TeSS-projektet:

Udvikling af et system til eksperimentel undersøgelse af brugergrænseflader til edb-baseret tekstsøgning

Jette Holm Broløs, Erik Frøkjær, Morten Hertzum,
Marta Kristín Lárusdóttir, Kristian Bang Pilgaard,
Flemming Steen Sørensen

Datalogisk Institut
Københavns Universitet 1993

DIKU rapport 93/18

Forord

TeSS er udviklet af en projektgruppe på DIKU bestående af de datalogistuderende Jette Holm Broløs, Kristian Bang Pilgaard, Marta Kristín Lárusdóttir og Flemming Steen Sørensen, i samarbejde med Erik Frøkjær og Morten Hertzum som videnskabelige medarbejdere. I starten benyttede vi betegnelsen 'Brugergrænseflade-gruppen' for projektgruppen. Senere blev 'TeSS-gruppen' et mere naturligt navn.

Datalogistuderende Ketil Perstrup har støttet TeSS-gruppen med værdifulde diskussioner og konstruktive ideer undervejs; det samme gælder medarbejderne i DIKU's edb-afdeling, specielt Claus Engdahl, Lars G. Jakobsen og Carl-Lykke Pedersen. Kim Høglund, også i DIKU's edb-afdeling, ydede en stor indsats for projektet ved installeringen af Mjølner Beta systemet, det objektorienterede programudviklingssystem som projektgruppen valgte til realisering af TeSS.

Vi takker Mjølner Informatics, Århus, for udlånet af Mjølner Beta systemet. Systemet har fungeret yderst tilfredsstillende under arbejdet med design og implementation af TeSS — også i samspillet med de øvrige benyttede værktøjer: X Window systemet med Athena Widgets, C og relationsdatabasesystemet Ingres. Når projektgruppen enkelte gange løb ind i problemer med Beta, har Jørgen Lindskov Knudsen, DAIMI, Århus Universitet, og Kim Jensen Møller, Mjølner Informatics, hver gang ydet os effektiv hjælp.

Til slut en særlig tak til de mange Datalogi 2-studerende, der - som led i den karaktergivende rapportopgave K2 i foråret 1993 - besluttede at lade TeSS-gruppen føre en detaljeret, anonym log over deres interaktion med TeSS-systemet. Vi håber, at der herved er indsamlet forsøgsdata, som kan bidrage til at kaste mere lys over, hvad mennesker under arbejde med søgning i tekstdata kan have brug for af datamatstøtte. Her forestår for Erik Frøkjær og Morten Hertzum et analysearbejde, som vil blive rapporteret senere. Læsere, der er interesserede heri, kan meddele det til Erik Frøkjær (e-mail: erikf@diku.dk) eller Morten Hertzum (e-mail: morten@diku.dk). De vil så modtage besked, når den endelige evalueringsrapport foreligger.

TeSS-gruppen

DIKU
Universitetsparken 1
2100 København Ø

24. maj 1993

Indhold

1 Introduktion til TeSS-projektet

Vi har designet og implementeret TeSS - et tekstsøgesystem. I dette kapitel redegøres først for formålet med TeSS-projektet, for tilrettelæggelsen af det og for opbygningen af denne rapport. Derefter behandles valget af de programmeringsværktøjer, vi har brugt i implementeringen. Kapitlet afsluttes med en beskrivelse af, hvordan TeSS-systemets hoveddele er placeret i DIKU's maskinmiljø.

1.1 TeSS-projektets formål og tilrettelæggelse

Formålet med TeSS-projektet har været at designe og implementere et system til eksperimentel undersøgelse af brugergrænseflader til edb-baseret tekstsøgning. Mere specifikt blev projektets indhold og form tilrettelagt ud fra et ønske om at bringe primært følgende tre hensyn i 'positiv samklang':

- I foråret 1993 fik emnet 'brugergrænsefladedesign' for første gang en selvstændig placering på datalogistudiets grunduddannelse med et omfang svarende til en studiebelastning på ca. et halvt semester, se kursusbogen Frøkjær & Hessellund (1993). Som et led heri skal indgå en rapportopgave, som giver de studerende lejlighed til at arbejde selvstændigt med problemstillinger af væsentlig betydning inden for brugergrænsefladedesign, se opgaveformuleringen Frøkjær (1993), gengivet i bilag D.
- Emnet 'edb-støtte til fagfolks tekstsøgning' er den centrale forskningsmæssige interesse for Morten Hertzum i hans PhD-studium. Erik Frøkjær har i de senere år arbejdet inden for feltet, bl.a. med et designforslag til et tekstsøge- og redigeringssystem til Karnovs Lovsamling og med undersøgelser af såkaldte ekspertsystemer. Som led i denne forskning har Morten Hertzum og Erik Frøkjær ønsket at opbygge et system, der muliggør eksperimenter med forskellige brugergrænseflader i edb-baserede systemer til støtte for fagfolks informationssøgning.

- Blandt de studerende på datalogistudiets overbygningsuddannelse er der ofte interesse for at deltage i lidt større forsknings- og udviklingsprojekter. En sådan studieaktivitet kan tilrettelægges som et sædvanligt, kreditgivende skriftligt projekt, hvor der kan forventes en særlig aktiv opbakning og støtte fra vejlederside. Ofte vil sådanne projekter kunne være direkte specialeforberedende. Et yderligere perspektiv i dette projekt har været, at et vellykket resultat af en fælles udviklingsindsats skulle indgå som grundlag for en ambitiøs rapportopgave for de studerende på Datalogi 2.

På denne baggrund - og efter indledende at have sonderet om DIKU's edb-afdeling kunne tilvejebringe de nødvendige materielle og programmelmæssige ressourcer for et interessant projekt - besluttede Erik Frøkjær og Morten Hertzum i slutningen af september 1992 at invitere interesserede andendelsstuderende til at gå ind i projektet. På det tidspunkt var projektet endnu meget åbent, hvilket gav de studerende mulighed for at deltage i de afgørende beslutninger om designet af tekstsøgesystemets grafiske brugergrænseflader og om valget af programmeringsværktøjer. Følgende overordnede valg lå dog som rammer for udviklingen af TeSS.

1. Der skulle benyttes grafiske brugergrænseflader

Kravet om grafiske, vinduesbaserede brugergrænseflader baseret på 'direkte manipulation' følger af, at denne interaktionsform i dag har en meget fremtrædende plads i arbejdet med udvikling af mere brugervenlige edb-systemer, se f.eks. Shneiderman (1992) og Mandelkern (1993). Fra starten af projektet var det klart, at nogle af de interaktionsformer, vi ønskede i TeSS, var inspireret af og faktisk krævede en grafisk, vinduesbaseret brugergrænseflade. Ligeledes var det ønskeligt, at de studerende under løsningen af rapportopgaven i et eller andet omfang - i år eller senere - kunne arbejde selvstændigt med programmering af en sådan brugergrænseflade.

2. Der skulle benyttes relationelle databaseteknikker

Selve tekstdatabasen skulle baseres på relationelle databaseteknikker, så projektet kunne give mulighed for fortsatte undersøgelser af bæredygtigheden af de ideer, som Erik Frøkjær og Morten Hertzum arbejder med i tilknytning til tekstsøgesystemer for fagfolk Hertzum et al. (1993). Endnu i begyndelsen af 1990'erne er databaserne i flertallet af de eksisterende tekstsøgesystemer baseret på inverterede filer; relationsdatabaserne, introduceret af Codd (1970), blev i løbet af 1980'erne efterhånden det foretrukne hjælpemiddel til at håndtere og lagre strukturede data. Morten Hertzum og Erik Frøkjær ønsker at undersøge, hvilke fordele og omkostninger, der knytter sig til at benytte relationsdatabaser også i forbindelse ustrukturerede data som tekst og figurer.

3. Objektorienterede programmeringsteknikker blev anset for ønskelige

Vi ønskede at benytte objektorienterede programmeringsteknikker til design og konstruktion af TeSS ud fra det overordnede synspunkt, at disse teknikker ville støtte os i at opnå et overskueligt programdesign med en struktur, der er bekvem ved senere modifikation og udbygning af systemet. Brooks (1986) udpeger objektorienteret programmering som en af de mere lovende udviklinger indenfor programmeringssprog, og generelt vil et objektorienteret programmeringsmiljø med et vel-udbygget objekt-bibliotek give os mulighed for effektivt genbrug af programmel.

TeSS skulle iøvrigt gerne kunne anvendes af de studerende ved løsning af opgaver, som på en naturlig måde kunne indgå som et led i undervisningen i brugergrænsefladedesign. Kun herved kunne vi forvente, at de studerende ville arbejde seriøst med tekstsøgesystemet. En realistisk mulighed forekom at være at udnytte offentligt tilgængelige tekster, der allerede forelå på elektronisk form. Den primære ide var at finde tekster fra edb-manualer, der omhandlede programmeringsværktøjer til brug ved programmering af grafiske brugergrænseflader. Denne ide er blevet realiseret; men undervejs har andre muligheder også været inde i billedet, f.eks. *ACMs Guide to Computing Literature*, der findes som CD-ROM i en udgave med mange årgange samlet.

Formål

Sigtet med TeSS-projektet er at designe og implementere et tekstsøgesystem, som udover at kunne bruges til tekstsøgning også er egnet til eksperimentel undersøgelse af brugergrænseflader til tekst-søgesystemer. Systemet tænkes benyttet af de studerende på Datalogi 2, som omdrejningspunktet for deres rapportopgave i brugergrænsefladedesign. Der skal endvidere etableres sådanne faciliteter, at dat2ernes interaktion med TeSS - under bevarelse af anonymitet - kan logges. De herved opsamlede data skal benyttes i senere analyser, som falder uden for dette projekt.

TeSS skal omfatte flere forskellige brugergrænseflader til de inkluderede edb-manualer, dvs. flere forskellige muligheder for søgning i manualerne. To væsensforskellige typer brugergrænse-flader til tekstsøgesystemer er udviklet i hver deres miljø og ses kun sjældent kombineret. I for-bindelse med hypertekstsystemer er der udviklet en række faciliteter til at bevæge sig rundt i det netværk af tekststykker, som udgør den pågældende hypertekst. Man læser en hypertekst ved at læse et tekststykke - helt eller delvist - og derudfra beslutte, om man vil følge en af referencerne videre til andre relaterede eller uddybende tekststykker. Alternativt kan man gå tilbage gennem de tekststykker, man kom fra, for at undersøge andre veje gennem teksten. Denne måde at læse eller søge i en tekst kaldes browsing.

Indenfor området informationssøgning har man traditionelt taget udgangspunkt i biblioteks-verdenen, hvor der ud fra forespørgsler skal fremfindes en lille mængde dokumenter ud fra meget store samlinger af dokumenter, se Rasmussen (1992). Søgning ved hjælp af denne type søge-systemer består i, at man formulerer en forespørgsel, som beskriver det man ønsker information

om. Som svar på forespørgslen leverer søgesystemet samtlige dokumenter, der opfylder forespørgslen. Udfra de fremfundne dokumenter kan forespørgslen reformuleres. I faglitteraturen er der imidlertid en tendens til at rette opmærksomheden mod faciliteter, der ikke udnytter muligheden for iteration, men sigter på at levere det optimale svar første gang, se Bates (1990).

Vi forestiller os, at tekstsøgning er en iterativ proces, hvor allerede fundne tekster og anden feedback spiller afgørende ind i søgningens videre forløb. Vi finder derfor browsing-faciliteter meget væsentlige, selvom vi betragter dem som utilstrækkelige i sig selv. Der er stor risiko for at miste overblikket og fornemmelsen af, hvorvidt man får det hele med. Søgning ved forespørgsler er udviklet til søgning i store datamængder. Søgning ved forespørgsler tilbyder endvidere en meget direkte vej til de relevante dokumenter, forudsat at man er i stand til at formulere en rammende og dækkende forespørgsel. Men der gives ingen hjælp til formuleringen af den indledende forespørgsel. Vi tror, at browsing og søgning ved forespørgsler vil supplere hinanden godt, hvis de sammentænkes grundigt og stilles til rådighed i samme søgesystem.

TeSS skal således tilbyde både browsing og søgning ved forespørgsler. Udover en traditionel udformning af søgning ved forespørgsler vil vi arbejde med en udformning, hvor der tilstræbes en enklere angivelse af forespørgslerne og en mere omfattende og nuanceret feedback fra systemet, som svar på de stillede forespørgsler.

Rapportens opbygning

Efter indledningen i dette kapitel beskrives og diskuteres i kapitel 2, hvordan TeSS er opbygget. Først præsenteres i afsnit 2.1 de bærende ideer i TeSS. I de efterfølgende afsnit beskrives TeSS-systemets delkomponenter mere detaljeret: først brugergrænsefladen, dernæst den underliggende tekstdatabase, og endelig objektmodellen for TeSS.

I kapitel 3 redegøres for det forsøg, som TeSS skulle indrettes på. Først behandles i afsnit 3.1 målsætninger med og tilrettelæggelse af forsøget. I afsnit 3.2 behandles dernæst, hvorledes logningen af brugerinteraktionen er etableret. I afsnit 3.3 beskrives, hvorledes TeSS blev forsynet med et forsøgsstyringsmodul, som leder brugerne gennem forsøgets opgaver på den særlige måde, som følger af det forskningsmæssige sigte.

I kapitel 4 sammenfattes projektet. Til sidst er vedlagt bilag, der indeholder brugervejledning til TeSS, definition af relationsdatabase, brugergrænsefladens objekttræ, opgaveformuleringen, opgaverne til forsøget og logningsformatet. Som sidste bilag opsamles erfaringer fra driften af TeSS under afviklingen af rapportopgaven og forsøget. Vi har valgt at placere bilagene til denne rapport i et særskilt bind, blandt andet fordi flere læsere kun vil have behov for hovedrapporten.

1.2 Valg af programmeringsværktøjer

I dette afsnit redegør vi for, hvilke programmeringsværktøjer vi har brugt ved udvikling af TeSS og det tilhørende forsøgsstyringsmodul. Vi fremlægger de bindinger, projektet var underlagt, og begrunder de valg, vi har foretaget.

Ved valg af værktøjer skulle følgende krav opfyldes:

- Det skulle være teknisk muligt at gennemføre projektet og implementere et system, som kunne muliggøre eksperimentel undersøgelse af interessante brugergrænseflader til edb-baseret tekstsøgning.
- De stramme tidsmæssige rammer for projektet skulle respekteres, dvs. en driftsmæssig forsvarlig version af tekstsøgesystemet skulle være klar ultimo januar 1993. Ligeledes var der grænser for projektgruppens mulige indsats af arbejdstid.
- Det skulle være teknisk muligt at gennemføre en interessant rapportopgave på Datalogi 2, i kombination med forsøget.
- Arbejdet skulle ligge i forlængelse af Morten Hertzums og Erik Frøkjærs arbejde med at opbygge tekstsøgesystemer med udgangspunkt i relationsdatabaser.

Vi stod overfor følgende valg: Vi skulle bestemme os for en hardware-platform og et vinduessystem. Endvidere skulle vælges databasesystem, programmeringssprog, og evt. et sæt biblioteksrutiner til brug ved opbygning af brugergrænsefladen. Vi ville også undersøge muligheden af at arbejde med et programmeringsværktøj, der er specielt indrettet til effektivt at opbygge grafiske brugergrænseflader. Sådanne værktøjer betegnes ofte 'Graphical User Interface Builder' (GUIB) eller 'User Interface Management System' (UIMS), men terminologi og definitioner er ret uklare. En introducerende og principiel beskrivelse af værktøjer til opbygning af grafiske brugergrænseflader kan findes i Bass & Coutaz (1991), i Myers (1989) og i Shneiderman (1992).

Vinduessystem
Da systemet skulle bruges i forbindelse med en rapportopgave på Datalogi 2 var det mest hensigtsmæssigt at udvikle det til DIKU's Unix-miljø (se afsnit 1.3). Heraf fulgte, at vinduessystemet skulle være X Windows. Det er i øjeblikket ikke realistisk at tilbyde dat2erne at køre på Macintosh, hvilket havde været en anden interessant platform for et system af denne art.

Databasesystem
Projektgruppen undersøgte mulighederne for at få bevilget relationsdatabasesystemet Sybase, så dette kunne benyttes i stedet for instituttets nuværende relationsdatabasesystem, Ingres. Årsagerne hertil var mange; men her skal alene fremhæves det forhold, at Ingres ikke stiller en datatype til

rådighed, som er egnet til håndtering af større tekster, tabeller eller billeder. Den maksimale feltstørrelse i Ingres er således 2 kbyte; i Sybase 2 gigabyte. Vi frygtede, at de måder, hvorpå vi kunne komme udenom problemet i Ingres, ville vise sig at have uheldige sidevirkninger, både ved at påføre projektgruppen et større programmeringsarbejde og ved at medføre ringere svartider i det færdige system. Projektgruppens ansøgning om Sybase blev imidlertid ikke godkendt, og projektgruppen besluttede at arbejde videre med tekstsøgesystemet under den betingelse, at Ingres skulle benyttes.

Programmeringssprog

Programmeringssproget skulle kunne bruges med de øvrige værktøjer, dvs. databasen og X Windows systemet. Sproget skulle også passe sammen med et eventuelt GUIB-værktøj og måtte desuden ikke være helt ukendt for projektgruppen. Det var vigtigt at udnytte så meget af vores tidligere erfaring som muligt, da vi havde kort tid til at udvikle systemet.

Vi havde to muligheder, C og Beta. I gruppen var der gode erfaringer med at programmere i Beta i forbindelse med en grafisk brugergrænseflade. Der var også erfaring med at kommunikere med en database fra Beta via C. Dog var disse erfaringer indhøstet i forbindelse med programmering på Macintosh, hvor der i dette projekt skulle benyttes Unix-arbejdsstationer. Fordele ved C var mindre usikkerhed om samspil med de øvrige værktøjer og større ekspertise at trække på, f.eks. fra edb-afdelingen, i tilfælde af problemer. Ulemperne ved C var, udover vores manglende erfaring, at vi var nervøse for omfanget af ekstra programmering ved udvikling af et ikke-objektorienteret program omkring en brugergrænseflade opbygget i et objektorienteret vinduessystem.

Med i overvejelserne om programmeringssprog var også den mulige opgave til dat2erne. Da de ikke har fået undervisning i Beta, kunne vi kun regne med at stille dem en opgave, hvor de skulle rette i systemets ressource-fil - ikke i programmet. På dette tidspunkt var vi ikke klar over, hvilke dele af brugergrænsefladen der kunne lægges i ressource-filen, og var derfor usikre på, om vi udelukkede en fornuftig rapportopgave ved at vælge Beta. På den anden side var vi, alene på grund af kravene til diskplads, usikre på, om det overhovedet ville kunne lade sig gøre at lade knap 100 dat2ere lave hver deres version af tekstsøgesystemet. Som ansvarlig for TeSS-systemets brug i forbindelse med rapportopgaven på Datalogi 2 anlagde Erik Frøkjær her det synspunkt, at projektgruppen alene skulle tage hensyn til, hvorledes et tekstsøgesystem hurtigt og mest sikkert kunne udvikles af projektgruppen. Herefter var det hans opgave at udforme en rimelig implementationsopgave som en del af den samlede rapportopgave - og det burde kunne gøres uafhængigt af hvilke programmeringsværktøjer, der blev anvendt. På den måde blev hensynet til implementationsopgaven klart underordnet hensynet til en hurtig og sikker realisering af tekstsøgesystemet.

Vi valgte Beta, primært fordi gruppens samlede erfaring med sproget var større end vores

erfaring med C, ihvertfald i forbindelse med udvikling af grafiske brugergrænseflader. Dernæst håbede vi, at Beta ville spare os kostbar tid ved selve programmeringsarbejdet. Vi forventede at kunne udnytte muligheden af at opbygge programmet om de samme objekter, som skulle optræde i brugergrænsefladen. Yderligere håbede vi, at vores indlæring og brug af X Windows systemet ville blive lettet i kraft af, at Mjølner Beta systemet har en objektorienteret grænseflade hertil. Valget af Beta betyder dog ikke, at vi slet ikke har brugt C; bl.a. foregør al kommunikation mellem Beta og Ingres via C-moduler.

Widget Set
Et Widget Set er en overbygning til X Windows, der implementerer en samling standardvinduer, menuer, knapper osv. Ved brug af et Widget Set spares megen programmeringstid, og man får væsentlig støtte til at sikre vigtige former for ensartethed i brugergrænsefladen. Vi havde mulighed for at vælge mellem Athena Widgets og Motif, der begge findes på DIKU. Mjølner Beta har lavet grænsefladepakker til begge disse widget sets. Vi valgte Athena, da Mjølner Beta systemets Motif-pakke ikke var gennemprøvet på større opgaver.

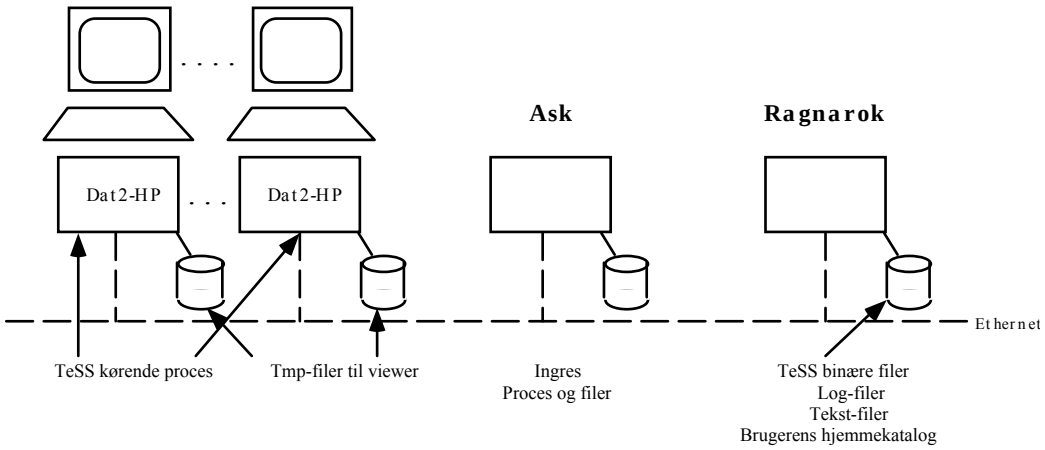
GUIB
Vi ville gerne arbejde med et sådant værktøj, da vi håbede, det ville spare os tid - den helt kritiske ressource i projektet. Desuden syntes vi, det ville være godt at vise dat2erne et GUIB-værktøj og give dem lejlighed til arbejde med det. En ulempe ved at vælge et sådant system var, at vi ikke frit kunne vælge programmeringssprog - de systemer vi havde indenfor rækkevidde genererede C-kode. Hvis vi ville bruge Beta, kunne vi derfor kun udnytte de ressource-filer værktøjet genererede, ikke koden. Der ville derfor blive et vist dobbeltarbejde, som gevinsten ved at bruge en GUIB skulle opvejes imod. Vi kom dog ikke i denne valgsituation, da det ikke lykkedes os at finde et passende GUIB-værktøj. Vi fandt et 'public domain' system, Dirt, som vi fik edb-afdelingen til at installere. Det kom dog aldrig til at fungere stabilt, og vi opgav at undersøge flere systemer.

TeSS er altså en Unix-applikation under X Windows, implementeret ved hjælp af Beta, Athena Widgets, C og Ingres. Under udviklingen af TeSS har vi benyttet en række manualer for disse værktøjer. Det drejer sig om Beta (1992a, b, c, d) for Beta, Beta (1992 d) for Athena Widgets, Kernighan & Ritchie (1988) for C, og Ingres (1991a, b, c) for Ingres.

1.3 Edb-udstyr

TeSS skulle implementeres i DIKU's Unix-miljø på en sådan måde, at systemet kunne benyttes af de studerende via instituttets dat2-maskiner, dvs. de 15 arbejdsstationer af mærket HP 9000 serie 300. Af tilsvarende 'hårde' bindinger kan fremhæves, at relationsdatabasesystemet, Ingres, til

håndtering af tekstdatabasen skulle placeres på en server-maskine af mærket HP 9000 serie 400, kaldet Ask. Ingres-systemet var i forvejen placeret på denne maskine og skulle i perioden umiddelbart inden afviklingen af brugergrænseflade-rapportopgaven benyttes til undervisningen i databaser på Datalogi 1. Under afviklingen af brugergrænseflade-rapportopgaven ville Ask blive friholdt fra andre ressourcekrævende opgaver end netop TeSS. Herved fik projektgruppen på forhånd af edb-afdelingen stillet sådanne maskinressourcer i udsigt, at det forekom forsvarligt at forsøge at realisere et tekstsøgesystem efter de overordnede krav og retningslinjer, som var fastlagt ved projektstarten.



Figur 1.3-1: Mål-konfigurationen - det maskinmiljø, som projektgruppen sigtede på som grundlaget for den endelige driftsafvikling af TeSS-systemet.

På figur 1.3-1 ses en skitse af den maskin-konfiguration, som blev mål-konfiguration for TeSS-systemet, dvs. det maskinmiljø projektgruppen skulle sigte på som grundlaget for den endelige driftsafvikling af TeSS-systemet under dat2ernes rapportopgave. Udover den nævnte opdeling i klient-maskinerne, de 15 HP300-maskiner til afvikling af TeSS-applikationen, og Ask-databaseserveren, bemærkes en tredie maskine, kaldet Ragnarok. Ragnarok håndterer de studerendes hjemmekataloger og er således et nødvendigt led i deres logon-procedure. På denne maskine valgte vi tillige at lagre de filer og logdata, som skulle knyttes til TeSS-systemet. Her er tale om TeSS-applikationens binære kode, filer med logdata samt filerne med manualteksterne.

Når tekstfilerne ikke er placeret i databasesystemet skyldes det, at Ingres ikke stiller en datatype til rådighed, som kan håndtere større tekster. Ingres håndterer de inverterede lister og andre datastrukturer, der gør effektiv søgning mulig, mens de faktiske tekster hentes frem fra Unix-filsystemet. Disse ting findes nøje behandlet i afsnit 2.3.

Når brugeren under tekstsøgningen finder et stykke tekst frem for at læse i det, er det helt nødvendigt med korte svartider. Derfor var det tidligt inde i billedet, at fremfundne tekststykker temporært skulle overføres til brugerens arbejdsstationen, og lagres dér for inspektion i en 'viewer'. Også det er med markeringen 'tmp-filer til viewer' illustreret på figur 1.3-1.

