Modellens abstraktion opfatter et faktisk datamatsystem, som om det logisk var opbygget af et hieraki af lag. Hvert lag i hierakiet definerer en delmængde af de funktioner, der er nødvendige for sammenkobling af datamatsystemer. OSI modellen omfatter 7-lag. Lagene nummereres fra nummer 1 til 7, idet det fysiske lag, der udgøres af de elektriske forbindelser, er lag nummer 1 og anvendelseslaget er lag nummer 7.

Det kan nu opfattes således, at hvert lag i den ene datamat kommunikerer med det tilsvarende lag i den anden datamat. Kommunikationen foregår via de underliggende lag, - idet den

faktiske kommunikation jo kun foregår i det nederste lag: det fysiske lag. F. eks. vil et anvendelsesprogram skulle gennem alle underliggende lag i begge maskiner, for at kommunikere med et anvendelsesprogram i den anden maskine.

De nedre lags funktioner er nødvendige for, at de øvre lag kan udføre deres funktion. Det udtrykkes i OSI modellen ved at sige, at ét lag udnytter tjenester fra et tilgrænsende nedre lag; mens det tilbyder tjenester til det tilgrænsende øvre lag.

Med baggrund i begreberne i OSI modellen kan den kommunikation, der er tale om i forbindelse med et DIIUS, illustreres således (figur 4-11):



Figur 4-11: Kommunikation i virtuel-niveaudelt model.

På det operationelle og det formelle niveau foregår en virtuel kommunikation, den reelle kommunikation foregår, - på det reelle niveau!

Vi skal nu se, hvorledes denne sammenkobling af de virtuel-niveaudelte funktionelle moduler kan foregå i et DIIUS:

Begrebsmæssig ramme for et undervisningssystem.

4.3.2. Sammenhægtning/integration

Det er ønskeligt, at et DIIUS fungerer som ét samlet arbejdsmiljø til udvikling af undervisningsprogrammer. Det vil sige, at forskellige funktionelle moduler skal sammenhægtes. Det er nødvendigt, at der etableres kommunikation imellem modulerne på alle niveauer: operationel, formel og reel.

For at de funktionelle moduler kan fungere i sammenhæng, er det vigtigt, at deres indbyrdes integration undersøges. Den sammenhægtning, der er nødvendig, skal etablere mulighed for overførsel af data imellem modulerne.

Hvert funktionelt modul har f.eks. sine egne data-typer med tilhørende data, dvs. sine egne beskrivelser på det formelle og reelle niveau. Derudover har hvert funktionelt modul sine egne beskrivelser af operationer, svarende til beskrivelser på det operationelle niveau.

For at data fra et modul kan anvendes i et andet er det således nødvendigt, at der ved hjælp af en transformationsfunktion foretages en tilpasning af f.eks. data fra én formel definition til en anden. På den måde bliver data i det ene modul kompatibelt med data i det andet modul. Det programmel, der sørger for at etablere denne grænseflade, betegnes et interfacemodul.

Det skal derfor undersøges, hvilke moduler der har behov for indbyrdes kommunikation. Imellem disse moduler skal der derefter etableres interfacemoduler. F.eks. vil der være et behov for at brugergrænsefladem odulet kan kommunikere med anvendelsesmodellen, og indenfor anvendelsesmodellen skal f.eks. forløbsafviklingen kunne kommunikere med næsten alle delmoduler indenfor anvendelsesmodellen.

Indtil videre vil jeg udelade betragtninger med hensyn til det operationelle niveau og blot forudsætte, at der findes en én-éntydig afbildning fra operationer/kommandoer i det ene modul til operationer/kommandoer i det andet modul. Denne begrænsning er foretaget på dette sted for at lette

tilegnelsen af det følgende, men vi skal senere vende tilbage til en diskussion med relation til interface på det operationelle niveau.

Til at illustrere de problemer, der opstår ved kommunikation og tilhørende tilpasning imellem to moduler, anvendes følgende eksempel:

4.3.2.1. Interface: Eksempel.

Vi forestiller os følgende situation:

- der anvendes en brugergrænseflade med ikoner og udpegning ved hjælp af en mus. Et bestemt digitaliseret videobillede vælges til præsentation på skærmenved, at den tilhørende ikon udpeges med musen.

Til identifikation af et bestemt billede anvender brugergrænseflade modulet f.eks. følgende pascal erklæringer:

```
TYPE
   ACTIVE_ICON_TYPE  = RECORD
                          X : 1..15;
                          Y : 1..10;
                       END;
VAR
   ACTIVE_ICON       : ACTIVE_ICON_TYPE;
```

- idet der kan vises 15 X 10 ikoner på skærmen. Ikonerne anvendes altså her til at udvælge et billede.

Til identifikation af et bestemt billede anvender informationshåndteringsmodulet i anvendelsesmodellen f.eks. følgende pascal erklæringer:

```
TYPE
     FRAME_NO_TYPE        = 1..64000;
     DEFINED_FRAME_TYPE   = ARRAY(.FRAME_NO_TYPE.) OF BOOLEAN;

VAR
     VIDEO_FRAME          : FRAME_NO_TYPE;
     DEFINED_FRAME        : DEFINED_FRAME_TYPE;
```

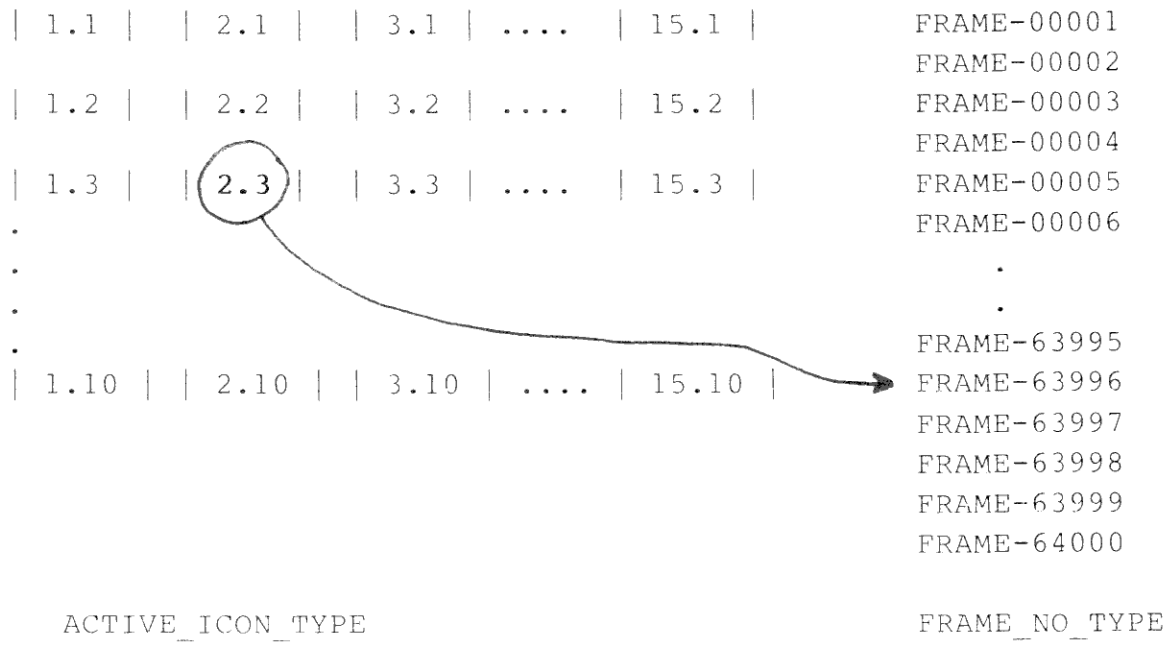
Idet DEFINED_FRAME(.F.) er sand, hvis frame nummer F faktisk findes. Det kunne nu være interfacemodulets opgave at udføre følgende:

```
X := TRANSFORMATION( ACTIVE_ICON );
IF NOT DEFINED_FRAME(.X.) THEN
     FRAME_ERROR
ELSE
     VIDEO_FRAME := X;
```

- idet funktionen "TRANSFORMATION"'s funktion er illustreret på figur 4-12 for ét tilfælde svarende til følgende faktiske data:

```
WITH ACTIVE_ICON DO BEGIN
     X := 2;
     Y := 3;
END;
```

Derudover skal variablen DEFINED_FRAME være tildelt faktiske data.



Figur 4-12: Eksempel - Transformations funktion.

I dette eksempel fås VIDEO_FRAME = 63996, når IKON (2,3) er udpeget.

Interfacemodulet skal, som det ses eksemplificeret her, have kendskab til den formelle beskrivelse indenfor de to moduler hvor imellem det befinder sig. For det andet skal interfacemodulet indeholde funktioner, der transformerer data fra den ene formelle repræsentation til den anden, som illustreret på figur 4-12.

Udvikling af et interfacemodul omfatter således: (1) fastlæggelse af det formelle niveau for de to funktionelle moduler og (2) udvikling af transformationsfunktionen.

4.3.2.2. Identifikation af interfacemoduler

Det vil være nødvendigt at identificerede de moduler/delmoduler, der har behov for kommunikation, - dermed kan interfacemodulerne identificeres.

Modellen af et givet undervisningsprogram befinder sig i anvendelsesmodellen. Brugeren kan ændre på denne model via brugergrænsefladem odulet og endelig kan en repræsentation af modellen lagres ved hjælp af lagringsmodulet. Det vil derfor minimum være nødvendigt at definere interfacemoduler således (figur 4-13):

BRUGER GRÆNSE-FLADE	**INTER-FACE**	ANVENDEL-SES MODEL	**INTER-FACE**	LAGRINGS MODUL

Figur 4-13: Nødvendige interfacemoduler.

4.3.3. Interface: Anvendelsesmodel <--> lagringsmodul.

Programmellet i lagringsmodulet indeholder faciliteter til at lagre og genfinde data, tilhørende anvendelsesmodellen, på baggrundslager: diskette, pladelager, bånd, CD eller lignende.

Data vil være organiseret på forskellig vis på de forskellige medier. Dette interface skal derfor have kendskab til en beskrivelse af strukturen for lagringsmediet, samt en beskrivelse af den interne struktur for anvendelsesmodellen.

Den situation, at data er organiseret i én struktur på lagermediet og i en anden i programmet, er velkendt fra følgende eksempel: Der anvendes et pladelager, organiseret i spor og sektorer. En datastruktur, der indeholder et register, kunne f.eks. være organiseret i poster, der f.eks. ønskes genfundet i alfabetisk orden. Det er derfor nødvendigt med en transformation fra den ene struktur til den anden.

En transformationsfunktion, der omsætter data fra anvendelsesmodellens struktur til lagringsmodulets struktur, skal derfor defineres og anvendes til overførsel af data.

Figur 4-14 viser grænsefladen/interface imellem anvendelsesmodellen og lagringsmodulet.

	ANVENDELSES-MODEL	INTERFACE <--->	LAGRINGS-MODUL
FORMELLE NIVEAU	DATA TYPER	TRANSFORMATIONS FUNK.	YDRE ENHED
REELLE NIVEAU	DATA	DATA TRANSFORMATION	LAGREDE DATA

Figur 4-14: Interface imellem anvendelsesmodel og lagringsmodul.

4.3.4. Interface: Brugergrænseflademodul <--> Anvendelsesmodel

Brugergrænseflademodulet håndterer kommunikation imellem menneske og maskine. Det består af to adskilte dele: (1) præsentationsdel og (2) kommandodel. (ref. 15).

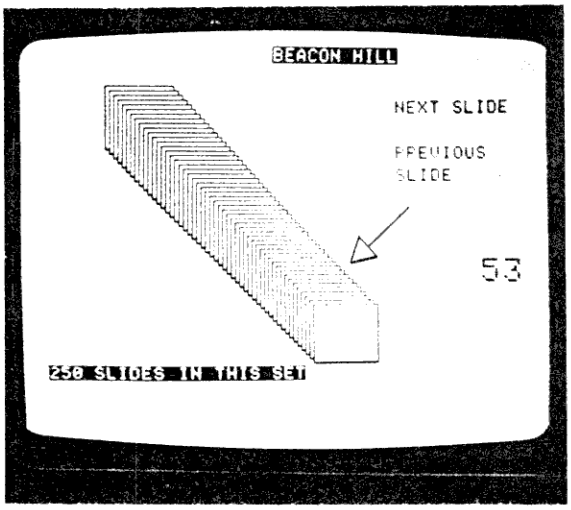
(1) Præsentations programmellet omformer anvendelsesmodellens data til en form, der lader sig præsentere (grafisk) for brugeren. Data modtages fra det interface, der er beliggende imellem brugergrænseflade modulet og anvendelsesmodellen.

Præsentationen og dermed transformationen, der skal foregå i interfacet, omfatter to hovedområder: (A) billede præsentation og (B) struktur præsentation.

(A) Billede præsentationen omfatter præsentation af de billeder som modellen indeholder, dvs. tekst, grafik og digitaliserede videobilleder. Transformationen kommer derved til at omfatte en transformation, som konverterer data ved hjælp af modellens formelle beskrivelse og den ydre enheds formelle beskrivelse.

(B) Struktur præsentationen omfatter en oversigtsmæssig eller strukturel præsentation af de objekter i modellen der ikke lader sig præsentere direkte, eller som ikke ønskes præsenteret direkte af oversigtsmæssige årsager.

Hvis f.eks. modellens delmodul til ekstern proceskommunikation indeholder styring af dias-fremvisning, kunne en struktur præsentation, når der refereres til et enkelt billede, være som vist på figur 4-15 (ref. 2):



Figur 4-15: Eksempel på struktur præsentation.

Transformationsfunktionen transformerer her data fra anvendelsesmodellens datastrukturer til de data, der f.eks. skal anvendes af en grafisk kerne for at danne billedet i figuren.

(2) Kommando programmellet transformerer kommandoer fra brugergrænseflade modulet til anvendelsesmodellen. Der foretages således transformation på det operationelle niveau. Derefter kan de operationer, der er defineret på det operationelle niveau for anvendelsesmodellen udføres i denne.

Vi skal i næste afsnit se nøjere på, hvilke definitioner det operationelle niveau omfatter, samt hvilke transformationer

der er behov for.

4.3.5. Interface på det operationelle niveau.

Er der forskelle på det operationelle niveau, vil der være behov for en transformation, - svarende til den vi så det for det formelle og det reeele niveau:

F.eks. kunne operationen "VIS BILLEDE PÅ SKÆRM" eksistere i brugergrænsefladem odulet og i anvendelsesmodellen. For at alle modulerne skulle kunne anvendes i én sammenhæng, skulle der derfor, naturligvis, være overenstemmelse imellem definitionerne af en given operation indenfor de forskellige moduler på alle virtuelle-niveauer.

Imidlertid vil implementationen af den enkelte operation være meget forskellig i hvert modul, og modulerne ville derfor være højst forskellige på det formelle og det reelle niveau, som vi tidligere har set, og som det ses af følgende eksempel.

F.eks. kunne en stregtegning med tekst repræsenteres ved (1) vektorgrafik og grafiske tegn i én model og ved (2) rastergrafik i en anden model. Altså en forskel på det formelle og reelle niveau. Et problem der måtte håndteres af interface modulet.

Hvis modulerne ikke udviser overensstemmelse på det operationelle niveau, må der endvidere foretages en tilpasning på det operationelle niveau, som det ses af følgende eksempel:

Hvis således en model, model (1), udviklet i et miljø, skulle anvendes i forbindelse med model (2), udviklet i et andet miljø, måtte det overvejes, om kendte operationer i model (2) ville være i stand til at emulere operationerne i model (1).

F.eks. kunne én model (1), arbejde med variable, der indeholder oplysninger om, hvor langt eleven er nået i forløbet, hans personlige niveau, opnåede points og lignende. I model (1) kunne der derfor eksistere operationer som: "GEM_AFVIK-

LINGS_FORLØBS_DATA" og "RETABLER_AFVIKLINGS_FORLØBS_DATA". Hvis en anden model (2) ikke omfattede disse operationer måtte det overvejes, hvor vidt det var muligt for model (2) at udføre operationerne v.h.a. dens egne kendte operationer.

Vi vil i det følgende afsnit betragte anvendelsesmodellen og brugergrænsefladem odulet.

4.3.5.1. Interface: Brugergrænseflade <--> anvendelsesmodel.

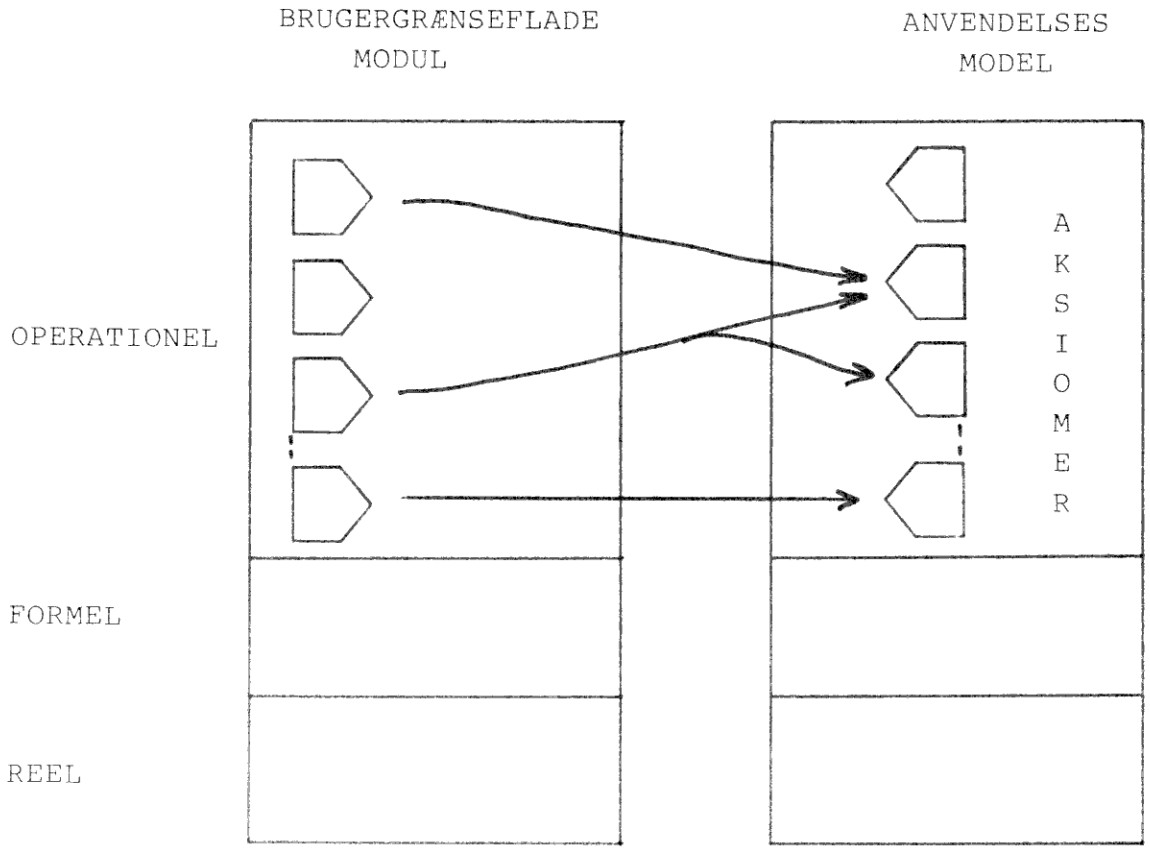
Det operationelle niveau for anvendelsesmodellen beskriver de grundlæggende operationer, der kan udføres på anvendelsesmodellen. Disse grundlæggende, primitive operationer udgøres af aksiomerne tilknyttet modellen. Operationerne er implementeret i overensstemmelse med den formelle opbygning af selve modellen, evt. af modellens delmoduler.

Det operationelle niveau for brugergrænsefladen beskriver de kommandoer til ændring af anvendelsesmodellen, som brugeren faktisk har til sin rådighed.

Forskelle på det operationelle niveau ville forekomme i den situation, at der ikke, som tidligere forudsat, er en éntydig sammenhæng imellem operationerne tilgængelige i brugergrænsefladem odulet og operationerne/aksiomerne i anvendelsesmodellen. Dette kunne f.eks. forekomme, hvis den samme brugergrænseflade anvendes på to forskellige anvendelsesmodeller.

Fastlæggelsen af det operationelle niveau for anvendelsesmodellen og brugergrænsefladem odulet er f.eks. af interesse, i fald enkelte operationer i brugergrænsefladen skal udføres ved hjælp af flere operationer/aksiomer i anvendelsesmodellen.

I dette tilfælde vil det være nødvendigt at etablere interface, med tilhørende transformations funktion, på det operationelle niveau.



Figur 4-16: Transformationsfunktion - operationelle niveau.

Som det ses af figur 4-16 sammenknyttes operationer i brugergrænseflademodulet med én eller flere aksiomer i anvendelsesmodellen.

I tilfælde af stor uoverensstemmelse på det operationelle niveau kan den situation, at det slet ikke er muligt at få de enkelte moduler til at fungere sammen, meget vel opstå. Situationer hvor anvendelsesmodellens aksiomer ikke er i stand til at udføre brugergrænsefladens operationer vil med andre ord være mulige.

Mest ønskeligt var det naturligvis med en slags "standard" ramme for udvikling af DIIUS systemer, således at alle modeller af undervisningsprogrammer var i overensstemmelse på det operationelle niveau. Der tænkes her på standard i den betydning, at dette begreb anvendes i ISO's OSI model

for en datakommunikationsmodel, specielt for de øvre lag.

På det formelle og reelle niveau, hvor der er tale om transformation af data i forskellige formater og af forskellig type, vil det normalt være muligt at konstruere et interface.

Nødvendigheden for fastlæggelse af en standard på det operationelle niveau er derfor af afgørende betydning som grundlag for en konstruktion af systemer, der lader sig anvende i én sammenhæng.

4.4. Infra-struktur programmellet.

Infra-struktur programmellet (ref. 15) anvendes ved design, konstruktion, programmering og afprøvning af supra- og intermediær-struktur programmellet.

Infra-strukturen udgør også fundamentet for integrationen af de funktionelle moduler i supra-struktur programmellet.

Infra-strukturen, set på den måde, udgør endvidere fundamentet for det overliggende lag, idet det stiller basale værktøjer til rådighed. Det kunne f.eks. dreje sig om en grafisk kerne eller et databasesystem. I denne situation foretages der egentlige kald til infra-strukturen under udførelse af supra-struktur programmerne.

Infra-struktur programmel kunne f.eks. omfatte:

(1) Programmeringssprog
(2) Grafiske kerner
(3) Databasesystemer
(4) Kommunikationssystemer
(5) Brugergrænseflade systemer
(6) Lænke- og ladeprogrammer, debuggere, assemblere etc.