2 Tekstsøgesystemet TeSS

TeSS er et eksperimentelt system til tekstsøgning, udviklet specielt med henblik på studier af brugergrænsefladen. I dette kapitel beskrives, hvordan TeSS er bygget op. Vi beskriver brugergrænsefladens udseende og dens funktionalitet; organiseringen af de tekster, TeSS tilbyder søgning i; og den model, hvorefter selve programmet er bygget op. Den version af TeSS, der udvikles gennem dette kapitel, har som sigte at støtte programmører og systemudviklere ved på en enkel og bekvem måde at give adgang til en række edb-manualer. TeSS omfatter tre manualer for værktøjer til udvikling af X-applikationer med grafiske brugergrænseflader:

- Xlib - C Language Interface
- X Toolkit Intrinsics - C Language Interface
- Athena Widget Set - C Language Interface

Senere versioner af TeSS vil eventuelt give mulighed for søgning i yderligere manualer, eller i helt andre typer tekster.

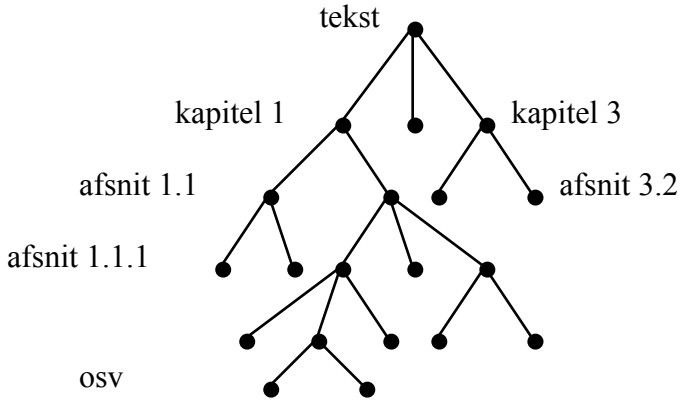
Udviklingen af TeSS har været centreret om to hovedspørgsmål: “Hvordan skal teksterne repræsenteres?” og “Hvordan skal der kunne søges i dem?” Disse to spørgsmål går igen gennem hele dette kapitel. Derfor vil vi - inden vi vender os mod udviklingen af brugergrænsefladen, organiseringen af den underliggende tekstdatabase osv.- gøre rede for de overordnede ideer i vores svar på de to hovedspørgsmål.

2.1 Grundlæggende ideer i TeSS

TeSS skal være et system til at søge i tekster med. Forskellige typer tekster vil resultere i forskellige krav til søgesystemet, f.eks. vil tekstens struktur have stor indflydelse på, hvilke browsing-faciliteter, systemet skal stille til rådighed. Vi må derfor gøre os klart, hvad vi vil forudsætte om de tekster, TeSS skal indeholde.

Det er afgørende, at TeSS er velegnet til edb-manualer, men vi har også lagt vægt på at systemet kan bruges til andre typer tekster. Edb-manualer er blot een type tekster; andre tekster er anderledes opbygget og skrevet til andre brugssituationer. Så at sige al teknisk litteratur er hierarkisk opdelt; i modsætning til f.eks. et leksikon. I visse hierarkisk opdelte tekster er opdelingen endvidere så grov, at den ikke giver tilstrækkeligt overblik over tekstens struktur, f.eks. en roman opdelt i kapitler.

Et stort antal tekster, inklusive de manualer TeSS skal indeholde, kan repræsenteres hierarkisk. En forudsætning om at teksterne er hierarkisk opbygget, dvs. opdelt i kapitler, afsnit og underafsnit, eller en anden lignende opdeling, er derfor rimelig.



Figur 2.1-1. Hierarkisk opdelt tekst.

En søgning i teksterne består i at udvælge en eller flere dele af tekstsamlingen til nærmere undersøgelse. Denne udvælgelse kan i TeSS ske ved browsing eller ved forespørgsler. De tekster, der udvælges ved en søgning, kan præsenteres på skærmen til læsning.

Browsing

Browsing er en søgeteknik, der specielt kendes fra hypertekst-systemer. At browse (dansk: bladre, skimme) kan være at søge efter noget, man ikke ved hvad hedder (eller ikke engang ved hvad er), at orientere sig i en tekst, eller blot at lade sig inspirere ved at bladre rundt og "se hvad der er". Forskellige faciliteter til browsing er f.eks. beskrevet i Conklin (1987), Thompson & Croft (1989) og Nielsen (1990). Browsing-faciliteter kan være mulighed for at bevæge sig rundt i en teksts hierarkiske struktur; mulighed for at følge referencer (links) rundt i et netværk af tekststykker; eller mulighed for at hoppe til andre steder i teksten, uden at der på forhånd er defineret referencer dertil - såkaldte dynamiske referencer.

TeSS bygges op omkring tekster, der er hierarkisk struktureret. Det afspejles i browsing-faciliteterne, der skal give mulighed for at bevæge sig rundt i den træstruktur, hierarkiet udgør, og se den på forskellige detaljeringsniveauer, f.eks. kapitelniveau eller afsnitsniveau. Det er derimod

ikke muligt at følge referencer i teksten, ved f.eks. at klikke på “se afsnit 11.4”.

Søgning ved hjælp af forespørgsler

Søgning ved hjælp af forespørgsler adskiller sig fundamentalt fra browsing. Mens browsing kan karakteriseres som en udforskende søgeteknik, er søgning ved forespørgsler specificerende. Brugeren specificerer gennem forespørgslen det problem, det emne, den konkrete detalje eller lignende, han er interesseret i. Derefter fremfinder søgesystemet de tekster, der matcher forespørgslen. Den mest udbredte type søgning ved forespørgsler er boolsk søgning, hvor forespørgslerne opbygges af søgeord og boolske operatorer. Denne søgeform er behandlet bl.a. i Belkin & Croft (1987) og Radecki (1988). Boolsk søgning er en eksakt match teknik, idet en tekst enten matcher forespørgslen og bliver fremfundet, eller ikke matcher den og ikke bliver fremfundet. Udover disse teknikker findes en række partiel match teknikker, hvor teksterne sorteres efter i hvor høj grad, de matcher forespørgslen, og præsenteres for brugeren i denne rækkefølge. Een teknik til partiel match kan f.eks. være at lade brugeren angive en tekst, som de ønskede tekster skal ligne. En oversigt over såvel eksakt match som partiel match søgeteknikker findes i Belkin & Croft (1987).

Der er forskellige problemer forbundet med forespørgsler opbygget af søgeord og boolske operatorer, se f.eks. Cooper (1988). Det har vist sig at være vanskeligt for brugere at formulere forespørgsler ved hjælp af de boolske operatorer. Ofte vil brugere opfatte "eller" som bindende stærkere end "og". De vil skrive forespørgsler som "forældre eller fædre og orlov", hvormed der menes "(forældre eller fædre) og orlov", mens systemet vil være opbygget så det tolker spørgsmålet som "forældre eller (fædre og orlov)". Den modsatte situation er naturligvis også mulig.

Et andet problem er valg af søgetermer. Ved eksakt match teknikkerne har brugeren behov for respons fra systemet om, hvilke af de angivne søgeord der er årsagen til et dårligt resultat - alt for mange eller få fremfundne tekster - og denne respons fås ikke ved traditionel boolsk søgning.

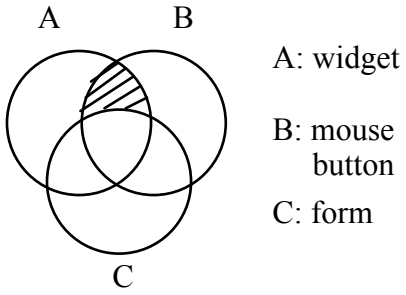
I TeSS foretages søgning ved forespørgsler ud fra en eksakt match teknik. En forespørgsel angiver således, hvilke ord der *skal* være i de tekster, der findes frem, og hvilke ord der *ikke må* forekomme. TeSS omfatter to forskellige udformninger af denne teknik, idet forespørgslerne enten kan formuleres som logiske udtryk eller ved hjælp af et mængdediagram, også kaldet et Venn-diagram. Forespørgsler formuleret som logiske udtryk er traditionel boolsk søgning; en sådan forespørgsel kan se således ud:

```
widget AND (mouse OR button) AND NOT form
```

Eksemplet her resulterer i fremfinding af de tekster, der indeholder ordet *widget* og mindst et af ordene *mouse* og *button*, mens ordet *form* ikke findes i teksten.

Venn-diagrammer blev i det 19. århundrede præsenteret af matematikeren John Venn. Ideen om at formulere forespørgsler ved hjælp af Venn-diagrammer, istedet for logiske udtryk, blev

tidligt præsenteret af Smith (1976). Hun forestillede sig dem imidlertid tegnet på papir som et led i dialogen mellem bibliotekar og låner, før bibliotekaren foretog selve søgningen. Senere har Michard (1982) designet et søgesystem, hvor forespørgslerne blandt andet formuleres ved hjælp af Venn-diagrammer. Med inspiration fra hans brugergrænseflade kan den samme forespørgsel som ovenfor specificeres således:



Figur 2.1-2 Forespørgslen "widget AND (mouse OR button) AND NOT form" formuleret ved hjælp af Venn-diagram.

Hver af de tre cirkler repræsenterer en delmængde af hele tekstmængden. I det efterfølgende bruger vi betegnelsen "grundmængde" for en sådan delmængde. På figuren repræsenterer grundmængde A altså de tekster, der indeholder ordet *widget*; grundmængde B repræsenterer de tekster, der indeholder mindst et af ordene *mouse* og *button*; og grundmængde C repræsenterer de tekster, der indeholder ordet *form*.

Det samlede søgesystem

Browsing og søgning ved forespørgsler skal af hensyn til forsøget kunne bruges hver for sig; men da vi bl.a. vil undersøge om de supplerer hinanden, skal de også kunne kombineres. Brugeren skal have mulighed for at bruge browsing-faciliteterne til at udvælge den mængde tekst, en efterfølgende forespørgsel skal give anledning til søgning i. Denne facilitet vil give brugeren mulighed for at fokusere sin søgning, hvis han på forhånd ved noget om, hvor han skal søge svar på sin forespørgsel. Værdien af at kunne fokusere søgningerne vil vokse, efterhånden som TeSS udvides til at omfatte flere manualer.

Browsing-faciliteterne kan også benyttes til at udforske teksterne og finde egnede søgeord, før brugeren formulerer en forespørgsel. Efter formulering og udførsel af en forespørgsel kan browsing-faciliteterne bruges til at se, hvordan de fremfundne tekster er placeret i den samlede tekst, f.eks. ved at deres overskrifter markeres i træstrukturen. Med udgangspunkt i de fundne tekster kan brugeren bevæge sig ned til mere detaljerede afsnit, op til mere overbliksgivende beskrivelser, eller blot ud til de omkringliggende tekster. Et væsentligt aspekt i det følgende, detaljerede design af TeSS er at indtænke dette samspil mellem de to overordnede søgeteknikker.

2.2 Brugergrænsefladen i TeSS

Formålet med dette kapitel er at give indblik i hvilke begrundelser, TeSS-gruppen havde for designet af brugergrænsefladen. Som beskrevet i 2.1 *Grundlæggende ideer i TeSS* skal TeSS give mulighed for:

- At læse tekst fra skærmen.
- At søge efter relevant tekst med browsing-teknikker.
- At søge ved forespørgsler, formuleret enten som logiske udtryk eller ved hjælp af et Venn-diagram.
- At kombinere disse søgemåder.

Afsnit 2.2 opdeles udfra disse punkter i underafsnit. I hvert underafsnit opridses nogle væsentlige alternativer for design og det valgte design beskrives. For et sammenhængende eksempel på brug af TeSS se brugervejledningen i bilag A.

Designbeslutninger indenfor de enkelte områder har naturligvis haft indflydelse på hinanden og på helheden. Alligevel beskriver vi de enkelte områder hver for sig. Der er dog nogle af beslutningerne angående systemets enkeltdele, der først kan begrundes, når det samlede system præsenteres i afsnit 2.2.5 *Sammenkobling af systemets enkeltdele*.

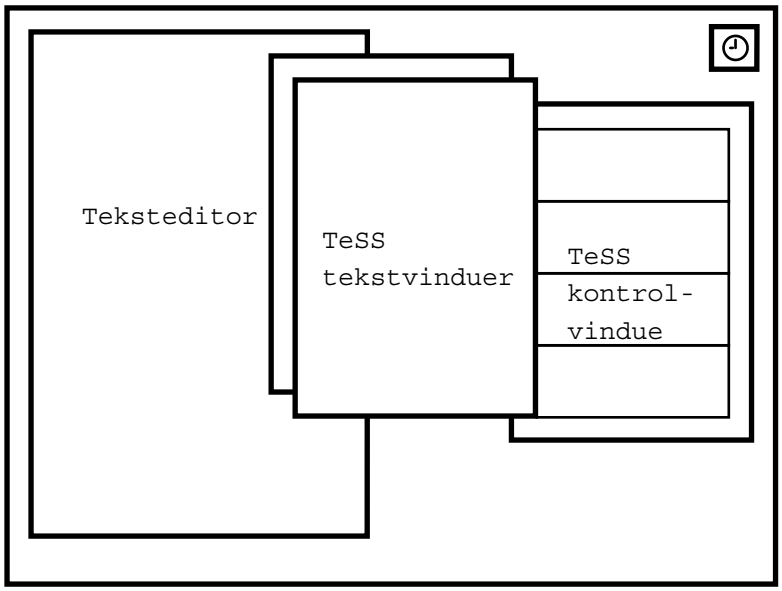
2.2.1 Overordnede retningslinier

I dette afsnit vil vi præsentere nogle overordnede retningslinier, vi har fulgt i designarbejdet. Retningslinierne er ikke blevet til ved at vi fra starten har diskuteret og fastlagt dem udfra nogle teoretiske overvejelser. Designarbejdet er foregået ved at vi har arbejdet udfra konkrete forslag til brugergrænseflade og funktionalitet. I løbet af dette arbejde har der været yderst forskellige forslag fremme, og efterhånden har der udkrystalliseret sig nogle retningslinier. Dem har vi brugt som ledesnor, men ikke fulgt i alle tilfælde.

- Hvad brugeren kan gøre, skal så vidt muligt kunne ses direkte. Der skal for eksempel bruges knapper til funktioner i stedet for menuer og dobbeltklik.
- Objekter med samme funktion skal have samme udseende, for eksempel skal funktionsknapper altid være firkantede.
- Hvis funktionerne bliver irrelevante på grund af brugergrænsefladens tilstand, skal de være utilgængelige, og dette skal kunne ses direkte.
- Der skal være en opdeling af funktionerne i TeSS, sådan at hver funktion får den plads, den behøver, og ikke let forveksles med andre relaterede funktioner.
- Der skal ikke tilbydes flere måder at gøre det samme på.

- Vi bruger kun een af musens tre knapper.
- Når dele af brugergrænsefladen har ensartet udseende, skal de også fungere ensartet.
- Der skal ikke være for mange vinduestyper. Hvis der skal være mange vinduer, skal de let kunne skelnes fra hinanden.

Disse retningslinier kan dels begrundes i vores egne personlige holdninger og smag, og dels i den sammenhæng, TeSS skal indgå i. TeSS er rettet mod edb-fagfolk, ikke søgeeksperter såsom bibliotekarer. Systemet skal kunne bruges efter en kort introduktion, og det skal derfor være være umiddelbart gennemskueligt. Det skal endvidere fungere i en arbejdssituation hvor også andre programmer indgår, f.eks. en teksteditor. Brugerens primere opgave er programmering ikke tekstsøgning. Han koncentration må ikke forstyres af et tekstsøgesystem, hvor han skal huske eller slå op hvordan han skal betjene det.



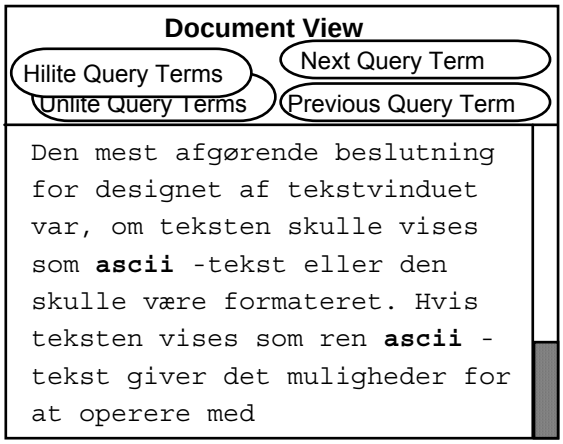
Figur 2.2.1-1 TeSS i en tænkt arbejdssituation, med TeSS' kontrolvindue, flere tekstvinduer med manualtekster og en teksteditor fremme på skærmen.

2.2.2 Tekstvinduet - Text Viewer

Et af kravene til TeSS er, at den relevante tekst skal kunne læses på skærmen. Efter at brugeren har fundet et afsnit, der virker interessant, skal han nemt kunne få hele teksten at se. For at gøre det lettere for brugeren at koncentrere sig om teksten, har vi valgt at præsentere den i et selvstændigt vindue, en "Text Viewer". Brugeren kan have flere tekstvinduer på skærmen samtidigt.

Den mest afgørende beslutning for designet af tekstvinduet var, om teksten skulle vises som ren ascii-tekst eller den skulle være formateret. Hvis teksten vises formateret, skal den tegnes som grafik, hvilket giver en "død" tekst. Hvis teksten vises som ren ascii-tekst giver det muligheder for

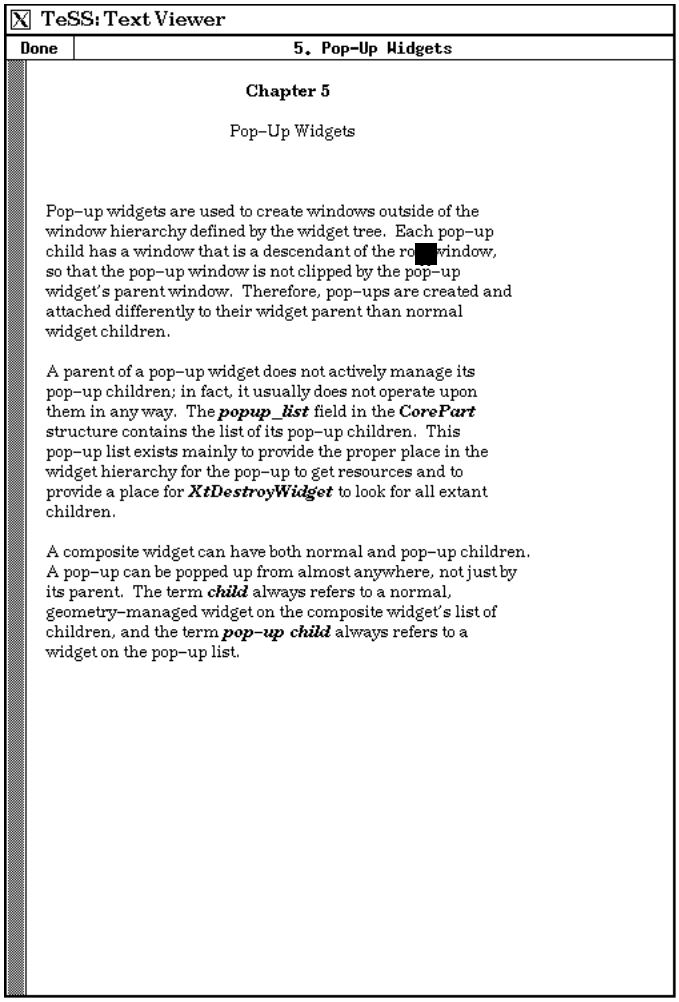
at operere med teksterne i tekstvinduet. Der kan søges på ord, og de søgeord, der forekommer i den viste tekst, kan markeres.



Figur 2.2.2-1: Et af vores tidlige forslag til tekstvinduet. Teksten vises som ren ascii-tekst. Kun een af de to knapper "Hilite Query Terms" og "Unlite Query Terms" er synlige ad gangen.

Vores valg

Vi har valgt at beholde formateringer i teksterne.De manualer, der skal lægges ind i TeSS, er skrevet udfra at formatering (fed, kursiv osv.) er til rådighed. Der er derfor fra forfatternes side lagt betydning i formateringerne - udover de æstetiske - f.eks. bestemt udseende af funktions- og klassenavne. Teksten mister mening hvis formateringen fjernes. Tabeller bliver uoverskuelige og strukturen i teksten sværere at fange. Det færdige tekstvindue har så følgende udseende:



Figur 2.2.2-2 Det endelige design af tekstvinduet "Text Viewer".

Brugeren kan anvende rulle-skakten i venstre side af vinduet til at flytte teksten op og ned i vinduet. Titellinien øverst i vinduet viser tekstens overskrift. Tekstvinduet lukkes ved at klikke på "done" knappen.

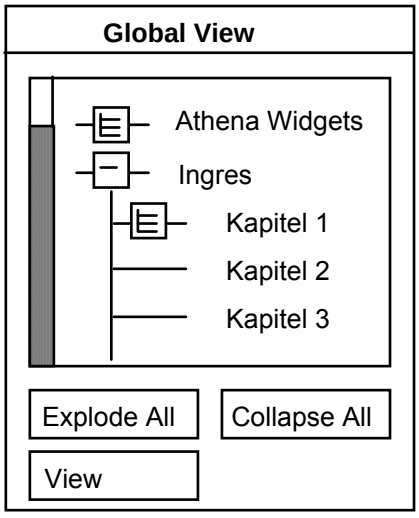
Beslutningen om at bevare teksternes formatering betød at vi fravalgte muligheden for at operere på teksterne i tekstvinduet. At det kan lade sig gøre at have begge dele ved vi f.eks. fra tekstbehandlingsprogrammer på Macintosh og under Windows, men det var ikke muligt for os at finde lignende programmer under X Windows. Vi skulle derfor selv have kodet det hele, hvilket ikke var muligt med den tid, vi havde til rådighed.

2.2.3 Browsing

Browsing-faciliteterne i TeSS skal støtte brugeren i at orientere sig i tekstsamlingen. Der kan bladres i indholdsfortegnelserne for manualerne, og skimmes igennem relevant tekst i tekstvinduer. Browsing-søgeformen er oplagt, når brugeren søger et centralt begreb eller et emneområde i manualerne, da centrale emner som regel vil kunne findes i indholdsfortegnelsen. Et eksempel kunne være at brugeren vil finde tekst om menuer og muligheder for deres udseende og udarbejdning. I indholdsfortegnelsen for Athena Widgets, en af TeSS' manualer, finder han kapitel 4, der hedder *Menus*. Brugeren får dette kapitel vist i et tekstvindue og skimmer den for at finde den relevante tekst. I underafsnit 4.2 beskrives alle muligheder for at bestemme menuers udseende.

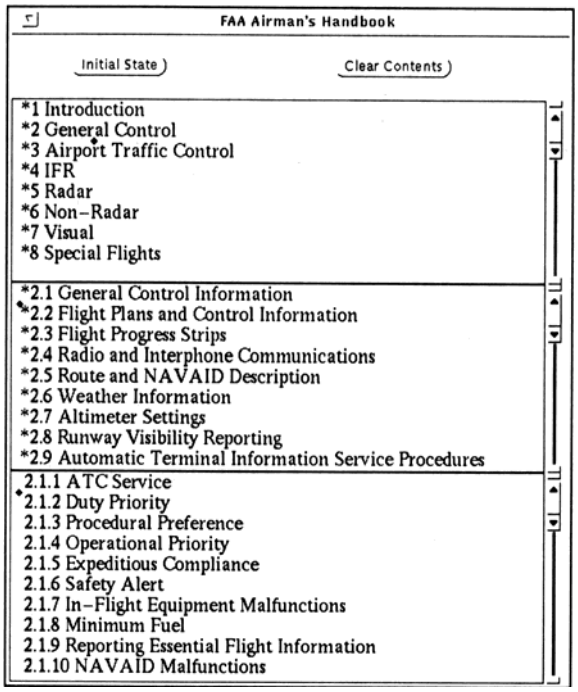
Manualerne i TeSS er struktureret hierarkisk og kan præsenteres som et træ, som beskrevet i afsnit 2.1 *Grundlæggende ideer i TeSS*. Til enhver knude i træet knyttes en overskrift fra indholdsfortegnelsen og den tilhørende tekst indtil den næste overskrift. Browsing-faciliteterne i TeSS giver brugeren mulighed for at vælge, i hvilket omfang træets overskrifter vises i brugergrænsefladen.

Der findes mange måder at lave en brugergrænseflade til at arbejde med denne træstruktur. Een er grafisk, sådan at relationer mellem kapitler og underkapitler beskrives med en linie imellem dem. Ikoner på linierne kan så vise status for grenene i træet, hvor meget der er udfoldet eller synligt. En sådan fremstilling giver et overskueligt billede af tekstens struktur.



Figur 2.2.3-1. Eksempel på et af vores tidlige forslag til grafisk fremstilling af træstrukturen. Symbolerne viser at manualen *Athena Widgets* kan udfoldes. Manualen *Ingres* er udfoldet, og kapitlerne 2 og 3 kan ikke udfoldes yderligere.

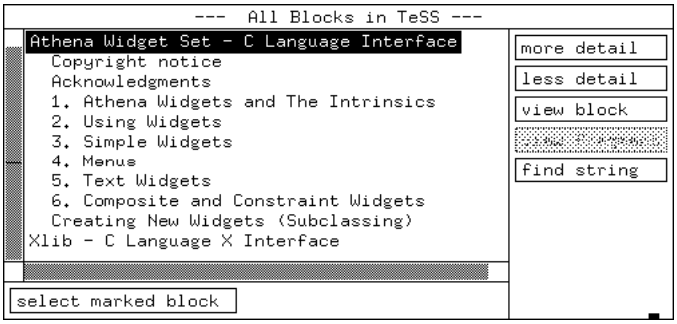
En anden repræsentation af en træstruktur er en liste med overskrifterne, hvor niveaudelingen af teksten gøres tydelig med en fordeling imellem delvinduer sådan at hvert delvindue indholder et bestemt niveau, det første kapitelniveau, det andet afsnitniveau og det tredje underafsnitniveau.



Figur 2.2.3-2 Niveaudelt browsing vindue, Shneiderman (1992). Det øverste delvindue viser kapitlerne i en tekst. Det midterste delvindue viser underafsnit til kapitel 2, og det sidste delvindue viser underafsnit til afsnit 2.1. En stjerne viser at et afsnit kan udfoldes, og et ruder-tegn viser hvilket afsnit der er udfoldet.

Vores valg

Vi har valgt at vise indholdsfortegnelsen i et delvindue, som en liste af overskrifter. Indrykning af overskrifterne viser deres niveau i træstrukturen. Delvinduet "All Blocks in TeSS" har følgende udseende:



Figur 2.2.3-3 Det endelige design af delvinduet "All Blocks in TeSS" med træstrukturen repræsenteret ved indrykning.

Brugeren bladrer i indholdsfortegnelsen ved at vælge en linie af gangen og folde niveauerne ud eller folde dem sammen ved at klikke på "more detail" og "less detail" knapperne. På figuren er manualen for Athena Widgets valgt, og der er klikket på "more detail". For at se hele indholdsfortegnelsen for Athena Widgets kan man klikke 4 gange til på "more detail", da denne manual har 5 niveauer i indholdsfortegnelsens opdeling.

Med funktionen "find string" kan der søges efter tekststrenge i indholdsfortegnelsens overskrifter. Søgningen begynder ved den markerede overskrift, og fortsætter igennem alle overskrifter i træet. Der bliver søgt i alle overskrifter, både synlige og usynlige.

Når brugeren har fundet en interesant overskrift, kan et tekstvindue kaldes frem ved brug af "view block" og "view fragment" knapperne og teksten kan læses på skærmen. Ved klik på "view fragment" i ovenstående eksempel vises teksten i Athena Widgets-manualen indtil 1. kapitel. Dette svarer til at kun teksten for den markerede knude i træstrukturen vises. Derimod vises hele manualen ved klik på "view block", det vil sige "view block" viser teksten for den markerede knude samt alle knuderne i undertræet.

I TeSS kan der kun vælges og opereres med een linie ad gangen i indholdsfortegnelsen, mens der i andre systemer ofte er mulighed for at vælge flere linier ad gangen. Derved bliver brugergrænsefladen mere effektiv i den forstand, at der ikke behøves så mange museklik for at gøre de samme ting. Vi havde svært ved at definere en klar og gennemskuelig funktionalitet af at ud- og sammenfolde flere blokke samtidig. Vi antog derfor at det også ville være sværere for brugeren at forudsige hvorledes funktionerne fungerer, når flere linier er markerede.

Et alternativ til at have en browser og et seperat tekstvindue kunne være at kombinere de to, således at selve teksten kom frem når man udfoldede indholdsfortegnelsens nederste niveau, hvilket blandt andet kendes fra hypertekstsystemet Guide. Dette kan kun lade sig gøre hvis teksten vises uden formateringer. Desuden mener vi, at man nemmere kan fare vild i teksten med et sådant vindue, hvor overskrifter af pladsårsager ikke altid vil være synlige på skærmen.

2.2.4 Søgning ved hjælp af forespørgsler

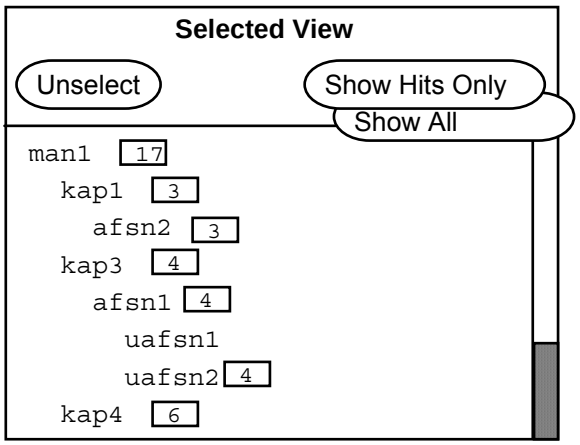
Denne søgeform støtter brugeren i at finde tekster der indeholder bestemte søgeord. Følgende elemneter indgår i:

- Brugeren udvælger de dele af tekstmængden, han ønsker at søge i.
- Brugeren formulerer forespørgslen.
- Brugeren får respons fra TeSS om hvormange og hvilke afsnit der opfylder søgekriteriet.
- Brugeren kan læse eller skimme de fundne tekster.
- Brugeren kan ændre på den udvalgte delmængde og på forespørgslen indtil søgningen giver et tilfredsstillende resultat.

I dette afsnit beskriver vi hvordan tekster udvælges til søgning, hvordan forespørgsler formuleres og hvordan TeSS giver respons om hits, dvs. afsnit der opfylder søgekriteriet.

Overskrifter for de udvalgte tekster og for tekster med hit skal vises i brugergrænsefladen. Overskrifterne vises som træstrukturer ligesom den samlede tekstmængde. Dette gør brugergrænsefladen mere ensartet, idet overskrifter altid optræder i en træstruktur.

En mulighed ville være at vise udvalgte blokke og blokke med hit i samme træstruktur:



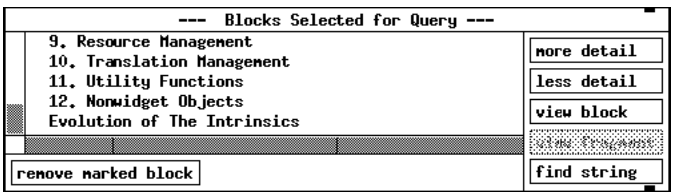
Figur 2.2.4-1 Eksempel på udvalgte blokke og blokke med hit i samme træstruktur. En kasse med antal viser hvormange hit der er det pågældende afsnit. Ved klik på "Show Hits Only" forsvinder blokke uden hit fra træstrukturen, og knappen "Show Hits Only" erstattes af knappen "Show All".

En fordel ved denne sammenkobling kunne være, at brugergrænsefladen kommer til at omfatte et område der er mindre, end hvis de to træstrukturer vises hver for sig. Endvidere bliver det muligt direkte at se, hvilke af de udvalgte afsnit, der ikke indeholder nogen hit.

Vores valg

Vi har valgt at vise udvalgte blokke og blokke med hit i hver sin træstruktur. Vi mener at dette vil hjælpe brugeren med at koncentrere sig om een del af søgeprocessen ad gangen.

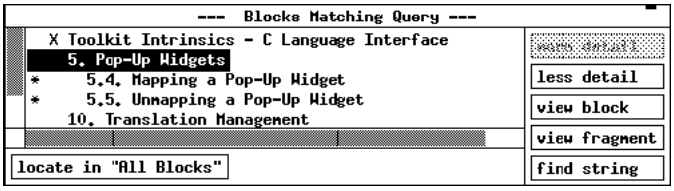
Udvalg af tekstmængde for søgningen sker fra delvinduet "All Blocks in TeSS", ved at markere en linie og klikke på knappen "select marked block", dermed vises den udvalgte linie i delvinduet "Blocks Selected for Query":



Figur 2.2.4-2 Endeligt design af delvinduet "Blocks Selected for Query", hvor blokke udvalgt til søgning vises.

En træstruktur for dette delvindue bygges på den måde op af brugeren. Udvalgte linier kan fjernes fra træstrukturen ved at klikke på knappen "remove marked block". Visning af hvilke blokke, der

opfylder søgekriteriet, sker i delvinduet "Blocks Matching Query":



Figur 2.2.4-3 Endeligt design af delvinduet "Blocks Matching Query", hvor resultatet af en forespørgsel vises.

De afsnit som opfylder søgekriteriet er markeret med en stjerne. De andre afsnit er taget med for at vise, hvor i den samlede tekstmængde de fundne blokke hører til, hvilket svarer til at vise alle knuder over en knude med hit i.

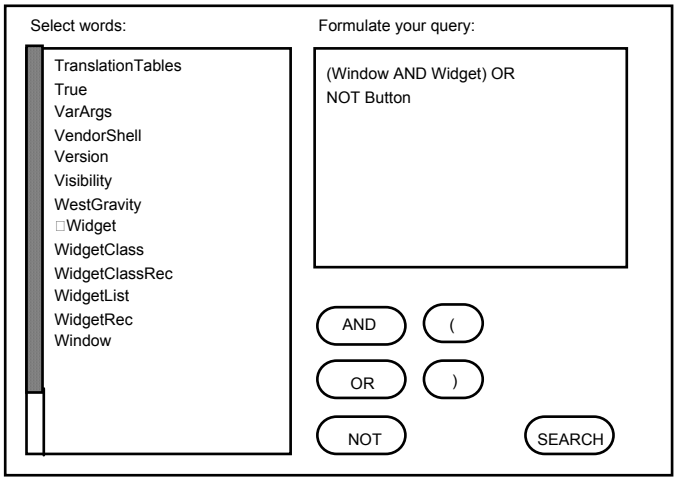
Forespørgsler kan i TeSS formuleres på to måder: Med logiske udtryk og ved brug af Venn-diagrammer. De to måder at formulere forespørgsler på behandles i de to efterfølgende afsnit.

Forespørgsler ved logiske udtryk

I denne søgeform formuleres forespørgsler ved brug af de logiske operatorer, AND, OR og NOT samt paranteser (). Hvis brugeren vil se de afsnit i manualerne, der indeholder ordene "menu" og "button", eller "width", formuleres det som:

(menu AND button) OR width.

I brugergrænsefladen skal det være muligt at formulere en sådan forespørgsel. Dette kunne gøres ved at vælge søgeordene fra en liste og sammensætte dem ved at klikke på knapper.

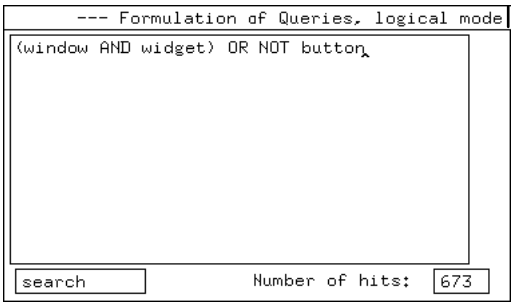


Figur 2.2.4-4 Designide til formulering af logiske udtryk

Her fås hjælp til at afgøre hvilke søgeord der findes, og knapperne viser hvilke operatorer man har til rådighed ved opbygning af udtryk.

Vores valg

Vi har valgt at give brugeren adgang til et tekstfelt hvor forespørgslen tastes ind. Eftersom vores brugere er fortrolige med formulering af logiske udtryk med boolske operatorer, syntes vi ikke det var nødvendigt at have støtteknapper til valg af disse. Listen over søgeord har vi fravalgt fordi mængden af søgeord i TeSS er for stor (> 61.000). Delvinduet har følgende udseende:



Figur 2.2.4-5. Søgning ved hjælp af logiske udtryk

Brugeren indtaster kombination af søgeord og boolske operatorer, f.eks. "(window AND widget) OR NOT button". Derefter klikker han på "search" knappen. Systemet viser nu hvormange blokke, der opfylder søgekriteriet, og i delvinduet "Blocks Matching Query" vises en oversigt over alle disse blokke. I eksemplet er paranteserne overflødige, da den sædvanlige præcedens mellem operatorer gælder.

Der kan benyttes jokertegn i søgningerne, "*" for et vilkårligt antal tegn og "?" for eet tegn. Hvis brugeren for eksempel vil finde alle ord der indeholder "window" taster han "*window*". Der kan kun søges på ord som begynder med bogstav eller "_" (underscore) og derefter indeholder bogstaver, "_" eller cifre. Systemet er altså ikke indrettet på søgning efter vilkårlige tekststrenge. Se nærmere beskrivelse af gyldige søgeord i afsnit 2.3 *Design og etablering af tekstdatabasen*.

Forespørgsler ved Venn-diagram

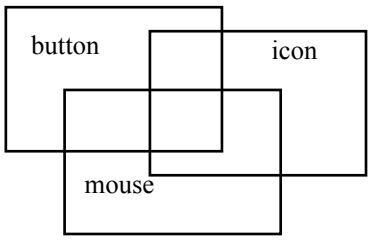
I denne søgeform formuleres forespørgsler ved brug af et Venn-diagram. Diagrammet giver mulighed for at søge "i 2 faser":

- Der indtastes søgeord for et antal grundmængder. På diagrammet viser systemet, for hver kombination af grundmængderne, det antal blokke der matcher det tilsvarende udtryk. Udfra antallene alene kan brugeren vurdere kvaliteten af sin søgning, og justere grundmængderne.

- Når brugeren vurderer at antal hits i den ønskede delmængde er passende, udføres den egentlige søgning, og herved får brugeren adgang til de fundne tekster.

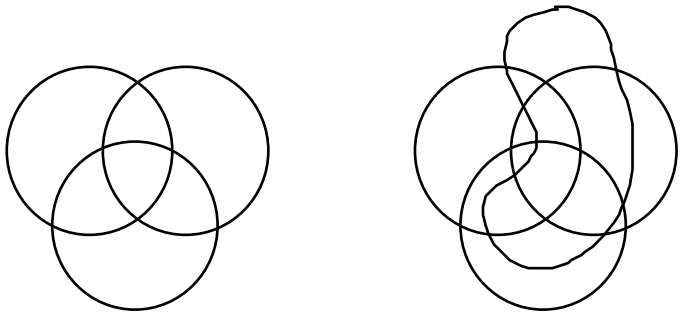
På denne måde støttes brugeren både i at formulere forespørgslen, og i sin evaluering af resultatet af en søgning. Istedet for at indtaste den samlede forespørgsel skal brugeren indtaste søgekriterier for de enkelte grundmængder, og derefter angive den ønskede kombination af disse. Forespørgslen nedbrydes herved til enkeltdele, der er mere overskuelige end den samlede forespørgsel, samtidig med at det ikke er nødvendigt at have fuldstændigt kendskab til brug af de boolske operatorer AND, OR og NOT. Når brugeren skal evaluere kvaliteten af sin søgning, støttes han af væsentlig flere informationer end hvis han blot fik antal hit for den samlede forespørgsel. På diagrammet kan han direkte se, hvilke grundmængder der giver for mange eller for få hits.

Mængdediagrammer kan tegnes på flere forskellige måder, f. eks. ved brug af rektangler.



Figur 2.2.4-6 Eksempel på mængdediagram lavet med rektangler

En anden måde at illustrere mængdediagrammer er med cirkler, som det ofte er brugt indenfor den matematiske verden. Hvis man skal have mere end 3 mængder, kan man ikke bruge cirkler, idet mindst een af mængderne må være konkav.

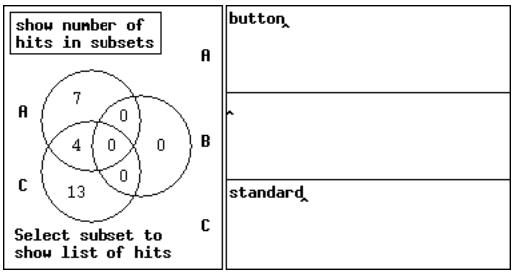


Figur 2.2.4-7 Mængdediagrammer med 3 og 4 mængder. Ved diagrammet med 4 mængder kan ikke alle mængderne tegnes som cirkler.

Et diagram med mere end 3 mængder er imidlertid ikke overskueligt. En anden mulighed for at arbejde med mere end tre mængder er en iterativ søgeproces, hvor resultatet af een søgning kan bruges som den ene af grundmængderne i en ny søgning.

Vores valg

Vi har valgt at lave et mængdediagram med 3 cirkler. Brugeren kan få vist antallet af hits ved hjælp af en knap eller få udført en søgning ved at klikke i en af delmængderne på diagrammet. Der er et indtastningsfelt til hver grundmængde.



Figur 2.2.4-8 Det endelige design af Venn-diagrammet. Her vises antal hits for søgeordene "button" og "standard", hvor 4 af de udvalgte blokke indeholder begge søgetermer. For at få overskrifterne for disse 4 blokke frem i "Blocks Matching Query" klikkes på fællesmængden mellem A og C.

Det er muligt at indtaste flere søgeord til hver grundmængde. Disse søgeord bindes sammen af den boolske operator OR. Man kan således opbygge en grundmængde af tekster, der omhandler eet begreb, skønt dette begreb måske kaldes ved flere forskellige navne, ved i indtastningsfeltet at angive en række synonymer.

Udover dette ville vi gerne give mulighed for søgning efter fraser, dvs. flere ord der står ved siden af hinanden. Dette kan f.eks. være relevant i engelske tekster, hvor begreber ofte skrives i flere ord som f.eks. "data base". Dette ville vi lave således at ord på samme linie tolkedes som fraser. Af tidsmæssige årsager er dette imidlertid ikke færdigimplementeret.

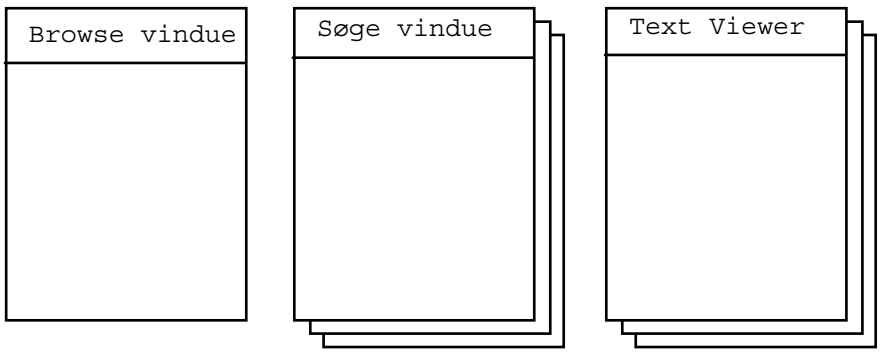
I den nuværende version af Venn-diagrammet fortolkes det derfor forskelligt om ord står på samme linie eller hver sin linie. Mellem hver linie underforståes et OR, og ord på samme linie knyttes sammen af et AND. Søges der efter "data base" fås derfor alle afsnit, der indeholder ordene "data" og "base" - blandt andet de afsnit hvor "data base" forekommer.

2.2.5 Sammenkobling af systemets enkeltdele

Det, der er beskrevet i de foregående afsnit, er de delelementer, der skal indgå i det endelige design af TeSS. I dette afsnit behandler vi sammenkoblingen af delelementerne til det færdige system. Den mest betydende beslutning for sammenkoblingen er at bestemme, hvor mange vinduer TeSS skal bestå af. Det er allerede beskrevet at "Text Viewer" skal være et selvstændigt vindue, hvoraf der kan være flere fremme samtidigt.

Til en søgning ved forespørgsel hører: En udvalgt delmængde af den samlede tekstmængde, en forespørgsel formuleret enten ved logisk udtryk eller ved Venn-diagram, samt et resultat bestående af en mængde tekster der opfylder søgekriteriet. Disse dele udgør en helhed, og

delvinduerne "Blocks Selected for Query", "Formulation of Queries" og "Blocks Matching Query" skal derfor findes i samme vindue. TeSS kunne så opbygges af tre vinduestyper:

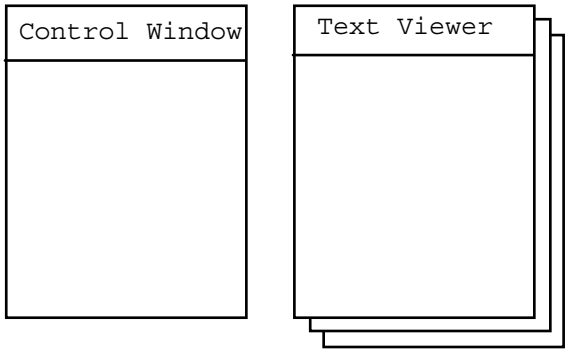


Figur 2.2.5-1 Skitse af et samlet system: Eet vindue til at browse i, flere søgevinduer og flere tekstvinduer.

På denne måde ville det være muligt at gemme interessante søgninger med tilhørende søgemængde og resulterende hits, og eventuelt arbejde videre på dem senere.

Vores valg

Vi har valgt at browsing og søgning ved forespørgsler skal være i samme vindue. Hermed får TeSS kun to vinduestyper, kontrolvinduet og tekstvinduet. Derved får vi mulighed for at støtte kombinationen af de to forskellige søgeteknikker browsing og forespørgsler. Det bliver desuden lettere at håndtere systemet i en arbejdssituation, hvor også andre systemer indgår.

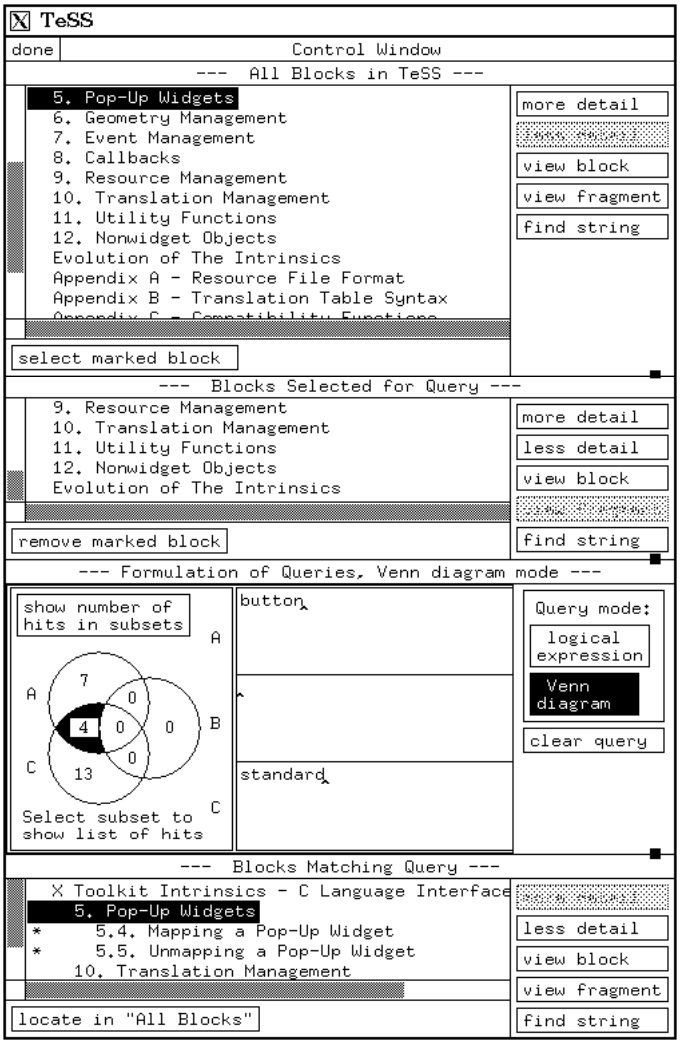


Figur 2.2.5-2 Skitse af det samlede TeSS: Eet vindue til browsing og søgning ved forespørgsler samt flere tekstvinduer.

Vi har endvidere valgt at browsing-faciliteterne skal være til rådighed også i de to træstrukturer der hører til forespørgsler, nemlig udvalgte blokke og hits. En yderligere sammenkobling mellem browsing og forespørgsler er funktionen "Locate in All Blocks". Funktionen bruges til at vise, hvor i den samlede tekstsamling, "All Blocks in TeSS", en blok med hit i hører til.

Da TeSS skal kunne bruges i en arbejdssituation med andre systemer er det vigtigt at hver af

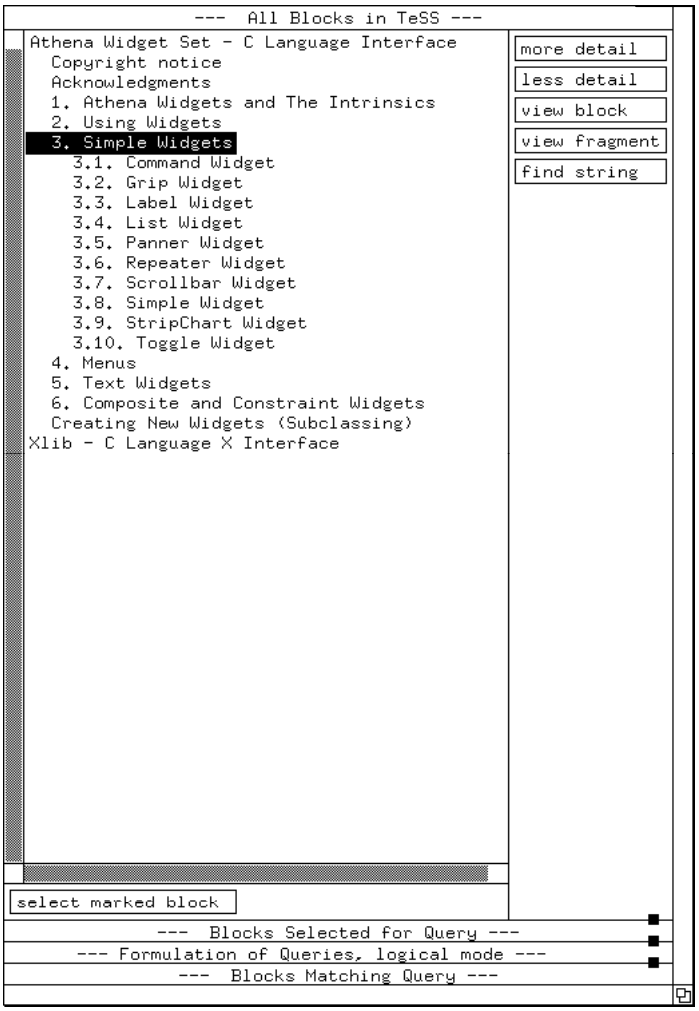
vinduerne "Control Window" og "Text Viewer" ikke fylder mere end halvdelen af skærmen. På den måde er det f.eks. muligt at arbejde med søgning og tekstlæsning samtidig, eller læsning samtidig med at der bruges en teksteditor. Vinduernes størrelse initielt er under en halv skærm i bredden og cirka en hel skærm i højden. Det endelige design af kontrolvinduet ser således ud:



Figur 2.2.5-3 Det endelige design af kontrolvinduet i TeSS.

På figur 2.2.5-3 ses TeSS' kontrolvindue. I feltet "All Blocks in TeSS" er hele manualen for X Toolkit udvalgt og kopieret til feltet "Blocks Selected for Query". Det logiske udtryk for forespørgslen er "button AND standard", da fællesmængden mellem mængde A og C er valgt, som den mængde der ønskes udsøgt. I feltet "Blocks Matching Query" vises det, at der blandt andet er hits i afsnittene 5.4 og 5.5.

Kun en af de to måder, at formulere forespørgsler på, er fremme ad gangen. Det er muligt med de små håndtag (eng.: panes), at ændre på delvinduernes størrelse. For eksempel kan det øverste delvindue udvides til at fylde næsten hele vinduet:

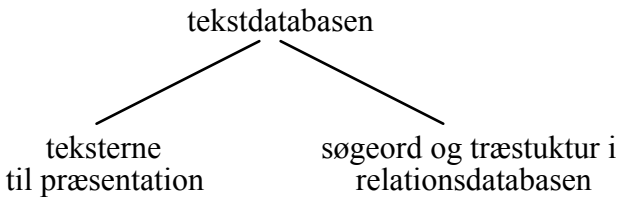


Figur 2.2.5-4 Kontrolvinduet med fokus på feltet "All Blocks in TeSS".