De funktionaliteter som infra-struktur programmellet omfatter er: værktøjer til opbygning, integration og understøttelse af de funktionelle moduler i intermediør- og supra-struktur programmellet.

Den funktionelle opdeling af infra-strukturen svarer til den vi har set for supra-strukturen og omfatter, som vi tidligere har set, en opdeling i tre funktionelle moduler:

(1) Brugergrænsefladem odul
(2) Anvendelsesmodellen
(3) Lagringsmodulet

Vi skal i det følgende se, hvad infra-struktur programmellet omfatter i forbindelse med de tre funktionelle moduler.

4.4.1. Infra-struktur: Brugergrænseflademodul.

Udvikling af brugergrænseflade har hidtil været en opgave der ikke er blevet afsat specielle resourcer til, - det har blot været en mere eller mindre tilfældig del af selve udviklingsarbejdet!

Ideen om at opdele modellering og præsentation, har givet anledning til først udvikling af grafiske standard systemer og sidenhen til egentlige brugergrænseflade systemer til understøttelse af brugergrænseflade udvikling. (UIMS: User Interface Management Systems).

Grafiske systemer til støtte for udvikling af brugergrænseflader bringer os et stykke ad vejen, men ikke langt nok. F.eks. savnes muligheden for en virkelig åben og dynamisk brugergrænseflade, der i høj grad lader sig justere efter individuelle ønsker og behov.

F.eks. giver Apple/MacIntosh og Digital Research/GEM (Graphics Environment Manager) på mange måder en tilfredsstillende brugergrænseflade, - men man er dog bundet til den præsentation og de kontrolmuligheder, der ligger fast indbygget i systemerne.

Ideen i Mac/GEM konceptet er bl.a., at der kan udvikles programmer med en ensartet brugergrænseflade. Det giver den fordel, at brugeren umiddelbart er fortrolig med systemet. Dette udelukker dog på ingen måde, at brugergrænsefladen evt. kunne tilpasses individuelle ønsker. Men denne mulighed for dynamisk udvikling af brugergrænseflade findes ikke i Mac/GEM i den udstrækning der her tænkes på.

Brugergrænseflade systemer, UIMS (User Interface Management System) er et godt skridt på vejen til at udbygge disse muligheder.

4.4.2. Infra-struktur: Anvendelsesmodel.

Anvendelsesmodellen udgøres bl.a. af det programmel, der fastlægger den formelle repræsentation af anvendelsesmodel-

lens data. Infra-struktur programmel til at understøtte opbygningen af denne formelle beskrivelse: datastrukturer og abstrakte datatyper, findes i en række højniveau programmeringssprog.

Muligheden for at konstruere generelle uafhængige anvendelsesmodeller, er ikke tilstede idag.

Der findes i operativsystemerne hjælpemidler til at understøtte det reelle niveau, - dvs. til at understøtte håndtering af de data, der tilhører den aktuelle opgave. Operativsystemerne indeholder f.eks. faciliteter til virtuel lagring, der giver mulighed for at arbejde med store datamængder. Samtidig hermed giver den teknologiske udvikling til stadighed forøgede muligheder for adgang til lagerplads, til faldende pris.

Aksiomerne udgør det operationelle niveau i anvendelsesmodellen, og dermed de operationer, der kan udføres på anvendelses modellen. Infra-struktur programmel der kan definere disse primitive operationer på akstrakte datatyper findes ikke i dag. Måske vil femte-generationssprog ("descriptive languages") omfatte disse muligheder (ref. 15).

4.4.3. Infra-struktur: Lagringsmodul.

Database systemer (DBMS: Data Base Management System) findes i dag til understøttelse af lagring og genfinding af data i lagringsmodulet.

Et moderne databasesystem omfatter endvidere muligheden for at specificere det interface imellem anvendelsesmodellen og lagringsmodulet, som er beskrevet tidligere.

4.5. Intermediær-struktur programmel.

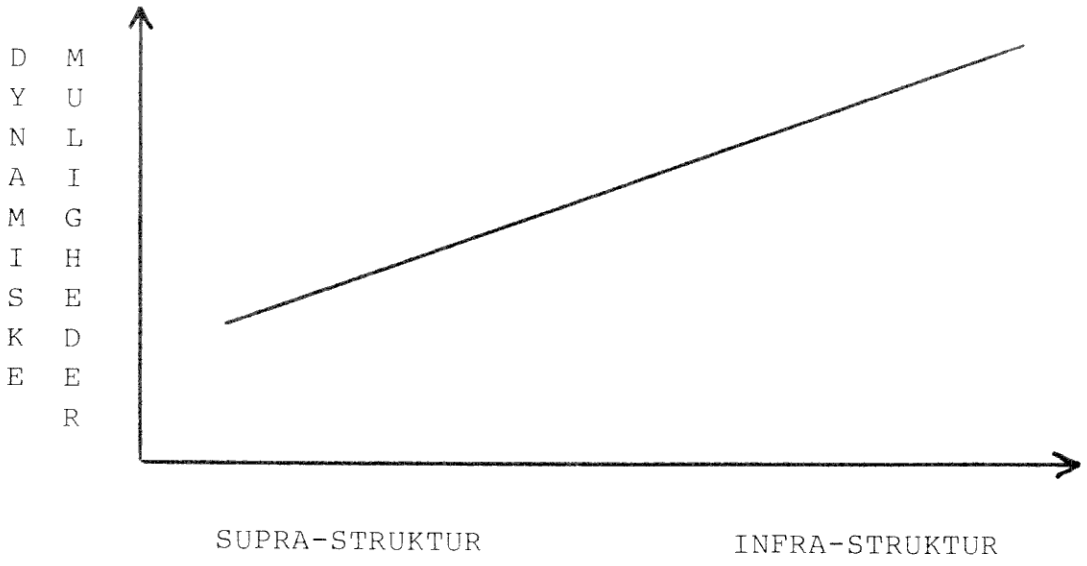
I dette afsnit skal det vises, at det ved hjælp af et strukturelt lag, den intermediære-struktur beliggende imellem infra- og supra-strukturen, er muligt at tilføre systemet åbenhed og dynamik.

Åbenhed/dynamik anvendes her i betydningen, mulighed for at foretage ændringer på selve DIIUS. Ændringerne kunne f.eks. omfatte en tilpasning af brugergrænsefladen efter individuelle ønsker, eller f.eks. en tilpasning af de mulige operationer/kommandoer, hvormed anvendelsesmodellen kan ændres. Ændringerne kunne også omfatte muligheden for at ændre på de faktiske aksiomer til påvirkning af modellen, således at der f.eks. oprettes nye aksiomer.

4.5.1. Dynamiske muligheder i DIIUS.

Indledningsvis skal vi se på et par parametre, der er karakteriserende for det DIIUS, der er beskrevet indtil nu. Endvidere vil tre eksempler belyse problemerne omkring åbenhed/dynamik.

Fordelingen af dynamiske muligheder i det hidtil beskrevne DIIUS kan skitseres således (figur 4-17):

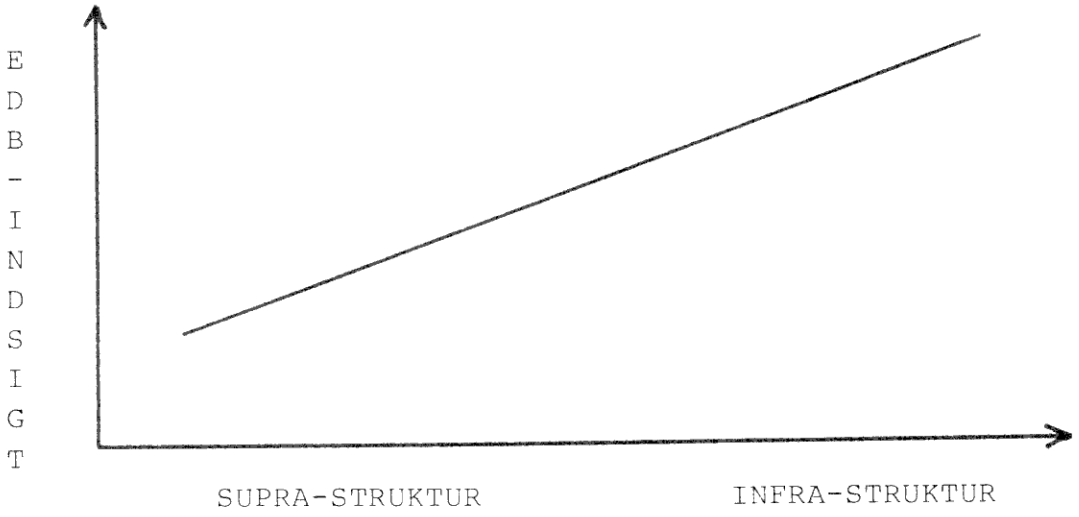


Figur 4-17: Supra<>Infra: Dynamiske muligheder.

Med dynamiske muligheder menes her den totale mængde af tilstedeværende muligheder for at ændre på et givet DIIUS, -

denne mængde vil naturligvis begrænses, når man bevæger sig imod et mere specifikt system. Her bevæger vi os fra de generelle værktøjer i infra-strukturen til realisationen af det specifikke system i supra-strukturen.

Samtidig sker der, desværre (!), en markant ændring i kravet til EDB-mæssig indsigt på systemniveau for brugeren af henholdsvis supra- og infra-struktur programmellet (figur 4-18):



Figur 4-18: Supra<>infra: EDB-indsigt.

Problemet tangerer det klassiske problem i system design og konstruktion: hvorledes sikres det, at den model der udvikles, faktisk svarer til brugerens model. Brugeren af DIIUS vil f.eks. kun have en EDB baggrund, der sætter ham i stand til at anvende supra-struktur programmellet, dvs. det konstruerede DIIUS. Modellen er udviklet v.h.a infra-struktur programmel. Spørgsmålet er derfor om den udviklede model svarer til brugerens ønske.

Her ønsker vi at undersøge mulighederne for at beskrive en konceptuel ramme for et system, et DIIUS, der lader sig justere efter slutbrugerens ønsker.

Begrebsmæssig ramme for et undervisningssystem.

Situationen er den, at der kræves systemprogrammører til konstruktion af DIIUS ved anvendelse af infra-struktur programmel. Slutbrugeren vil vi definere som den bruger, der kun har kendskab til anvendelsen af det færdige system.

Et ønske om maksimale dynamiske muligheder for at kunne ændre eller tilpasse et givet DIIUS, må med andre ord afvejes overfor hvor mange resourcer, man ønsker at tildele brugeruddannelse. Et kompromis synes at være en mulighed, - den mulighed der vælges i lignende situationer.

Inspireret af ref. 18 vil vi introducere begrebet en "specialistbruger". Han skal f.eks. kunne konstruere brugergrænsefladen for andre brugere efter individuelle ønsker og behov og foretage en tilpasning fra de operationelle muligheder, der gives i brugergrænsefladen, til anvendelsesmodellens aksiomer. Han skal kunne foretage ændringer i opbygningen af anvendelsesmodellen.

Specialistbrugeren tænkes at være en person, der ikke har nogen egentlig EDB faglig baggrund, men faktisk arbejder med undervisningssystemet på lige fod med andre operationelle brugere. Derudover skal han besidde lidt mere indsigt i og have nogen interesse i EDB på systemniveau. Han skal blot være den person, der ofte vil kunne findes i organisationen, som er i stand til at sætte sig ind i lidt mere avancerede operationer med systemet.

Det vil være nødvendigt at stille programmel, der befinder sig på et strukturelt niveau imellem infra- og suprastrukturen, til rådighed for specialistbrugeren, - det er netop intermediær-struktur programmellet.

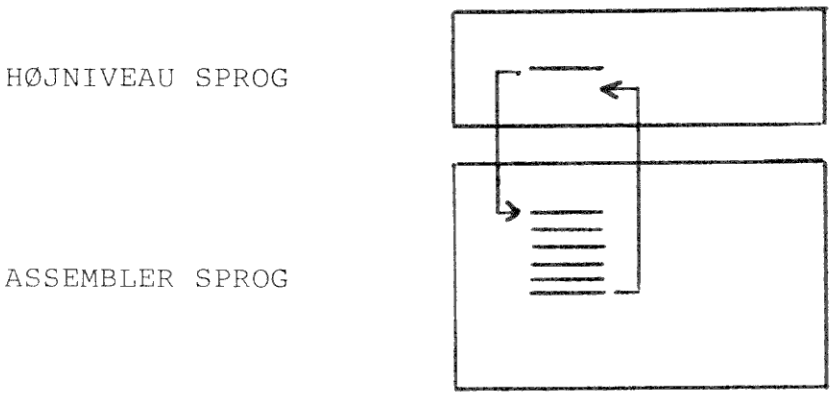
Dette programmel skal indeholde et værktøjssæt, der befinder sig på et mellemniveau med hensyn til, dels hvor kompliceret det er at anvende, og dels med hensyn til hvilke muligheder det giver for at ændre opbygningen af et givet DIIUS.

De følgende eksempler vil belyse hvilke muligheder der f.eks. kunne være ønskelige at finde i intermediær-struktur programmellet:

Begrebsmæssig ramme for et undervisningssystem.

4.5.1.1. Eksempel 1: Højniveau-sprog/assembler.

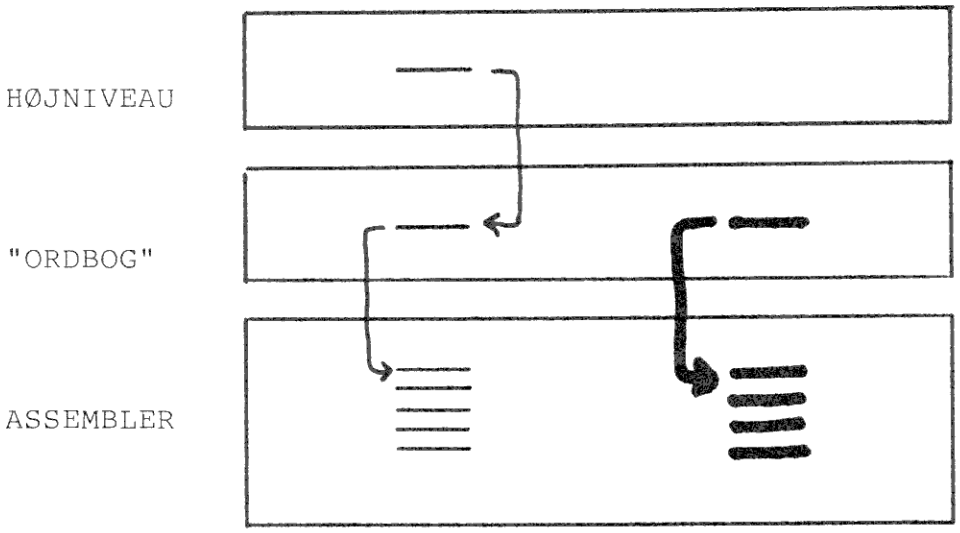
Som bruger af et højniveau sprog har man en række sproglige faciliteter til sin rådighed. Disse muligheder er fastlagt dels i sprogets definition dels i den faktiske implementation. Højniveau sproget bygger på en virtuel maskine, der er implementeret på assembler niveau (figur 4-19):



Figur 4-19: Eksempel - Højniveau..Assembler.

Hvis situationen nu var den, at brugeren af højniveau sproget ønskede at få udvidet sproget med f.eks. en ny kommando, hvilke muligheder havde han så? Hvis han ønsker udvidelsen som en reel udvidelse af sprogets implementation på assembler niveau, vil han i praksis oftest ikke have nogen mulighed derfor!

Hvis spoget imidlertid var et sprog som FORTH (ref. 3, 4 og 5), ville situationen være denne (figur 4-20):



Figur 4-20: Eksempel - Dynamisk højniveau sprog.

Ændringen, der kunne tænkes udført ved hjælp af intermediærstruktur programmel, er på denne figur og de følgende vist med fed skrift.

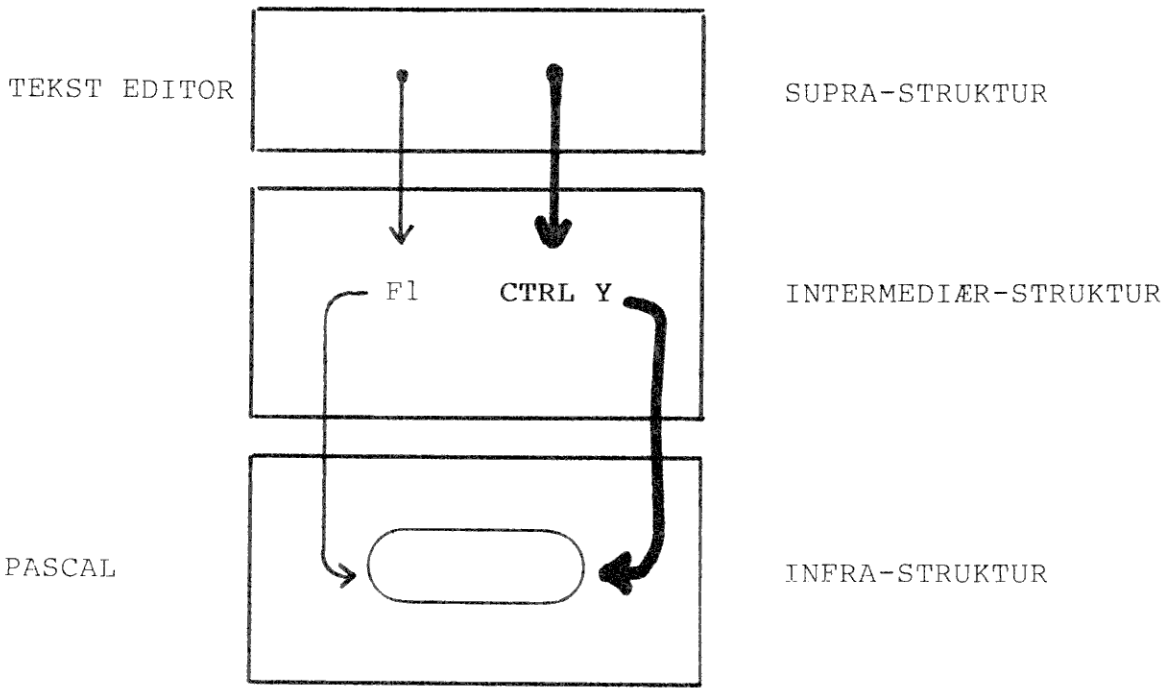
I sproget FORTH gives der mulighed for at udvide "ordbogen" med rutiner i maskinsprog og for at kombinere ord til nye ord. Oftere betegnes FORTH som et system til generering af sprog, end et egentligt højniveausprog i sig selv.

FORTH er altså et eksempel på et system, her et programmeringssprog, hvor der gives mulighed for at ændre i systemets funktion.

Herefter ses på mulighederne for anvendelse af intermediærstruktur i forbindelse med DIIUS:

4.5.1.2. Eksempel 2: Teksteditor.

I en teksteditor kunne f.eks. findes en kommando der bevirker, at "én linie slettes". Denne operation kunne f.eks. aktiveres ved tryk på funktionstast "F1", - hvis brugeren ønskede at funktionen blev aktiveret ved kommandoen "CTRL Y" kunne situationen illustreres således (figur 4-21):

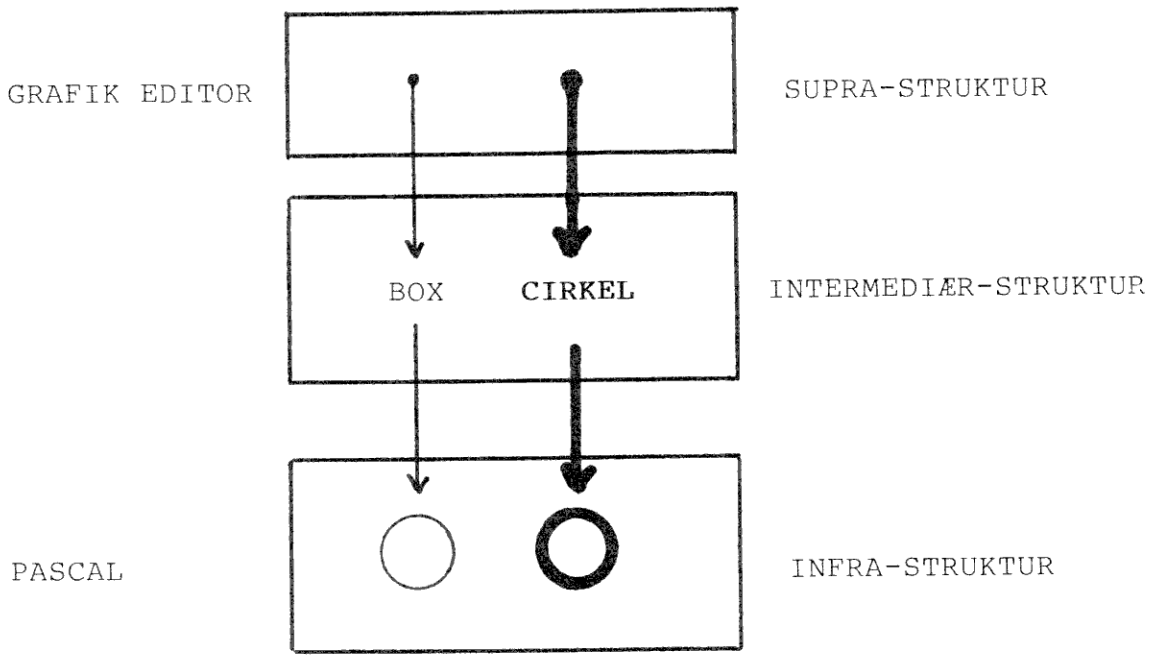


Figur 4-21: Eksempel - Tekst editor.

Som det antydes på figur 4-21, er det forholdsvis enkelt at foretage en mindre ændring i den intermediære-struktur. Ændringen bevirker, at en anden operation i brugergrænsefladen, via intermediær-strukturen, bindes til det samme aksiom i anvendelsesmodellen, - realiseret med infra-struktur programmel.

4.5.1.3. Eksempel 3: Grafisk system.

Lad os forestille os et grafisk system, hvor operationen "BOX" er kendt som en operation i brugergrænsefladen og er tilknyttet et aksiom i anvendelsesmodellen. Hvis situationen nu er den, at man ønsker en operation "CIRKEL" og at denne operation ikke findes som et aksiom i den eksisterende model, kunne situationen illustreres således (figur 4-22):



Figur 4-22: Eksempel - Grafik editor.

I denne situation ville det således ikke være muligt blot at ændre på tilpasningen fra brugergrænsefladen til anvendelsesmodellens operationelle niveau. Det ville være nødvendigt at ændre på selve opbygningen af anvendelsesmodellen, idet den skulle tilføjes et nyt aksiom.