Som det kan ses af designet figur 2.2.5-3 står indtastningsfelterne til mængderne på Venn-diagrammet samlet istedet for et felt ved hver cirkel. Det var ikke den oprindelige ide. Det var meningen at felterne skulle relatere sig til cirklerne uden at det var nødvendigt at navngive cirklerne og indtastningsfelterne. Det viste sig imidlertid ikke muligt at få plads til felterne ved cirklerne, da der var sat en grænse for hvor bredt hele vinduet måtte være, så her spillede den overordnede ide om hvor store vinduerne måtte være ind, og gjorde det endelige design knap så godt.

2.3 Design og etablering af tekstdatabasen

Tekstdatabasen omfatter tre edb-manualer til udvikling af X-applikationer med grafiske brugergrænseflader, ialt omkring 3 megabyte tekst. Derudover omfatter tekstdatabasen søgeord og træstruktur, der muliggør effektiv søgning i teksterne, se figur 2.3-1. Først diskuteres lagringen af teksterne til præsentation, og processen bag etableringen af tekstdatabasen beskrives. Derefter fastlægges de datastrukturer i relationsdatabasen, der skal støtte browsing og søgning ved forespørgsler. Disse overvejelser munder ud i, at vi opstiller datamodellen for tekstdatabasen. Endelig afsluttes afsnittet der gør rede for den valgte løsning og overvejelserne bag den, med en beskrivelse af nogle af vores bestræbelser på at optimere relationsdatabasen.



Figur 2.3-1: Tekstdatabasens to dele: Teksterne formateret til præsentation på skærmen og behandlet for relationsdatabase, der muliggør effektiv søgning.

2.3.1 Behandling af teksterne

Tekstdatabasen omfatter de formaterede tekster, lagret med henblik på præsentation i et tekstvindue. En fil kan potentielt indeholde tekst, såvel som tabeller og figurer billeder, lyd eller endda film. Edb-manualer indeholder ofte tabeller med oversigter over options og lignende; nogle indeholder også figurer med f.eks. eksempler på skærmbilleder. Mulighederne for at håndtere tabeller og figurer afhænger primært af Text Vieweren; den skal kunne præsentere tekst kombineret med grafik. Hvis tabellers og figurers ikke-tekstmæssige indhold ikke blot skal kunne præsenteres, men også skal være søgbart, kræves imidlertid helt nye faciliteter. Flertallet af de kommercielt tilgængelige søgesystemer kan ikke håndtere figurer, og i de, der kan, er søgemulighederne som regel reduceret til, at man kan søge på tabel- og figurteksterne, se Ashford (1986) og Basch (1989). Vi vil afgrænse os til at behandle tekst og dermed kun behandle figurer og tabeller, som består af ord.

Originalteksternes forbehandling

Hvor nemt overførslen af tekst fra original til tekstdatabase kan ske afhænger af originaltekstens repræsentation. Originalteksterne indenfor de områder, vi søgte tekster i (udviklingsværktøjer til grafiske brugergænseflader, brugergrænsefladers standardbiblioteker, operativsystemsmanualer etc.) foreligger på DIKU sammen med programpakkerne; altså på maskinlæsbar form.

Formateringen af teksterne er dog vidt forskellig, selvom vi udelukkende søgte dokumenter rettet mod Unix-baserede systemer. Nogle vælger det meget finkornede sidebeskrivelsessprog Postscript, andre Tex, men hyppigst tekstformatering forberedt til programmerne Nroff og Troff.

Vi opstillede tre krav til teksterne; vi skulle kunne uddrage søgeord og træstruktur fra teksten, vi skulle kunne vise teksterne i et tekstvindue med formateringen bibeholdt, og endelig skulle arbejdsbelastningen ved denne proces være rimelig. Processen kunne opdeles i to faser; først konvertering til et fælles udgangsformat og derefter opsplitning i tekst til tekstvinduet og tekst til relationsdatabasen.

Dokumentation lagret i Postscript kunne vi ikke finde værktøjer til at filtrere den søgbare tekst ud af. Der findes programmer, der gør det muligt at vise Postcript-filer på skærmen, men er alt for langsomt til vores formål. Indscanning og tegngenkendelse var for arbejdskrævende samtidig med at genkendelsesprocenten ikke var 100%. Tilsvarende kunne vi ikke let konvertere dokumentation fra Tex format til tekstvinduer og relationsdatabase. Den eneste synlige realistiske lå derfor indenfor dokumentation formateret til Nroff.

Tekst formateret med Nroff kan vises med formatering af en lang række programmer, hvoraf f.eks. Man, Xman og Less er hyppigt brugt. Der findes tillige kommandoer til at fjerne formateringerne og derved filtrere teksten fra til brug for søgningerne.

De tre edb-manualer er netop skrevet for behandling gennem Nroff. Behandlingen af filerne bestod i at indsætte delingsinformation før overskrifter, formatere teksterne til tekstvindue/relationsdatabase, splitte teksterne op i lagringsenheder og til slut fjerne delingsinformationen igen. Dette arbejde blev for en stor del udført automatisk ved vekslende brug af bl.a. Awk, Nroff, Troff og Csplit. Ikke alle delingsinformationer kunne indsættes automatisk og originalteksterne måtte redigeres manuelt på enkelte punkter for f.eks. at undgå sidenumre. Tilsvarende måtte teksterne til databasen redigeres for at fjerne overflødige tegn, der var sluppet gennem tekst-filteret. Forfattere anvender Nroff forskelligt, og det giver derfor ikke mening at redegøre for den præcise redigering der er foretaget.

Lagringen af teksterne efter forbehandling

Lagringen af teksterne forudsætter fastlæggelse af en enhed for lagringen, dvs. fastlæggelse af hvordan teksterne skal opdeles før de lagres. Valget af lagringsenhed skal ske under hensyntagen til både de ønskede søgemuligheder, og at udfinding af tekst til præsentation i en Text Viewer samt bladring i denne teksts overskrifter skal kunne implementeres effektivt.

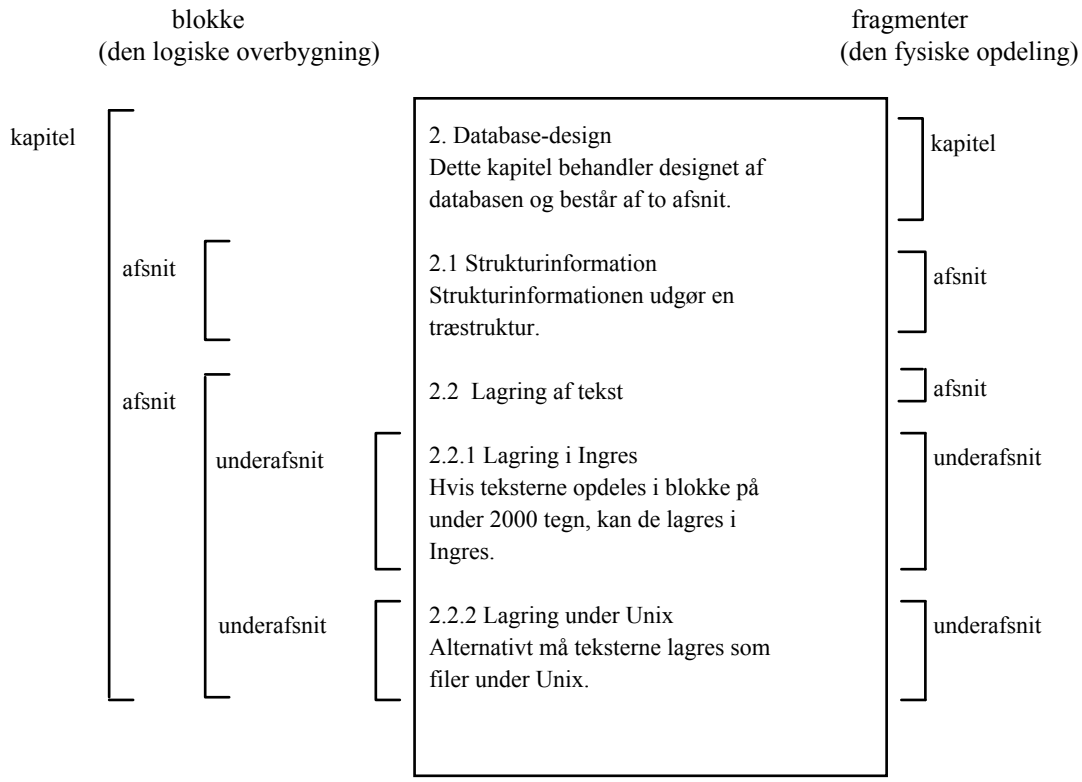
Des mindre lagringsenhed, des flere enheder skal der læses for at fylde en Text Viewer og bladre i den. Det taler for at vælge lagringsenheden, så der - i det mindste i nogle tilfælde - kun skal læses een enhed: Lagringsenheden bør minimum være den mindste enhed, man kan få vist i et tekstvindue. I tekstsøgesystemer som BRS og Folio Views lagres hele tekstsamlingen i een fil, og al adressering af tekststykker foregår ved pointere indenfor denne fil, se Andersen et al. (1992).

Herved gøres størrelsen af de blokke, der læses, til det afgørende, fremfor størrelsen af lagringsenheden. Vi har fundet det mere hensigtsmæssigt at vælge fragmentet som lagringsenhed. Med dette valg kan tekst til et tekstvindue genereres udelukkende ved manipulation af hele lagringsenheder; det involverer ikke pointere ind i de lagrede tekststykker. Ved at lagre hvert fragment for sig er der endvidere gode muligheder for at slette, rette eller tilføje tekst til tekstsamlingen.

Lagringen af teksterne kan ske enten som filer under Unix eller som strenge i en Ingres-relationsdatabase. Unix-filer kan være ligeså store som diskkapaciteten tillader; Ingres tillader derimod kun strenge på indtil 2000 tegn, se Ingres (1991a). Det kan bemærkes, at flere andre relationsdatabasesystemer ikke har denne begrænsning: Oracle tillader strenge på indtil 64 Kb og Sybase på indtil 2 Gb. Fragmenterne har meget forskellig størrelse; men de holder sig langtfra alle indenfor 2000 tegn (ca en A4-side). Lagringen må derfor ske i filer under Unix.

2.3.2 Strukturinformation

Ved siden af filerne med teksterne etableres de datastrukturer, der skal muliggøre effektiv søgning. En af disse er strukturinformationen. Strukturinformationen skal dels muliggøre browsing, dels rumme oplysninger, der kan placere hittene fra en forespørgsel i den totale tekstsamling. Det opnås ved, at strukturinformationen ovenpå teksterne etablerer en træstruktur, som definerer tekstsamlingens opdeling i blokke:



Figur 2.3.2. Opdelingen af et stykke tekst i blokke og fragmenter. Blokke defineres i træstrukturen og omfatter teksten fra og med deres overskrift til, men ikke med, den næste overskrift på samme niveau. Fragmenter afspejler den fysiske opdeling og omfatter teksten fra og med deres overskrift til, men ikke med, den næste.

En blok består af en overskrift og den tekst, der logisk hører under overskriften, dvs. en blok er teksten fra og med dens overskrift til, men ikke med, den næste overskrift på samme niveau. Træstrukturen giver en opdelingen af teksterne i ligeså mange blokke, som der er overskrifter. Kapitler består imidlertid ofte af en indledning uden egen overskrift - i det følgende betegnet manchetten - efterfulgt af en række afsnit med hver deres overskrift. I nogle tilfælde er manchetten faktisk et selvstændigt afsnit - afsnit 0 - som det ville være relevant at kunne referere til. I TeSS er manchetten imidlertid ikke en blok, da den ikke har nogen overskrift. For at gøre manchetten tilgængelig er det i TeSS ikke blot muligt at få en blok, men også et fragment, op i en Text Viewer. En manchet giver anledning til et fragment, der består af manchetten og den umiddelbart foranstående overskrift. Til en kapiteloverskrift er således knyttet en blok, der indeholder hele kapitlets tekst, og et fragment der indeholder en eventuel manchet. Fragmenter bruges ikke kun i forbindelse med kapitlers manchetter; overskrifter på andre niveauer kan også have manchetter. Generelt er et fragment teksten fra og med dets overskrift til, men ikke med, den næste overskrift. Som alternativ til fragmenter kunne man have introduceret selvstændige overskrifter til manchetterne. Det har vi undladt, primært fordi det vil udvide træstrukturens omfang væsentligt uden at tilføje ret megen ny information.

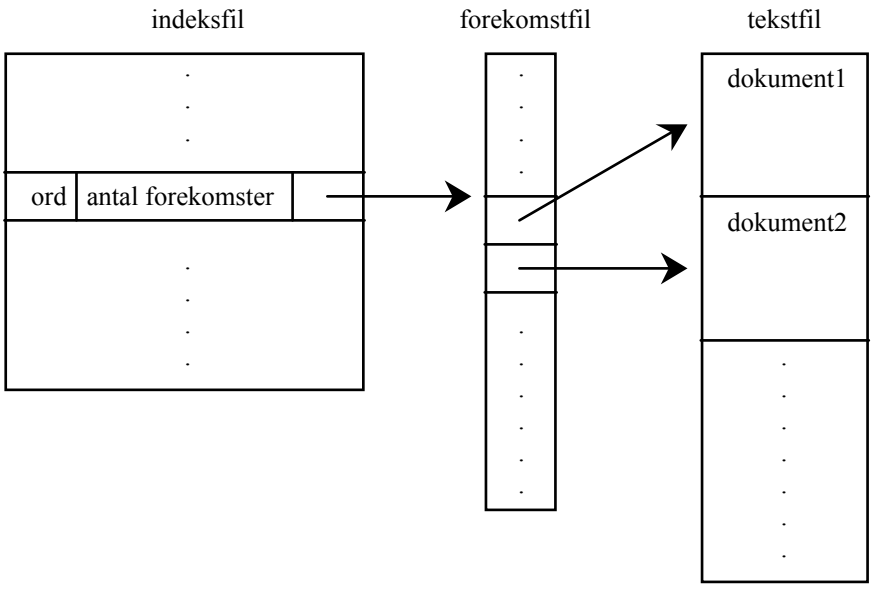
I TeSS er grundenheden for søgning fragmenter. Forespørgslen skal være opfyldt indenfor

et fragment, og resultatet af en søgning er således en række fragmenter. En nærliggende udvidelse ville være at give brugeren mulighed for at angive, at resultatet af en søgning skulle være blokke på et givet niveau. I så fald kunne brugere med behov for en bredere behandling af et emne angive, at de søgte efter hit indenfor hele kapitler; mens brugere med behov for en specifik oplysning kunne søge indenfor f.eks. afsnit. Som led i en sådan udvidelse af TeSS kunne der knyttes navne, f.eks. "kapitel" og "afsnit", til de niveauer af blokke, hver enkelt manual er opdelt i. Alternativt kunne der vælges en niveauangivelse; øverst , næstøverst, næstnederst, nederst. Med sådanne navngivne niveauer kunne der også blive mulighed for at angive, at resultatet af søgning i en eller flere referencemanualer skulle være blokke på niveauet "kommandobeskrivelser".

2.3.3 Søgeord

Den enkleste måde at lagre tekst på med henblik på direkte søgning er at lagre teksten i en sekventiel fil og foretage søgningerne ved at skanne denne fil. Fordelene ved denne lagringsteknik er, at der ikke bruges anden lagerplads end den, der går til lagringen af teksten, og at den for simple søgeformer er meget let at implementere. Den afgørende ulempe er de lange søgetider. Fuldtekstskanning var tilstrækkeligt, dengang et stort antal forespørgsler kunne samles og behandles i batch-kørsler, men er utilstrækkeligt nu, hvor forespørgslerne skal behandles enkeltvis og interaktivt, se van Rijsbergen (1979).

Den mest udbredte afløser for sekventielle filer og fuldtekstskanning er inverterede filer. Invertering svarer til oprettelse af et indeks ved siden af den eller de filer, dokumenterne er lagret i. En inverteret fil består af en indgang for hvert ord i teksten og hægter til alle ordets forekomster i teksten. Typisk opdeles en inverteret fil i to: en indeksfil og en forekomstfil, se figur 2.3.3-1. Prisen for de meget mere effektive søgninger er, at det kun er muligt at søge på tegnfølger, der er optaget i indeksfilen. Indeksfilen kan f.eks. omfatte alle ord i teksterne, eller blot en række udvalgte ord; men den indeholder kun ord. Det er således kun muligt at søge efter ord, ikke vilkårlige tegnfølger.



Figur 2.3.3-1: Typisk filstruktur ved anvendelse af inverterede filer, efter Faloutsos (1985). Indeksfilen indeholder de inverterede ord; forekomstfilen angiver ordenes placering i teksten; og teksten ligger i en fil for sig.

I TeSS er direkte søgning baseret på en ordtabel genereret ved invertering af teksten. Før ordtabellen kan dannes, skal det fastlægges, hvad der udgør et ord. Vi definerer ord ved de tegn, de indeholder: Ord er de følger af alfanumeriske tegn, der begynder med et bogstav. Udover 'A'-'Z' og 'a'-'z' betragtes også '_' (underscore) som et bogstav. Eksempler:

søgbare ord	**ikke-søgbare tegnfølger**
init1	10
_init	10_str
init_DB	C++
hex2dec	!=

Ord behøver ikke være adskilt med blanktegn. Et ord begynder, når der forekommer et bogstav, og slutter, når det næste tegn ikke er alfanumerisk. Tegnfølgerne '10_str' og 'exit(0)' vil således give anledning til søgeordene '_str' og 'exit'.

Både på dansk og på engelsk kan ord sammenskrives ved hjælp af en bindestreg. Brugen af bindestreger varierer ofte fra person til person, således at et ord som 'database' på engelsk forekommer såvel skrevet i to adskilte ord, som skrevet med bindestreg og skrevet ud i eet. I TeSS betragtes bindestreg som et skilletegn. Ord skrevet med bindestreg optages således i ordtabellen som to adskilte ord. Alternativt kunne sådanne sammenskrevne ord optages både som eet ord skrevet med bindestreg og som to adskilte ord. I TeSS skal brugeren søge på de adskilte ord, også når han eller hun er i tvivl om, hvorvidt ordet eventuelt skrives med bindestreg.

Det er som regel en fordel, hvis søgningerne er ufølsomme overfor variationer i brugen af store og små bogstaver. I enkelte tilfælde afhænger et ords betydning imidlertid af brugen af store og små bogstaver, f.eks. 'ROM', 'Rom' og 'rom'. Vi mener, der er ret få af disse tilfælde, og tillader os derfor at se bort fra dem. Alle bogstaver konverteres til små bogstaver, før de lægges ind i ordtabellen.

Ordtabellen behøver ikke indeholde alle de ord, der forekommer i teksterne, men kun dem, en bruger kan finde på at anvende som søgeord. Da en lille gruppe af denne type ord forekommer meget hyppigt, kan en stopliste - en liste med ord der ikke tillades som søgeord - reducere antallet af tupler i ordtabellen væsentligt. Stoplister bruges udelukkende for at reducere antallet af tupler i ordtabellen, herved spares lagerplads, og det tager mindre tid at afgøre, om et ord forekommer i en tekst. Normalt betyder en stopliste, at antallet af tupler i ordtabellen reduceres med 40-50%, se Salton & McGill (1983). I TeSS er reduktionen på 48%.

Udgangspunktet for TeSS' stopliste er en stopliste for generel engelsk tekst udviklet af Fox (1989). Vi gennemgik denne stopliste, der omfatter 421 ord, for at frasortere ord, vi ønskede skulle være søgbare. Det resulterede i, at 79 ord blev slettet fra listen. Derefter blev en tredjedel af teksterne inverteret i en prøveversion, og alle ord i denne tekstmængde blev udskrevet sammen med den frekvens, de forekom med. Den derved genererede liste blev gennemgået for ord, som forekom med høj frekvens og som vi ikke forventede brugerne ville søge efter. Det resulterede i 6 nye stopord. TeSS' stopliste omfatter således 348 ord.

2.3.4 Datamodel

Teksterne til Text Vieweren er lagret som filer under Unix; datastrukturerne til søgning udgøres af en relationsdatabase. Hele datamodellen for denne relationsdatabase, i form af de SQL-sætninger der definerer den, er gengivet i bilag B; i det følgende gøres rede for overvejelserne bag den valgte datamodel.

Først etableres korrespondancen mellem relationsdatabasen og filerne med teksterne. Da et fragment svarer til een fil, kan det fragment, der en knyttet til en given overskrift, identificeres blot ved angivelse af et filnavn. En foreløbig modellering af denne korrespondance kan skitseres således (af historiske årsager kaldes tabellen bloktabellen, selvom fragmenttabellen ville være mere korrekt):

Bloktabel:

blokid
bloknavn
filnavn

Med denne tabel er der skabt sammenhæng mellem relationsdatabasen, hvor fragmenter identificeres ved såkaldte blokid'er, og filerne. Dernæst skal træstrukturen modelleres, så fragmenterne kan placeres i manualerne, og blokkene defineres. Angivelsen af et fragments placering i træstrukturen består i en angivelse af, hvilken manual, blokken forekommer i, og en lignende angivelse for hvert niveau i opdelingen af manualen. Vi finder det rimeligt at sætte et maksimum for, hvor mange niveauer en manual kan underdeles i. Sættes dette maksimum til syv, kan en foreløbig datamodel for træstrukturen skitseres således:

Strukturtabel:

blokid
manual
niv1
niv2
niv3
niv4
niv5
niv6
niv7

En blok er et deltræ i træstrukturen. Det vil sige en blok består af de fragmenter, der hører til samme manual og indtil et vist niveau også til den samme af træstrukturens underdelinger. Hvis manual 1 er opdelt i kapitler, afsnit og underafsnit, består den blok, der udgør 3. afsnit af manualens 2. kapitel, altså af de fragmenter, der i strukturtabellen har manual = 1, niv1 = 2 og niv2 = 3, uanset indholdet af niv3-niv7.

Ordtabellen skal for hvert ord, der forekommer i teksterne og ikke er et stopord, indeholde oplysning om, hvilket fragment ordet forekommer i. For et ord i 3. afsnit af kapitel 2 i manual 1 angives altså, at det forekommer i blok(1, 2, 3); der suppleres ikke med de dubblerende oplysninger om, at ordet dermed også forekommer i blok(1) og blok(1, 2). Fra starten var det intentionen, at TeSS skulle give mulighed for søgning på vendinger. For at muliggøre det suppleres oplysningen om, hvilken blok et ord forekommer i, med oplysning om ordets placering indenfor blokken. Ordtabellen kan skitseres således:

Ordtabel:

blokid
ordnr
ord

Som datamodellen er skitseret hidtil, må der for at folde en overskrift i træstrukturen ud foretages en *join* af strukturtabellen, der fortæller hvilke underdelinger en given overskrift har, og bloktabellen, der indeholder navnene på disse underdelinger. Hvis TeSS senere skal udvides med mulighed for, at resultatet af en søgning ikke blot kan være fragmenter, men f.eks. også kapitler, vil AND-operatoren have flere mulige betydninger. Hvor kravet nu er, at de involverede ord forekommer 'indenfor samme fragment', vil kravet i en søgning, der skal resultere i kapitler, være 'indenfor samme kapitel'. For at afgøre om to ord forekommer i samme kapitel kræves en *join* af ordtabellen og strukturtabellen.

Det ville betyde en anseelig forbedring af svartiderne, hvis disse *joins* kunne undgås. En umiddelbar mulighed for at opnå det er at kode oplysningerne i strukturtabellen ind i blokid'et og derved eliminere strukturtabellen. Blokid'et defineres derfor som en 16-tegns streng med to tegn til angivelse af manualen og to tegn til angivelse af hver af niveauerne:

blokid: mm nn nn nn nn nn nn nn

hvor	mm	=	00-99,	manualnummer
	nn	=	00,	ikke-eksisterende overskriftsniveau
	nn	=	01-99,	bloknummer på det pågældende niveau

I manual 1 har 3. afsnit i kapitel 2 således blokid = '0102030000000000'. Når træstrukturen foldes ud og ind, vil det endvidere være bekvemt at have direkte adgang til oplysning om, hvilket niveau en blok hører til på. Det vil ligeledes være bekvemt at have direkte adgang til oplysning om, hvorvidt en blok er underdelt og dermed kan foldes ud. Begge disse oplysninger kan udledes af blokid'et; men for at forenkle programmeringen og reducere svartiderne vælger vi at repræsentere dem eksplicit. Vi vælger endvidere at bruge blokid'et som filnavn for filerne med teksterne. Den modificerede datamodel ser således ud:

Bloktabel:

blokid
blokniveau
blokdelt
bloknavn

Ordtabel:

blokid
ordnr
ord

For at gøre stoplisten let at ændre lægger vi også den ind som tabel. Der føjes altså en tredje tabel til datamodellen:

Stopliste:

ord

2.3.5 Optimering

En væsentlig optimering af relationsdatabasen bestod i at definere tabellernes primære indeks. Uden et sådant indeks foretages alle søgninger i en tabel ved at gennemløbe den fra ende til anden. Fastlæggelsen af indeksene tog udgangspunkt i forespørgslerne til de enkelte tabeller. Tabellens attributter blev delt op i dem, hvis værdi var angivet i forespørgslerne, og dem, hvis værdi blev hentet frem af forespørgslerne. For de fleste tabeller er det de samme attributter, der angives i alle forespørgsler; i disse tilfælde blev der oprettet indeks på disse attributter. For ordtabellen blev der imidlertid genereret både et primært og et sekundært indeks, da forespørgslerne faldt i to grupper, der angav to forskellige attributter, ord og blokid.

Med indeks på tabellerne afhænger den måde søgninger foretages på af den lagringsstruktur, der er valgt for hver enkelt tabel. Ingres tilbyder fire lagringsstrukturer - hob, hash, isam (index sequential access method) og b-træ - med meget forskellige forcer, se Ingres (1991b, kap.11). Tabeller oprettes pr default som hobe; men nøje overvejelse af, hvilken lagringsstruktur der er mest hensigtsmæssig, viste sig, specielt for ordtabellen, at have stor betydning. For ordtabellen kunne to lagringsstrukturer komme på tale: isam og b-træ. Begge kan håndtere store tabeller og forespørgsler, hvor de attributter der er indeks på, kun er delvist specificeret. Med isam udføres generelt færre disk-operationer, og isam er bedst egnet til statiske tabeller. Med b-træer er en række hyppigt forekommende sorteringer derimod overflødige, da tuplerne altid behandles i rækkefølge ud fra de attributter, indekset er baseret på. Efter nogle eksperimenter valgte vi isam til ordtabellens primære indeks på ord og b-træ til det sekundære indeks på blokid.

Da det blev klart, at TeSS ikke ville komme til at tilbyde søgning på fraser, lå en væsentlig optimering lige for. Når der ikke skal søges på fraser, er ordnr'et i ordtabellen overflødigt. Fordelen ved at fjerne ordnr er primært, at antallet af tupler i tabellen reduceres kraftigt, i vores tilfælde fra 152553 tupler til 61874. Det skyldes, at mange ord forekommer flere gange i samme fragment. Sålænge ordenes rækkefølge spiller en rolle, giver hver forekomst af ordet anledning til en tupel. Hvis det derimod kun betyder noget, om ordet er eller ikke er i fragmentet, så giver ordet kun anledning til een tupel, uanset for mange gange det forekommer i et fragment. Reduktionen af ordtabellen giver et godt billede af, hvor ressourcekrævende det er at tilbyde søgning på fraser.

Endelig kunne det forhold, at relationsdatabasen er statisk, udnyttes til at optimere såvel pladsforbruget som svartiderne. For hver tabel kan det angives, hvor stor en del af hver diskblok, der udfyldes fra starten, og hvor meget der reserveres til senere modifikationer af tabellens indhold. Når en tabel oprettes sættes pr default en masse plads af til senere modifikationer. Det betyder, at tabellen spredes over flere diskblokke, og dermed at der skal læses flere diskblokke for at

gennemføre en søgning, end hvis diskblokkene blev fyldt helt ud. Da TeSS' database er statisk, angav vi, at diskblokkene fra starten skulle fyldes helt ud.

2.4 Objektmodel af tekstsøgesystemet

Med udgangspunkt i beskrivelse af brugergrænsefladen og repræsentation af manualteksterne opbygges en objektmodel af tekstsøgesystemet. Modellen består af en række klasser, deres data og funktioner, og deres indbyrdes relationer. Endvidere redegøres for, hvordan vi har brugt Xt/Athena-klasserne til at opbygge brugergrænsefladen.

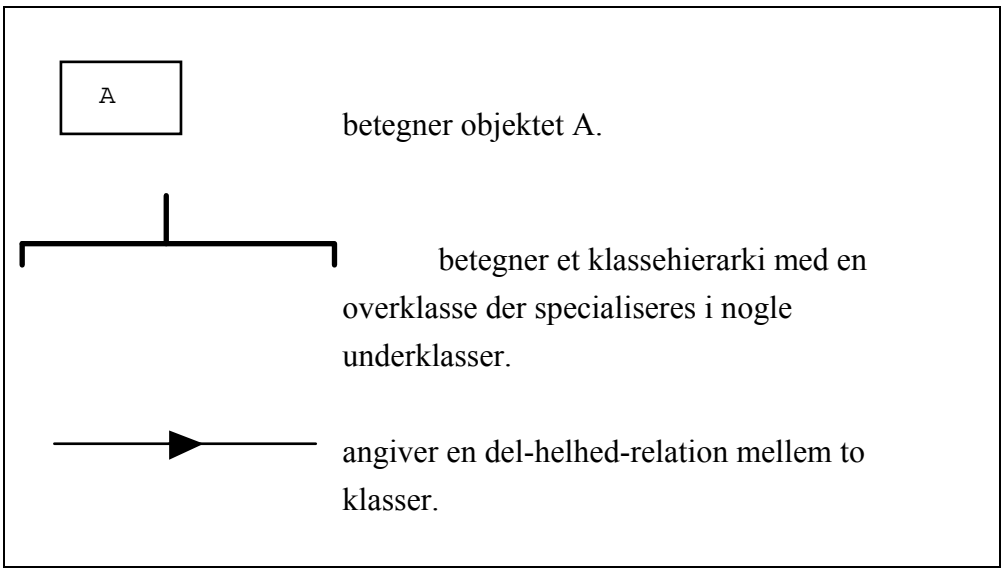
TeSS' brugergrænseflade er opbygget af en række objekter. TeSS udvikles i et objektorienteret sprog, Beta, ved brug af et objektorienteret brugergrænsefladeværktøj, Xt/Athena. Herved fås mulighed for at have de samme objekter i program og i brugergrænseflade. Vi har derfor valgt at starte implementationsarbejdet med at opbygge en objektorienteret model af systemet. Modelleringen skal omfatte identifikation og beskrivelse af et antal klasser, deres sammenhæng med hinanden, og deres data og funktioner. Ved modelleringen tager vi udgangspunkt i de designbeslutninger, der er fremlagt i afsnit 2.2 *Brugergrænsefladen i TeSS* og i 2.3 *Design og etablering af tekstdatabasen*.

Vi har fundet hovedparten af modellens objekter og sammenhænge udfra brugergrænsefladens layout og funktionalitet. Alle vinduer og felter på skærmen var mulige klasser i modellen, og ting der på skærmen har ensartet udseende eller funktionalitet var mulige klassehierarkier. Udfra design af tekstdatabasen har vi modelleret programmets repræsentation af manualteksterne. Herunder har vi især bygget på forudsætningen om en hierarkisk strukturering af teksterne, lagring i databasen af alle blokke, deres titel og niveau, samt invertering af teksterne og lagring i databasen af alle søgbare ord. I forbindelse med tekstdatabasen findes endvidere beslutningen om lagring af selve manualteksterne i filer. Dette var dog ikke prægende for modellen, idet den var åben overfor hvordan teksten blev vist på skærmen. Objektet `TextViewer` skulle blot sørge for, givet en blok, at vise den tilhørende tekst på skærmen.

Vi vil nu fremlægge den model, TeSS er opbygget udfra. Først vil vi introducere notationen, der er brugt i vores figurer. Derefter giver vi en oversigt over den samlede model, og de fire hoveddele som modellen kan inddeles i gennemgåes grundigere i de efterfølgende afsnit.

2.4.1 Notation

I næste afsnit viser vi en oversigt over den samlede objektmodel. På denne oversigt bruges notationen vist på figur 2.4.1-1.



Figur 2.4.1-1 Notation der bruges i oversigten over den samlede model

Del-helhed-relationerne læses på følgende måde:

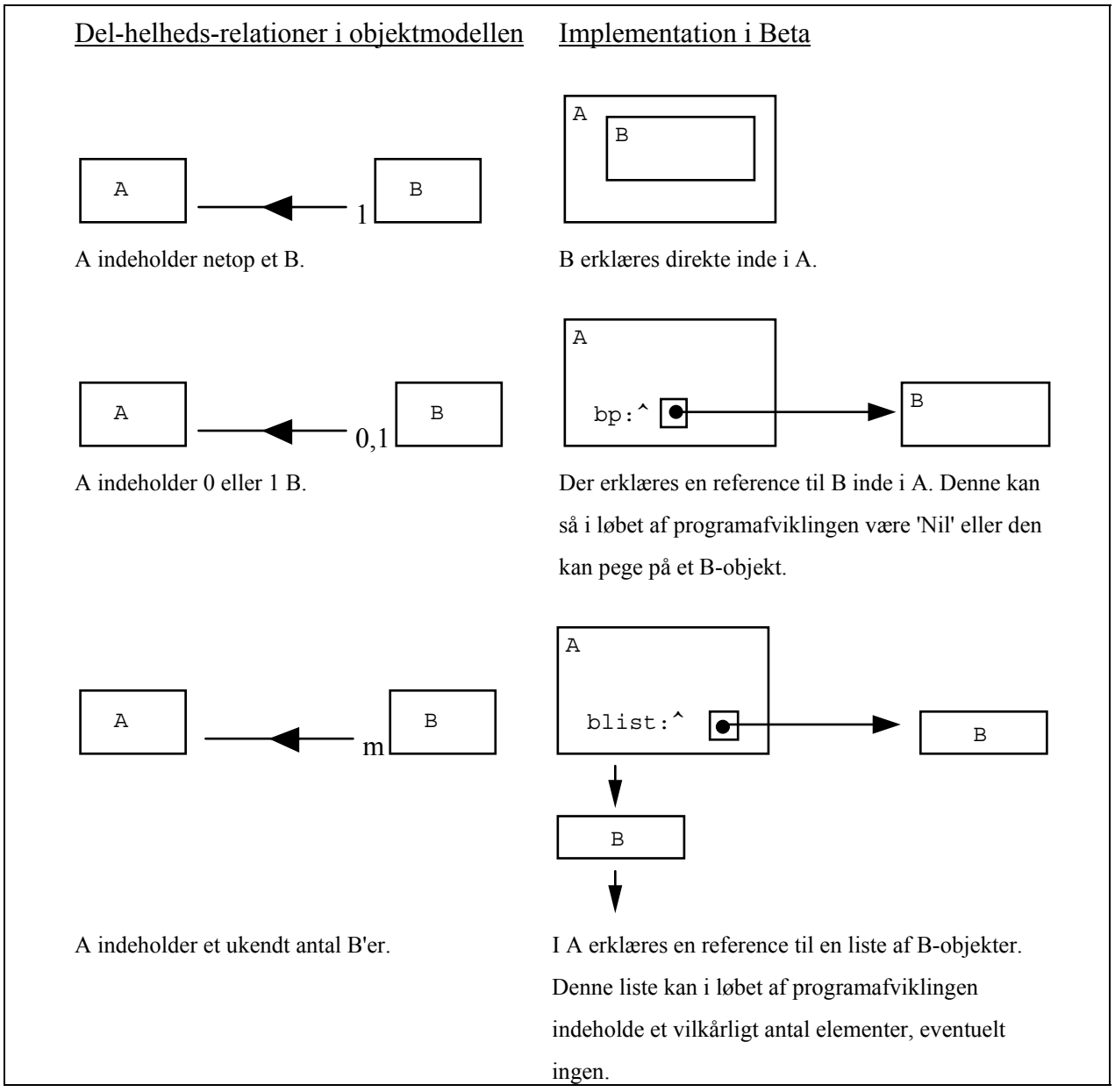
"et A-objekt indeholder et B-objekt".
Når der ikke er antal på pilen, betyder det at et A indeholder netop et B, og et B indgår i netop et A.

"et A-objekt kan indeholde m B-objekter".

For eksempel indeholder et `GraphicsQueryField`-objekt 3 `InputTextField`-objekter, et `ShowButton`-objekt og et `SetDiagram`-objekt.

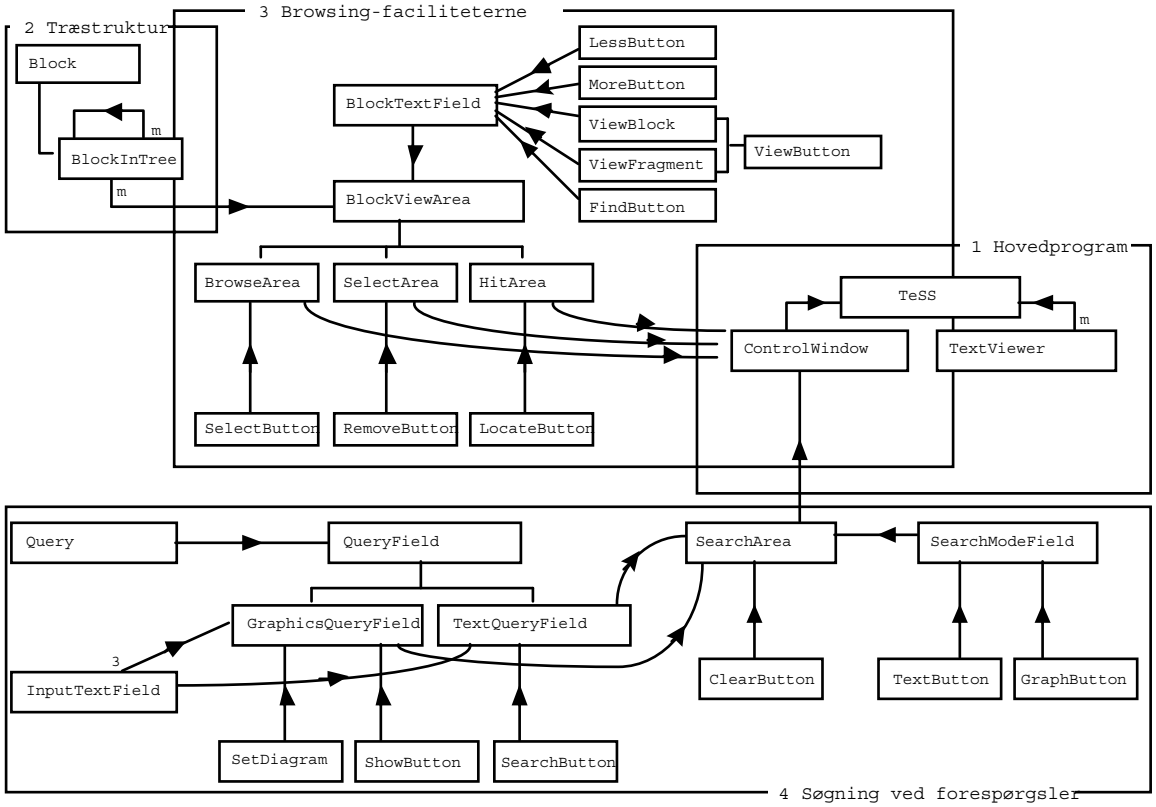
Afsnit 2.4.3 til 2.4.6 beskriver klasserne i objektmodellen i detaljer. De indledes alle med en figur der viser strukturen mellem klasserne, og for hver enkelt klasse vises hvilke væsentlige attributter (variable) den indeholder. Disse attributter opstår først og fremmest ved at vi implementerer alle del-helhed-relationerne, dvs. at vi beslutter hvordan hver enkelt relation skal repræsenteres i programmet. Der findes to forskellige måder at erklære variable i Beta: Som en forekomst af en

klasse og som en reference til en forekomst (dette svarer til f.eks. records og pointers i Pascal). Vi har valgt at implementere del-helhed-relationerne i objektmodellen som vist på figur 2.4.1-2.



Figur 2.4.1-2: Implementation af del-helhed-relationer.

2.4.2 Samlet model af systemet



Figur 2.4.2-1 Den samlede model for TeSS

Modellen kan inddeles i fire hoveddele:

- Første del repræsenterer TeSS-hovedprogrammet. TeSS vil altid omfatte eet kontrolvindue. Derudover kan det omfatte et vilkårligt antal tekstvinduer.

- Anden del afspejler den træstruktur, som teksterne i TeSS er organiseret i. Hver blok i en manual findes som et `Block`-objekt i modellen. En træstruktur opbygges af `BlockInTree`-objekter, der hver refererer til et `Block`-objekt. Disse klasser har vi udledt af beslutningen om den hierarkiske tekststruktur samt databasens bloktabel.

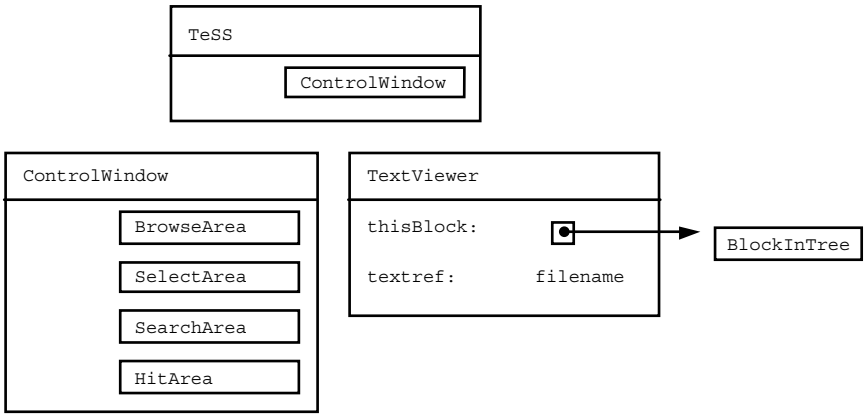
- Tredie del repræsenterer de funktioner i TeSS, som tager udgangspunkt i at manualerne er organiseret i en træstruktur. Her findes browse-faciliteterne, udvælgelse af blokke til forespørgsler samt præsentation af blokke fundet ved en forespørgsel. Klasserne svarer til brugergrænsefladens tre listefelter og de tilhørende knapper.

- Fjerde del repræsenterer søgning ved forespørgsel. Her findes indlæsning, behandling og udførsel af forespørgsler. Klasserne er udledt dels fra brugergrænsefladens forespørgselsdel, og dels fra vores beslutninger om hvordan søgning efter ord skal fungere i TeSS.

Vi vil nu beskrive de fire dele af modellen mere indgående. Hvert afsnit indledes med en tegning af de klasser delmodellen består af, deres attributter og deres indbyrdes struktur. For hver klasse beskriver vi derefter dens rolle i helheden og dens væsentligste procedurer.

2.4.3 TeSS hovedprogrammet

TeSS hovedprogrammet består først og fremmest af et kontrolvindue. Derudover kan der oprettes et vilkårligt antal tekstvinduer.



Figur 2.4.3-1 Klasser der udgør TeSS hovedprogrammet

`TeSS`

Hovedprogrammets eneste opgave er at starte et `ControlWindow`. Herefter overlades kontrollen til X-systemets hovedløkke, der håndterer hændelser i skærmbilledet.

`ControlWindow`

Denne klasse repræsenterer TeSS' kontrolvindue. Der findes kun een forekomst af klassen, svarende til hovedvinduet i applikationen. `ControlWindow` indeholder fire forskellige arealer med hver deres funktionalitet. Objektet sørger for formidling af procedurekald mellem de fire arealer, ved hændelser i eet areal der har indflydelse på et eller flere af de andre. Endvidere formidles kontakten til Text Viewer-delen af programmet. Ved klik på "View fragment" eller "View block" knapperne oprettes således en ny forekomst af `TextViewer`-klassen.

`TextViewer`

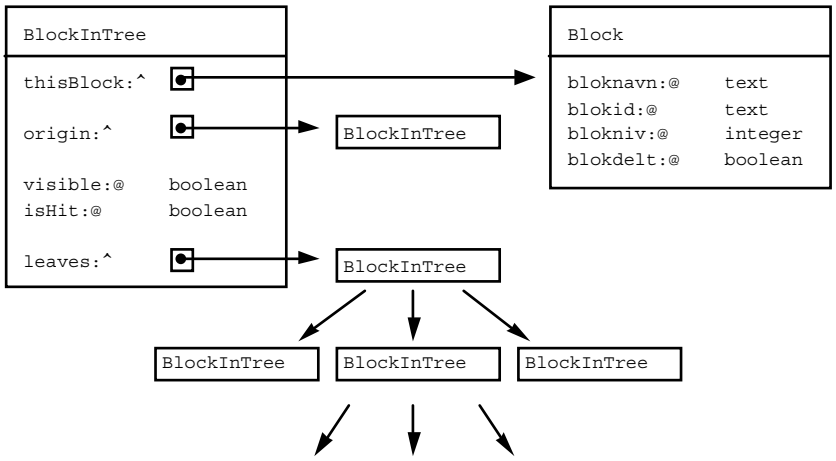
Denne klasse repræsenterer TeSS' "Text Viewer"-vindue. Heri vises teksten hørende til en blok.Ved oprettelse modtager objektet et `BlockInTree`-objekt, samt oplysning om hvorvidt der skal vises et fragment eller en hel blok. Udfra det deltræ som `BlockInTree`-objektet udpeger findes de filer der skal vises. Hvis der skal vises mere end een fil dannes en temporær fil, der indeholder al den tekst der skal vises.

Selve oprettelsen af tekstvinduet, og visning af teksten heri, er lavet som et selvstændigt program. Årsagen til dette var primært at Beta ikke kunne fungere sammen med den del af Xman, vi brugte til at vise teksterne, se nedenfor. Text Viewer-programmet tager udgangspunkt i den (evt. temporære) fil der er identificeret. TeSS-modellen er altså uafhængig af hvordan teksten vises.

Udfra beslutningen om at teksternes formatteringer var væsentlige at vise har vi valgt at udnytte en del af et eksisterende system, Xman. Dette system kan vise tekster der er formateret og behandlet med Unix's Nroff-program. "Text Viewer"-vinduet har derfor samme udseende som Xman-systemets tekstvindue. Kildeteksten til Xman-systemet er offentligt tilgængelig, og vi har forholdsvis nemt kunnet identificere den del af systemet der viser teksterne.

2.4.4 Træstrukturen

Denne del af modellen afspejler manualteksternes hierarkiske struktur.



Figur 2.4.4-1 Klasser der udgør træstrukturen

Vi har tidligere redegjort for hvordan vi opfatter en manual som struktureret i et træ. Et sådant træ repræsenteres i modellen af de to klasser `Block` og `BlockInTree`. `BlockInTree` udgør strukturen og `Block` dataindholdet i knuderne. Hvert af de tre listefelter i brugergrænsefladen modsvares af et træ. Hvis en blok vises i alle tre felter, vil den i programmet være repræsenteret ved tre `BlockInTree`-objekter men kun eet `Block`-objekt.

`Block`

Et `Block`-objekt repræsenterer en blok i en manual, dvs. at et `Block`-objekt svarer til en post i Bloktabellen i databasen. De fire attributter er alle hentet direkte fra denne tabel.

Når TeSS starter vil der blive opbygget et træ indeholdende samtlige blokke der findes i databasen. Dette træ svarer til øverste listefelt, "All Blocks in TeSS". Ved opbygning af træet indlæses fra databasens bloktabel oplysninger for samtlige blokke, og alle `Block`-objekter oprettes. Programmet vil derfor ikke senere behøve at læse blok-oplysninger fra databasen.

`BlockInTree`

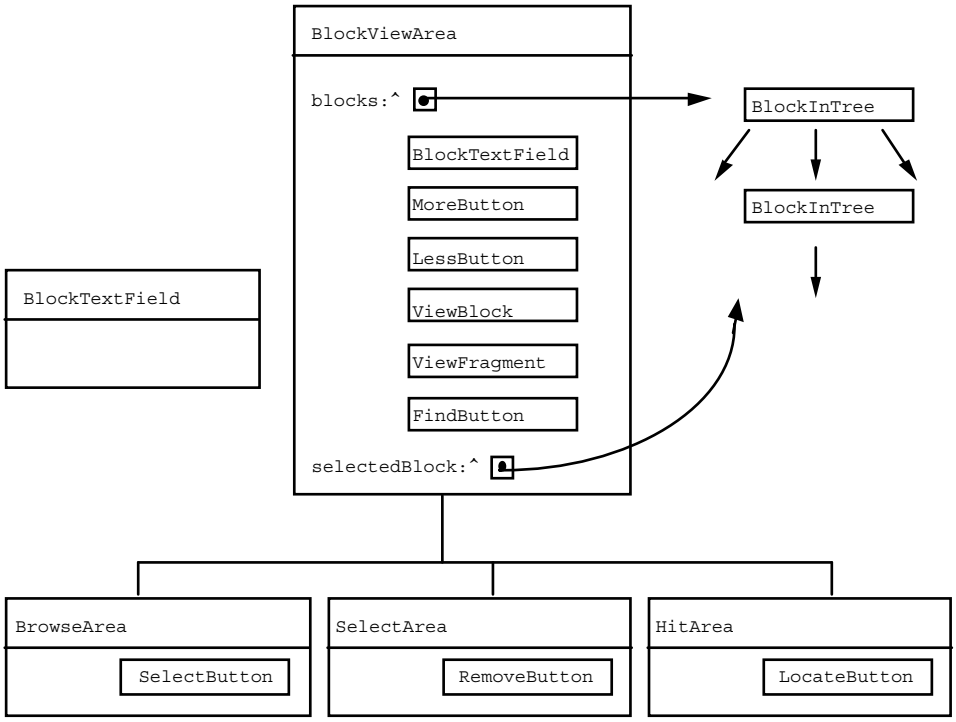
Denne klasse svarer til en knude (evt. roden) i et træ. Klassen indeholder alle de funktioner der opererer på træstrukturen. Eksempler på dette er gennemløb af træer, søgning efter bestemte blokke, indsættelse af et deltræ, kopiering af et deltræ.

`BlockInTree`-objekter kan være synlige eller usynlige. Dette bestemmer om den tilhørende overskrift vises i listefeltet på skærmen. Attributten "`visible`" viser om blokken skal med ud på skærmen. Udfoldning og sammenfoldning, funktionerne *more detail* og *less detail*, foregår i programmet ved at ændre på værdien af "`visible`". Denne konstruktion er nødvendig af hensyn til `SelectArea` og `HitArea` (se næste afsnit), idet vi ikke må fjerne blokke fra træet når der foldes sammen. Hvis vi gør det ved vi nemlig ikke længere hvilke blokke der var udvalgt/fundet.

Attributten "`origin`" peger på objektets forælder i træet. Ved mange af de træoperationer vi har brug for er det nødvendigt at kunne gå både op og ned i træet.

2.4.5 Operationer med udgangspunkt i træstrukturen

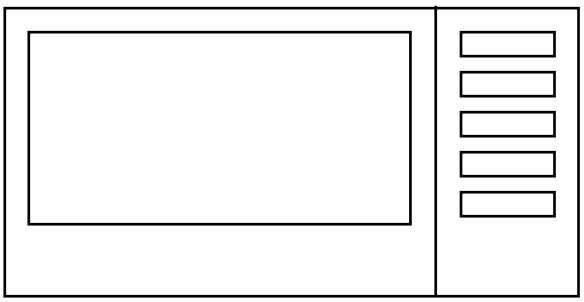
Denne del af modellen svarer til de tre listefelter i brugergrænsefladen og deres tilhørende funktioner. Kernen i denne del er klassen `BlockViewArea` og dennes tilhørende træ af blokke.



Figur 2.4.5-1 Klasser til operationer på træstrukturen

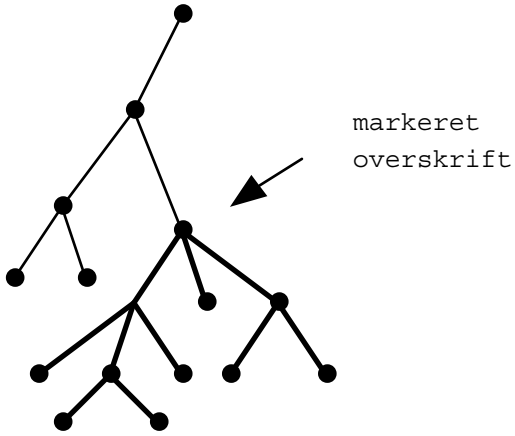
`BlockViewArea`

Denne generelle klasse bruges til at lave browse-, select- og hitarealerne. Klassen indeholder de dele af modellen der er fælles for de tre arealer. Et `BlockViewArea` indeholder fem knapper: `MoreButton, LessButton, ViewBlock, ViewFragment` og `FindButton`. Desuden indeholder det et felt hvor bloktitlerne vises, `BlockTextField`.