Infra-struktur programmel til at uderstøtte denne mulighed, via intermediær-struktur programmel, findes ikke i dag, - men der er plads til begrebet i denne begrebsmæssige ramme for et DIIUS.

En anden og simplere situation ville være den, at anvendelsesmodellen faktisk indeholdt et aksiom "CIRKEL", men at ingen operation i brugergrænsefladen var tilknyttet dette aksiom. Situationen ville her være delvis parallel til situationen i eks. 2, her kræves blot, at der oprettes en operation i brugergrænsefladen, der derefter knyttes til det eksisterende aksiom "CIRKEL".

4.5.2. Udvidede strukturel opdeling.

Den udvidede og endelige strukturelle opdeling omfatter derefter tre moduler således:

(1) Supra-struktur programmel
(2) Intermediær-struktur programmel
(3) Infra-struktur programmel

- hvor den intermediære-struktur anvendes af specialistbrugeren til at foretage justeringer af systemet. Mulighederne for justeringer er illustreret i eksemplerne og kunne iøvrigt omfatte følgende indenfor de tre funktionelle moduler (figur 4-23):

BRUGERGRÆNSEFLADE MODUL	-ændring af kommandoer hørende til eksisterende operationer
	-oprettelse af nye operationer med tilhørende nye kommandoer
ANVENDELSES MODEL	-oprettelse af nye aksiomer
LAGRINGS MODUL	-ændring af operationelle muligheder i eksisterende database.

Figur 4-23: Intermediær-struktur.

Den totale systematik for et DIIUS er hermed beskrevet stykvis, og vi kan nu præsentere den samlede systematik:

4.6. Den samlede 3-D systematik for DIIUS.

Systematikken er hidtil beskrevet dimension for dimension ved hjælp af ialt 3 dimensioner. På figur 4-24 ses den samlede 3-D systematik, idet der er beskrevet en opdeling af

DIIUS ad tre akser: funktionel, strukturel og virtuel-niveau:

Som det ses af figur 4-24 er DIIUS ialt blevet opdelt i 15 moduler. Den samlede struktur er omfattende og måske vanskelig at overskue uden en grundig forståelse af det foregående. Flere interessante forhold kan imidlertid iagttages i den samlede struktur.

Kommunikationen foregår imellem funktionelle moduler og i supra-strukturen, på det reelle virtuelle niveau. Dvs. imellem modulerne, der kan angives ved koordinaterne (1,3,1), (2,3,1) og (3,3,1). Kun (3,3,1) er delvis synlig.

Da den faktiske kommunikation foregår i suprastrukturen, er det kun nødvendigt med interface på dette strukturelle niveau. Interfacegrænsen ses på figur 4-24. Dette interface kan imidlertid opfattes som om det foregår i tre virtuelle-niveauer. Der er således interface imellem modulerne: (1,3,1-3), (2,3,1-3) og (3,3,1-3). Disse moduler består faktisk af tre delmoduler, men vi har tidligt fastlagt den meget brede definition af termen modul.

Endnu et interessant aspekt skal nævnes:

Anvendelsessystemets behov tilfredsstilles af den totale samling af informationsbehandlings funktionaliteter i DIIUS, - repræsenteret ved hele den geometriske DIIUS-model i figuren. Disse funktioner kan betragtes fra forskellige synspunkter, - repræsenteret ved forskellige moduler i DIIUS-modellen.

Man kan opfatte systemet på den måde, at brugerens synspunkt er modulet (1,3,3), herfra ser han imod systemet. Systemets synspunkt er modulerne (1-3,1,1), herfra ser systemet imod brugeren.

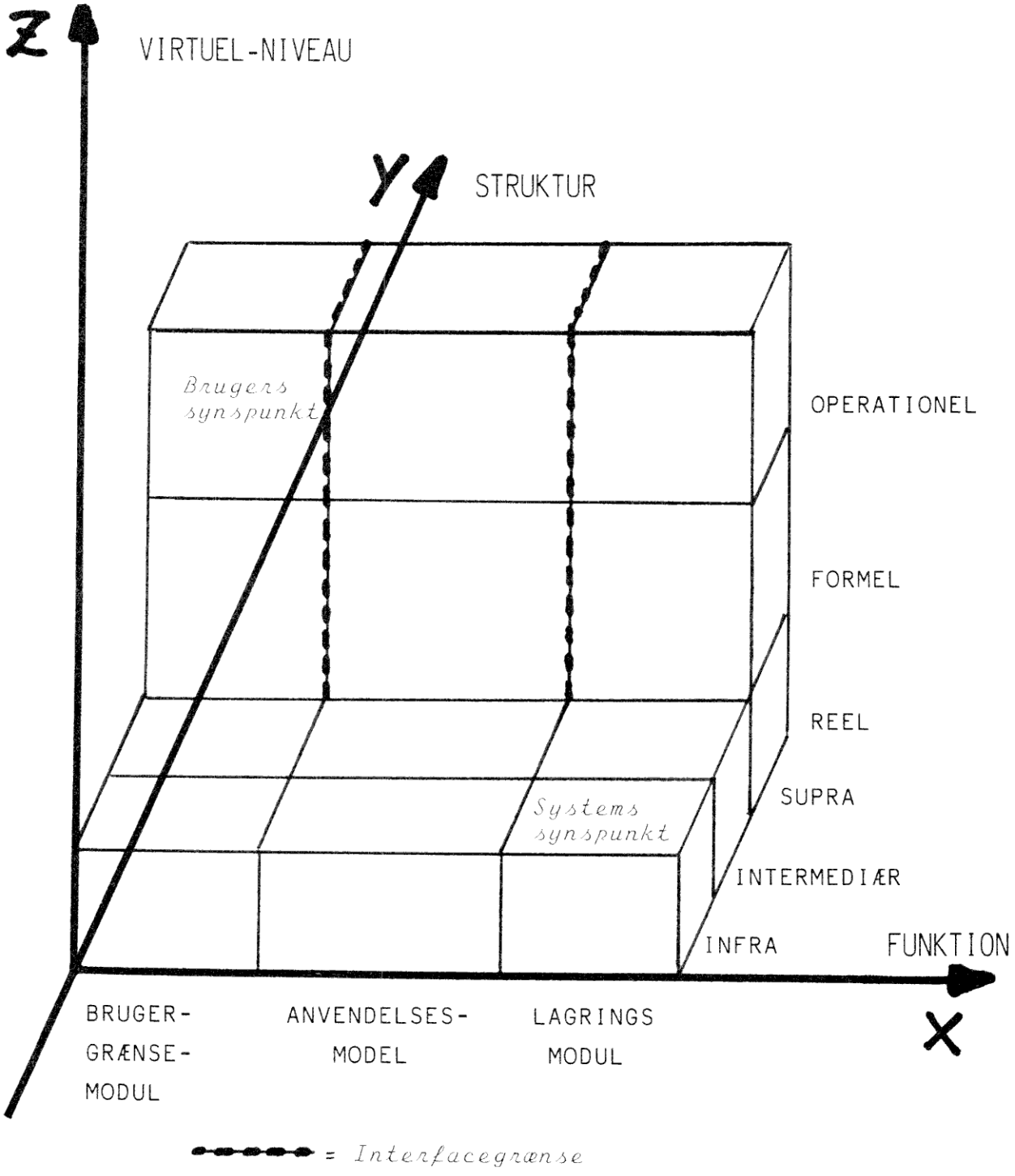
Fra brugerens synspunkt kan funktionaliteterne i DIIUS ses som en mængde brugerspecifikke funktioner, som systemet tlbyder brugeren via brugergrænsefladen.

Fra systemets synspunkt tilfredsstilles brugerens krav dels

ved hjælp af brugerspecifikke funktioner, dels ved systemspecifikke funktioner, der er specifikke for mange forskellige anvendelser. Alle funktioner er synlige for systemet!

Med andre ord, nogle funktioner er specifikke for den aktuelle anvendelse af DIIUS, og må specificeres af brugeren, - f.eks. v.h.a. intermediær-struktur programmel. Andre funktioner er fælles for mange forskellige anvendelser og findes til at understøtte bruger specifikke funktioner. De realiceres gennem funktioner i DIIUS, som kun er synlige på systemprogrammerings niveau. De specificeres ikke direkte af brugeren. Endelig findes systemspecifikke funktioner, der tilfredstiller brugerens krav med hensyn til udnyttelse af maskinel faciliteter tilgængelige for systemet.

Disse forhold illustrer samtidig det velkendte systemudviklingsprincip om, at programmelsystemet skal tilnærme system og bruger.



Figur 4-24: Den samlede 3-D systematik for DIIUS.

Begrebsmæssig ramme for et undervisningssystem.

4.7. Undervisningsprogrammet og modellen.

I det foregående er beskrevet systematikken for et undervisningssystem, der anvendes til at udvikle et undervisningsprogram, og som bl.a. omfatter en anvendelsesmodel. Hvordan sammenhængen er imellem undervisningsprogram og anvendelsesmodel vises i det følgende.

Jeg vil indlede med at drage en parallel til den situation, hvor der skal udvikles et traditionelt EDB-program i et højniveausprog som f.eks. Pascal eller Basic.

Modellen er også i denne situation en logisk model af et stykke virkelighed, som vi ønsker at udvikle en datamatisk løsning for.

Hvis det drejer sig om udvikling af et pascal program, udgøres det kørende program af kildeteksten oversat til maskinkode. Det oversatte pascalprogram indeholder så at sige en realisation af modellen. Men derudover indeholder det kørende program en række andre funktioner. Disse funktioner er indeholdt i det såkaldte køretidsmodul (eng. "run-time-package"). Køretidsmodulet udgør en del af selve udviklingsmiljøet, i dette tilfælde pascalsystemet. Køretidsmodulet lægges ind i maskinkodeprogrammet under oversættelsen. Køretidsmodulet håndterer forskellige forhold under kørsel af programmet f.eks. forhold som køretidsfejl eller input/output fejl.

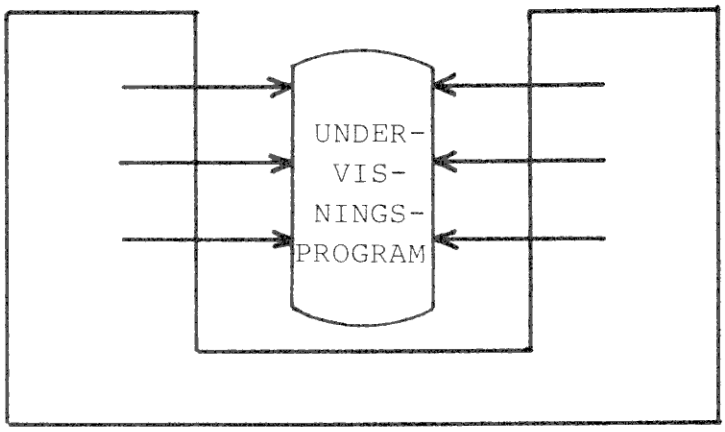
Er der tale om udvikling af et program skrevet i et fortolkende sprog, som f.eks. basic, er forholdet et noget andet. I denne situation udgøres selve programmet af, hvad der faktisk svarer til kildeteksten for pascal programmets vedkommende. Men når programmet skal køres, er det nødvendigt at hele udviklingssystemet er tilstede. Dvs. brugeren skal være i besiddelse af hele udviklingsmiljøet, f.eks. fortolkeren, editoren, load/save/slet funktioner etc., som således kan opfattes som køretidsmodulet i denne situation.

Ud fra denne parallel ser vi, at programmet ikke blot er en realisation af de funktionaliteter, der kan uddrages af modellen. Programmet indeholder yderligere en række funktio-

naliteter, der kan siges at hidrøre fra udviklingssystemet.

Lad os nu vende os imod teorien for DIIUS og her specielt anvendelsesmodellen. Anvendelsesmodellen omfatter, som tidligere behandlet, dels modellen af undervisningsforløbet, realiseret ved undervisningsprogrammet, dels de tilhørende aksiomer.

Som det ses af figur 4-25, kan vi opfatte det, som om undervisningsprogrammet "fødes" i anvendelsesmodellen:



ANVENDELSESMODEL MED AKSIOMER

Figur 4-25: Undervisningsprogrammet "fødes".

Anvendelsesmodellen omfatter således både undervisningsprogrammet, og de tilhørende aksiomer, der anvendes ved konstruktion, opbygning, ændring eller behandling af undervisningsprogrammet, - illustreret ved pilene på figur 4-25.

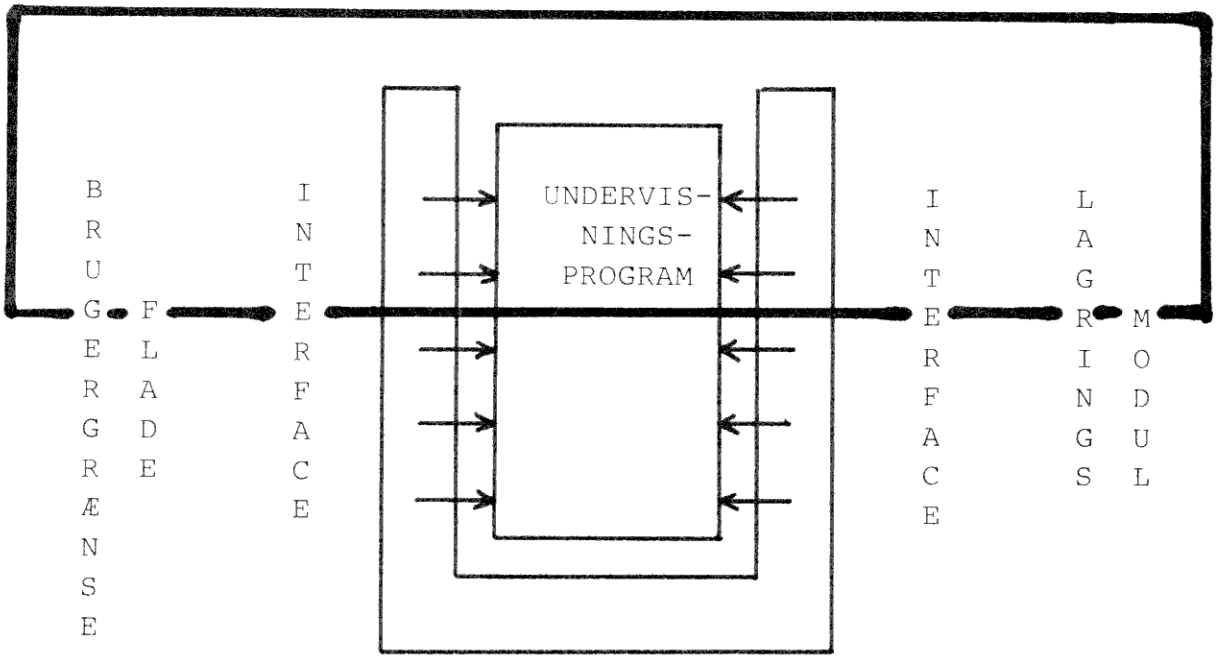
Hvor stor en del af vort DIIUS skal det kørende undervisningsprogram omfatte? Ud fra anvendelsesmæssige kriterier vil det også i denne situation være ønskeligt med stor grad af åbenhed/dynamik. Det vil være ønskeligt, at der er fri mulighed for at lade en større eller mindre del af DIIUS indgå i det kørende undervisningsprogram.

Begrebsmæssig ramme for et undervisningssystem.

Et eksempel: I SOFUS/COMUS systemet er det muligt at knytte hele udviklingsmiljøet, bestående af samtlige editorer med tilhørende brugergrænseflade, til selve undervisningsmaterialet. Der er imidlertid også mulighed for kun at give eleven adgang til den del af systemet, der håndterer selve afviklingen.

Med baggund i disse betragtninger vil det således være ønskeligt, at der er frihed til at lade en større eller mindre del af DIIUS indgå sammen med undervisningsprogrammet i det system, som vi kan betegne det kørende undervisningsprogram.

Denne situation kan illustrere således:



Figur 4-26: Det kørende undervisningsprogram.

Det kørende undervisningsprogram kan således omfatte en større eller mindre del af modulerne vist på figur 4-26. F.eks. det angivne område.

Begrebsmæssig ramme for et undervisningssystem.

I det følgende skal vi se på mere konkrete værktøjer, der kunne udfylde systematikken for DIIUS.

5. IDENTIFIKATION OG PLACERING AF VÆRKTØJER

I dette kapitel vil kriterier for identifikation af specifikke datalogiske værktøjer, der synes at være relevante i en undervisnings anvendelsessammenhæng, blive beskrevet.

Ved datalogiske værktøjer forstås programmer/værktøjer der kan indpasses i den tidligere beskrevne systematik for DIIUS, og som endvidere er konstrueret ud fra de kriterier der er anført i følgende systematik.

Vi skal se, hvordan det er afgørende på hvilket niveau, det enkelte værktøj befinder sig. Ved niveau forstås i denne sammenhæng en skala, der går fra rene datalogiske værktøjer til rene undervisningsmidler. I den forbindelse vil det blive antydet, på hvilket niveau kendte værktøjer befinder sig i dag.

Det er ikke målet at identificere en udtømmende liste med værktøjer. En eventuel liste kan og vil på ingen måde være udtømmende. Det bør netop være et mål, at man har mulighed for at arbejde frit med konstruktion af nye værktøjer, indenfor den ramme der udgøres af det DIIUS, hvis systematik er beskrevet tidligere. Det skal derfor være muligt løbende at udvide "værktøjssamlingen" med nye værktøjer.

5.1. Systematik.

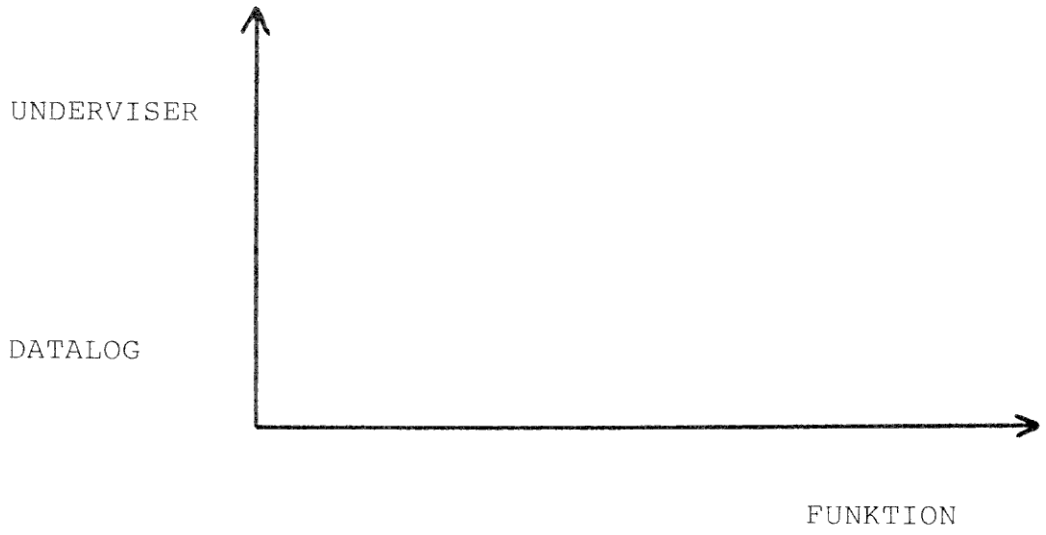
Værktøjerne indplaceres på en flade udspændt af to akser: en vertikal og en horisontal:

(1) Den horisontale dimension angiver mulige værktøjer. Denne dimension svarer til den funkticnelle dimension ved beskrivelsen af systematikken for et DIIUS.

(2) Placeringen ad den vertikale akse er af central betydning i denne sammenhæng. Denne dimension angiver i hvor høj grad, der er tale om et værktøj der knytter sig til datalogens begrebsverden eller et værktøj der knytter sig til underviserens begrebsverden. Dermed er denne dimension i nogen grad parallel til den strukturelle dimension ved be-

skrivelsen af systematikken af et DIIUS. Her fokuseres der imidlertid på det "undervisningsmæssige" kontra det "datalogiske" aspekt ved værktøjet.

Værktøjer indplaceres således i disse to dimensioner (figur 5-1):



Figur 5-1: Systematik - placering af værktøjer.

Eksempel (1): Et generelt programmeringssprog kan defineres som et rent datalogisk værktøj, der ikke giver anledning til nogen praktisk begrænsning i den undervisningsmodel, der kan konstrueres med det.

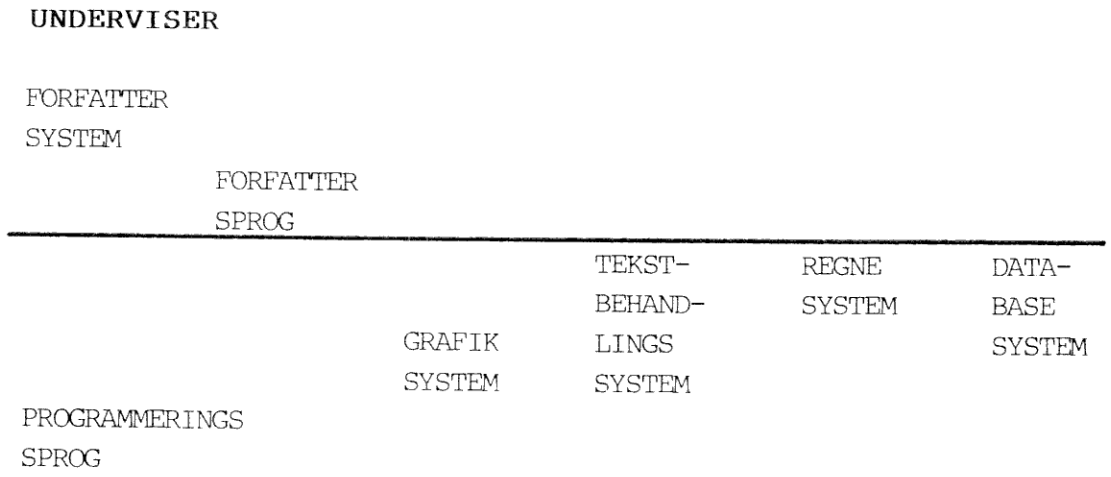
En indvending der ofte mødes herimod er, at enhver anvendelse af datamater i forbindelse med undervisningen udgør et bånd. Det er også korrekt, - men ikke under de forudsætninger jeg har gjort i indledningen til rapporten, hvor det er forudsat, at man faktisk har valgt at anvende en datamat i forbindelse med undervisning. Iøvrigt henvises til rapportens indledning for en nærmere afklaring af denne for rapporten meget væsentlige forudsætning!

Eksempel (2): Et forfattersystem eller et andet undervisningssystem, der kun behandler én specifik og begrænset model, kan betegnes som et værktøj der er stærkt knyttet til undervisning. Der er i dette tilfælde i høj grad foretaget

en række pædagogiske valg.

På figur 5-2 ses en række eksempler på værktøjer, der kendes i dag, og som i større eller mindre grad faktisk indgår i undervisningssystemer. Værktøjerne er skønsmæssigt placeret på den vertikale akse.

Begrebet værktøj er her anvendt i en noget bredere og mindre stringent betydning end den, vi senere skal nå frem til. Indtil videre må vi således acceptere at tage udgangspunkt i eksisterende systemer, der kan falde ind under et noget bredere defineret værktøjsbegreb.



Figur 5-2: Eksempler på placering af værktøjer.

Vi skal nu se, hvorledes det er muligt at identificere værktøjer ved de to kriterier, der er knyttet til den horisontale og vertikale akse.

5.2. Horisontal opdeling: Hvilke værktøjer.

I henhold til den netop omtalte placering af værktøjer ved hjælp af en horisontal og en vertikal akse, skal vi nu se på den horisontale placering, dvs. identifikation af værktøjer

Identifikation og placering af værktøjer

samt deres indbyrdes afgrænsning.

Der findes næppe nogen entydig definition af, hvorledes de enkelte værktøjer skal identificeres og adskilles. Under udviklingen af en systematik for DIIUS blev en mulig funktionel opdeling beskrevet både for det samlede DIIUS og for anvendelsesmodellen. Det var tanken, at denne opdeling skulle være rummelig og generel. Et minimumskrav vil derfor være, at et værktøj ikke placerer sig tværs over opdelinger i henhold til denne grundlæggende opdeling.

I det tilfælde at der sker en integration af en række delelementer, der kunne have eksisteret som enkeltelementer med en veldefineret grænseflade, sker der en vis fastlåsning af værktøjerne.

Et eksempel: TenCore systemet indeholer en ret anvendelig grafikeditor, der genererer en række koder, der er specifikke for TenCore. Problemet er imidlertid, at vi ikke er i stand til at anvende denne grafikeditor i en anden sammenhæng. Den eksisterer som en integreret del af TenCore systemet. Dette er ikke specielt en kritik af TenCore systemet, men snarere en kritik af måden at løse problemerne på!

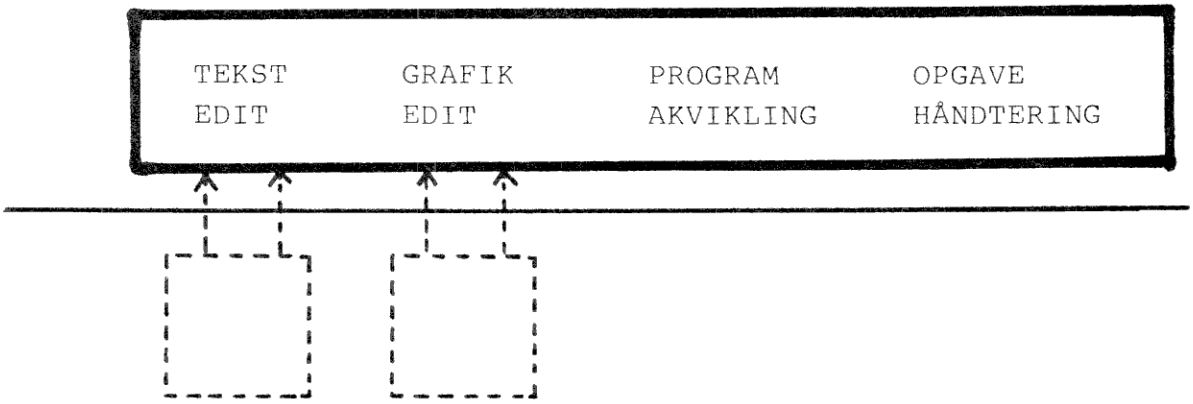
I dette eksempel kunne man ønske sig at grafikeditoren var frigjort fra resten af systemet og dermed kunne anvendes som et generelt værktøj til generering af grafiske billeder.

Det anvendte design, hvor der genereres kode under det interaktive arbejde med editoren, kan diskuteres. Det skal dog bemærkes, at i det tilfælde editoren var designet som et generelt værktøj, havde det været at foretrække, at der f.eks. genereredes koder til et mere generelt grafisk operativsystem, f.eks. GKS (Graphics Kernes System) eller GSX (Graphics System Extension), der er meget udbredt på mikrodatamater.

På figur 5-3 ses den negative effekt ved integration af værktøjer, eller ved design af værktøjer, der omfatter for mange komponenter, der "naturligt" kunne opdeles i mindre:

Identifikation og placering af værktøjer

UNDERVISNING



DATALOGI

Figur 5-3: Blokering af datalogiske værktøjer.

Det er illustreret, hvorledes værktøjerne teksteditor og grafikeditor, fastlåses eller blokeres, når de integreres eller kobles tæt sammen med andre værktøjer, specifikke for f.eks. et bestemt undervisningsforløb. Det kunne f.eks. dreje sig om moduler, der håndterer en helt speciel afvikling af undervisningsprogrammet eller afvikler specielle opgavetyper i forbindelse med undervisningsforløbet. Et eksempel er SOFUS/COMUS, hvor der findes en ordbogsfunktion, der imidlertid kun kan anvendes som en del af informationssystemet. Vi støder her på en effekt, der samtidig griber ind i den vertikale placering, som vi senere skal vende tilbage til.

Vurderingerne af, hvor grænsen skal trækkes imellem de enkelte moduler minder om den sondring, der skal foretages, når et modul i et EDB program skal identificeres. Opdelingen vil altid bero på en vurdering.

I denne sammenhæng finder jeg det derfor afgørende, at det som et minimumskrav kræves, at moduler, der indeholder egentlige overvejelser over specifikke undervisningsmæssige problemer i deres design, ikke kobles sammen med andre moduler.

Identifikation og placering af værktøjer

I dag er situationen stort set den, at det under alle omstændigheder vil være et stort skridt på vejen, hvis man blot begyndte at bevæge sig væk fra integrerede systemer, - hvor alt er fastlåst i ét system!

Det er efter min overbevisning ikke nogen løsning på problemerne omkring udvikling af undervisningsprogrammer, at et udviklingsværktøj til undervisningssystemer, f.eks. et forfattersprog eller forfattersystem, har en vis åbenhed. F.eks. mulighed for at kunne aktivere andre programmer under det anvendte styresystem, eller for forfattersprogets vedkommende indeholder sprogfaciliteter, der giver en åbning for en mere generel programmering ved hjælp af det pågældende system.

Konklusionen på denne diskussion er følgende anbefaling ved afgrænsning af værktøjer:

(1) Moduler, der indeholder design, som bygger på specifikke undervisningsmæssige overvejelser bør adskilles helt fra rene datalogiske moduler.

(2) Funktioner, der kunne tænkes anvendt i helt forskellige undervisningsmæssige sammenhænge, bør tilhøre separate moduler.

(3) De datalogiske moduler bør afgrænses, således at de indeholder beslægtede funktioner, med hensyn til hvilken behandling der foretages, og således at de udviser en klart defineret grænseflade med hensyn til udveksling af data.

(4) Identifikation af separate værktøjer vil altid bero på et skøn. Et skøn, der vil kunne gøres mere kvalificeret, når der gøres erfaringer og udføres anvendelsesforsøg indenfor dette område.

5.3. Vertikal placering: Undervisning <--> datalogi.

Hensigten med anvendelsen af datalogiske værktøjer er, at de skal bygge bro imellem maskinen og visionen om undervisningsprogrammet. Den række værktøjer, der skal bygge bro fra

datalogiske problemstillinger til undervisningsmæssige skal gradvist nærme sig undervisningsprogrammet begrebsmæssigt.

Vi har tidligere set, hvorledes infra-struktur programmellet mest knytter sig til en ren datalogisk anvendelse og hvorledes man via intermediær-struktur og supra-struktur nærmer sig det programmelsystem, et DIIUS, der kan anvendes af undervisere ved opbygning af undervisningprogrammer.

Det er således supra-struktur programmellet, der henvender sig til læreren. Spørgsmålet er derefter, hvor langt skal de værktøjer, der udgør supra-struktur programmellet, række sig imod læreren.

Det er vigtigt at få indplaceret det enkelte værktøj på det ideelle niveau, dvs. løftet så langt op, at det lader sig anvende i en undervisningsmæssig sammenhæng uden for store vanskeligheder. Her tænkes på, hvor meget der kræves på EDB-systemniveau af den lærer der er systemudvikler. På den anden side, skal det også placeres så langt nede, at det er frigjort fra undervisningsmæssige overvejelser i den betydning, som vi tidligere har været inde på, - således at datalogen ikke fristes til at binde læreren unødvendigt.

Værktøjer, der befinder sig for lavt på datalog-underviser aksen, vil være mere eller mindre uanvendelige for læreren. Værktøjerne skal bygges til et niveau, der netop tangerer underviserens begrebsverden.

Bliver niveauet for højt, opstår to problemer: værktøjerne fastlåses i specifikke undervisningsmæssige sammenhænge. De er ikke længere generelle værktøjer, - ikke længere datalogiske værktøjer. Samtidig stiger risikoen for at disse værktøjer, konstrueret af dataloger, ikke beskriver visionerne af den model, som underviseren har i tankerne.

Problemet kunne f.eks. skematisk eksemplificeres således (figur 5-4):

Identifikation og placering af værktøjer

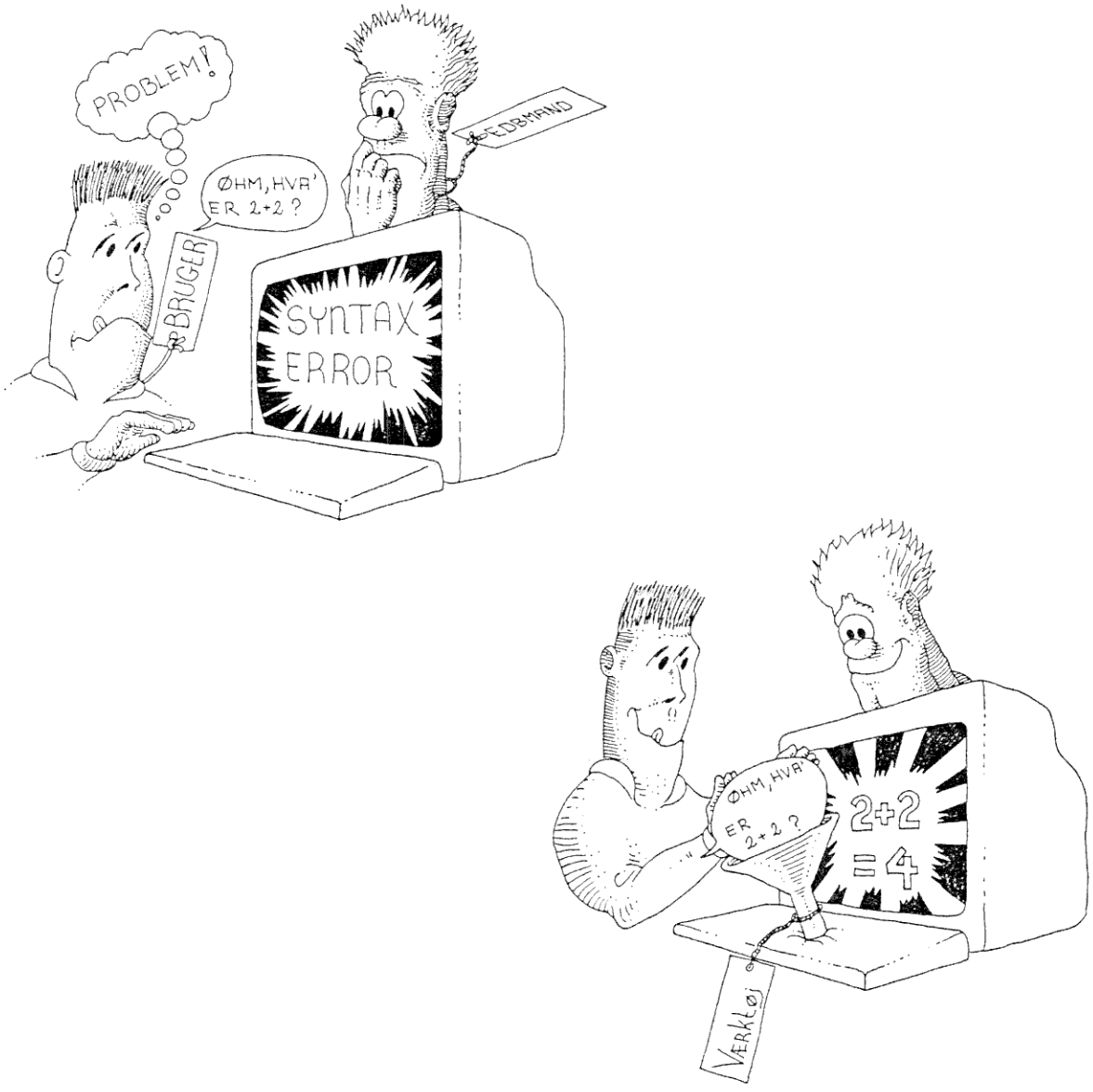
Figur 5-4: Visions kommunikation?

Helt så galt går det vel ikke, - men en tilstrækkelig tilnærmelse imellem infra-struktur, der jo f.eks. kunne omfatte højniveau sprog, grafiske kerner, databasesystemer og lig-

nende, og underviserens vision om et undervisningssystem er nødvendig.

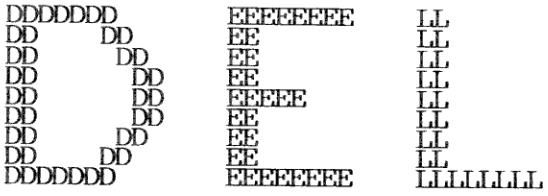
Den ideale placering af datalogiske værktøjer er derfor kort sagt: så højt at de kan anvendes af læreren, og så lavt at de stadig kan betegnes som generelle værktøjer.

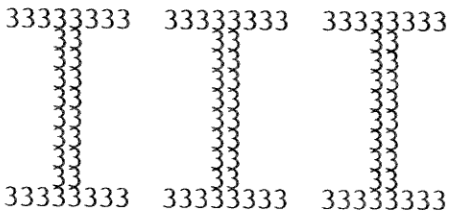
Den ønskede situation kan illustreres således (figur 5-5):



Figur 5-5: Tilnærmelse til brugerens begrebsverden.

Identifikation og placering af værktøjer





E K S I S T E R E N D E D A T A L O G I S K E V Æ R K T Ø J E R

6. EKSISTERENDE DATALOGISKE VÆRKTØJER.

Som nævnt tidligere, må målet for datalogen være at kunne stille værktøjer for design og konstruktion af undervisningsprogrammer til anvendelse i forbindelse med undervisning til rådighed for læreren. Det må derefter være datalogens opgave at informere og præsentere læreren for værktøjerne. Hvilke værktøjer findes der? Spørgsmålet afklares i dette kapitel. I næste kapitel præsenteres en anvendelse af et intermediær-struktur værktøj, der kan anvendes ved tilpasning af brugergrænsefladen efter individuelle ønsker.

Der findes naturligvis undervisningsprogrammer, ligesom der findes undervisningssystemer, - som f.eks. beskrevet i del 1. Men vi vil her interessere os for værktøjer til udvikling af undervisningsprogrammer i betydningen fastsat i del 2. Dvs. datalogiske værktøjer, der er i overensstemmelse med kravene til (1) hvorledes værktøjer kan identificeres og placeres (kapitel 5). Endvidere må vi (2) indpasse eventuelle værktøjer i DIIUS systematikken (kapitel 4). Indpasningen i systematikken giver god forståelse for hvilke muligheder, der findes i dag og hvilke muligheder, der i særdeleshed ikke findes.

Som beskrivelsesramme anvendes den funktionelle hovedopdeling i brugergrænsefladem odul, anvendelsesmodel med aksiomer og lagringsmodul, - samt den strukturelle opdeling i supra- og intermediær-struktur programmel.

Når vi her ønsker at tale om eksisterende anvendelsessystemer, vil vi ikke interessere os for infra-struktur programmellet, der anvendes til opbygning og understøttelse af disse systemer.

De centrale moduler i et undervisningssystem er anvendelsesmodellen på supra- og intermediær-struktur niveau. Disse programmelmoduler eksisterer ikke! Problemet er, som nævnt tidligere, at der i dag ikke findes infra-struktur programmel til modellering, idet der ikke findes programmel, der kan definere aksiomerne, dvs. de primitive operationer, på abstrakte datatyper.

Situationen minder om den situation, som databaser befandt sig i før end der var udviklet en standard for opbygning af databaser. Da CODASYL arbejdsgruppens rapport fremkom i 1969, var grundlaget lagt for en standard til opbygning af databaser.

I dag findes der veludviklet teori for databasesystemer og til dels for grafiske systemer. Det har bevirket, at der er blevet udviklet velfungerende systemer indenfor disse to områder. Systemerne omfatter supra- og intermediær-struktur programmel. Forholdsvis let tilgængelige systemer findes meget udbredt på mikrodatamater.

Endvidere har værktøjer som tekstbehandlingssystemer og regnesystemer ("regneark", eng.: "spreadsheet") vundet stor udbredelse med mikrodatamaternes fremkomst. De er via deres udbredelse til en meget stor og bred skare af brugere blevet udviklet til systemer, der er meget åbne, let tilgængelige og velfungerende. Omvendt må det bemærkes, at det ikke er tilfældigt at disse systemer har fundet så stor udbredelse, - årsagen må antages at være særlige kvaliteter ved systemerne.

De generelle datalogiske værktøjer, der kan stilles til rådighed for læreren er derfor: databasesystemer, grafiske systemer, tekstbehandlingssystemer, og regnesystemer.

Disse fire generelle systemer lader sig endvidere forholdsvis let integrere*), således at systemerne kan arbejde på fælles data, og således at de behandlede data kan præsenteres samlet.

*)Af kommercielle integrerede systemer kan f.eks. nævnes: Lotus-1-2-3, og IBM Assistent serien. GEM (Graphics Environment Manager) der omfatter tekstsystem og grafisk system med egentlige tegnefaciliteter.

Som undervisningssystemer er defineret i denne sammenhæng, udgør disse værktøjer således, hvad der kan stilles til rådighed for læreren. Det er datalogens opgave at informere læreren om disse muligheder! Endvidere må det naturligvis være datalogens opgave at arbejde med udvikling af egentlige

undervisningssystemer, der også omfatter funktionaliteter indenfor anvendelsesmodellen med aksiomer.

Jeg skal ikke her give den beskrivelse af værktøjerne, der f.eks. kunne videregives til en underviser. Jeg vil blot fremlægge nogle af de muligheder, der ligger i disse værktøjer set i overensstemmelse med den beskrevne systematik for DIIUS, - der jo er hovedemnet for dette arbejde.

De fire værktøjer: databasesystemer, grafiske systemer, tekstbehandlingssystemer og regnesystemer indeholder i større eller mindre udstrækning intermediær-struktur programmel. Dvs. det er muligt for "specialistbrugeren", som vi introducerede under beskrivelsen af intermediær-struktur programmellet, at foretage mindre tilretninger af systemerne, før end de f.eks. anvendes af en større gruppe brugere.

På grund af værktøjernes åbenhed og generelle egenskaber kan datalogen informere om værktøjerne uden at skulle tage stilling til deres eventuelle anvendelse ved undervisning. Et eksempel viser bedst mulighederne. Formålet med eksemplet er blot at give et indtryk af de muligheder, der kunne indgå som en del af informationen til en lærer, - det er ikke hensigten at give en fyldestgørende information. Som en del af eksemplet indgår mere eller mindre konkrete anvisninger på, hvorledes datalogen forestiller sig at værktøjerne evt. kunne anvendes. Disse forslag er blot tænkt som inspiration, - det må være lærerens opgave at undersøge hvordan og til hvad værktøjerne skal anvendes, - hvis de skal anvendes.

6.1. Eksempel.

Lad os forestille os, at man er i besiddelse af en fil, der indeholder en database med oplysninger om en række lande. Ved hjælp af databasesystemet vil det være muligt at udtrække en mængde deloplysninger f.eks. omhandlende lande i norden. Disse data kunne derefter behandles af regnesystemet, hvor der kunne foretages passende beregninger. Resultatet af beregningerne kunne derefter behandles af grafiksystemet, hvor der kunne tegnes kurver og diagrammer over data. Udtrækket af databasen, resultatet af beregninger og kurverne

kunne behandles i tekstbehandlingssystmet, hvor alt materiale kunne redigeres sammen med tekst. Derefter kunne alt præsenteres på skærm eller på skriver.

Manuelt:	Indsamling af data
DATABASESYSTEM:	LAGRING - SØGNING - UDTRÆK
REGNESYSTEM:	BEREGNING
GRAFIKSYSTEM:	TEGNING - KURVE - DIAGRAM
TEKSTSYSTEM:	REDIGERING - UDSKRIVNING

Figur 6-1: Eksempel: Anvendelse af integreret system.

Før end denne anvendelse er mulig, skulle data være indsamlet, derudover kunne der være arbejdet en del med datasystemerne, således at processen i figur 6-1 var blevet forberedt, f.eks.:

1) Database oprettes, poster og felter defineres, nøglefelter bestemmes etc.

2) Udtræk af database fastlægges evt. og designes med hensyn til format o.l.

3) Beregningsmodel fastlægges i regnesystemet.

4) Grafisk behandling fastlægges: kurve/diagram type etc.

Det er faktisk muligt at fastlægge operationerne 1-4 på forhånd med de eksisterende systemer. Dette er muligt fordi de nævnte systemer omfatter intermediær-struktur programmel herfor. Dette arbejde kunne udføres af den tidligere omtalte "specialistbruger", en lærer der har sat sig lidt ekstra ind

i systemernes anvendelse.

6.2. Supra-struktur for undervisningssystemer.

Det eksisterende supra-struktur programmel, som vi har set indpasset i DIIUS systematikken samt systematikken for identifikation af værktøjer, er programmel, der er udviklet til kontorsektoren og administrative opgaver. Eventuel brug i forbindelse med undervisning er et biprodukt af forholdsvis generelle egenskaber.

Regnesystemerne er således primært designet med henblik på budget simulering, tekstbehandlingssystemerne med henblik på brev og rapport skrivning, grafiksystemerne med henblik på såkaldt "business"-grafik, databasesystemer med henblik på registre organiseret i poster. Derudover er de grafiske systemer i de seneste år udviklet til egentlige tegneredskaber.

Systemerne er med deres forholdvis generelle konstruktion, der bl.a omfatter intermediær-struktur programmel, i stand til at blive anvendt bredt, f.eks. også i undervisningsmæssige sammenhænge.