Figur 2.4.5-2 Det generelle blokfelt `BlockViewArea`.

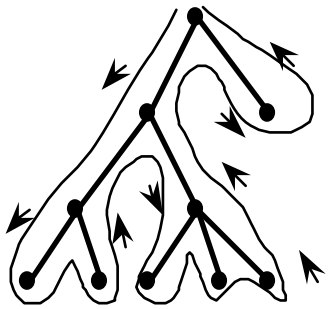
Til et `BlockViewArea` hører en træstruktur over de blokke, der på et givet tidspunkt findes i det tilhørende listefelt. Da der i TeSS kan optræde flere manualer, har vi for hvert listefelt, dvs. for hvert `BlockViewArea`, indført en "pseudo-rod", der indeholder en gren for hver manual. Endvidere indeholder `BlockViewArea` en reference til den overskrift (knude i træet) der for øjeblikket er markeret.



Figur 2.4.5-3 Træ af blokke, og markeret deltræ.

Funktionerne *more detail* og *less detail* tager udgangspunkt i den markerede overskrift. Ud- og sammenfoldning foregår ved at ændre værdien af `BlockInTree`-objekternes "`visible`"-variabel i deltræet.

Funktionen *find string* søger efter en tekststreng i alle overskrifter, både synlige og usynlige. Ved søgningen startes ved den markerede overskrift. For at søgningen af brugeren vil blive opfattet som sekventiel ned gennem listen skal programmet gennemløbe træet dybde-først:



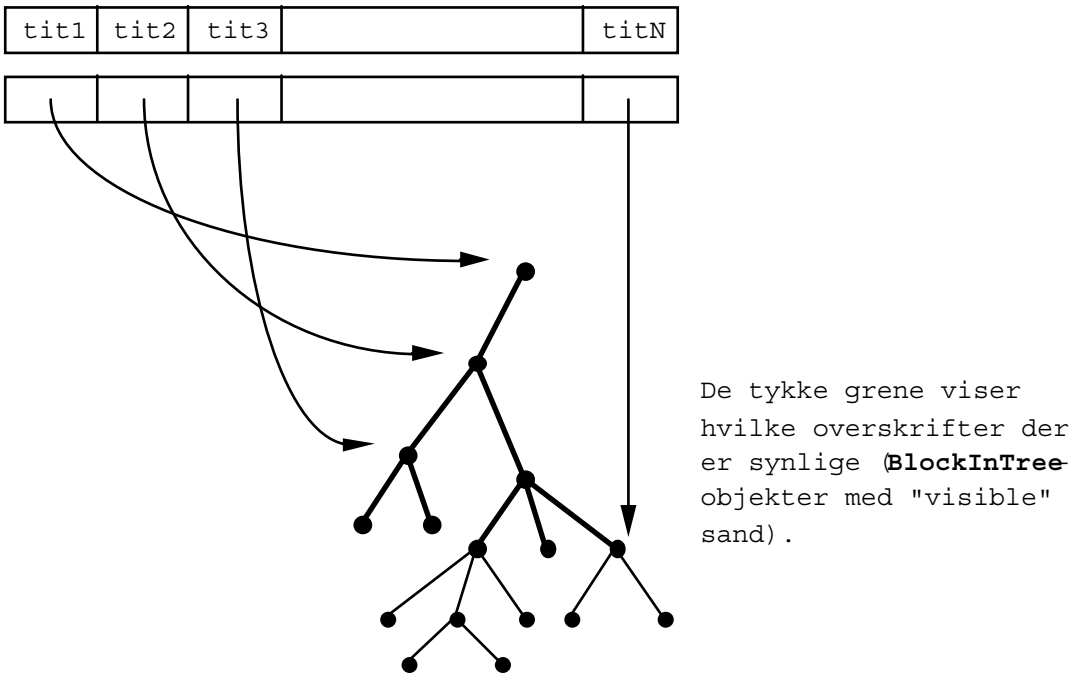
Figur 2.4.5-4 Søgning dybde-først i træet.

Hvis strengen findes i en blok, der er sammenfoldet (usynlig), vil blokken, og de af dens forfædre der måtte være sammenfoldede, blive foldet ud.

BlockTextField

Klassen svarer til selve listen af overskrifter. Den indeholder et felt på skærmen, hvor alle blokkenes titler vises, og hvor en blok kan udvælges ved klik på en titel. Til dette bruger vi Athena-objektet, der hedder `ListWidget`. Det er et felt på skærmen, med en eller to scroll-bars, hvori der vises en liste af tekststrenge.

Objektet indeholder også en datastruktur, der bruges til at udveksle information med `ListWidget`-objektet. Denne datastruktur består af to tabeller, der er lige lange. Den ene tabel indeholder bloktitler, der kan bruges af `ListWidget`. Den anden tabel indeholder referencer til `BlockInTree`-objekter, så det er nemt at finde blokken svarende til en titeltekst. Ved klik på en linie returnerer `ListWidget` nemlig et tabelindex.



Figur 2.4.5-5 Datastruktur til kommunikation med `ListWidget`

BrowseArea

Denne klasse udgør det øverste delvindue, "All Blocks In TeSS", i brugergrænsefladen. Den er en specialisering af `BlockViewArea`, og udover de nedarvede dele indeholder klassen en knap, der aktiverer *select*-funktionen i `SelectArea`.

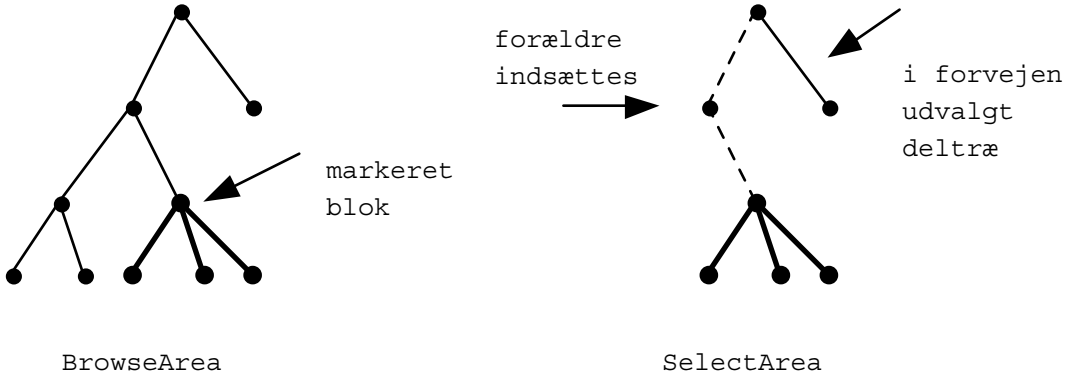
Træet der hører til `BrowseArea` er statisk. Det indlæses ved programmets start. For hver `BlockInTree`-knude oprettes da et `Block`-objekt. Når træerne hørende til felterne "Blocks Selected for Query" og "Blocks Matching Query" senere udvikler sig dynamisk efter brugerens handlinger, sker det ved at `BlockInTree`-objekter dannes og indsættes. Disse objekter refererer ikke til nye `Block`-objekter, men til dem der allerede fra start er indlagt i træet i `BrowseArea`.

Funktionen *locate* tager udgangspunkt i den overskrift (det deltræ) der er udpeget i `HitArea`. Den samme overskrift skal lokaliseres i `BrowseArea`, hvor den skal gøres synlig og blive den nye markerede overskrift. Endvidere skal alle dens forfædre og søskende i træet gøres synlige.

`SelectArea`

Denne klasse udgør det næstøverste delvindue, "Blocks Selected for Query", i brugergrænsefladen. Den er en specialisering af `BlockViewArea`, og udover de nedarvede dele indeholder klassen en knap, der aktiverer *remove*-funktionen.

Funktionen *select*: Et deltræ er udpeget i `BrowseArea`, og dette deltræ skal kopieres og indsættes i træet i `SelectArea`. Hvis der i dette træ mangler nogle af deltræets forfædre, vil disse blive kopieret og indsat.



Funktionen *remove*: Et deltræ er udpeget i `SelectArea`, og dette deltræ skal fjernes. Nyt aktuelt deltræ bliver deltræets forælder. Hvis det deltræ, der fjernes, er på øverste niveau (en hel manual) vil der ikke være noget udvalgt bagefter.

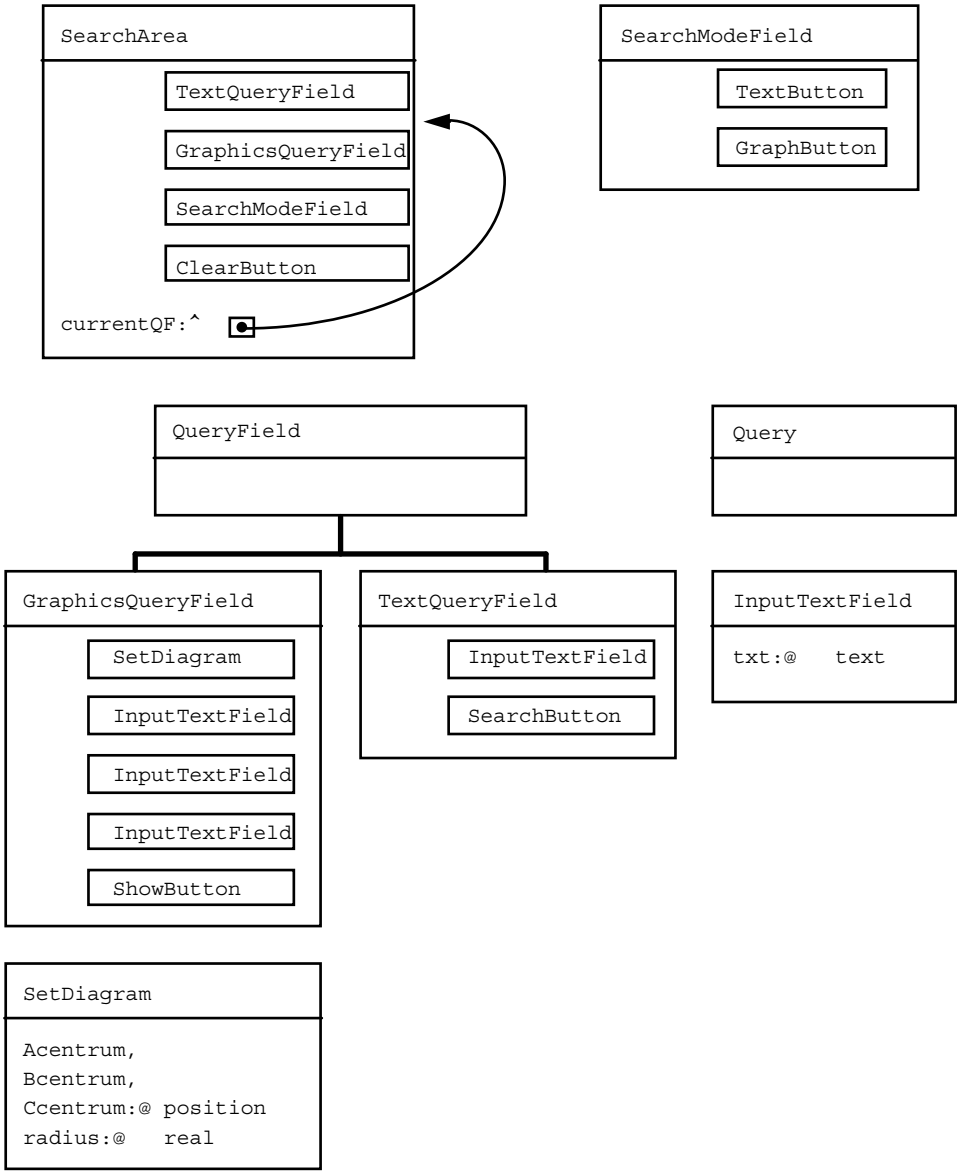
`HitArea`

Denne klasse udgør det nederste delvindue, "Blocks Matching Query", i brugergrænsefladen. Den er en specialisering af `BlockViewArea`, og udover de nedarvede dele indeholder klassen en knap, der aktiverer *locate*-funktionen.

Funktionen *show hits* kaldes, når en forespørgsel er gennemført. De fundne blokke vises i listefeltet. Også forfædre til fundne blokke vises, selvom de ikke opfylder forespørgslens betingelser.

2.4.6 Søgning ved forespørgsler

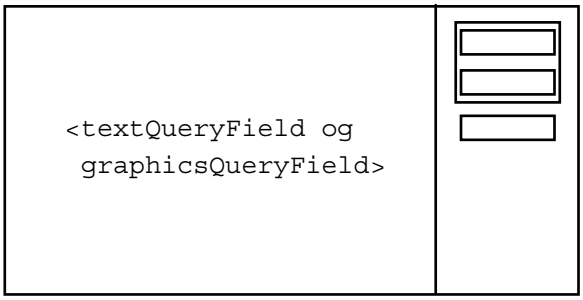
Denne del af modellen omfatter alle funktioner vedrørende forespørgsler til databasen.



Figur 2.4.6-1 Klasser der udgør søgning ved forespørgsler.

`SearchArea`

Denne klasse svarer til det samlede forespørgselsfelt, "Formulation of Queries", i brugergrænsefladen.



Figur 2.4.6-2 Delvinduet `SearchArea`

Klassen indeholder et felt til at vælge forespørgsels-måde og en knap til nulstilling af det aktive forespørgselsfelt. Endvidere indeholder klassen de to objekter `TextQueryField` og `GraphicsQueryField`, der svarer til felterne hvor forespørgsler formuleres. Der er kun eet af disse fremme på skærmen ad gangen. Dette afspejles ved referencen `"currentQF"` der peger på et af de to `QueryField`-objekter.

`QueryField`

Dette er en generel klasse, der bruges til at lave de to objekter `TextQueryField` og `GraphicsQueryField`.

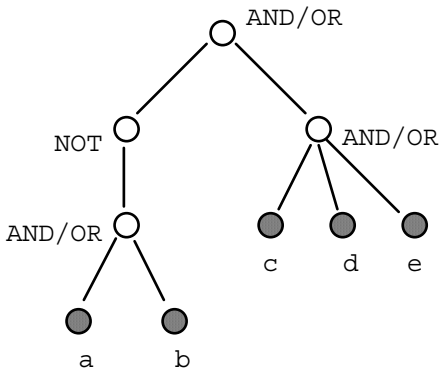
I klassen er erklæret en virtuel procedure *parser*, der udvides i `TextQueryField` og `GraphicsQueryField`. Denne procedure bruges til at indlæse og syntaks-checke søgeudtryk.

Funktionen *doSQLsearch* indlæser fra et indtastningsfelt og udfører den tilhørende forespørgsel. Funktionen udfører følgende delprocesser:

- Feltet parses vha. den virtuelle *parser*-procedure. Der dannes herved et `Query`-objekt.
- Udfra denne `Query` checkes om forespørgslen indeholder stopord.
- Det logiske udtryk omformes vha. proceduren *flatten* i `Query`.
- Der konstrueres en SQL-forespørgsel der udføres via dynamisk SQL.
- Ved modtagning af resultatet indlæses fra en SQL-cursor, og der testes om en funden blok findes i de udvalgte blokke.
- Den handling, der herefter skal ske med den enkelte blok, er virtuelt defineret og specificeres ved kald af *doSQLsearch*.

`Query`

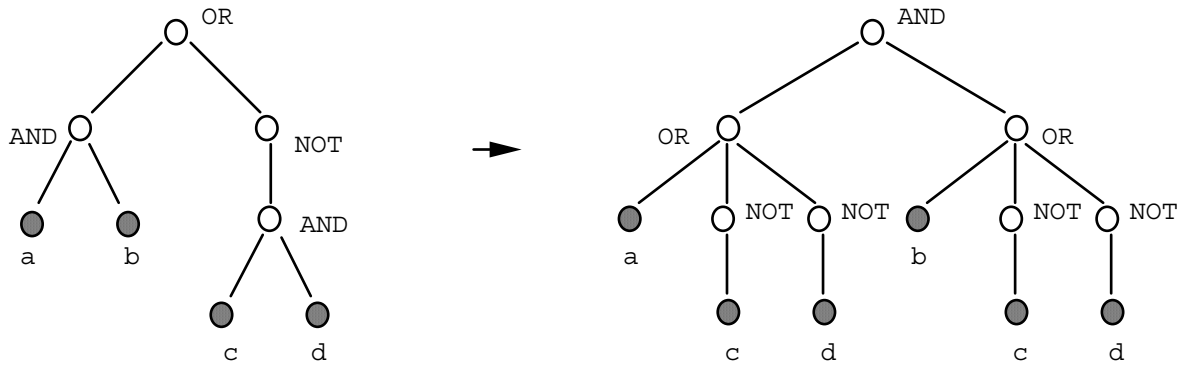
En forespørgsel formuleres enten i "logical expression mode" eller i "Venn diagram mode". Alle forespørgsler skal oversættes til et eller flere SQL-kald til Ingres. Vi har defineret en fælles form som forespørgsler i begge "modes" kan omformes til, og derefter oversættes fra denne form til SQL. Denne fælles form findes i programmet som datastrukturen `Query`, der er logiske udtryk formuleret som træstrukturer, hvor knuderne repræsenterer AND, OR eller NOT, og bladene søgeord.



Figur 2.4.6-3 Logisk udtryk repræsenteret som træstruktur

Træet på figuren repræsenterer udtrykket: (NOT (a OR b)) AND (c OR d OR e).

Funktionen *flatten* omformer en `Query` til en ny `Query` hvor der er højst een AND-knude. Denne vil altid være roden i træet. Alle OR-knuder vil være direkte under AND-knuden, og NOT-knuder vil således kun have søgetermer under sig.



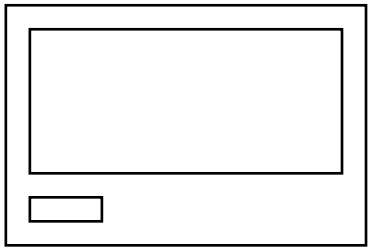

```
(a AND b) OR NOT (c AND d)
```

```
(a OR NOT c OR NOT d) AND
(b OR NOT c OR NOT d)
```

Figur 2.4.6-4 Eksempel på omformning med *flatten*.

`TextQueryField`

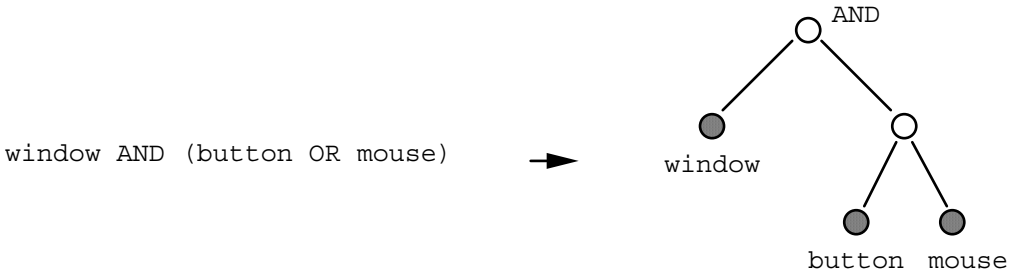
Klassen svarer til brugergrænsefladens felt til formulering af forespørgsler som logiske udtryk.



Figur 2.4.6-5 Felt til indtastning af logiske udtryk

Klassen indeholder et tekstfelt hvor forespørgslen indtastes, og en knap, `SearchButton`, hvormed søgning igangsættes.

Funktionen *parser* omdanner den indtastede forespørgsel til et `Query`-objekt.

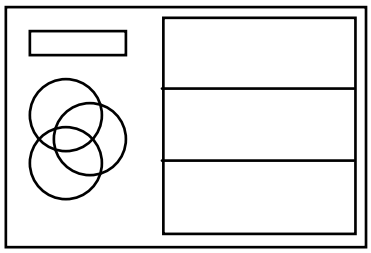


Figur 2.4.6-6 Eksempel på parsning af logisk udtryk

Funktionen *doSearch* aktiverer *doSQLsearch*, og opsamler hits i et træ af `BlockInTree`-objekter der vises i "Blocks Matching Query".

`GraphicsQueryField`

Klassen svarer til brugergrænsefladens felt til formulering af forespørgsler via Venn-diagrammer.



Figur 2.4.6-7 Feltet til forespørgsler vha. Venn-diagram

Det grafiske forespørgselsfelt indeholder en tegning af de tre grundmængder, A, B og C, samt alle kombinationer af dem (ialt 7 felter), tre felter til indtastning af de søgeord, der bestemmer de tre grundmængder, og knappen "show number of hits".

Ved søgning via Venn-diagrammet skal der for hvert af de tre indtastningsfelter dannes et søgeudtryk, og derefter skal de tre udtryk kombineres. Hvis funktionen *showHitNumbers* aktiveres skal antal hits bestemmes for hver af de syv kombinationer (kombinationen hvor ingen af udtrykkene gælder beregnes ikke). Hvis der klikkes i en af delmængderne på diagrammet bestemmes antal hits i de syv kombinationer, og mængden af hits overføres til `HitArea`.

Ved kombination af de tre udtryk for grundmængderne havde vi to løsninger at vælge imellem:

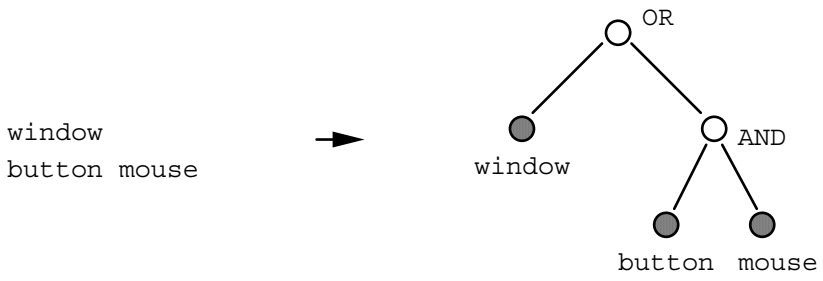
1) Danne syv SQL-forespørgsler
2) Danne tre SQL-forespørgsler, indlæse resultatet i tre datastrukturer, og danne de syv delmængder udfra dette.

Vi valgte løsning 2, dels fordi vi var nervøse for svartiderne i løsning 1, og dels fordi løsning 2 giver mulighed for genbrug. Hvis brugeren f.eks. kun retter i B-feltet og så spørger igen, behøver programmet kun at danne SQL-forespørgsel for dette felt, og derefter genberegne de syv kombinationer. Resultatet af søgningerne A og C er stadig ajour.

Til hver af de tre grundmængder indeholder klassen derfor en liste af blokid'er, der bruges til at huske resultatet af forespørgsler for de tre søgeordsfelter. Ved aktivering af et søgeordsfelt (man taster i det) vil den tilsvarende blokid-liste blive nulstillet.

Funktionen *parser* danner en `Query` udfra eet af indtastningsfelterne i Venn-diagrammet. I Venn-diagrammets indtastningsfelter vil ord på samme linie medføre en AND-knude og hver linie

udover den første vil medføre en OR-knude.



Figur 2.4.6-8 Parsning af indtastede søgeord i Venn-diagram

Funktionen *showHitNumbers*: For de tre søgeordsfelter undersøges om der allerede findes et aktuelt søgeresultat. Ellers dannes det. Udfra de herved dannede blokid-lister findes antal hits i de forskellige delmængder. De syv tal der findes vises på diagrammet.

Funktionen *doSearch* kaldes fra `SetDiagram` når der klikkes i en af de syv delmængder på diagrammet. For de tre søgeordsfelter undersøges om der findes et aktuelt søgeresultat. Ellers søges ved kald af *doSQLsearch*, hvorved der opbygges en blokid-liste. Udfra disse tre blokid-lister konstrueres et træ af `BlockInTree`-objekter der vises i `HitArea`.

`SetDiagram`

Indeholder selve tegningen af mængderne, samt felter til at vise antal hits. Tegningen er følsom overfor klik med musens venstre knap. Der beregnes en boolsk tripel (A-ramt,B-ramt,C-ramt), der viser hvilke af de tre grundmængder der er ramt af klikket. For en cirkel med centrum (cx,cy) ved klik i (x,y):

$$ramt = (cx-x)^2 + (cy-y)^2 \leq r^2$$

Hvis det ikke resulterer i (falsk,falsk,falsk) skal der kaldes *doSearch*.

2.4.7 Brug af Xt og Athena Widgets

Vi har nu præsenteret en samlet objektmodel for TeSS. I modellen svarer de fleste objekter direkte til objekter i brugergrænsefladen. Dette er en naturlig følge af, at modellen er opbygget med udgangspunkt i definitionen af brugergrænsefladen.

Selve skærmbilledet er implementeret ved Xt/Athena-objekter. Når man opbygger en brugergrænseflade ved hjælp af Xt/Athena indgår samtlige elementer i skærmbilledet i et træ. Roden er programmets hovedvindue, og dette underinddeles i felter, der igen underinddeles osv. Ethvert felt er således barn af netop eet andet felt. Disse forælder-børn-relationer svarer i de fleste

tilfælde direkte til del-helhed-relationer i modellen. Felter kan have et vilkårligt antal børn, dog afhængigt af felttypen. Disse felttyper kaldes i Xt/Athena klasser, og er ordnet i et hierarki.

I Mjølner Betas grænseflade pakke er Xt/Athena-klasserne implementeret som almindelige Beta-klasser. Beta's interface til Xt/Athena giver os derfor mulighed for at definere nye klasser som er direkte specialiseringer af Xt/Athena-klasser. Det har vi udnyttet ved implementation af brugergrænsefladen, idet vi har kunnet samle fælles udseende-mæssige egenskaber i nogle generelle klasser, lavet som specialiseringer af nogle Xt/Athena-klasser. Disse er derefter specialiseret til de enkelte dele af skærmbilledet. De af modellens objekter, der svarer direkte til noget på skærmbilledet, er simpelthen implementeret som en specialisering af en passende Xt/Athena-klasse. Derudover har vi tilføjet nogle Xt/Athena-objekter som "mellemlag" for at opnå det ønskede layout. I bilag C findes det samlede Xt/Athena Widgets objektræ.

Vi vedlægger ikke kildeteksten til programmet i denne rapport, fordi den fylder for meget, men hvis nogen er interesseret i få den, er de velkomne at kontakte enten Erik Frøkjær eller Morten Hertzum.