Der vil meget tænkeligt kunne findes andre systemer med tilsvarende generelle egnenskaber, udviklet for andre målgrupper, som også vil kunne anvendes i forbindelse med undervisning. Dette forhold er imidlertid ikke af afgørende betydning!

Det der savnes er en tilbundsgående undersøgelse af hvilke generelle værktøjer, der skal konstrueres for at opfylde lærerens behov og ønsker.

Set i relation til det beskrevne DIIUS er de eksisterende værktøjer blot "små glimt i mørket". Der savnes egentlige undervisningssystemer, der er i stand til at bearbejde en model af et datamatisk undervisningsforløb. De eksisterende værktøjer kunne meget vel indgå som byggestene i et undervisningssystem. Men næppe i den form de kendes i dag, hvor de til trods for muligheder for integration er for selvstæn-

dige og lukkede systemer.

Undersøgelse af lærerens behov og ønsker for generelle datalogiske værktøjer er et område, der burde underkastes grundig analyse.

6.3. Infra-struktur for undervisningssystemer.

Fundamentet for konstruktion af undervisningssystemer, realiseret ved supra- og intermediær-struktur programmel, er det dertil nødvendige infra-struktur programmel.

Vi har tidligere i afsnit 4.4 set, hvilket infra-struktur programmel der ønskes til konstruktion af undervisningssystemer. Nogle dele af dette programmel mangler helt, som vi også har set. Men spørgsmålet er derefter, hvor velegnet er det eksisterende programmel til konstruktion af de ønskede supra- og intermediær-strukturer.

F.eks. vil det ikke nødvendigvis forholde sig sådan, at eksisterende programmel til opbygning af databasesystemer er velegnet i denne sammenhæng. Hvilken type databaser har læreren behov for? Hvilken type programmeringssprog er velegnet til konstruktion af denne type programmel? Hvilke grafiske systemer? Dette er uafklarede spørgsmål.

På baggrund af analyser og afklaring af hvilket supra- og intermediær-struktur programmel der ønskes, vil der være behov for grundig analyse af hvilket infra-struktur programmel, der er velegnet i denne sammenhæng.

-oOo-

Det er særdeles begrænset, hvad der findes af eksisterende datalogiske værktøjer til udvikling af undervisningsprogrammer, - som er emnet for denne del af rapporten. Supra- og intermediær-struktur programmel for anvendelsesmodellen sav-

nes helt.

Der findes imidlertid, til anvendelse for mikrodatamater, et interessant intermediær-struktur programmel for brugrergrænsefladem odulet. Næste kapitel omfatter anvendelse af dette intermediær-struktur programmel til design og implementation af en individuel brugergrænseflade.

7. INTERMEDIÆR-STRUKTUR: BRUGERGRÆNSEFLADE DESIGN

Formålet med denne implementation er at vise hvilke muligheder, der findes for design og implementation af individuelt designet brugergrænseflade under anvendelse af intermediærstruktur programmel. Samtidig demonstrerer denne anvendelse generelle egenskaber ved intermediær-struktur programmel. Endvidere er det formålet at vise hvilke generelle problemer, der opstår ved en sådan implementation.

Der vil blive konstrueret en brugergrænseflade, der anvender udpegning ved hjælp af mus. Brugergrænsefladen designes for undervisningssystemet SOFUS. Det anvendte intermediærstruktur programmel er DESIGNER POP-UP MENUS fra MOUSE SYSTEMS, samt den tilhørende PC-MOUSE, - uddrag af beskrivelsen af dette programmel findes i appendiks C. Det udviklede rullegardin menusystem vil i det følgende blot blive betegnet menusystemet. Det program, hvortil der designes menuer, betegnes anvendelsesprogrammet.

7.1. Indledning.

Ønsket eller kravet om at kunne tilpasse brugergrænsefladen efter individuelle behov udspringer af den generelle struktur, der er præsenteret for et DIIUS. Det er det datalogiske krav om at udvikle et system, der er generelt og åbent/dynamisk.

Arbejdet med udvikling af brugergrænseflade tænkes udført af den tidligere beskrevne "specialistbruger". Dette kræver ifølge DIIUS systematikken adgang til passende intermediærstruktur programmel. Menusystem programmellet fra "Mouse Systems" viser sig at opfylde kravene til intermediærstruktur programmel. Det er meget simpelt at anvende, og det giver mulighed for udvikling af brugergrænseflade, uden at der foretages indgreb i det eksisterende anvendelsesprogram, - her SOFUS undervisningssystemet.

Den type brugergrænseflade, der kan udvikles med "Mouse Systems" programmellet, beskrives i det følgende afsnit. (Afsnit 7.1.1 er baseret på ref. 10).

7.1.1. Grafiske brugergrænseflader.

Idag fokuseres der i stigende grad på anvendelse af grafik ved kommunikation i almindelighed til mennesker. Således ses der også i stigende grad anvendelse af grafik på nyere udviklede brugergrænseflader til f.eks. mikrodatamater.

Almindelige standard operativsystemer til mikrodatamater er normalt udstyret med en yderst primitiv brugergrænseflade. Ved enhver kommunikation med et program, f.eks. et operativsystem, skal der sendes informationer fra brugeren til systemet, og fra systemet til brugeren. Det traditionelle operativsystem meddeler sig til brugeren ved hjælp af tekst, og modtager informationer fra brugeren som tekst. Teksten indtastes normalt i en enkelt linie med simple rettefaciliteter.

Anvendelsesprogrammer anvender ofte én af to metoder: menuer eller funktionstaster. I begge tilfælde meddeler brugeren sig til systemet ved et tastetryk, i det første tilfælde ved et valgtegn, dvs. et tal eller bogstav svarende til valget i menuen, i det andet tilfælde ved tryk på en funktionstast.

Med en grafisk brugergrænseflade, udvides de mulige kommunikationsmåder mellem bruger og system ganske væsentligt. Vi skal se, hvorledes der kan anvendes udpegning ved hjælp af musen i forbindelse med rullegardinmenuer.

7.1.1.1. Rullegardinmenuer.

En såkaldt rullegardinmenu (eng. "pop-up menu") er en menu, der viser sig på skærmen, når der er behov for den, - når en mulighed er valgt i menuen, fjernes den igen. Fordelen er således, at den kun optager plads på skærmen, når der er behov for den. Samtidig giver individuel design af menuer mulighed for en meget fleksibel tilpasning af brugergrænseflade til individuelle ønsker og behov.

Jeg har valgt betegnelsen "grafisk" brugergrænseflade for

denne type brugergrænseflade, idet jeg finder det uvæsentligt, om der anvendes grafiske symboler eller tekstsymboler, - det afgørende er, at man meddeler sig til systemet ved at udvælge symboler på skærmen.

Valg i menuen foretages ofte ved hjælp af udpegning med mus:

7.1.1.2. Anvendelse af udpegning på skærmen.

Når brugeren meddeler sig til systemet, kan han afgive sin kommando f.eks. ved hjælp af et tastetryk, men grundideen ved rullegardinmenuerne er muligheden for at anvende udpegning på skærmen.

Til udpegning anvendes en såkaldt "mus". Anvendelse af menuer, er jo kendt fra talrige applikationsprogrammer, men valg af mulighed i rullegardinmenuen foregår ved hjælp af udpegning med musen. Vi skal nu først se på anvendelsen af den ydre enhed musen:

Traditionelt anvendes markørtaster til at bevæge markøren på skærmen. Det er imidlertid også muligt at anvende en mus hertil.

Udpegning foregår ved hjælp af musen. Man kunne opfatte det som om, musen befandt sig i det punkt, hvor markøren peger.

På musen findes mindst en tast, - antallet af taster afhænger af fabrikatet. Ønsker man at udpege en menu mulighed, peges der først med markøren på den ønskede mulighed, derefter gives der et tryk på musen, og muligheden er valgt.

7.1.1.3. Teknikken og historien bag musene.

Arbejdet med udvikling af mus startede i Xerox PARC (Palo Alto Research Center) for ca. 15 år siden. Samtidig opstod ideen med at bruge musen i forbindelse med de såkaldte rullegardinmenuer eller "pop-up" menuer. Til PC'ere anvendes der i dag i det væsentligste to typer mus: "Micro Soft Mouse" (="MS-mouse") og "PC-Mouse".*) Lad os se lidt på

teknikken bag de to typer mus:

*) Fra henholdsvis firmaerne MicroSoft og Mouse Systems.

Micro Soft Mouse fungerer mekanisk og var en af de første, der fremkom til IBM PC og dermed kompatible maskiner.

Den virker på den måde, at en kugle drejes, når musen bevæges over en overflade, f.eks. skrivebordet. På to af kuglens sider i en vinkel af 90 grader, er der placeret ruller, der roterer, når kuglen ruller. Rullerne står igen i forbindelse med elektriske censorer, der sender signalet videre til en mikroprocessor. Denne processor kan enten sidde i musen, eller være placeret i et udvidelseskort i selve datamaten. Musen kan eventuelt blot tilsluttes en seriel port på datamaten. (RS 232 C).

PC-Mouse, som vil blive anvendt i denne sammenhæng, indeholder ingen bevægelige dele, det er en såkaldt optisk mus. Den anvender to sæt af fototransistorer og lysdioder. En PC-Mouse tilsluttes altid til en seriel port.

Brug af musen kræver anvendelse af et særligt reflekterende underlag på ca. 20 x 25 cm. Underlaget er forsynet med striber i to retninger, således at der dannes et netværk på underlaget. Lysdioderne sender lys ned mod underlaget, som reflekterer lyset og sender det igennem et linsesystem, således at det rammer fototransistorerne.

De lodrette og vandrette linier på underlaget har hver sin farve, og da de to fototranssistorer er følsomme overfor hver sin farve lys, kan det således registreres i hvilken retning musen bevæges.

Den stigende anvendelse af mus tyder på, at anvendelse af en mus til udpegning er den bedste mulighed. Alternative måder er f.eks. den såkaldte berørings-følsomme skærm, hvor det ved hjælp af et "net" af infrarøde lysstråler registreres, hvor på skærmen der peges.

En anden mulighed kunne være en lysfølsom pen, hvor det dog også var nødvendigt at pege på selve skærmen. Disse mulighe-

der har ikke fundet større udbredelse, hvorimod musen finder stadig større udbredelse.

7.1.2. PC-Mouse systemet.

PC-Mouse anvendes i forbindelse med operativsystemet MS/PC-DOS. Programmellet består bl.a. af disse komponenter:

MOUSESYS.COM	Mus drivprogram/system
MSC .EXE	Menu oversætter
M_<anv> .MSC	Menu kildetekst for anvendelsesprogrammet <anv>
M_<anv> .COM	Oversat menu definition for <anv>

Fra DOS version 2.0 og senere blev der åbnet mulighed for konstruktion af egne drivprogrammer. MOUSESYS.COM er et sådant drivprogram til kontrol af en ydre enhed: musen. Drivprogrammet køres som en almindelig programfil og forbliver derefter resident i det indre lager. Drivprogrammet aktiveres, når der modtages afbrydelser fra den ydre enhed, - her musen. Dvs. hvis musen bevæges, eller hvis dens taster aktiveres, sendes der en afbrydelse.

Den oversatte menudefinition køres også som en programfil, derved lagres også denne permanent i det indre lager. Drivprogrammet vil derefter være i stand til at kalde menudefinitionen. Drivprogram og menudefinition fylder typisk 10 - 15 Kb tilsammen.

Denne meget enkle mulighed for design af egne drivprogrammer i MS/PC-DOS giver netop gode muligheder for design af intermediær-struktur programmel. Det ydnyttes at drivprogrammet er resident, og at det kan kaldes ved hjælp af afbrydelser på et tidspunkt, hvor anvendelsesprogrammet er kørende.

Der er mulighed for kald direkte fra en applikation til MOUSESYS drivprogrammet og der er endvidere mulighed for anvendelse af et MSMOUSE drivprogram, der får musen til at emulere en MS-Mouse. Disse muligheder omtales ikke nærmere i denne forbindelse. Vi skal nu se nærmere på design af egne rullegardinmenuer.

Menuerne defineres ved hjælp af et meget simpelt programmeringssprog. Se dokumentation i appendiks C. Der anvendes en almindelig teksteditor til redigering af kildeteksten, der derefter oversættes med MSC oversætteren.

Jeg mener at programmeringssproget er meget simpelt at anvende. Man oplever faktisk ikke syntaksfejl, køretidsfejl er ikke mulige. Logiske fejl, det vil i dette tilfælde sige om menuerne fungerer efter hensigten, er det egentlige problem. Det viser sig, at det er meget let at få menuerne til at fungere, som man har besluttet det. Det egentlige problem er derfor, hvordan menuerne skal designes for at fungere hensigtsmæssigt i forbindelse med anvendelsesprogrammet. Dermed må problemerne siges at være, hvor de hører hjemme, nemlig koncentreret om design og ikke om system og teknik.

7.1.3. Mulighederne ved design af menuer.

En række funktioner er på forhånd fastlagt, når der skal designes menuer. Der er med andre ord indgået et kompromis ved udvikling af systemet til konstruktion af menuer, hvor design muligheder og åbenhed er afvejet overfor det forhold at gøre programmeringssproget til design af menuerne simpelt.

F.eks. er menuernes grafiske fremtoning i det væsentligste fastlagt: rammen omkring menuen er fast, tekst centreres altid, én linie i top og bund er ikke menuvalg men tekstfelter, der kan anvendes til identifikation af menuen etc.

Gennem programmeringssproget er det muligt at fastlægge menuens bredde, negativ skærmskrift eller ikke, placering på skærmen. Hvis disse muligheder ikke anvendes af programmøren vælges standard værdier. Nogle muligheder, f.eks. om menuen skal fremstå i negativ skrift, kan kun angives som globale parametre, dvs. gældende for alle menuer.

En række funktioner ved anvendelse af menuerne er ligeledes fastlagt på forhånd: markøren i menuerne, der anvendes til at udpege det felt, der er valgt med musen, angives med en bjælke i negativ skrift i forhold til resten af menuen.

Intermediær-struktur: Brugergrænseflade design

Markørens hastighed i menuerne er fast. Muligheden i menuen skal altid vælges med den tast på musen, der aktiverede den pågældende menu. Efter valg af mulighed i menuen fjernes denne, uanset hvilket menuniveau man befandt sig på. Når en given menu fremkommer på skærmen vil menumarkøren pege på den mulighed, der blev valgt ved sidste anvendelse af menuen. Dette er en fordel ved gentagne anvendelser af en kommando.

På den anden side er det også her muligt at styre nogle af disse forhold ved hjælp af programmeringssproget til definition af menuerne. F.eks. kan man vælge, at den nederste linie i menuen anvendes til at forlade menuen, uden at der er foretaget et valg i menuen. Teksten i denne linie er standard sat til "Exit Pop-Up menu", men teksten kan ændres. Man kan vælge, om det skal være muligt at fjerne en menu, ved at markøren flyttes til højre eller venstre, ud af den aktuelt viste menu.

Ved udvikling af menuerne skal man fastlægge to væsentlige forhold: Hvad skal der ske, når musen bevæges, og hvad skal der ske, når der trykkes på en kombination af musens tre taster.

Aktivering af en mus tast (eller en kombination af taster), valg af menu mulighed og musbevægelse kan give anledning til følgende aktioner, - eller evt. en kombination af følgende aktioner:

1. Fremkaldelse af menu
2. Afsendelse af tegn til anvendelsesprogrammet
3. Ændring af tidligere definitioner for:
    1. Taster
    2. Musbevægelse

Normalt vil man lade musbevægelse give markørflytning, og lade tastetryk give rullegardinmenu på skærmen. Disse to egenskaber kan ændres efter tastetryk, - f.eks. i en menu, således at der f.eks. vises en anden menu når samme tastetryk anvendes senere.

De centrale dele i programmet er definitionen af menuerne.

Intermediær-struktur: Brugergrænseflade design

Dvs. fastlæggelse af hvilke tekstlinier menuen skal indeholde og hvilke taste-tryk systemet skal simulere afgivet, når den pågældende mulighed vælges.

-oOo-

Vi vil nu vende os imod det egentlige design af menuer for SOFUS forfattersystemet. Der vil blive lagt vægt på forhold, der er generelle ved design af denne type brugergrænseflade. Beskrivelsen formidler derfor en række erfaringer, der vil være anvendelige for kommende design af denne type menuer, - erfaringerne kunne f.eks. udnyttes i forbindelse med information til lærere om denne type intermediær-struktur programmel.

7.2. Design: Fremgangsmåde.

I dette afsnit opsummeres nogle af de erfaringer, der er gjort med henblik på at fastlægge en anvendelig fremgangsmåde ved arbejdet med udviklingen af menusystemet.

Tidsfordelingen ved udvikling af menuer med dette system har være således, - idet tallene er afrundet til nærmeste med 5 delelige procent:

	PCT.
1) Træning med menusystemet	5%
2) Design af menuer	25%
3) Programmering/oversættelse/aftestning	20%
4) Forsøg med og afprøvning af menuer og anvendelsesprogram	50%
IALT:	100%

Af denne tidsfordeling ses det, at det er meget hurtigt at sætte sig ind i anvendelsen af systemet, - denne del ville vel være større for en ikke datalogisk bruger. Den anvendte

tid til design af menuer er faktisk for stor. I praksis viser det sig, at det er vanskeligt at planlægge og designe menuerne på forhånd, - det er først ved kørsel med systemet, at det er muligt at vurdere menuerne. Den anvendte tid på programmering omfatter i det væsentligste indtastning. Selv begyndere vil næppe skulle rette mange syntaksfejl. Oversætteren er hurtig og anvender kun det indre lager under arbejdet. Der er anvendt forholdsvis meget tid på kørsel og forsøg med anvendelsesprogram og menusystem. I praksis vil man ofte vende tilbage senere for at foretage ændringer i sine menuer, efterhånden som krav og ønsker til brugergrænsefladen ændres.

Konklusionen på denne tidsfordeling, samt erfaringerne med udviklingen er, at der i høj grad er tale om interaktiv udvikling, der bør omfatte mange forsøg med menusystem og anvendelsesprogram.

Det vil ikke være muligt eller rimeligt at foretage en tilbundsgående analyse før end programmeringen foretages. Hvad der imidlertid er vigtigt er, at man har analyseret kommandostrukturen og funktionen for det anvendelsesprogram, hvortil der skal konstrueres menuer.

Afprøvningen vil i det væsentligste begrænse sig til en vurdering af om systemet virker tiltalende at bruge i praksis. En vanlig systematik ved denne kørsel og vurdering er selvfølgelig nødvendig. Den klassiske interne og eksterne aftestning for fejl bør naturligvis indgå i denne systematik.

Under alle omstændigheder kan det fastslås, at det der designes er en individuel opfattelse af én løsning på et menu design, - og det er netop, hvad der er hensigten!

7.3. Design: Afklaring af generelle problemer.

I dette afsnit afklares en række forhold, der må karakteriseres som værende af generel karakter for udvikling af rullegardinmenuer med PC-Mouse systemet.

Intermediær-struktur: Brugergrænseflade design

7.3.1. Forhold ved "påklistret" brugergrænseflade.

Der er en række forhold, der gør sig gældende i en situation, hvor der anvendes en brugergrænseflade, der eksisterer sammen men den oprindelige.

Hvilke kommandoer, der kan accepteres af anvendelsesprogrammet, afhænger af, hvilken tilstand programmet befinder sig i. Programmets tilstand afhænger igen af hvilke kommandoer, der tidligere er afgivet til programmet. Det er derfor ønskeligt, at det ved hjælp af menuerne kun er muligt at afgive kommandoer, der kan accepteres af programmet. De menuer, der er mulighed for at kalde frem ved hjælp af musens taster, skal afhænge af programmets tilstand, og dermed af tidligere kommandoer.

Med andre ord, det er ønskeligt, at den/de aktuelle menuer kun indeholder kommandoer, der er lovlige at afgive til anvendelsesprogrammet i den aktuelle tilstand for anvendelsesprogrammet.

Det er desværre uundgåeligt, at brugeren kan "overhale" menusystemet ved at bruge anvendelsesprogrammets standard brugergrænseflade, f.eks. funktionstaster. Anvendelsesprogrammet vil dog altid kunne "komme i trit med" menuerne igen, f.eks. ved at der afgives kommandoer med menuerne. Dette vil imidlertid kræve afgivelse af kommandoer, der er uacceptable for anvendelsesprogrammet i den aktuelle tilstand, idet programmet netop er ude af trit med menuerne. Spørgsmålet er, om dette er muligt, - eller om det bevirker en fejlmelding eller lignende i anvendelsesprogrammet. En anden mulighed for at komme i trit er at anvende standard brugergrænsefladen for at få anvendelsesprogrammet tilbage i den tilstand, der svarer til menuernes.

Det er således brugerens ansvar kun at anvende menuerne, - og det er menu programmørens ansvar at udvikle menuer, hvis opbygning følger anvendelsesprogrammets opbygning af kommandostruktur! En forudsætning herfor er et grundigt kendskab til anvendelsesprogrammets kommandostruktur og funktion.

7.3.2. Valg af menutekst.

Valg af tekststrenge i menuerne til identifikation af tilhørende kommando bør i høj grad være baseret på individuelle ønsker, herunder nationale sprog. Som et udgangspunkt vil det være rimeligt at anvende de betegnelser, der er anvendt f.eks. i brugervejledningen, på funktionstastoverlæg eller i programmmets skærmtekster.

Det bør tilstræbes, at ofte anvendte valgmuligheder befinder sig i den øvre del af menuen, - idet menumarkøren befinder sig i øverste linie, hvis der ikke tidligere er foretaget valg i den pågældende menu. Hvis der tidligere er foretaget valg, vil det være mest sansynligt at finde markøren i det øvre område, - idet systemet automatisk placerer markøren i den sidst udpegede menulinie.

7.3.3. Anvendelse af kommandoer.

Hvilke kommandoer der skal medtages i menuerne må igen bero på individuelle ønsker. Det må i den forbindelse overvejes, om menuerne skal indeholde alle programmets mulige kommandoer. Til begynder brugeren kunne det evt. være relevant at anvende et begrænset menusystem.

Endvidere må det overvejes, om der skal findes kommandoer i menuerne, der er en kombination af eksisterende kommandoer. F.eks.:

$$K_m = k_{p_1} + k_{p_2} + ... + k_{p_n}$$

- hvor K_m er en kommando i menuen, der er sammensat af en række kommandoer i anvendelsesprogrammet. Svarende til hvad der kunne betegnes som en kommando-macro.

Ved design af denne type menuer vil det være af interesse at samle passende grupper af kommandoer i samme menu. Det kunne da være af interesse at lade samme program kommando k_p forekomme i flere menuer, og dermed optræde som flere kommandoer i menuerne:

$$K_{m_1} = k_p \quad \text{og} \quad K_{m_2} = k_p$$

7.3.4. Menu niveauer.

Det er må bero på en individuel vurdering, om man foretrækker flade eller dybe menuer. Flade menuer vil sige et menusystem med et begrænset antal niveauer, - men med mange valgmuligheder i menuerne. Dybe menuer vil sige et menusystem med mange niveauer, - men med et begrænset antal valgmuligheder i menuerne.

7.3.5. Ensartede funktioner.

Programmeringen af menuerne bør foretages således, at funktionerne i taster og mus bevægelse er ensartet i hele menusystemet. F.eks. kan venstre tast på musen altid anvendes til at fremkalde menuen, højre tast altid afgive tegnet <return>. Endvidere bør det overvejes, hvor på skærmen menuen skal fremkomme, - hvis der ikke er et særligt argument imod, bør alle menuer fremkomme på samme sted på skærmen.

7.3.6. Følsomhed og hysterese.

Følsomhed og hysterese er to forhold, der er af betydning for, hvorledes markøren bevæger sig på skærmen i forhold til, hvordan musen bevæges på underlaget.

7.3.6.1. Følsomhed.

Musen afgiver 100 impulser, for hver tomme den bevæges (1 tomme = 25,4 mm). Underlaget, hvorpå musen skal bevæges, er 225 x 193 mm, musen er 65 x 100 mm. Dette bevirker, at musen effektivt kan bevæges indenfor et rektangel på 160 x 93 mm. Dette medfører at der kan afgives:

160 / 25,4 * 100 X 93 / 25,4 * 100 impulser

= 630 X 366 impulser.

Det er muligt at nedsætte følsomheden individuelt i X og Y retningen, idet der angives faktorer n_x og n_y, der angiver nedsættelsen af impulsantal.

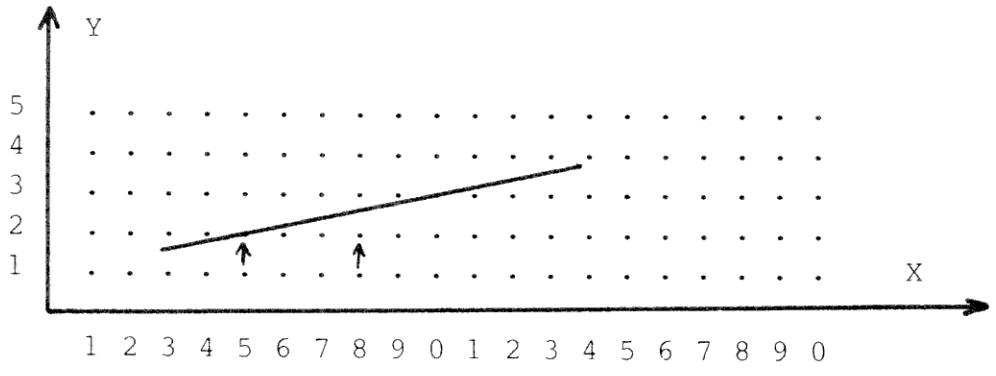
Hvis man f.eks. ønsker at tilpasse musunderlaget til skærmen, således at en fuld bevægelse fra venstre til højre og fra top til bund giver en fuld bevægelse på skærmen, og skærmen har 80 kolonner og 25 linier, beregnes således - idet der afrundes til heltal:

$$n_x = 630 / 80 = 8 \quad \text{og} \quad n_y = 366 / 25 = 15$$

Det er imidlertid i praksis ikke muligt at udnytte hele musunderlaget, men et areal på f.eks. 500 X 280 mm synes passende. Dette ville give $n_x = 6$ og $n_y = 11$.

7.3.6.2. Hysterese.

Hysterese angiver trægheden, hvormed markøren bevæger sig. Hvis man ønsker at kunne bevæge markøren i en ret linie i X-retningen, er det f.eks. af interesse af have nogen træghed mod Y-bevægelse, mens musen bevæges i X-retningen.



Figur 7-1: Hysterese.

Forudsæt at følsomheden i Y-retningen er nedsat med n_y = 100, svarende til 100 impulser for 1 enhed på Y-akse.

Det ses af figur 7-1, at X bevægelse fra 5 til 8 ikke har flyttet markøren én enhed på Y-aksen.

Der er imidlertid opsamlet en række impulser fra musen, på figur 7-1 ca. 70. Hvis hysterese-X er sat til 3, og der som her er bevæget 3 enheder i X-retning, men kun opsamlet 70 impulser i Y-retning, vil disse 70 impulser blive bortkastet, - svarende til at Y-bevægelsen negligeres! Bemærk, den største hysterese fås med hysterese-X og hysterese-Y lig med 1.

Hvis musen bevæges diagonalt på underlaget, i den hensigt at bevæge markøren diagonalt på skærmen, opstår der et andet problem. Hvis der anvendes stor hysterese eller træghed, f.eks. hysterese-X lig med 1, og markøren tænkes bevæget i et kvadratisk net, som i figur 7-1, og bevægelsen påbegyndes i nederste venstre hjørne og der bevæges mod højre og samtidig opad i en vinkel på blot lidt mindre end 45 grader, vil al bevægelse i Y-retningen blive negligeret. Hysterese skal derfor ikke ukritisk sættes op.

7.4. Design: SOFUS - Specielle problemer.

Som tidligere anført er hovedformålet med denne implementation og tilhørende dokumentation, at få kendskab til generelle problemer ved implementation af rullegardinmenuer ved hjælp af PC-Mouse programmellet.

I dette afsnit vil udvalgte eksempler fra design og implementation af menuer for forfattersystemet SOFUS blive behandlet. Det er søgt at give fremstillingen en sådan form, at den kan læses uden større kendskab til SOFUS systemet, - men udbyttet vil være større hvis læseren har kendskab til SOFUS.

Beskrivelsen er delt op i et afsnit om generelle problemer, der knytter sig til hele menusystemet og en række afsnit, der knytter sig til de enkelte SOFUS-editorer.

Afklaringerne i dette afsnit bygger i det væsentligste på de erfaringer, der er nået gennem forsøg og arbejde med og vurdering af menusystemet i funktion i forbindelse med SOFUS systemet.

Programlistning for det udviklede program til menudefinition findes i appendiks D.

7.4.1. Fra eksisterende brugergrænseflade til menusystem.

SOFUS systemet består af fem editorer samt en afviklingsdel. Video editoren er imidlertid ikke behandlet her. Den eksisterende brugergrænseflade i SOFUS systemet anvender funktionstaster, hvor der anvendes fire forskellige overlæg til funktionstasterne. Det samme overlæg anvendes til to forskellige editorer. Til markørflytning på skærm anvendes markørtasterne.

I dette design af menuer, der er tænkt som et eksempel på, hvorledes et system kunne designes, er der anvendt en hovedopdeling af kommandoer i menuer, der stort set svarer til den opdeling, der findes på funktionstastoverlæggene.

Intermediær-struktur: Brugergrænseflade design

7.4.2. Generelt.

Det er valgt, at musens venste tast altid giver menu, og at højre tast altid giver tegnet <return>. Bevægelse af musen giver altid markørbevægelse.

Muligheden for markørbevægelse v.h.a. musen giver en stor fordel ved arbejdet med SOFUS, idet systemet i høj grad giver mulighed for at brugeren frit kan skrive et vilkårligt sted på skærmen og f.eks. selv vælge hvilke felter, der skal udfyldes i et skema.

I SOFUS vil man ofte skifte imellem at skulle udfylde tabeller med 5-9 kolonner og 15-17 linier, og at skulle skrive på et fuldt skræmbillede med 80x25 tegn, - svarende til det informationsbillede eleven senere præsenteres for. Dette giver et ønske om at kunne ændre mus følsomheden i overenstemmelse hermed.

Det vil sige, at vi har en situation, hvor menusystemet, og her specielt følsomheden, skal følge anvendelsesprogrammet. Problemet er imidlertid, at arbejdet med et helt skærmbillede i SOFUS ofte afsluttes ved tryk på <return>, - hvorefter der vendes tilbage til arbejdet med en tabel. Hvis man ønsker, at der skal ske et skift i følsomhed i denne situation, bliver det nødvendigt at skabe en ny kommando, der bedst kunne være en menu mulighed. Kommandoen skal bevirke: "<return>_skift_følsomhed", hvor <return> bevirker "afslut_skærmbillede_retur_til_tabel". Den generelle kommando <return> kan ikke kombineres med skift af følsomhed, - idet den anvendes i andre sammenhænge, hvor skift af følsomhed ikke ønskes.

I praksis kan der imidlertid godt arbejdes med tabellen under anvendelse af den samme høje følsomhed, som den der anvendes ved fuldt skærmbillede. Det kan imidlertid også være af interesse at arbejde med lav følsomhed på hele skærmbilledet. Jeg har derfor valgt at give en mulighed for ændring af følsomhed med en menu, der kan kaldes frem i alle program tilstande, og hvor der kan vælges ud fra en række følsomheder.

Problemet hysterese viser sig ved forsøg at være meget lille. Der anvendes højst en opløsning på skærmen på 80x25 punkter, med denne opløsning og en følsomhed der tilpasser hele skærmen til hele musunderlaget, er det ikke vanskeligt at bevæge markøren i rette linier, - selv med lav hysterese. Af tidligere nævnte årsager er der en anden årsag til at anvende en lav hysterese, - idet det med lav hysterese er muligt at bevæge markøren diagonalt v.h.a. diagonal musbevægelse. Forsøg viser at en hysterese på (10,10) er passende.

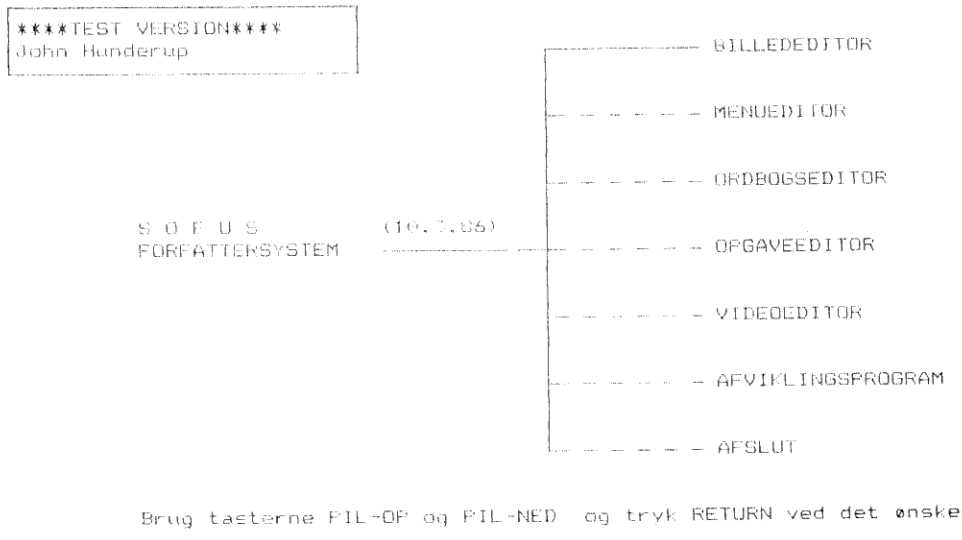
Det er valgt at lade alle menuer være placeret centreret på skærmen for at give en så rolig fremtoning af menuer som muligt.

Hvor en menulinie giver anledning til, at der skriftes til en undermenu, er det valgt at omgive menuteksten med to "*"er.

I det følgende behandles de problemer, der knytter sig til de fire editorer og til afviklingsdelen, samt SOFUS hovedmenu, der vælger imellem de fire editorer, afvikling, afslutning af SOFUS og videoeditoren, - som ikke behandles her.

7.4.3. **Hovedmenu.**

SOFUS anvender selv en hovedmenu, hvor en bjælke flyttes med pil-op eller pil-ned tasterne, og der trykkes <return> ved den valgte mulighed. Se figur 7-2.



Figur 7-2: SOFUS Hovedmenu.

Med henblik på at opnå, at menusystemet følger SOFUS systemet så meget som muligt, har jeg valgt også at lade valget i SOFUS hovedmenuen foregå med en rullegardinmenu. Skal f.eks. afviklingsprogrammet startes, skal der tastes pil-ned 5 gange og derefter <return>. Det viser sig imidlertid, at menusystemet tilsyneladende afsender disse ialt 6 tegn for hurtigt, således at programmet ikke når at fange tegnene! Dette problem kan løses ved, at der imellem pil-ned tegnene sendes et tegn, der ikke reageres på i SOFUS hovedmenuen, f.eks. blanktegn.

For at undgå at brugeren uforvarende kommer til at flytte bjælken i SOFUS-menuen, er al markørbevægelse v.h.a. musen slået fra i denne menu.

7.4.4. Billededitor.

I SOFUS' brugergrænsefladen gives der i billededitoren kommandoer ved hjælp af funktionstaster eller ved alfanumeriske tegn. Billeder defineres ved dels at arbejde i en tabel, dels ved at arbejde på selve skærmbilledet. Vælges f.eks. i tabellen kopi, skal der afgives kommandoen placer (F7) for

at komme til selve skærmbilledet, og der angive det område, der skal kopieres. Tabellen ses i figur 7-3.

B I L L E D E D I T O R

BILLEDNUMMER: 1000

DEL NR.	TYPE	PLACERING X1	Y1	X2	Y2	X3	Y3	VIDERE	LÆNGDE 128	OK	BEMÆRKNINGER (gemmes ikke)
1	RAMME	13	7	53	11	0	0	0		**	
2	TEKST	15	9	51	10	0	0	R	74	**	
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											

TYPE: Tekst Lin. Ram. Pil Flyt Kopi Slet Vid. Attr. Hop Bil. Col.

Figur 7-3: Billededitor - tabel.

Billederne opbygges af en række såkaldte billedelementer: tekst, linie, ramme, pil, flyt, kopi, slet, video, attribut, hop, billede og color. Valg af billedelementer foregår ved at taste det pågældende bogstav. Ikke alle billedelementer har lige sigende navne, et forhold der kan bedres ved design af menuerne.

At der er anvendt color og ikke det danske farve skyldes sammenfald af valgtegnet f for farve og flyt. Dette sted er der derfor i menuen valgt at fravige beslutningen om at anvede samme betegnelser i menuer som i anvendelsesprogram og brugervejledning, - derfor anvendes der i menuen farve, - og der afgives tegnet c ved valg af menumuligheden.

Efter de billedelementer, der altid kræver anvendelse af kommandoen placer, f.eks. slet, tekst, flyt, kopi, farve eller attribut (dvs. fremhæv, blink, invers, understreg og normal), - er det valgt at lade menuen afgive kommandoen

placer umiddelbart. Vælges således f.eks. kopi i menuen, afgives tegnene k og F7. Dette er et eksempel på, at menuen indeholder en kommando, der er summen af flere anvendelsesprogram kommandoer.

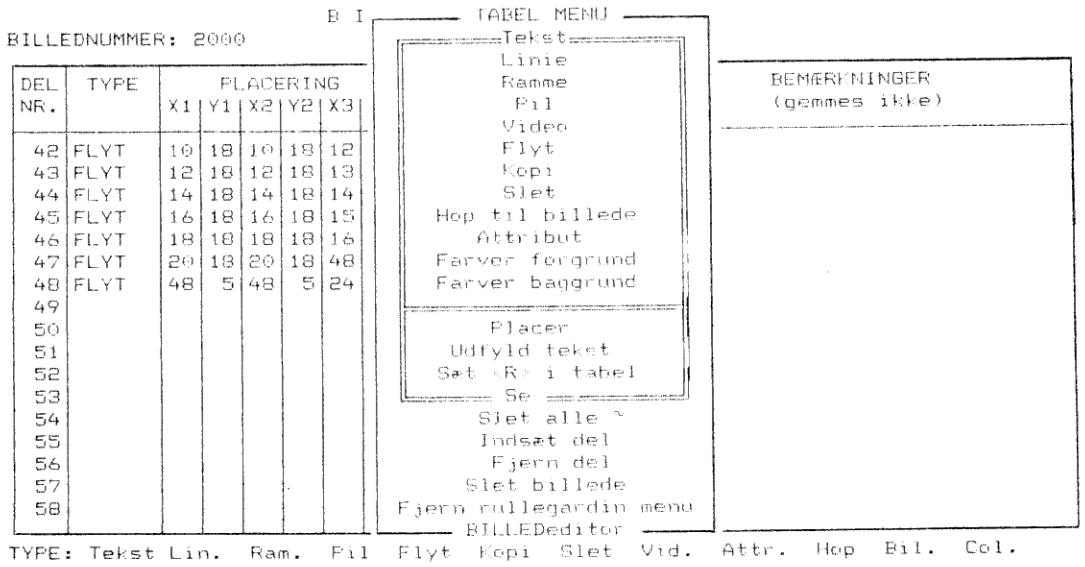
Antallet af mulige kommandoer i hele billededitoren er ret omfattende. Der findes således 22 funktionstast kommandoer, hertil kommer valg af billedelementer med et bogstav (12 mulige) og valg af farve (8 mulige) og attribut (5 mulige), dvs. ialt 47 kommandoer. Princippet om at samle kommandoer hørende til samme overlæg i samme menu er derfor fraveget her.

De 22 funktionstastkommandoer er fastlagt i funktionstast overlægget som ses i figur 7-4.

SHIFT+						papir tabel	papir billed	billed nummer skærm	billed nummer print
ALT+		semigrafik on	semigrafik off	indsæt ~	fjern ~	indsæt linie	fjern linie	slet alle ~	tilbage
slet billede	stop	nyt nummer	nyt billede	indsæt del	fjern del	placer	udfyld	gem	se
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10

Figur 7-4: Billededitor funktionstastoverlæg.

Der er således designet en tabel-, tekst-, farve-, attribut- og system-menu. Farve og attribut findes i forvejen som undermenuer i SOFUS. I figur 7-5 ses tabelmenuen i anvendelse sammen med billededitorens tabel.



Figur 7-5: Rullegardinmenu - Billededitor tabel.

Med henblik på grafisk at samle alle kommandoer, der vælger billedelementer i tabelmenuen, er det valgt at indramme valgmulighederne med en dobbeltstreg. Dobbeltstregen findes som en del af IBM-PC tegnsættet. Det giver et problem i den linie i menuen der dermed blot kommer til at indeholde en dobbelstreg. Ved imidlertid at knytte en aktion der vælger tabelmenuen ved udpegning af denne linie, vil man forblive i menuen ved evt. udpegning af dobbelt stregen!

Tekstmenuen omfatter de kommandoer, der udelukkende anvendes ved arbejdet med tekst i selve skærmbilledet. Denne tilstand nås ved kommandoen udfyld tekst i tabelmenuen. Efter denne kommando skal der derfor skiftes til tekstmenuen, således at den fås ved tryk på musens venstre tast. Heldigvis er det således, at der returneres til tabellen med en særlig kommando tilbage. Denne kommando kan derfor anvendes til samtidig at bevirke skift til tabelmenuen i musens venstre tast.

Forholdene omkring følsomhed i henholdsvis tabel og skærmbillede er omtalt tidligere, idet en generel mulighed for skift af følsomhed blev valgt.

I praksis vil man under anvendelse af menusystemet faktisk altid arbejde enten i kolonne 1 i tabellen, der ialt har 9 kolonner, - idet kolonnerne "længde" og "OK" udfyldes af systemet. Se figur 7.3. Derudover er der ofte behov for at sætte et "R" i kolonnen "videre".

Denne funktion er det valgt at lade indgå som en mulighed i menuen, da det p.g.a. problemer med mus følsomhed er vanskeligt at bevæge markøren i tabellen.

Ved hjælp af 8 gange højre-pil kan der bevæges helt til højre i tabellen, - uanset hvor man måtte befinde sig, én venstre-pil flytter til kolonnen "videre", hvor "R<return>" derefter kan skrives.

Når der anvendes 9 gange højre pil i programmet skyldes det et tidligere nævnt problem, nemlig at der kan tabes tegn, idet anvendelses programmet ikke kan nå at modtage alle tegn fra menuen.

Systemmenuen omfatter kommandoer, der anvendes sjældnere, og som ikke omfatter egentlig billedtegning.

7.4.5. **Ordbogseditor.**

Her findes en mulighed for at bladre i ordbogens ord, idet pil-op giver det foregående ord og pil-ned det følgende. Pil-op og pil-ned afgives, som det tidligere er besluttet, af musens bevægelse. Forsøg viser imidlertid, at det er vanskeligt at bladre ét ord ad gangen på denne måde. Derfor er det besluttet at fravige det generelle princip om at musens højre tast altid afgiver <return>. Midterste og højre tast kan derfor også anvendes til at bladre frem og tilbage i ordbogen.

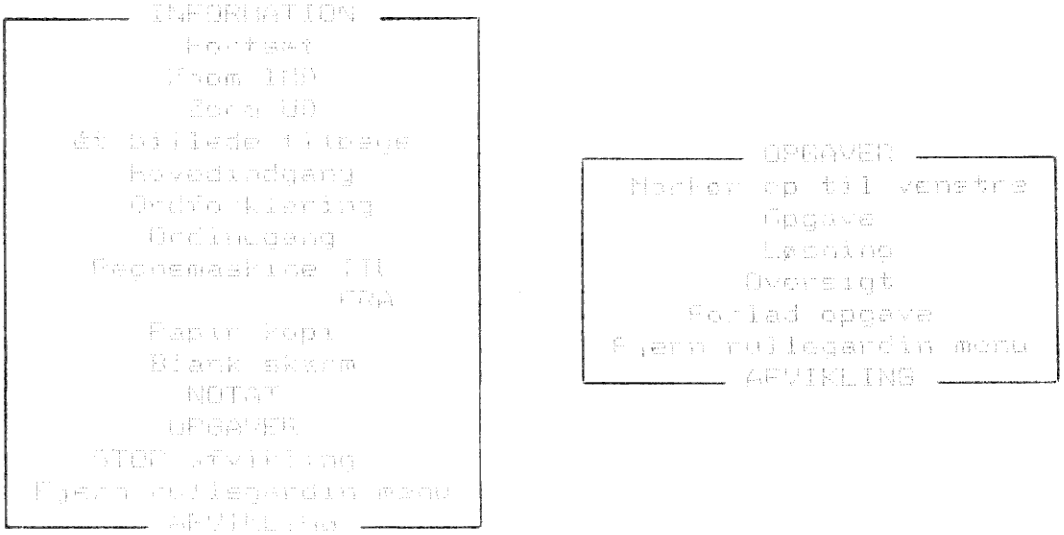
7.4.6. **Afvikling.**

Det er skønnet at brug af kommandoerne for opgaver samt notater er væsentligt mindre brugt en kommandoerne, der styrer informationssøgningen. Det er derfor valgt at opdele

afviklingen i tre menuer: informations-, notat- og opgavemenu. Her brydes princippet om, at menuerne skal følge programmets tilstand, idet det er valgt at begrænse antallet af mulige kommandoer. Situationen er den, at det er lovligt at afgive kommandoer fra alle tre menuer under al anvendelse af afviklingsprogrammet.