3 Forsøget

TeSS er udviklet til undersøgelse af brugergrænseflader til tekstsøgesystemer. I det foregående kapitel er design og implementation af selve TeSS behandlet. I dette kapitel behandles det forsøg, TeSS indgår i, og de tilføjelser til systemet, forsøget giver anledning til. Brugergrænsefladen evalueres gennem et forsøg, der udføres i tilknytning til rapportopgaven om brugergrænseflader på Datalogi 2, foråret 1993. TeSS er omdrejningspunktet for denne rapportopgave; de studerende arbejder med TeSS, hvadenten de deltager i forsøget eller ej. De studerende deltager i forsøget ved at give tilladelse til, at deres interaktion med TeSS under løsning af rapportopgaven logges.

I næste afsnit behandles tilrettelæggelsen af forsøget. Indsamlingen af data fra forsøget - logningen - behandles i kapitlets andet afsnit. Endelig behandler kapitlets tredje afsnit den forsøgsstyringsskal, der etableres udenom TeSS. Analysen af forsøgsmaterialet falder udenfor denne rapports rammer.

3.1 Tilrettelæggelse af forsøg

Forsøget er tilrettelagt ud fra et generelt og et specifikt sigte.

Generelt ønsker vi at tilvejebringe empirisk primærmateriale til studier af menneske-maskine interaktion. Det betyder, at vi ikke på forhånd kan udpege en lille gruppe forhold og sammenhænge, vi ønsker at indsamle data om. Der skal således indsamles detaljerede data om brugernes interaktion med TeSS. Det giver mulighed for, at materialet kan bruges i forskelligartede studier, som vi eller andre - efter aftale med os - ønsker at gennemføre.

Specifikt ønsker vi at undersøge og sammenligne teknikker til browsing og søgning ved forespørgsler: Hvordan bruges browsing og søgning ved forespørgsler i forbindelse med forskellige søgninger? Hvilke søgninger er de to typer søgeteknikker egnede til? Vi ønsker endvidere at undersøge, hvorvidt - og i givet fald hvordan - de to typer teknikker kan støtte forskellige dele af een og samme søgning. Hvis de to typer teknikker i højere grad supplerer end dubblerer hinanden, må søgesystemer, der tilbyder begge muligheder, forventes at være signifikant bedre end systemer med blot een af mulighederne.

Afsnittet indledes med en beskrivelse af den involverede brugergruppe og en diskussion af

brugernes anonymitet. Derefter diskuteres de forskellige konfigurationer af TeSS, der indgår i forsøget, og der gøres rede for de fire opgavesæt, brugerne stilles overfor. Det andet opgavesæt er det mest omfattende, og forsøgets specifikke sigte står og falder med, at det gennemføres uden at der introduceres systematiske skævheder. Afsnittet afsluttes med, at der udarbejdes en detaljeret forsøgsplan for dette opgavesæt.

3.1.1 Brugergruppe

TeSS er et eksempel på et tekstsøgesystem designet til fagfolk. I dette tilfælde er der tale om programmører og systemudviklere, i andre tilfælde er der tale om jurister, se f.eks. Hertzum & Søes (1992), forskere indenfor matematik, se f.eks. McAlpine & Ingwersen (1989), og lignende. Fagfolk besidder faglig kompetence og står dermed i modsætning til faglige novicer. Fagfolk kan derimod ikke forventes at have særlige forudsætninger for informationssøgning og står dermed i modsætning til søgeeksperter som f.eks. bibliotekarer. Programmører og systemudviklere adskiller sig fra andre fagfolk ved at have indgående kendskab til datamaskiner og boolsk algebra. På disse punkter har de bedre forudsætninger for at benytte sædvanlige edb-baserede tekstsøgesystemer, end andre fagfolk har.

I forbindelse med dette forsøg er brugerne ikke fungerende fagfolk, men datalogistuderende i slutningen af deres grunduddannelse. De studerende vil typisk allerede have en vis praktisk erfaring med programmering og systemudvikling fra deres erhvervsarbejde. Vi vil derfor tillade os at betragte de studerende som snart-fagfolk og ikke skelne skarpt mellem dem og fungerende fagfolk. Man skal dog være opmærksom på, at i en uddannelsesmæssig sammenhæng er kriterierne for gode løsninger anderledes end for personer, der er ansat som programmører eller systemudviklere. For studerende er f.eks. svartider og behandling af fejlsituationer typisk ikke centrale problemstillinger, mens der lægges stor vægt på velargumenterede og 'rene' løsninger. Som ansat programmør eller systemudvikler er praktiske hensyn de helt afgørende, og løsningerne dikteres ofte af et krav om at udvikle systemer, der kører til tiden og ikke går ned.

De studerende vil i størstedelen af forsøget bruge TeSS individuelt. Een persons informationssøgning er en relevant, kompliceret og langtfra afklaret proces. Design og evaluering af brugergrænseflader til individuel tekstsøgning er både påkrævet og udfordrende. Vi er imidlertid klar over, at både studerende og andre fagfolk ikke blot arbejder individuelt, men også i projektgrupper. Vi har ikke ønsket at komplicere undersøgelsen yderligere ved også at gøre gruppearbejde til et vigtigt aspekt. Vi har kun meget begrænsede muligheder for at kontrollere, om nogen alligevel løser opgaverne i fællesskab, og vi er klar over, at den tætte forbindelse mellem forsøg og rapportopgave gør det svært at trække skarpe grænser. Vi kan kun anmode de studerende om at respektere forsøget og dets præmisser. Derudover er deltagelse i forsøget en frivillig sag, så vi håber og tror, at de, der deltager, vil have forståelse for de forskningsmæssige aspekter.

Det er sjældent, der er mulighed for at få en større gruppe af fagfolk til at bruge og evaluere et søgesystem i deres opgaveløsningssituation. Det er derfor ret unikt, at vi kan udføre vores forsøg i forbindelse med en af datalogistudiets rapportopgaver og med knapt 100 brugere.

3.1.2 Anonymitet

Forsøget er tilrettelagt som en evaluering af TeSS' brugergrænseflade, ikke en test af brugerne. Vi er ikke interesserede i at sætte navn på besvarelserne af opgaverne. Når spørgsmålet om anonymitet alligevel melder sig, skyldes det to forhold: For det første er det svært at sikre anonymitet i en undersøgelse, der udføres i et miljø, hvor alle brugere skal have konto og brugernavn. For det andet er det afgørende for realismen i undersøgelsen, at brugerne kan løse opgaverne i flere adskilte sessioner. Der er derfor behov for at kunne identificere og samle sessioner, der hører til samme bruger.

Et vigtigt aspekt i vores bestræbelser på at gennemføre forsøget anonymt har været at sikre, at de studerende kunne være helt trygge ved at deres deltagelse i forsøget ikke kobles med deres besvarelse af rapportopgaven. Hverken vi eller de studerende kan leve med den usikkerhed, der kan opstå om, hvorvidt deltagelse i undersøgelsen influerer på karakteren for rapportopgaven, eller hvorvidt ønsket om en god karakter for rapportopgaven skaber særlig tolerance og ihærdighed ved deltagelsen i undersøgelsen. Den første forudsætning for at etablere disse rammer om forsøget er, at deltagelse er frivillig. Derudover søger vi at sikre brugernes anonymitet på følgende måde:

Ved den forelæsning, hvor rapportopgaven udleveres, trækker alle studerende en seddel med et brugerid og et password. Dette brugerid og password er den studerendes nøgle til TeSS og skal bruges, hvadenten han eller hun deltager i forsøget eller ej. En nøgle giver imidlertid ikke adgang til TeSS, før den studerende har angivet, om han eller hun ønsker at deltage i forsøget ved at tillade, at sessionerne med TeSS logges. Tilladelsen til eller afvisningen af logning sker under brugerid og uden angivelse af hvilken navngiven person, der har trukket det pågældende brugerid. Adgang til TeSS opnås derefter ved først at logge ind på Unix-systemet på sædvanlig vis og dernæst logge ind på TeSS med den særlige TeSS-nøgle.

Udfra brugerid'et kan de forskellige dele af en studerendes besvarelse sammenkædes. Ved både at have et brugerid og et password kan vi også være temmelig sikre på, at en studerende ikke - tilsigtet eller utilsigtet - får adgang til TeSS under en andens brugerid. Brugerid'erne er en fortløbende række tre-cifrede tal. Password'ene er genereret ud fra indekset til en Unix-manual: For hver indgang i indekset på mindst syv bogstaver er genereret et password bestående af bogstav fire til syv.

Vi ved, hvilke navngivne personer der løser rapportopgaven, og vi ved, hvilke nøgler der er i brug. Sålænge disse oplysninger ikke sammenkædes er anonymiteten intakt. Vi troede længe, vi helt kunne undgå denne sammenkædning ved hjælp af de muligheder, Unix giver for automatisk at

ændre på, hvem der sættes som ejer af de filer, en applikation genererer. Alle de forsøgte løsninger viste sig imidlertid at have uacceptable bivirkninger. Sådan som TeSS er implementeret registreres oplysninger, der i en kort periode muliggør at TeSS-gruppens medlemmer foretager sammenkædningen. Det kan ske ved at tage en logfil for et givet brugerid og se, hvilken Unix-bruger der ejer denne fil. Det skal dog bemærkes, at disse oplysninger kun var tilgængelige for TeSS-gruppen og ikke den enkelte bruger. I løbet af en eller to dage ændrede vi imidlertid ejerskabet på logfilerne (i stedet for den oprindelige ejer sættes et af TeSS-gruppens medlemmer som ejer). Efter dataindsamlingen var afsluttet, og analysen af de indsamlede data skulle til at begynde, var brugerne kun identificeret ved deres TeSS-brugerid. Vi finder derfor, at de studerendes anonymitet er sikret på betryggende vis.

3.1.3 Konfigurationer

Forsøget skal indrettes på at give indblik i, hvorledes teknikker til browsing og til søgning ved forespørgsler bruges i og er egnet til forskellige søgninger. Der findes flere forskellige teknikker til såvel browsing som søgning ved forespørgsler. TeSS omfatter een teknik til browsing og to teknikker til søgning ved forespørgsler.

Vi ønsker både at sammenligne forskellige søgeteknikker og at se på samspillet mellem dem. For at kunne gøre det skal forsøget omfatte:

- Sammenligning af et søgesystem med een søgeteknik og et andet søgesystem med en anden søgeteknik. Her har hver enkelt bruger kun een søgeteknik til rådighed, og søgningerne er derfor uafhængige af brugernes subjektive vurdering af, hvilken søgeteknik der ville være bedst egnet i de pågældende tilfælde. Det giver mulighed for at sammenligne brug af de forskellige søgeteknikker på et ensartet grundlag.

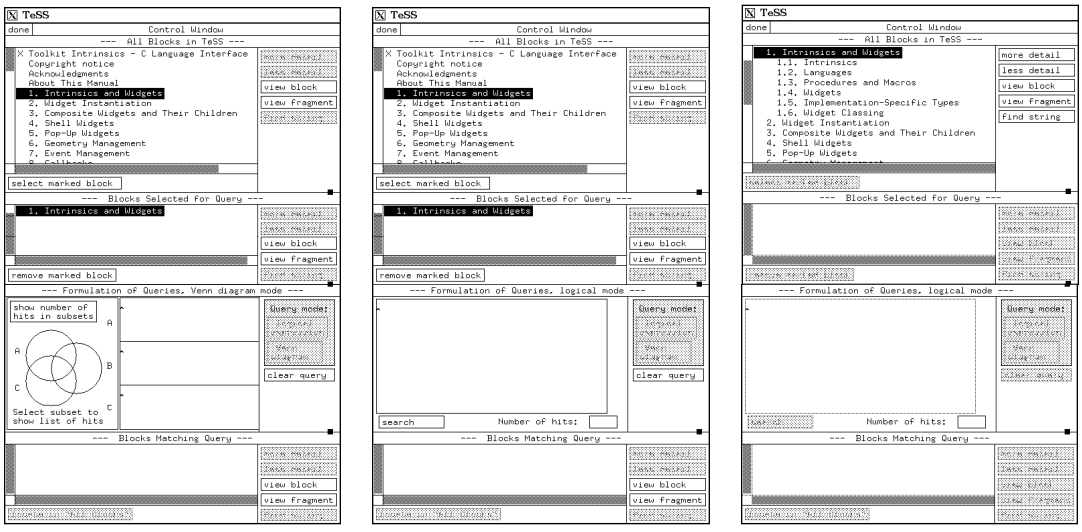
- Undersøgelse af brugen af de enkelte søgeteknikker og samspillet mellem dem i et søgesystem, der tilbyder en kombination af søgeteknikker. Her vil brugernes præferencer komme til udtryk, da de selv vælger, hvilken søgeteknik eller kombination af søgeteknikker de vil anvende. Vi får derfor mulighed for at undersøge samspillet mellem søgeteknikkerne, og for at sammenligne med de søgninger, hvor brugerne kun har een søgeteknik til rådighed.

Det er endvidere vigtigt, at forsøget indeholder et referenceniveau, der giver mulighed for at sætte vores forsøg i relation til andre forsøg og andre hjælpemidler til tekstsøgning. For at lette sammenligning med andre edb-baserede tekstsøgesystemer omfatter TeSS en variant af den absolut mest udbredte søgeteknik: boolsk søgning hvor forespørgslerne angives ved logiske udtryk. Vi

finder det derudover relevant at sammenligne brug af TeSS med manuel søgning i papirkopier af de involverede manualer. Forsøget omfatter således to referenceniveauer. Samlet omfatter forsøget fem forskellige konfigurationer - i form af fire forskellige konfigurationer af TeSS samt papirkopier af manualerne:

1. Browsing
2. Boolsk søgning ved logiske udtryk
3. Boolsk søgning ved Venn diagrammer
4. Hele TeSS, dvs. de tre ovennævnte konfigurationer på een gang
5. Manuel søgning i papirkopier af manualerne

Den fjerde konfiguration er TeSS som beskrevet i kapitel 2. De tre første konfigurationer er begrænsede konfigurationer af TeSS, fremkommet ved at gøre dele af brugergrænsefladen inaktiv. I den første konfiguration, browsing, er kontrolvinduets øverste delvindue, "All Blocks in TeSS" aktivt; de tre øvrige delvinduer er inaktive. I den anden konfiguration, boolsk søgning ved logiske udtryk, er alle faciliteter til at bevæge sig rundt i træstrukturen utilgængelige. Træstrukturen i delvinduet "All Blocks in TeSS" er fra starten foldet ud til kapitelniveau. Kapitler kan udvælges til og fjernes fra en søgning; men derudover består søgemulighederne udelukkende i at formulere forespørgsler ved logiske udtryk og bringe tekster op i Text Viewer'e. Den tredje konfiguration, boolsk søgning ved Venn diagrammer, svarer til den anden konfiguration, pånær at forespørgslerne skal formuleres ved hjælp af Venn diagrammer.



(a) (b) (c)

Figur 3.1.3-1 De tre begrænsede konfigurationer af TeSS. (a) Kun browsing, (b) Kun søgning ved forespørgsler formuleret som logiske udtryk og (c) Kun søgning ved forespørgsler formuleret ved Venn-diagrammer.

3.1.4 Opgavesæt

Forsøget består i, at hver bruger løser en række på forhånd givne opgaver med henblik på at opbygge viden om udvikling af X-applikationer med grafiske brugergrænseflader. Brugerne skal løse fire opgavesæt, der beskrives overordnet i det følgende. De konkrete opgaver, der indgår i de enkelte opgavesæt, er gengivet i bilag E.

Tænke-højt forsøg og initiel oplæring

Dette opgavesæt har to formål: (1) De studerende skal have lejlighed til at prøve en enkel og effektiv teknik til evaluering af brugergrænseflader - de skal planlægge og gennemføre et tænke-højt forsøg. Dette formål med opgavesættet er primært rettet mod rapportopgaven. (2) Opgavesættet skal gøre de studerende nogenlunde fortrolige med TeSS. Dette andet delformål er primært rettet mod forsøget. De opgaver, der skal løses, er enkle og sigter på at få brugerne til at prøve mange af søgesystemets faciliteter. Dette opgavesæt kræver mindst to personer - en forsøgsleder og en forsøgsperson - og gennemføres i rapportopgavegrupperne.

Informationssøgningsopgaver

Det andet opgavesæt udgør kernen i indsamling af data til forsøget. Data fra denne del af forsøget skal kunne bruges til sammenligningerne mellem:

- Browsing og forespørgsler
- Forespørgsler formuleret ved logiske udtryk og Venn-diagram
- Søgesystemer med een og flere søgeteknikker

Opgavesættet består af en række små, veldefinerede informationssøgningsopgaver. Sammen med hver opgave foreskrives, hvilken konfiguration der skal bruges ved løsningen. For at undgå systematiske skævheder i forsøgsresultaterne varierer det, hvilken konfiguration den enkelte bruger skal bruge til løsningen af hvilke opgaver. Opgaverne skal løses i rækkefølge og er søgt udformet så det vil tage 10-20 minutter at besvare hver af dem. Opgavesættet omfatter alle fem konfigurationer og løses individuelt. Den nærmere planlægning af gennemførelsen af dette opgavesæt behandles nedenfor.

Implementationsopgaver

Det tredje opgavesæt består i løsning af tre mindre implementationsopgaver. For at kunne løse disse opgaver må brugerne benytte TeSS til at skaffe sig viden om kommandoer og muligheder i det benyttede programudviklingssystem. Med dette opgavesæt undersøges brugen af systemet, når det skal fungere i samspil med udvikling af programmer. I forbindelse med dette opgavesæt stilles brugeren frit i valget af konfiguration, pånær at der ikke er mulighed for manuel søgning i papirkopier af manualerne.

De ideer, spørgsmål og problemer, der giver anledning til søgningerne under løsning af tredje opgavesæt, vil være mange og variere fra studerende til studerende. Sammenhængen mellem

brugernes intention og de foretagne søgninger er imidlertid ikke tilgængelig ved evalueringen af de loggede data. De loggede data kan derfor primært bruges til eksplorative undersøgelser af, hvad brugerne faktisk har gjort, f.eks. hvilke faciliteter og hvor lang tid har de brugt. Endvidere løses implementationsopgaverne på grund af deres størrelse i rapportopgavegrupperne. Det er ikke realistisk at tro, at de studerende i en rapportopgaveperiode kan afse tid til at løse opgaverne individuelt. Det kunne give anledning til et ønske om, i forsøget, at kunne sammenkæde de studerende, der udgør en gruppe. Det vil vi imidlertid afstå fra.

Spørgeskema

Det fjerde og sidste opgavesæt er et spørgeskema, der skal tjene to formål: For det første skal det give oplysninger om brugernes køn, alder, studie- og erhvervserfaring og lignende. For det andet skal det opfordre brugerne til at give udtryk for deres erfaringer med brug af TeSS og deres subjektive bedømmelse af de forskellige konfigurationer. Spørgeskemaet udleveres sammen med rapportopgaven og udfyldes individuelt. Det kunne også være et elektronisk dokument; men da skemaet for en stor del omhandler vurderinger og præferencer, finder vi det hensigtsmæssigt at give de studerende frihed til at overveje og udfylde det hjemme.

I rapportopgaven indgår endvidere en designopgave og en visionsopgave. De er ikke knyttet til forsøget og vil ikke blive behandlet yderligere her.

3.1.5 Forsøgsplan for informationssøgningsopgaverne

Det andet opgavesæt, informationssøgningsopgaverne, er det mest omfattende og kræver en særlig omhyggelig planlægning. Opgavesættet skal tilrettelægges på en måde, der ikke på forhånd introducerer skævheder i forsøgsmaterialet. Det vi vil sammenligne er de forskellige søgeformer. Det eneste, der må variere, er derfor hvilken konfiguration der benyttes ved løsningen af opgaverne - alle andre faktorer skal være neutrale i forhold til forsøget.

For at sikre at eventuelle sammenhænge mellem opgaverne påvirker forsøgsresultaterne på en ensartet måde, skal alle de studerende løse de samme informationssøgningsopgaver i samme rækkefølge. Det giver også mulighed for bevidst at lade opgaverne bygge på hinanden og derved skabe en vis sammenhæng i opgavesættet.

Hyppige skift mellem konfigurationerne vil gøre det svært for de studerende at danne sig selvstændige billeder af de forskellige konfigurationer. Det vil endvidere give en række praktiske problemer, specielt i forbindelse med de opgaver der skal løses ved manuel søgning i papirkopier af manualerne. For at løse disse opgaver må brugeren forlade terminalen. Det er i sig selv et afbræk i arbejdet og kan blive ledsaget af ventetid, både for at få adgang til papirkopier af manualerne og for bagefter at få adgang til en terminal igen. For at undgå disse problemer vil vi undlade hyppige skift mellem konfigurationerne og i stedet lade opgavesættet bestå af fem blokke af opgaver, hvor alle

opgaver i en blok løses ved hjælp af samme konfiguration. De fem blokke skal indeholde lige mange opgaver, da vi ellers vil skulle tage højde for denne forskel i erfaring med konfigurationerne, når forsøgsmaterialet analyseres.

Forsøgspersonerne skal altså bruge hver konfiguration til løsning af een blok af opgaver. Vi skal nu finde frem til i hvilken rækkefølge den enkelte forsøgsperson skal møde de 5 konfigurationer. Fordelingen af konfigurationer på blokke kan foretages på 5! (120) forskellige måder. Med knap 100 brugere kommer vi således ikke alle fordelinger igennem. Vi kan enten vælge at give alle de studerende forskellige fordelinger for derved at komme så langt som muligt i retning af at dække alle fordelinger, eller vi kan vælge at samle de studerende i grupper, der møder konfigurationerne i samme rækkefølge. Vi vælger den sidste mulighed, da den både giver mulighed for at sammenligne data fra brugere, der har været i samme forsøgssituation, og fra brugere, der har været i forskellige forsøgssituationer.

Vi formoder, at det er de konfigurationer, brugeren møder først, der påvirker hans eller hendes oplevelse af hele forsøget mest. Det er således vigtigt, at alle fordelinger af konfigurationer på de første blokke bliver dækket. Der er 5*4 (20) mulige fordelinger af konfigurationer på de to første blokke. Hvis der knyttes een gruppe af studerende til hver af disse 20 fordelinger, vil hver gruppe komme til at omfatte knap 100/20 (5) studerende. Vi finder, det er en rimelig balance mellem at sprede de studerende på mange fordelinger og holde en vis minimumsstørrelse på grupperne. De studerende deles altså i 20 grupper, der hver dækker een af de 20 mulige kombinationer af konfigurationer på de to første blokke af opgaver.

Med de to første konfigurationer valgt mangler vi endnu at fastlægge rækkefølgen af de tre resterende konfigurationer. Det varierer fra gruppe til gruppe, hvilke konfigurationer der endnu ikke er knyttet til en opgaveblok; men for alle 20 grupper kan de tre resterende konfigurationer opskrives sorteret ud fra en nummering af konfigurationerne. Hvis denne rækkefølge blev brugt direkte, ville det give systematiske skævheder i fordelingen, f.eks. ville alle studerende, der ikke har haft konfiguration 5 som en af de to første, få den tilsidst. Derfor varieres rækkefølgen af de tre resterende konfigurationer ved at cyklisk at anvende de seks permutationer af en 3-mængde. Resultatet er følgende fordeling af konfigurationer på blokkene af opgaver:

gruppe 1:	k_1	k_2	k_3	k_4	k_5
gruppe 2:	k_1	k_3	k_2	k_5	k_4
gruppe 3:	k_1	k_4	k_3	k_2	k_5
gruppe 4:	k_1	k_5	k_3	k_4	k_2
gruppe 5:	k_2	k_1	k_5	k_3	k_4
gruppe 6:	k_2	k_3	k_5	k_4	k_1
gruppe 7:	k_2	k_4	k_1	k_3	k_5
gruppe 8:	k_2	k_5	k_1	k_4	k_3
gruppe 9:	k_3	k_1	k_4	k_2	k_5
gruppe 10:	k_3	k_2	k_4	k_5	k_1
gruppe 11:	k_3	k_4	k_5	k_1	k_2
gruppe 12:	k_3	k_5	k_4	k_2	k_1
gruppe 13:	k_4	k_1	k_2	k_3	k_5
gruppe 14:	k_4	k_2	k_1	k_5	k_3
gruppe 15:	k_4	k_3	k_2	k_1	k_5
gruppe 16:	k_4	k_5	k_2	k_3	k_1
gruppe 17:	k_5	k_1	k_4	k_2	k_3
gruppe 18:	k_5	k_2	k_4	k_3	k_1
gruppe 19:	k_5	k_3	k_1	k_2	k_4
gruppe 20:	k_5	k_4	k_1	k_3	k_2

For at støtte de studerende i at anvende den foreskrevne konfiguration ved løsningen af opgaverne har vi automatiseret tildelingen af den konfiguration, der skal anvendes ved løsningen af hver opgave. Det er gjort ved udenom selve TeSS at implementere en skal, der varetager forsøgsstyringen, se afsnit 3.3 *Forsøgsstyringsmodulet*.

3.2 Logning

Logningen har til formål at registrere brugernes aktivitet under interaktionen med TeSS. Den foretages for at muliggøre en detaljeret evaluering af brugergrænsefladen, dvs. logningen indgår i den videre udvikling af brugergrænsefladen.

De to hovedspørgsmål i dette kapitel er at finde ud af, hvad der skal logges, og hvordan det skal gøres. Svaret på det første spørgsmål består i at fastlægge, hvilke kriterier vi vil evaluere brugergrænsefladen efter, og finde ud af hvilke muligheder der er for at indsamle data om disse kriterier. Vi vil indlede behandlingen af dette spørgsmål med en oversigt over evalueringskriterier, der har været anvendt i andre undersøgelser. Svaret på det andet spørgsmål bestemmes af sikkerheds- og performance-hensyn.

3.2.1 Kriterier i evaluering af brugergrænseflader

Valget af de kriterier, der skal bruges i evalueringen af en brugergrænseflade, afhænger af mange faktorer. Vi vil nævne tre væsentlige og hyppigt forekommende:

- Kriterierne skal kunne måles. Det betyder i mange tilfælde, at evalueringen udelukkende baseres på kvantitative kriterier.
- De nødvendige data skal kunne indsamles med et relativt lille forbrug af ressourcer.
- Kriterierne skal være relevante i forbindelse med det system, der skal evalueres.

I faglitteraturen findes flere bud på generelle sæt af kriterier; men dels er disse kriterier ikke alle relevante i alle sammenhænge, dels vil det ofte være relevant at inddrage specielle kriterier i evalueringen.