Når der f.eks. vælges OPGAVER i informationsmenuen, har jeg valgt at omdefinere venstre mustast, således at opgavemenuen forbliver tilknyttet denne tast. Opgavemenuen bliver en såkaldt "sticky"-menu. Det er derfor nødvendigt med en menu mulighed, der giver mulighed for at slippe af med opgavemenuen i venstre tast. Hertil er anvendt kommandoen Forlad opgave, der således ikke er en SOFUS kommando, - men blot en nødvendighed i PC-Mouse systemet. Man kunne vælge ikke at omdefinere tasterne, således at man skulle gennem informationsmenuen hver gang én af kommandoerne i opgavemenuen skulle anvendes. Forsøg viser imidlertid, at denne mulighed ikke virker tiltalende.

På figur 7-6 ses menuerne afvikling-information og afvikling-opgave.



Figur 7-6: Afvikling - information og opgave menu.

Hvis eleven under anvendelse af afviklingsprogrammet ønsker at løse en opgave eller at se et tidligere skrevet notat, anvendes i begge tilfælde først kommandoen markør op til venstre. Denne kommando findes derfor både i opgave- og notatmenu, - et eksempel på, at to menukommandoer udgøres af samme anvendelsesprogram kommando.

7.5. Menudefinitions programmet.

Det udviklede program kan ses i appendiks D. Beskrivelsen af programmeringssproget findes i appendiks C. For at kunne referere til linier i programmet, er programlistningen udvidet med linienumre. Reference til linie nr. <nnnn> angives således: "L:<nnnn>".

Det udviklede PC-Mouse program indeholder følgende programdele:

BESKRIVELSE	NØGLEORD
1. Globale definitioner	*)
2. Markør definitioner	CURSOR
3. Musetast definitioner	BUTTON
4. Initiel mus aktioner	MOUSE
5. Menu definitioner	MENU

*) Globale parametre kan omfatte følgende nøgleord: Comment, ExitMenu, Sensitivity, Hysteresis, ReverseVideo, FixedMenu, MenuCenter, Position, EnableBeep, NoviceMode og ExpertMode.

Funktionen af de enkelte programdele kan summeres således:

Ad 1) Forhold der er gældende for hele det udviklede menusystem.

Ad 2) En række definitioner for markørens bevægelse ved musbevægelse.

Ad 3) En række definitioner for aktioner ved tryk på mustast.

Ad 4) Fastlægger de definitioner, der skal være aktive initielt ved aktivering af systemet.

Ad 5) Menutekst samt aktioner ved valg af menumulighed for en række menuer.

7.5.1. Globale definitioner (L:0001-0012).

Hysterese og position anvendes her globalt, - kunne alternativt indgå i henholdsvis markør definition og menudefinition.

7.5.2. Markør definitioner (L:0014-0057).

Alle definitioner i ARROWKEYS og ARROWKEYS_2 - 8 fastlægger musbevægelse ved markørflytning, men med faldende følsomhed.

NOKEYS giver ingen markørflytning ved musbevægelse, - anvendes i SOFUS hovedmenu.

7.5.3. Musetast definitioner (L:0071-0097).

Følgende systematik er anvendt ved valg af etiketnavne:

<M>_<LMR>B - hvor

<M> angiver den menu, hvor tastdefinition er gældende:

S: Start
H: Hoved
BT: Billed Tabel
BX: Billed teXt
BS: Billed System
M: Menu (editor)
O: Ordboq
OP: OPgave
A: Afvikling
AO: Afvikling Opgave
AN: Afvikling Notat
ALL: Alle menuer

<LMR> angiver hvilke(n) musetast(er), evt. tastkombination, definitionen er gældende for:

L: Left
M: Middle
R: Right

7.5.4. Initiel mus aktioner (L:0060-0068).

LEFT, MIDDLE og RIGHT refererer til musetaster. LEFTRIGHT (L:0067) angiver venstre og højre tast. Alle taster er tilknyttet S_LMRB tast definitionen, der (L:0073) giver startmenuen.

7.5.5. Menu definitioner (L:0099-0477).

Programmet omfatter følgende MENU definitioner:

L:0101	CURSOR_SENSIV *)
L:0113	START
L:0131	VEJLEDNING
L:0169	HOVED
L:0234	BILLED_TABEL
L:0278	BILLEDE_ATTRIBUT
L:0293	BILLEDE_FARVE
L:0312	BILLEDE_TEKST
L:0330	BILLEDE_SYSTEM
L:0353	MENUEDIT
L:0376	ORDEDIT
L:0400	OPGAVEEDIT
L:0421	AFVIKLING
L:0451	AFVIKLING_NOTAT
L:0466	AFVIKLING_OPGAVE

*) "markør" og "følsomhed" ikke tilladte symboler.

Bemærk f.eks.: Anvendelse af dummy tegn (L:0205-0209). L:0253 aktionen MENU(BILLED_TABEL) bevirker, at man forbliver i denne menu ved evt. valg af linien.

Omdefinition af taster ses f.eks. L:0176-0178, hvor billededitor startes, og L:0343-0345 hvor billededitor forlades.

Her ses også aktivering af markørdefinition L:0179, og deaktivering af markør bevægelse L:0346.

7.6. Vejledning for PC-Mouse/SOFUS rullegardinmenuer.

Denne kortfattede vejledning forudsætter kendskab til PC-Mouse systemet.

Menuerne aktiveres på følgende måde. PC-Mouse drivprogrammet loades med DOS kommandoen:

```
A>MOUSESYS/M:15000
```

- dermed afsættes der plads til 15.000 byte menudefinitioner. SOFUS menuerne loades med kommandoen:

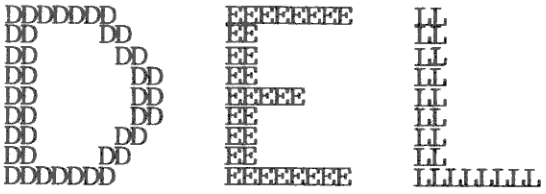
A>M_SOFUS

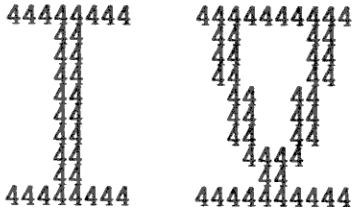
Alle kommandoer til SOFUS, herunder start af SOFUS bør foregå med mus og rullegardinmenuerne.

Tastaturet anvendes udelukkende til indtastning af tekst, - i alle sammenhænge i SOFUS systemet!

Tryk på en vilkårlig musetast bevirker, at startmenuen fremkommer. Her kan vælges en kort vejledning eller start af SOFUS. Derefter vil alle musetaster give hovedmenuen, hvorfra SOFUS editorerne kan startes.

Venstre tast anvendes til at fremkalde hovedmenuen for den pågældende editor. Midterste tast anvendes som oftest ikke, - i billededitoren giver den dog billededitor-systemmenu. Højre tast giver tegnet <return>, - i ordbogseditoren anvendes højre og midterste tast dog til at "bladre" i ordbogen. Venstre og højre tast samtidig giver altid menu for ændring af følsomhed.





K O N K L U S I O N O G V I D E R E U N D E R S Ø G E L S E R

8. KONKLUSION.

Efter afslutningen af dette projekt kan bl.a. drages følgende konklusioner:

Der er betydelige vanskeligheder ved udvikling af undervisningsprogrammel. Dette gælder såvel i skolesektoren som i industrien.

Udvikling af undervisningsprogrammer bør foretages af lærere. Udviklingen bør ske under anvendelse af så begrænsede resourcer, og med en så tilstrækkeligt kort udviklingstid, at læreren har mulighed for, dels at anvende ny teknologi, dels for hurtigt og smidigt at kunne afprøve nye ideer til datamatiske undervisningsforløb.

Dette kræver at lærerne har tilstrækkelige værktøjer til rådighed. Værktøjer der ligger tæt på lærerens begrebsverden.

Datalogen bør præsentere og informere lærerne om værktøjer - udviklet under gensidig inspiration.

Værktøjerne bør være åbne og let tilgængelige for lærerne, - idet undervisningsprogrammer bør udarbejdes af lærere ved hjælp af værktøjer udviklet af dataloger.

Der er stor afstand fra eksisterende undervisningssystemer, som f.eks. forfatterredskaber, til det DIIUS (Datamatisk Interaktivt Integreret Undervisnings System), hvis systematik er beskrevet.

Få værktøjer kan indplaceres i den beskrevne systematik for DIIUS og den tilhørende systematik for identifikation af datalogiske værktøjer. De eneste værktøjer, der kan indpasses i systematikken, er værktøjer som databasesystemer, grafiske systemer, regnesystemer og tekstbehandlingssystemer.

Karakteristisk for disse værktøjer er, at der enten er udviklet en systematik og metodologi for dem, eller at de via deres store udbredelse er specielt gennemarbejdede sy-

stemer. Denne baggrund mangler undervisningssystemer.

Dette projekt har givet en del af den teoretiske baggrund for undervisningssystemer, - nemlig en systematik og nomenklatur.

Karakteristisk er det også, at de værktøjer, der kan indpasses i DIIUS systematikken, er systemer, der ikke omfatter modellering af anvendelsesmodellen for et datamatisk undervisningsforløb. Denne mulighed kræver udvikling af infrastruktur programmel herfor.

Den væsentligste mangel ved eksisterende værktøjer indenfor rammerne af det beskrevne DIIUS er netop mangel på muligheder for modellering af undervisningsmodellen. Endelig vil der være behov for en egentlig metodik for opbygning af undervisningssystemer. Fremtidige undersøgelser bør omfatte disse områder.

De eksisterende programsystemer til udvikling af undervisningsprogrammer savner intermediær-struktur muligheder.

Erfaringerne med udvikling af alternativ brugergrænseflade med PC-Mouse programmellet viser, at det er muligt at konstruere gode intermediær-struktur værktøjer til anvendelse for brugergrænsefladen, - det var dog ønskeligt med et større "sortiment".

Det må fortsat anses for væsentligt at undersøge og arbejde med eksisterende undervisningssystemer og programmer herunder brugertest og anvendelsesforsøg, som baggrund for fortsatte arbejder indenfor dette område.

I næste kapitel omtales et bredt spektrum af de mulige videre undersøgelser, - som jeg finder kunne være af interesse.

Konklusion.

9. VIDERE UNDERSØGELSER.

Der skal her anføres en række mulige videre undersøgelser. Det skal understreges, at der naturligvis vil være mulighed for en meget lang række undersøgelser med udgangspunkt i dette projekt, - idet områderne undervisningssystemer og interaktive systemer generelt er et forholdsvis dårligt undersøgt område. De mulige projekter, som jeg anfører i det følgende, er derfor projekter, jeg finder har umiddelbar tilknytning til dette projekt, er særlig relevante eller som er af personlig interesse.

Dette projekt tog sit udgangspunkt i praktiske erfaringer med undervisningssystemer og udvikling af undervisningsprogrammer. Derefter blev der udviklet en teori omfattende bl.a. en systematik for et DIIUS, og endelig blev der foretaget en implementation af programmel til understøttelse af en individuel brugergrænseflade med "mus" og "rullegardinmenuer".

Som allerede anført i indledningen til dette projekt, vil jeg gerne understrege vigtigheden af, at også fremtidige arbejder indenfor dette område sker i konstant tilknytning til undersøgelser af praktiske erfaringer med brugernes anvendelse af systemerne. Anbefalingerne for fremtidige undersøgelser omfatter således dels teoretiske dels praktiske undersøgelser. Anbefaling af anvendelses forsøg er medtaget for netop at understrege vigtigheden af bruger erfaringer, - selvom sådanne forsøg næppe primært bør foretages af dataloger.

Herefter følger en beskrivelse af en række mulige projekter for fremtidige undersøgelser. Rækkefølgen angiver ikke nogen prioritet.

9.1. Mulige fremtidige projekter

1. Metodologi for design og implementation af undervisningssystemer.

På baggrund af den opstillede systematik og nomenklatur

for et DIIUS udvikles en metodologi for design og implementation af et DIIUS. Metodologien udvikles med udgangspunkt i den systematik for et DIIUS jeg tidligere har udviklet. Metodologien udvikles sideløbende med arbejdet omfattende praktisk udvikling af delmoduler af DIIUS (2, 3 og 4) samt indsamling af systemerfaringer (5).

2. Design og implementation af test systemer.

Der designes og udvikles test programmel, der udgør moduler i et DIIUS. Udviklingen foretages i overensstemmelse med DIIUS systematik, samt en evt. metodologi for design og implementation af DIIUS (1). Målet er ikke at designe og implementere et færdigt DIIUS, men at udvikle programmel, der eksempificerer systematikken samt evt. metodologi for design og implementation. Det udviklede programmel vil således udgøre en eksempelsamling til teorien for DIIUS.

3. Brugergrænseflade tilpasning.

Udvikling af PC/MS-DOS programmel, der understøtter udvikling af alternative brugergrænseflader, der kan tilpasses individuelle behov og ønsker. Det udviklede programmel skal være uafhængigt af anvendelsesprogrammet.

Der tænkes anvendt MS/PC-DOS faciliteten til at lade et program forblive i lageret, nemlig afbrydelse 27H ("Terminate-but-stay-resident"). Dermed kan et program anvendes af andre programmer ved afbrydelses håndtering, - programmet kan derved komme til at udgøre et supplement til det kørende anvendelsesprogram program.

Der tænkes her på programmel af den type, der f.eks. findes til anvendelse i forbindelse med "mus", hvor der ved hjælp af et begrænset sprog kan udvikles en brugergrænseflade, der anvender udpegning ved hjælp af mus og rullegardinmenuer.

Videre undersøgelser

Der er flere muligheder for udvikling af alternative brugergrænseflader. F.eks. (1) fri definition af tastsekvens for en given kommando, (2) kommando afgivelse med verbum og substantiv ("slet linie") snarere end funktionstaster/tastsekvens.

4. Multimedia undervisningsprogrammer.

Der undersøges datatekniske muligheder for anvendelse af f.eks:

* video
* audio
* dias
* processtyring
* kommunikation til databaser

- i forbindelse med datamatiske undervisningsforløb. Der skal dels undersøges elektroniske interfaceforhold dels udvikles datalogiske værktøjer til understøttelse af medierne.

5. Indsamling af systemerfaringer.

Der indsamles løbende praktiske erfaringer med eksisterende undervisningssystemer specielt i Danmark. F. eks. erfaringer med forfattersystemer og forfattersprog. Erfaringsindsamlingen baseres dels på personlige kontakter til skolesektoren og erhvervslivet, dels på litteraturstudier.

6. Beskrivelse af systematik for modeller af undervisningsforløb.

Der foretages en udvidelse af den systematik, der er opbygget for DIIUS, specielt for anvendelsesmodellen med tilhørende aksiomer. Anvendelsesmodellen udgør modellen af undervisningsforløbet.

Videre undersøgelser

7. Træning i anvendelse EDB-programmer.

En særlig type undervisningsprogram, der ofte betegnes et "tutor"-program, er et undervisningsprogram, hvis emne er undervisning i anvendelse af et bestemt EDB program. Det kunne dreje sig om et anvendelsesprogram f.eks. et tekstbehandlingssystem, et administrativt system, en banks terminalsystem eller lignende.

Det specielle ved denne type undervisningssprogram er, at det indeholder simulationer af kørsel med f.eks. tekstbehandlingssystemet eller terminalsystemet. Dette kan ske under udnyttelse af det faktum, at mediet for undervisningsprogrogrammet og for anvendelsesprogrammet er det samme, - nemlig en dataskærm.

Der foretages design evt. implementation af undervisningssystemer, der kan anvendes ved udvikling af denne type programmer.

8. Anvendelsesforsøg.

Uanset hvilket undervisningssystem, der anvendes til udvikling af et undervisningsprogram, vil undervisningsprogrammet realisere en bestemt model af et undervisningsforløb. Der bør anstilles en række anvendelsesforsøg med anvendelse af forskellige undervisningsprogrammer, der realicerer forskellige undervisningsmodeller. Ref. 1 omfatter rapport over anvendelsesforsøg for Danmarks Sparekasseforenings anvendelse af DFU.

9. Supra-struktur for lærere.

Det undersøges hvilket supra-struktur programmel, dvs. hvilke datalogiske værktøjer som en del af DIIUS, lærere finder relevante i forbindelse med datamatiske undervisningsforløb.

Videre undersøgelser

10. Infra-struktur for undervisningssystemer.

Det undersøges hvilket infra-struktur programmel, der er nødvendig for konstruktion, design og implementation af undervisningssystemer. Undersøgelsen kan f.eks. omfatte programmeringssprog, databasesystemer og grafiske kerner.

9.1.1. Projekternes indplacering i DIIUS systematik.

I den systematik, jeg har foreslået for et DIIUS, er der foretaget opdeleling bl.a. i tre funktionelle dele:

BRUGERGRÆNSEFLADE
ANVENDELSESMODEL
LAGRINGSMODUL

Idet delprojekterne kan karakteriseres som overvejende teoretiske eller overvejende praktiske, kan delprojekterne indplaceres i denne systematik således (figur 9-1):

BRUGERGRÆNSE-FLADE	ANVENDELSES MODEL	LAGRINGS MODUL	TOTALE DIIUS	
	(6)		(1) (9) (10)	**TEORI**
(3)	(4) (7)		(2) (5) (8)	**PRAKTIK**

Figur 9-1: Delprojekternes indplacering i forhold til DIIUS.

Punkterne (9) og (10) kan evt. blot tilknyttes ét delmodul.

Videre undersøgelser

A. REFERENCER

1) Danmarks Sparekasseforening, DFU-Anvendelsesforsøg, 1985.

2) Spatial data-management, MIT, 1979.

3) Tom Hogan, Discover Forth, McGraw-Hill, 1982.

4) Leo Brodie, Starting Forth, Prentice-Hall, 1981.

5) Z-80 Forth, Laboratory Microsystems, june 1984.

6) Udgået.

7) Günter E. Pfaff ed., User interface managemet systems, Springer-Verlag, 1983.

8) Sparekassen SDS, Udvikling af DFU, Vejledning til "SDS-modellen", dec. 1985.

9) John Hunderup, Undersøgelse af mikrodatamat forfatter systemer, DIKU projekt 85-6-7, 1985.

10) John Hunderup et. al., CAP: computer aided pedagogy, DIKU projekt 85-12-33, 1985.

11) R. A. Guedj et. al., Methodology of interaction, North-Holland, 1979.

12) ICL, COMUS, Dataformidlet Undervisning, Brugervejledning, juli 1984.

13) Computer Teaching Corporation, Introduction to TenCore, Operating System, Lauguage System.

14) Ivan Boserup og Jørgen Feder, MicroTutor, Museum Tusculanums forlag, 1984.

15) Hugh Tucker, Software infra-structure and interfaces

for graphic systems, International Electronic Image Week, Nice 1986.

16) Danmarks Sparekasseforening, Div. referater af foredrag om DFU, med tilhørende overheads, 1982-85.

17) Danmarks Sparekasseforening, CAT-Study trip to the U.S.A., jan 1984.

18) Hugh Tucker, Laying the Foundation for a DESIGN METHODOLOGY for Interactive System Design, Datalogisk Institut, Københavns Universitet.

19) John Hunderup et. al., OSI-modellens øvre lag, DIKU rapport "Blå Serie", nr. 85/5, 1985.

20) Andrew S. Tanenbaum, Computer Networks, kap. 1, Prentice/Hall International Inc., 1981.

21) Helle Frederiksen, Introduktion til Dataformidlet og Datamatstøttet Undervisning, Overseas Training Institute, Working Papers, 1984.

22) VIDA ApS, Notat om interaktiv video, 7 sider.

23) Viggo Sadolin, Myresnak, Teknisk forlag.

24) Viggo Sadolin, Simuler, Teknisk forlag.

25) Viggo Sadolin, Skomal, Teknisk forlag.

26) Danmarks Sparekasseforening, DFU Model. Kursus 00, juni 1985.

B. ORDLISTE

3-d systematik
Den samlede opdeling af DIIUS i tre dimensioner: funktion, struktur og virtuel-niveau. Side 96.

Aksiomer
Primitive operationer til ændring af anvendelsesmodellen. Omfattes af anvendelsesmodellen. Side 62.

Anvendelsesmodel
En funktionel del af DIIUS. Den datamatiske model af et datamatisk undervisningsforløb. Omfatter også aksiomerne. Side 62, 65, 87.

Brugergrænseflademodul
En funktionel del af DIIUS. Det modul der håndterer kommunikation med bruger. Side 62, 87.

Datalogisk værktøj
Et værktøj, f.eks. hele DIIUS, der kan indpasses i den beskrevne systematik for DIIUS, samt systematik for identifikation af værktøjer. Side 104.

Datamatisk undervisningsforløb
Et undervisningsforløb hvori indgår en datamat med tilhørende undervisningsprogram, - under én eller anden form. Side 11.

DBMS
Data Base Management System. Side 63, 88.

DIIUS
Datalogisk Interaktivt Integreret Undervisnings System. Side 58ff.

Dynamisk
I denne sammenhæng betegnet: åbent. Side 18ff.

Forfatterredskab
Eksisterende programsystem til udvikling af undervisningsprogrammer. Side 29ff.

Forfattersprog
Forfatterredskab der anvender et programmeringssprog til opbygning af undervisningsprogrammer. Et forholdsvis åbent system. Side 32.

Forfattersystem
Forfatterredskab der anvender menustyring eller lignende til opbygning af undervisningsprogrammer. Et forholdsvis lukket system. Side 33.

Formelt niveau
En virtuel-niveau del af DIIUS. Omfatter formelle databeskrivelser. Side 68ff, 70.

Funktionel
En dimension hvorefter DIIUS opdeles i brugergrænseflademodul, anvendelsesmodel og lagringsmodul. Side 62ff, 65ff.

Horisontal/vertikal
To dimensioner der anvendes ved identifikation af værktøjer. Horisontal: funktion. Vertikal: tilhørende datalogens eller underviserens begrebsverden. Side 104ff.

Informationsbillede
Et skærmbillede, evt. med animation o.l. i SOFUS/COMUS systemet. Side 51.

Infra-struktur
En strukturel del af DIIUS. Omfatter basalt programmel, der anvendes ved design og konstruktion af DIIUS. Side 86 ff.

Interface
Et modul der sørger for tilpasning imellem funktionelle moduler på supra-struktur niveau. Side 75.

-struktur
rukturel del af DIIUS. Omfatter programmel der
es af specialistbrugeren til at foretage mindre
ger i eksisterende DIIUS. Side 88ff.

ational Standards Organization. Side 73.

on
e sammenhæng udveksling af data imellem funktio-
moduler på supra-struktur niveauet via interface-
r og via en trelags model: operationel, formel og
Side 72ff.

ervisningsprogram
isningsprogrammet plus en større eller mindre del
ktionaliteterne fra undervisningssystemet. Side

ul
ktionel del af DIIUS. Omfatter programmel til
lagring og læsning af data. Side 62, 88.

rmbillede i SOFUS/COMUS systemet med en variation
n standard menu. Side 51.

mst[illegible]ling af metoder til anvendelse indenfor en
kab. Her metoder til udvikling af undervisnings-
er. Side 58, 152.

rre eller mindre del af DIIUS. Side 60.

blogi, fagligt navngivningssystem. I denne sam-
g indenfor DIIUS. Side 58ff.

t niveau
tuel-niveau del af DIIUS. Omfatter beskrivelser
rationer. Side 68, 70, 82.

Forfatte
Eks
nin

Forfatte
For
til
vis

Forfatte
For
de
hol

Formelt
En
bes

Funktion
En
dem
65f

Horisont
To
tøj
log

Informat
Et
sys

Infra-st
En
der
86

Interfac
Et
mod

Intermediær-struktur
En strukturel del af DIIUS. Omfatter programmel der anvendes af specialistbrugeren til at foretage mindre ændringer i eksisterende DIIUS. Side 88ff.

ISO
International Standards Organization. Side 73.

Kommunikation
I denne sammenhæng udveksling af data imellem funktionelle moduler på supra-struktur niveauet via interface-moduler og via en trelags model: operationel, formel og reel. Side 72ff.

Kørende undervisningsprogram
Undervisningsprogrammet plus en større eller mindre del af funktionaliteterne fra undervisningssystemet. Side 102.

Lagringsmodul
En funktionel del af DIIUS. Omfatter programmel til fysisk lagring og læsning af data. Side 62, 88.

Menubillede
Et skærmbillede i SOFUS/COMUS systemet med en variation over en standard menu. Side 51.

Metodologi
En fremstilling af metoder til anvendelse indenfor en videnskab. Her metoder til udvikling af undervisnings-systemer. Side 58, 152.

Modul
En større eller mindre del af DIIUS. Side 60.

Nomenklatur
Terminologi, fagligt navngivningssystem. I denne sammenhæng indenfor DIIUS. Side 58ff.

Operationelt niveau
En virtuel-niveau del af DIIUS. Omfatter beskrivelser af operationer. Side 68, 70, 82.

OSI
Open Systems Interconnection. Side 73.

Reelt niveau
En virtuel-niveau del af DIIUS. Omfatter beskrivelser af faktiske data. Side 68, 71.

Specialist bruger
En bruger af et undervisningssystem, der endvidere er i stand til at anvende intermediær-struktur programmel til at foretage mindre ændringer på systemet. Side 91.

Strukturel
En dimension hvorefter DIIUS opdeles i supra-, intermediær- og infra-struktur programmel. Side 61.

Supra-struktur
En strukturel del af DIIUS. Omfatter det egentlige undervisningssystem. Side 64ff.

Transformation funktion
En funktion der indgår som en del af et interface. Foretager transformation fra beskrivelser i ét funktionelt modul til ét andet. Side 78.

UIMS
User Interface Managemet System. Side 87.

Undervisningsmodel
En logisk beskrivelse af et undervisningsforløb. I denne forbindelse et datamatisk undervisningsforløb. Side 12, 100.

Undervisningsprogram
Det program der anvendes på datamaten til at styre den i forbindelse med et datamatisk undervisningsforløb. Side 12, 100.

Undervisningssystem

Det programmel der anvendes ved udvikling af undervisningsprogrammer. Side 12, 100.

Virtuel-niveau

En dimension hvorefter DIIUS, indenfor suprastrukturen, opdeles i et operationelt-, formelt- og reelt-niveau. Side 68, 69ff.

Åbent/lukket system

Begrebet åben=dynamisk er i denne sammenhæng knyttet til: undervisningsforløbet, programmet, modellen og systemet. Side 18ff.

Chapter 2

POP-UP MENU DEFINITION LANGUAGE

This chapter describes the language used to create pop-up menus. You'll find it helpful to refer to one of the menu definition files in Appendix C as an example of how the language is used. Before you attempt to make a personalized menu, be sure you have read both this chapter and Chapter 1.

OVERVIEW

A pop-up menu definition has five parts, each of which is defined explicitly later in this chapter. They are:

1. **Mouse Definition.** The mouse definition defines what moving the mouse or pushing the button will do when the menu is first loaded. Every menu definition must have one and only one mouse definition with at least one valid entry.

2. **Global Parameters.** Global parameters are settings that control how the mouse works with a pop-up menu. For example, you can control whether the menu appears in reverse video or not.

3. **Button Definitions.** A button definition is a list of actions that are to be performed when a button is clicked.

4. **Cursor Definitions.** A cursor definition defines the keystrokes that will be sent to the application when the mouse is moved. It can also define sensitivity and hysteresis.

5. **Menu Definitions.** A menu definition contains a list of items that will be displayed when a pop-up menu is invoked. Each item has a list of actions that are performed when that item is selected.

A button or item definition can have four types of actions. Actions are described in detail in the sections describing button and item definitions.

1. **Cursor.** The **cursor** action makes a new cursor definition active.

2. **Button.** The **button** action makes a new button definition active.

3. **Keys.** The **keys** action sends keystrokes to the application.

4. **Menu.** The **menu** action invokes a pop-up menu.

All parts of a pop-up menu definition use the same general syntax. See Appendix C of this manual for two examples of menu definition files.

GENERAL SYNTAX

The input to the MSC Menu compiler is a simple programming language. A word processor or text editor is used to create or modify the ASCII menu configuration file. If you are using a word processor, you should verify that it works with the MSC program (see chapter 1).

General Rules

White space characters (e.g., Spaces, Tabs, and Carriage Returns) may be freely used to improve readability of the menu definition file. Multiple spaces and tabs are equivalent to a single space character. White space characters are only significant within strings.

All comparisons involving special key names, labels, and keyword commands are "upper/lower case insensitive." For example, a label called MenuOne is the same as MENUONE or menuone.

Comments

Any text that is preceded by a semicolon is ignored by MSC until the end of the line. This comment character can be used at the beginning of lines for block comments, or on the same line as some other statement for in-line comments.

Comments that are used to document the menu definition file are not the same as the "Comment" parameter described later in this chapter.

Strings

Text strings consist of one or more ASCII characters enclosed within a matching pair of single quote marks or double quote marks. If a quote mark is required within a string, it can be included by using the other kind of quote marks around the string, or by two adjacent quote marks. Strings cannot be broken across line boundaries. The maximum string length is eighty characters.

Characters in the IBM extended character set (ASCII value greater than 127) will not be displayed accurately in graphics mode unless the graphics font is loaded into memory. Use the DOS command GRAFTABL to accomplish this. GRAFTABL is available for International DOS 1.1 and later, and for Domestic DOS 3.00 and later. Users of the IBM PCjr and the IBM Enhanced Graphics Card can display these characters in graphics mode without running GRAFTABL. This is in accordance with the standard method of supplying the extended character set.

Identifying Keystrokes

Two basic notations are used to specify what keystrokes are to be sent for cursor movement, button clicks, and menu item selection:

1. Keys in the typewriter group are given as quoted strings. For example, **"DIR"** denotes the three keystrokes for the D, I, and R keys. Use a blank in quotes to define the space bar.

2. Keys in the special group are given as a key name enclosed in square brackets ([and]). For example, **[F1]** denotes the keystroke for function key 1.

Both of these notations can be used in combination. For example:

```
[Esc] "DIR" [Enter]
```

Wherever possible, the name of the special key is identical to the legend used on the IBM keyboard. However, several keys use graphic symbols rather than names on their keytops. Also, some keys appear more than once on the keyboard. In these cases, we have used a descriptive name for the key.

Most keys can be specified in conjunction with a Shift, Control, or Alternate key. Thsè combinations are prefixed by **s-**, **c-**, and **a-**, respectively. For example, to specify Control-X, you would type **[c-X]**. For a complete list of all of the special key names, see Table 1.

It is also possible to specify keys as 16-bit hexadecimal or decimal numbers containing both the ASCII code and scan code for the key. In this case, keystrokes take the form [&H<number>] or [&D<number>]. For example, the keystroke **[&H3B00]** is another way to specify [F1]. Refer to the keyboard BIOS code in IBM Technical Reference manual for further details on this format.

Certain key combinations cannot be generated by the pop-up menu software. The IBM keyboard routines handle special keystroke combinations internally (e.g., Ctrl-Alt-Del, Ctrl-Numlock). In some cases, multiple keystroke combinations produce the same ASCII code and scan code pair (for example, Space, Shift-Space, Ctrl-Space, and Alt-Space).

Table 1. Special Key Names

Special Keys	Alt	Ctrl	Shift	Alt	Ctrl	Shift
[F1]	[a-F1]	[c-F1]	[s-F1]	[a-3]		
[F2]	[a-F2]	[c-F2]	[s-F2]	[a-4]		
[F3]	[a-F3]	[c-F3]	[s-F3]	[a-5]		
[F4]	[a-F4]	[c-F4]	[s-F4]	[a-6]	[c-6]	
[F5]	[a-F5]	[c-F5]	[s-F5]	[a-7]		
[F6]	[a-F6]	[c-F6]	[s-F6]	[a-8]		
[F7]	[a-F7]	[c-F7]	[s-F7]	[a-9]		
[F8]	[a-F8]	[c-F8]	[s-F8]	[a-A]	[c-A]	
[F9]	[a-F9]	[c-F9]	[s-F9]	[a-B]	[c-B]	
[F10]	[a-F10]	[c-F10]	[s-F10]	[a-C]	[c-C]	
[Esc]				[a-D]	[c-D]	
[Tab]				[a-E]	[c-E]	
[s-Tab]				[a-F]	[c-F]	
[Backtab]				[a-G]	[c-G]	
[Backspace]		[c-Backspace]		[a-H]	[c-H]	
[Enter]		[c-Enter]		[a-I]	[c-I]	
[Home]		[c-Home]		[a-J]	[c-J]	
[End]		[c-End]		[a-K]	[c-K]	
[PgUP]		[c-PgUP]		[a-L]	[c-L]	
[PgDn]		[c-PgDn]		[a-M]	[c-M]	
[UP]				[a-N]	[c-N]	
[Down]				[a-O]	[c-O]	
[Left]		[c-Left]		[a-P]	[c-P]	
[Right]		[c-Right]		[a-Q]	[c-Q]	
[Ins]				[a-R]	[c-R]	
[Del]				[a-S]	[c-S]	
[Plus]				[a-T]	[c-T]	
[Minus]				[a-U]	[c-U]	
		[c-PrtSc]		[a-V]	[c-V]	
	[a--]	[c--]		[a-X]	[c-X]	
	[a-0]			[a-Y]	[c-Y]	
	[a-1]			[a-Z]	[c-Z]	
	[a-2]	[c-2]			[c-[]	
					[c-\]	
					[c-]]	

Labels

A label names a menu, button, or cursor definition. Labels must start with an alphabetic character, and be unique in the first 12 characters. When labels are defined, they are followed by a colon (:).

Example:

```
Main_Menu:  Menu     ; Identifies a menu definition with the label
                     ; "Main_Menu"
```

Keyword Commands

The basic form of most commands is given by **keyword(**<argument(s)>**)**. The arguments to a given keyword command may be one or more numbers, flags (yes or no), strings, keystroke sequences, or other keyword commands. Some keyword commands are prefixed by label definitions. If multiple arguments are given, they can be separated by commas and optional white space characters.

GLOBAL PARAMETERS

The following commands are used to set "global" parameters used by MOUSESYS to control things like the sensitivity of the mouse and the appearance and placement of pop-up menus.

Comment Parameter

Format: Comment(<string>)

The comment parameter is not the same as the comments you use to document a pop-up menu definition. You can cause a message to appear on the screen using the Comment parameter. The message is displayed when the executable menu file is used to load its menu configuration, and when a MOUSESYS status command is performed. Multiple Comment commands may be given. All Comment strings are displayed one per line in the order given.

ExitMenu Parameter

Format: ExitMenu(<string>)

On all pop-up menus the "Exit Pop-up" item is automatically generated. The ExitMenu parameter is provided so that you can substitute other words in the item. For example, you might want your menus to say "Leave Menu" rather than "Exit Pop-up." (See also ExpertMode.)

Sensitivity Parameter

Format: Sensitivity(<Xinc>, <Yinc>)

This command sets the sensitivity of the mouse. The two numbers given <Xinc> and <Yinc> control how much mouse motion results in sending a cursor keystroke sequence. <Xinc> controls horizontal motion, while <Yinc> controls vertical motion. Since the mouse has a resolution of 100 counts per inch, both <Xinc> and <Yinc> are given in units of hundredths of an inch of mouse motion. The default settings are (10, 10). The sensitivity parameter affects cursoring in menus as well as programs. The Sensitivity command can also be used within a Cursor Definition, and so can be changed from within a program by changing which cursor definition is active.

Hysteresis Parameter

Format: Hysteresis(<AutoX>, <AutoY>)

This command allows the mouse motion to be "smoothed out". Since it is difficult to move the mouse in a perfectly horizontal or vertical line, the two numbers given (<AutoX> and <AutoY>) can be used to weight the mouse motion. If <AutoX> left or right cursor keystrokes are output without any up or down cursors being sent, any residual vertical mouse counts are discarded, and the cursor moves more horizontally. If <AutoY> up or down cursor keystrokes are output without any left or right cursors being sent, any residual horizontal mouse counts are discarded, and the cursor moves more vertically. The Hysteresis command can also be used within a Cursor Definition, and so can be changed from within a program by changing which cursor definition is active.

Example: Let's say that Sensitivity is set at (10,10), so ten mouse counts are required to generate a cursor keystroke, and <AutoX> is 2. You move the mouse 18 mouse counts to the right and 8 mouse counts down. The mouse will send 1 cursor keystroke to the right, and have 8 mouse counts left over in each direction. You move the mouse 2 more counts to right. The mouse adds the two counts to the eight left over and moves the cursor one more position to the right. The eight residual down counts are thrown away because two keystrokes were sent in the X direction with no keystrokes sent in the Y direction. You would have to move the mouse at least 10 more counts down before getting a keystroke in the down direction.

The smaller the value of <AutoX> or <AutoY>, the easier it is for the cursor to remain on a single screen row or column. The default settings are (2, 2).

ReverseVideo Parameter

Format: ReverseVideo(<flag>)

This command controls whether the pop-up menu appears in reverse video (black letters on a light background) or in normal video. The default setting is **Yes**. The other possible value is **No**.

FixedMenu Parameter

Format: FixedMenu(<flag>)

This command is included to provide compatibility with previous versions of the compiler. It has no effect. Menus are fixed if given a position, and will be centered on the cursor if not.

MenuCenter Parameter

Format: MenuCenter(<column>,<row>)

This command specifies where the menu will be displayed. The menu is centered over the given <column> and <row> numbers, if possible. The default settings are (0, 0). This command is included for compatibility with previous versions of the MSC compiler, and is a special case of the Position command.

Position Command

Format: Position(<control point>(<column>,<row>))

This command specifies where the menu will be displayed. The menu is placed so that the control point is at the specified column and row, if possible. The <column> and <row> numbers are relative to the upper left corner of the screen starting from 0. The available control points are:

UpperLeft	UpperCenter	UpperRight
CenterLeft	Center*Center*	CenterRight
LowerLeft	LowerCenter	LowerRight

For example, **Position(LowerRight(75,20))** would place the bottom right-hand corner of the menu at column 75, row 20. This command can also be used within menu definitions to control specific menus.

EnableBeep Parameter

Format: EnableBeep(<flag>)

This command enables or disables a warning beep for an incorrect mouse button sequence. If enabled, when a menu is displayed, any button click using a button other than the one that brought up the menu will result in a beep. If disabled, clicking the wrong button with a menu displayed will cause the menu to be dismissed, and the other button processed. The default setting is **No**. The other possible value is **Yes**.

NoviceMode Parameter

Format: NoviceMode(<flag>)

This command enables or disables novice mode. In novice mode, all left and right mouse motion in a pop-up menu is ignored. Note that in novice mode, the only way to get rid of an unwanted pop-up

menu is to select the "Exit Pop-up" menu item. The default setting is **No**. The other possible value is **Yes**. NoviceMode and ExpertMode cannot both be set to Yes.

ExpertMode Parameter

Format: ExpertMode(<flag>)

This command enables or disables expert mode. In expert mode, the "Exit Pop-up" menu item is not generated. Note that in expert mode, the only way to get rid of an unwanted pop-up menu is to move the cursor beyond the left or right border of the menu. The default setting is **No**. The other possible value is **Yes**. ExpertMode and NoviceMode cannot both be set to Yes.

DEFINITIONS

Mouse Definition

The Mouse definition describes the initial connection between the physical mouse motion and buttons and the Cursor and Button definitions by referencing the Button and Cursor definitions. This includes definitions of the chords produced by pushing more than one button. The general form of a mouse definition is given by:

```
Mouse
(
    Left(<label>)              ; Initial left button
                               ; definition
    Middle(<label>)            ; Initial middle button
                               ; definition
    Right(<label>)             ; Initial right button
                               ; definition
    LeftRight(<label>)         ; Initial left + right chord
                               ; definition
    LeftMiddle(<label>)        ; Initial left + middle chord
                               ; definition
    MiddleRight(<label>)       ; Initial right + middle chord
                               ; definition
    LeftMiddleRight(<label>) ; Initial left + right + middle
                               ; chord definition
    Cursor(<label>)            ; Initial cursor definition
)
```

Each of the possible statements in the mouse definition are optional, although a valid mouse definition must contain at least one.

The values in the mouse definition can be changed by the Cursor or Button Actions described below under "Button Definition" and "Menu Definition".

Button Definition

A button definition describes the action(s) to be performed when a mouse button is pressed and released. The set of actions is given by:

A keystroke sequence to send AND/OR
A menu to display AND/OR
A button action AND/OR
A cursor action

The button definition is used to specify a new set of actions to be performed the next time the mouse button is pressed. The initial button definition is established in the Mouse definition.

The general form of the button definition is given by:

```
<label>: Button
(
    Keys(<keystrokes>)          ; The keystrokes to send
    Menu(<label>)               ; The menu to display
    Button(<button>=<label>)    ; Redefine a button
    Cursor(<label>)             ; The new cursor definition to
                                ; use
)
```

The button definition must have a label. Note that the Menu, Button, and Cursor commands inside a button definition are **actions**, not **definitions**. A typical Button definition might look like one of the following:

Example 1:

```
Left_Button:  Button
(
    Keys([ESC])                 ; Send an Escape
    Cursor(NewCursor)           ; Change the sensitivity and
                                ; hysteresis
    Button(LeftRight=NewMB)     ; Redefine the Left+Right chord
```

Example 2:

```
NewMB:   Button    (Menu(Main))
```

The button action, "Button(<button>=<label>", makes a new button definition active for a button or chord where <button> is specified as Left, Right, Middle, LeftRight, LeftMiddle, MiddleRight, or LeftMiddleRight. These keywords represent all the buttons and chord combinations. The button indicator may be omitted so that the command reads "Button(<label>)". This is provided for compatibility with previous versions of Pop-up Menus and redefines the current button (i.e. the button that was pressed to initiate the action sequence). The label referenced must be a button definition.

Cursor Definition

The cursor definition describes the keystroke sequences to be sent for up, down, left, and right mouse motion. (Note that the LEFT and RIGHT commands do not refer to buttons, but to mouse movement.) It can also reset the sensitivity or hysteresis that was established with global parameters or in the initial cursor definition referenced by the mouse definition. The general form of this definition is:

```
<label>: Cursor
(
        Left(<keystrokes>)
        Right(<keystrokes>)
        Up(<keystrokes>)
        Down(<keystrokes>)
        Sensitivity(<Xinc>,<Yinc>)
        Hysteresis(<AutoX>,<AutoY>)
)
```

The cursor definition must have a label. Multiple cursor definitions are allowed in a menu configuration file. The initial cursor definition is specified in the mouse definition. A typical cursor definition might look like the following:

```
New_Cursor:   Cursor
(
    Left            ([Left])
    Right           ([Right])
    Up              ([Up])
    Down            ([Down])
    Sensitivity     (10,10)
    Hysteresis      (2,2)
)
```

Menu Definition

The menu definition describes the form and content of a pop-up menu. The general form of a Menu definition is given by:

```
<label>: Menu
(
    Title(<string>)                 ; The optional menu title
    Width(<number>)                 ; The minimum menu width
    Position(<control point>(<column>,<row>))
                                    ; The position of the menu
    Item(<string>, <actions>)       ; The first menu item
    Item(<string>, <actions>)       ; The second menu item
    .
    .
    .
    Item(<string>, <actions>)       ; The last menu item
    Footer(<string>)                ; The optional menu footer
)
```

The menu definition must have a label. The optional Title command specifies a string to be placed at the top of the menu. Similarly, the Footer command gives the string to be placed at the bottom of the menu. The Width command specifies the minimum width of the menu. The actual menu width is the maximum of the Width and the longest item text string. Each of the Item commands define the text to be placed in the menu, and the action(s) to be performed when the menu item is selected. An Item command is identical to a button definition with respect to actions. The general form of an item is given by:

```
Item
(
    <string>                    ; The text of the menu item
    Keys(<keystrokes>)          ; The keystrokes to send
    Menu(<label>)               ; A submenu to display
    Button(<button> <label>)    ; A new button assignment for
                                ; the mouse definition
    Cursor(<label>)             ; A new cursor definition
)
```

Note that the Menu, Button, and Cursor commands inside an item definition are **actions**, not **definitions**.

The Position command can be used inside a menu and works just like the global parameter except that it applies only to that menu.

```
Position(LowerRight(75,20))
```

would place the lower right-hand corner of the menu at column 75, row 20.

DESIGNER POP-UP MENU DEFINITION FILES

A menu definition file must contain, at minimum, a mouse definition which has one statement that points to a valid button or cursor definition. Thus it is acceptable to have a very small menu definition file that has very little functionality. Beyond this minimum, there is a great number of possible menu definitions with a wide range of possible functionality. We have included two sample menu definitions in Appendix C of this manual. We encourage you to examine these samples and to use them as a guide when developing your own Designer Pop-up Menus.

D. PROGRAMLISTNING: PC-MOUSE MENUER FOR SOFUS

I denne udgave findes programlistning ikke. Programlistning vil kunne fås ved henvendelse til forfatteren:

John A. Hunderup
Telf. 08 37 22 16.