Som baggrund for vores valg af kriterier vil vi i det følgende se på, hvilke kriterier andre har brugt eller anbefalet. Til det formål har vi valgt fem eksempler på studier af brugergrænseflader ud fra den meget omfattende litteratur på området:

Beyer et al. (1986) søger at anvise, hvordan brugervenlighed kan sikres en mere central plads i systemudviklingen. Med henblik på at der, som en del af kravspecifikationen, skal stilles kvantificerede krav angående systemets brugervenlighed, opstilles seks kriterier for brugervenlighed:

- *Indlæringstid:* Den tid det tager at lære at løse bestemte opgaver ved hjælp af systemet.
- *Effektivitet:* Hastigheden hvormed bestemte opgaver løses af bruger og system i forening.
- *Svartid:* Den tid det tager systemet at besvare bestemte forespørgsler.
- Genindlæringstid: Den tid det tager en bruger, som har været væk fra systemet et stykke tid, at løse bestemte opgaver.
- *Fejlhyppighed:* Antallet af fejlsituationer som brugerne kommer i, når de løser bestemte opgaver ved hjælp af systemet.
- *Subjektiv tilfredshed:* Brugernes subjektive tilfredshed med systemet.

Hauptmann & Green (1983) opstiller fem kriterier for deres sammenligning af en kommando-baseret, en menu-baseret og en naturligsprogs brugergrænseflade til et simpelt system til præsentation af statistiske data i form af søjlediagrammer, lagkagediagrammer og lignende:

- Den tid, det tager at gennemføre hver af de stillede opgaver.
- Antallet af fejl, der indtræffer under udførelsen hver af de stillede opgaver.
- Antallet af ord, som brugerne må angive for at gennemføre hver af de stillede opgaver.
- Antallet af linier, som brugerne må angive for at gennemføre hver af de stillede opgaver.
- Brugernes subjektive tilfredshed med systemet.

Campagnoni & Ehrlich (1989) sammenligner to søgeteknikker - browsing og søgning ved forespørgsler - og deres samspil i forbindelse med brug af hjælpesystemet til Sun386i. De søger gennem undersøgelsen at besvare følgende spørgsmål:

- Foretrækker brugerne browsing fremfor søgning ved forespørgsler, målt i antal opgaver de vælger at løse med hver af de to teknikker?
- Afhænger den tid, det tager at løse de stillede opgaver, af den valgte søgeteknik?
- I hvilken udstrækning bruges browsing som et forstadie til søgning ved forespørgsler, målt i antal forekomne tilfælde?
- Afhænger antallet af korrekt besvarede opgaver af den valgte søgeteknik?
- Afhænger valget af søgeteknik af hvordan opgaverne er formuleret, målt ved sammenligning af den faktiske fordeling og den, opgaveformuleringerne lægger op til?
- I hvor mange tilfælde bruges søgning ved forespørgsler som en sidste udvej?
- Afhænger brugernes valg af søgeteknik af deres rumlige visualiseringsevner, målt ved sammenligning af deres faktiske valg og resultatet af en standardiseret test til måling af rumlige visualiseringsevner?

Whiteside et al. (1988) sammenligner kommando-baserede, menu-baserede og ikon-baserede brugergrænseflader, ialt 11 forskellige brugergrænseflader. Grundlaget for deres undersøgelse er et aggregeret performance-mål og et spørgeskema til indhentning af data om brugernes subjektive tilfredshed:

- Brugergrænsefladen vurderes ud fra et performance-mål, der beregnes for hver testbruger:

$$M = \frac{1}{T} \; PC,$$

hvor M = performance-mål
T = tid brugt på opgaverne
P = procentdel af opgaverne der er gennemført
C = en konstant, sat til 5 min.

- Ud fra dette performance-mål testes hypoteser om sammenhæng mellem performance-mål, grad af erfaring og forskellige typer opgaver.
- Brugernes subjektive tilfredshed blev evalueret ved hjælp af et spørgeskema og sammenlignet med opnået performance-mål, erfaring og opgavetype.

Girill & Luk (1992) diskuterer hierarkiske søgefaciliteter i forbindelse med et

hypertekstsystem til on-line dokumentation. I diskussionen forud for deres egen undersøgelse refereres to empiriske studier, der anvendte følgende evalueringskriterier:

- Den tid det tager at løse de stillede opgaver.
- Det antal kommandoer brugerne udfører for at løse de stillede opgaver.
- Antallet af korrekt besvarede opgaver.
- Antal tekstsider brugerne læser/skimmer for at løse de stillede opgaver.

3.2.2 Hvad skal logges?

På baggrund af eksemplerne på kriterier, andre har brugt, kan vi pege på en række forhold, som det vil være relevant at undersøge. De fem ovenfor beskrevne eksempler viser imidlertid også, at der er et stort antal sammenhænge, som potentielt kan give værdifulde oplysninger om brugernes interaktion med TeSS. Hvis man på forhånd ved præcis hvilke sammenhænge, man ønsker at måle, kan logningen i vid udstrækning reduceres til optællinger af hyppigheden af en række specifikke hændelser. Vi ser det imidlertid som et mål i sig selv at indsamle empirisk primærmateriale til studier af menneske-maskine interaktion. Vi vælger derfor en omfattende logning (og en deraf følgende omfattende efterbehandling) fremfor på forhånd at forsøge at analysere og gætte os til en lille mængde særlig vigtige faktorer.

En vigtig parameter i evalueringen af en brugergrænseflade - og det underliggende system - er tiden. Denne parameter går igen i alle fem eksempler på kriterier, andre har brugt. En central forskel på browsing og søgning ved forespørgsler er, at brugeren kan browse uden først at skulle formulere, hvad der søges efter; til gengæld bruges der tid på at navigere i træstrukturen og læse/skimme tekster. Søgning ved forespørgsler kræver, at brugeren formulerer en forespørgsel, før han eller hun har set det første eksempel på et relevant dokument; til gengæld leverer systemet en fuldstændig liste over de dokumenter, der matcher forespørgslen. Logningen skal give mulighed for at undersøge denne forskel på browsing og søgning ved forespørgsler, dvs. for at se hvor brugeren bruger sin tid. For de kommandoer, der må forventes at have en mærkbar svartid, skal denne endvidere registreres særskilt, så den samlede tid kan opdeles i svartid og den tid, brugeren har brugt på at reflektere over resultatet og overveje 'næste træk'.

For at give mulighed for at se, hvor brugeren bruger sin tid, skulle logningen også registrere, hvor brugeren var. Vi har imidlertid ingen mulighed for at registrere, hvad brugeren tænker eller kigger på. Vi kan derimod registrere, hvilke kommandoer der udføres. En sådan registrering giver f.eks. mulighed for at gøre op, hvor hyppigt en kommando bruges, hvor mange kommandoer det kræver at løse en opgave, og om der er sekvenser af kommandoer, som ofte gentages. En stor del af kriterierne i de fem eksempler kan faktisk gøres op ud fra en registrering af hvilke kommandoer, der udføres. Antallet af læste/skimmede tekster pr. opgave fremgår f.eks. af, hvor mange tekstvinduer, der er blevet åbnet. Vi vil derfor lade loggen indeholde oplysning om enhver

kommando, der udføres.

Brugergrænsefladen kan opdeles i elementer, hvoraf de væsentligste er "All Blocks in TeSS", "Blocks Selected for Query", "Formulation of Queries", "Blocks Matching Query" og ingen, en eller flere Text Viewer'e. De kommandoer, der udføres, vil altid høre til eet af disse elementer. Ved for hver kommando at registrere hvilket af brugergrænsefladens elementer, den hører til i, kan man opdele besvarelsen af en opgave i sekvenser af kommandoer der hører til samme element. Det vil give et vist billede af, hvilke dele af brugergrænsefladen brugeren er beskæftiget med, samt hvor lang tid brugeren beskæftiger sig med hver enkelt del.

Endelig er det afgørende for vurderingen af de loggede data at vide, hvilken opgave de hører til, og hvor langt brugeren kom i løsningen af denne opgave. Det skal derfor registreres hver gang, brugeren påbegynder og afslutter en opgave, og ved afslutningen skal løsningen på opgaven også logges. Det første kan gøres automatisk i den forsøgsstyringsskal, der ligger udenom selve TeSS; det andet anmoder vi brugeren om at indtaste i forsøgsstyringsvinduets resultat-felt, se afsnit 3.3 *Forsøgsstyringsmodulet*.

Det ovenstående kan sammenfattes til, at loggen skal indeholde:

- Tidsstempling af alle oplysninger. For de kommandoer, der må forventes at have en mærkbar svartid, skal det både registreres, hvornår kommandoen afgives, og hvornår den er afsluttet.
- En registrering af alle udførte kommandoer, med eventuelle parametre.
- En registrering af hvilket af brugergrænsefladens elementer, kommandoerne hører til.
- En registrering af hvilken opgave, der arbejdes med.
- En registrering af hvilken løsning brugeren når til på de enkelte opgaver.

Denne logning er omfattende og kan kun gennemføres for de fire konfigurationer af TeSS, ikke for de opgaver, der løses ved manuel søgning i papirkopier af manualerne. Logningen af søgning i papirkopierne består blot i en registrering af:

- Hvor lang tid, der bruges på løsning af hver opgave.
- Hvilken løsning brugeren når frem til.

Selvom det er en grovkornet registrering, omfatter den to meget væsentlige aspekter for evaluering af et søgesystems anvendelighed.

Udover registreringen af sessionernes forløb er vi interesserede i oplysninger om brugernes subjektive tilfredshed med og vurdering af TeSS' brugergrænseflade i de forskellige konfigurationer. Disse oplysninger søges indhentet gennem det fjerde opgavesæt: spørgeskemaet.

3.2.3 Hvordan skal der logges?

Et overordnet valg i fastlæggelsen af, hvordan logningen skal foregå, er placeringen af de data, der logges. De loggede data kan enten lagres i den relationsdatabase, der indgår i tekstdatabasen, eller i filer. Vi har valgt at logge i filer. En forudsætning for valget af denne løsning var, at den kunne realiseres på en måde, hvor kun TeSS-gruppens medlemmer har adgang til de loggede data. To andre forhold spillede også en nævneværdig rolle: (1) Logning i filer er hurtigere end logning i en relationsdatabase. (2) Unix-filsystemet er meget stabilt, mens der har været mange problemer med DIKU's Ingres-installation. De data, der logges, er primærmateriale, som der ikke er nogen mulighed genskabe, hvis den første registrering kikser. Vi vægter derfor sikkerheden i logningsproceduren meget højt og har af samme grund etableret en backup-rutine omkring logfilerne, der tager højde for selv meget usandsynlige sammentræf af i sig selv usandsynlige hændelser.

Der oprettes en logfil for hver opgave, der løses ved hjælp af TeSS, og yderligere en logfil for hver Text Viewer, der åbnes. Hvis en opgave afbrydes, f.eks. for at brugeren kan komme hjem og sove, og senere genoptages, giver det endvidere anledning til en ekstra logfil. Forsøget giver således anledning til et meget stort antal logfiler. Den fuldstændige specifikation af logfil-formatet findes i bilag F; i det følgende gives en overordnet beskrivelse.

Af hensyn til efterbehandlingen er det vigtigt, at det er let at samle de logfiler, der hører til een bruger, og ordne dem kronologisk. Derfor indeholder logfilernes filnavne en brugeridentifikation og en tidsangivelse. En logfil indeholder en række loglinier, som hver registrerer een hændelse i brugerens interaktion med TeSS. De fleste hændelser er kommandoer, der udføres ved at klikke på en knap. Det giver f.eks. anledning til en loglinie, når brugeren folder en overskrift ud ved at klikke på "more detail". Det giver imidlertid også anledning til en loglinie, når TeSS udskriver en meddelelse til brugeren. Endelig er en del loglinier registreringer af, at brugeren påbegynder eller afslutter en opgave.

Eksempel på en linie i logfilen: En bruger har, efter at have markeret Xlib-manualens afsnit 3.2 i "Blocks Selected for Query", klikket på knappen "more detail". Det resulterer i følgende loglinie:

```
732554411:807148/0:0/S/more/0303020000000000$
```

Denne loglinie har samme format som alle andre loglinier:

- `732554411:807148` angiver, hvornår hændelsen er indtruffet (i sekunder og mikrosekunder siden 1. januar 1970).
- `0:0` er uden betydning i dette tilfælde; men angiver for nogle kommandoer, hvor lang tid TeSS har været om at udføre dem.

- `S` angiver, at den udførte kommando hører til i "Blocks Selected for Query". `more` angiver kommandoen.
- `0303020000000000` blokid'et for afsnit 3.2 i Xlib-manualen og angiver, hvilken overskrift der foldes ud.

Der er enkelte hændelser, vi ikke fandt en tilfredsstillende måde at logge på. Vi ville gerne registrere, hvornår der ændres i indtastningsfelterne til angivelse af forespørgsler, hvornår der scrolles, og hvornår det indbyrdes forhold mellem størrelsen af kontrolvinduets fire delvinduer ændres. Problemet er det samme i alle tre tilfælde: Hændelserne har en udstrækning over tid - de består ikke blot af et klik. Editering af et indtastningsfelt består f.eks. så godt som altid af en sekvens af indtastninger. Vi fandt ikke en måde at samle disse hændelser til een registrering for hver gang de forekom, og det er uacceptabelt med f.eks. een loglinie for hvert tegn der tastes i indtastningsfelterne.

3.3 Forsøgsstyringsmodulet

I dette afsnit behandles det modul, der blev udviklet for at automatisere afviklingen af forsøget. Forsøgsstyringsmodulet er lavet som et ekstra modul i TeSS. Det indeholder funktionalitet, der ikke har med tekstsøgningen at gøre, men udelukkende knytter sig til forsøget. Design og implementation af modulet er udført efter TeSS var færdigt - da vi vidste at vi havde tid til det.

Et centralt udgangspunkt i tilrettelæggelsen af forsøget har været, at de studerende ville have forståelse for de særlige krav, forsøget introducerer i rapportopgaven. Hensigten med forsøgsstyringsmodulet er at støtte de studerende i at gennemføre forsøget i overensstemmelse med disse krav. Forsøgsstyringsmodulet automatiserer en række ting, som de studerende ellers gennem hele forsøget selv skulle huske og kontrollere. Vi tror, forsøgsstyringsmodulet både øger sandsynligheden for, at forsøgets krav opfyldes, og gør disse krav mindre forstyrrende i de studerendes løsning af opgaverne. Som beskrevet i afsnit 3.1 *Tilrettelæggelse af forsøg* er følgende krav væsentlige ved afviklingen af forsøget:

- Der må kun logges, hvis brugeren har givet tilladelse til det.
- Forsøgspersonerne identificerer sig overfor TeSS, så logdata kan henføres til en bestemt fordeling af konfigurationer over opgaverne.
- Forsøgspersonerne skal have een opgave ad gangen, så de loggede interaktioner med TeSS kan henføres til løsning af netop den opgave.
- Forsøgspersonerne skal bruge bestemte konfigurationer ved løsning af de enkelte opgaver.
- Forsøgspersonernes svar på de enkelte opgaver skal registreres i umiddelbar sammenhæng med opgaverne, så svarene ikke påvirkes af de efterfølgende opgaver.

3.3.1 Funktioner i forsøgsstyringsmodulet

Forsøgsstyringsmodulet omfatter fire hoveddele til varetagelse af henholdsvis login, opgaveafvikling, konfigurationsstyring og modtagelse af resultaterne på opgaverne. Disse fire funktioner behandles nærmere i det følgende.

Login
Brugeren identificerer sig overfor TeSS ved i forsøgsstyringsmodulet at indtaste sit TeSS-brugerid og TeSS-password. Modulet kontrollerer, om disse svarer til hinanden, og om brugeren har taget stilling til, hvorvidt der må logges. Hvis begge dele er tilfældet, får brugeren adgang til selve forsøgsstyringen, og dermed til TeSS. Vi overvejede at lade forsøgsstyringsmodulet spørge brugerne om de vil tillade logning. Vi undlod imidlertid dette, da vi ikke mener, det spørgsmål skal stilles af en maskine. Det skal stilles af - og besvares overfor - de ansvarlige for forsøget.

Opgaveafvikling
Opgaverne præsenteres en ad gangen for brugeren og er til rådighed, også mens brugeren har TeSS kørende. Brugeren meddeler, når en opgave er løst, og modulet præsenterer derefter næste opgave. Istedet for at afslutte en opgave kan brugeren vælge at afbryde forsøgsstyringsmodulet. I så fald præsenteres brugeren for samme opgave næste gang, forsøgsstyringsmodulet startes, og kan så genoptage løsningen af den.

Konfigurationsstyring
Forsøgsstyringsmodulet afgør for en given kombination af brugerid og opgavenummer, hvilken konfiguration brugeren skal have til rådighed ved løsning af opgaven. Modulet sørger for, at det kun er muligt for brugeren at starte TeSS med netop denne konfiguration. For de opgaver, der skal løses ved manuel søgning i papirkopier af manualerne, er det ikke muligt at starte TeSS - den eneste tilgængelige funktion er muligheden for at afslutte den igangværende opgave. Brugeren får besked om, at opgaven skal løses ved hjælp af papirkopier af manualerne; men derudover yder forsøgsstyringsmodulet ingen støtte ved løsningen. Når brugeren har løst opgaven manuelt, melder han den afsluttet.

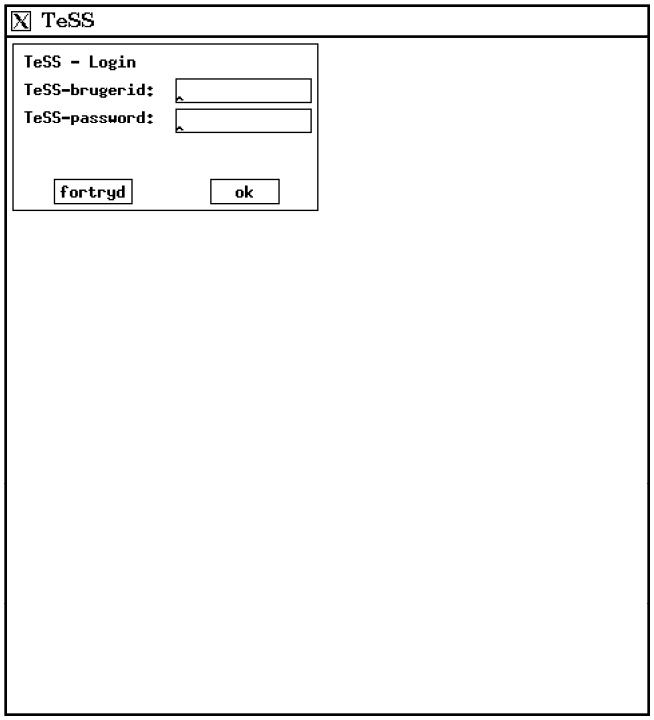
Modtagelse af resultat
Forsøgsstyringsmodulet giver mulighed for at indtaste svarene på de stillede opgaver direkte. Hvis en opgave afbrydes gemmes en eventuel påbegyndt besvarelse i en fil. Når opgaven senere genoptages kan brugeren fortsætte besvarelsen, hvor han slap. Han kan også rette i det, han måtte have skrevet, før han afbrød. Vi overvejede at indbygge mulighed for at få alle opgaver og besvarelser udskrevet, når den sidste opgave var besvaret. Det kunne være en støtte for de

studerende, når de, som en del af rapportopgaven, skulle beskrive deres oplevelse af og erfaringer med TeSS. Af tidsmæssige årsager blev dette imidlertid ikke implementeret.

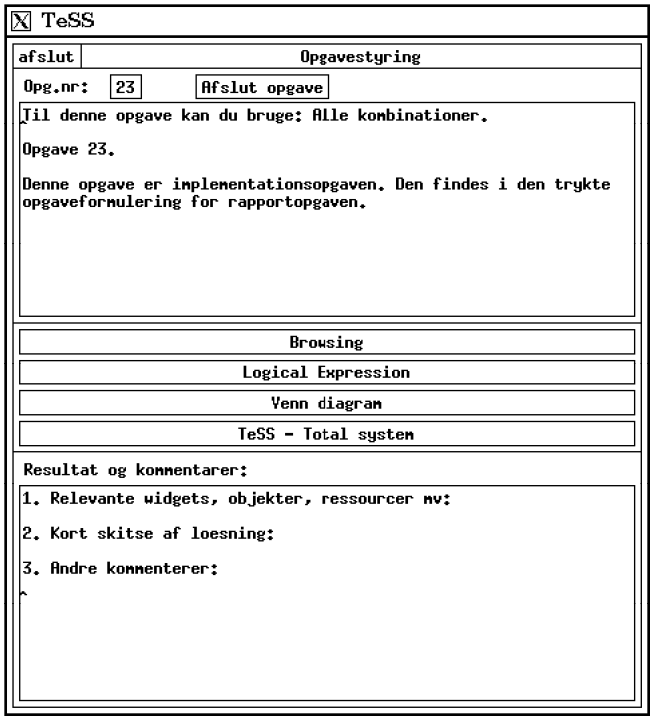
3.3.2 Brugergrænseflade for forsøgsstyringsmodulet

Vi har lavet brugergrænsefladen til forsøgsstyringsmodulet så simpel som muligt, uden at lægge så meget arbejde i at diskutere og udarbejde alternativer som vi gjorde med TeSS-brugergrænsefladen.

Forsøgsstyringsmodulet er, ligesom TeSS' kontrolvindue, holdt indenfor eet vindue. Ved start af systemet vil vinduet kun indeholde login-funktionen, hvor brugerid og password indtastes, se figur 3.3.2-1. Hvis de ikke stemmer overens, udskrives en meddelelse, og brugeren kan prøve igen. Hvis det ikke er registreret, hvorvidt brugeren har tilladt logning, udskrives en meddelelse om at brugeren skal sørge for at blive registreret, hvorefter programmet stopper.



Figur 3.3.2-1 Login-vinduet.



Figur 3.3.2-2 Forsøgsstyrings-vinduet.

Efter succesfuldt login skifter billedet til selve forsøgsstyringsmodulet, se figur 3.3.2-2. Forsøgs-styringsvinduet består af tre dele. Øverst vises opgavetekst og opgavenummer. Her findes også en knap, "Afslut opgave", der bruges til at færdigmelde en besvarelse og få næste opgave frem. Når den sidste opgave er påbegyndt er denne funktion ikke længere til rådighed. Dette er for at sikre at brugeren har TeSS til rådighed under hele rapportopgaveperioden, også efter at den sidste opgave er færdigmeldt.

I den midterste del af vinduet findes fire knapper, der alle starter TeSS. De repræsenterer de fire forskellige konfigurationer af TeSS: browsing, søgning ved logiske udtryk, søgning ved Venn-diagrammer og TeSS uden begrænsninger. Den 5. konfiguration, søgning i papirkopier af manualerne, er ikke repræsenteret her. Til hver opgave i de to første opgavesæt, initiel indlæring og informationssøgningsopgaverne, kan netop een af de fire konfigurationer vælges, undtagen hvis opgaven skal løses ved manuel søgning. Når de to første opgavesæt er besvaret, og det tredje, implementationsopgaverne, begynder, er alle fire konfigurationer af TeSS til rådighed.

Nederst i vinduet er feltet, hvor svar og kommentarer indtastes. I dette felt vil der ved start på en opgave stå en skabelon for svaret, i form af en række standard-punkter der kan besvares for hver opgave. Hensigten med dette er at lette besvarelsen og at gøre svarene korte og derved reducere arbejdet med at evaluere svarene.

Når både forsøgsstyringsmodulet og en konfiguration af TeSS kører, er det hverken muligt at starte andre eksemplarer af TeSS eller at afslutte opgaver. Før det kan ske, må den aktuelle konfiguration af TeSS lukkes. Mens TeSS kører har brugeren imidlertid mulighed for at indtaste sin

besvarelse i resultatfeltet, og for at stoppe hele programmet.

3.3.3 Databasetabeller til forsøgsstyringsmodulet

Den relationsdatabase, der indgår i tekstdatabasen, er blevet udvidet med tre tabeller af hensyn til forsøgsstyringsmodulet.

Brugertabel:

brugerid
pwd
logtilladt
sidsteopg

Her findes for hver bruger oplysning om password, om tilladelse til logning, og om hvilken opgave brugeren er igang med at løse.

Opgstyringstabel:

brugerid
opg
konfiguration

Her findes oplysning om hvilke konfigurationer hver bruger skal benytte til de enkelte opgaver. Feltet konfiguration kan have følgende værdier:

0 - frit valg, dvs. konfiguration 1-4 er til rådighed
1 - browsing
2 - søgning ved logiske udtryk
3 - søgning ved Venn-diagram
4 - Hele TeSS, dvs. kombinationen af konfiguration 1-3
5 - manuel søgning i papirkopier af manualerne

Opgavetabel:

opg
opgtekst

I denne tabel findes selve opgaveteksterne.

4 Sammenfatning

Vi har designet og implementeret TeSS, et system til eksperimentel undersøgelse af brugergrænseflader til tekstsøgesystemer. TeSS er udviklet både til brug i forbindelse med brugergrænsefladeundervisningen på Datalogi 2 og som en del af et forskningsprojekt om fagfolks tekstsøgning. Der har således været to nært sammenknyttede intentioner i TeSS-projektet, en uddannelsesmæssig og en forskningsmæssig. Den uddannelsesmæssige intention har været et ønske om at give de studerende på Datalogi 2 en oplevelse af og nogle erfaringer med grafiske brugergrænseflader, såvel brug heraf som evaluering og videreudvikling. Den forskningsmæssige intention har været et ønske om under kontrollerede forhold at logge en større kreds af brugeres tekstsøgning under brug af et tekstsøgesystem med væsensforskellige brugergrænseflader.

4.1 Tekstsøgesystemet TeSS

TeSS giver mulighed for søgning i tre manualer for værktøjer til udvikling af X-applikationer med grafiske brugergrænseflader. TeSS selv er et eksempel på en sådan applikation; brugergrænsefladen er implementeret ved hjælp af Athena Widgets; tekstdatabasen er opbygget i Ingres og Unix-filsystemet; og selve programmet er skrevet i Beta og - i mindre omfang - C.

Der er to væsensforskellige søgeteknikker til rådighed i TeSS: browsing og søgning ved forespørgsler. Browsing-faciliteterne giver mulighed for at søge blot ved at bevæge sig rundt i en indholdsfortegnelse for teksterne. Herved får teksternes struktur en central plads i søgningerne, og det bliver muligt at søge uden først at formulere, hvad der søges efter. Søgning ved forespørgsler giver mulighed for at søge på basis af præcis de ord, brugeren føler beskriver det, der søges efter. Det er imidlertid ofte svært at formulere rammende og dækkende forespørgsler, så den proces kan med fordel støttes. I TeSS består en stor del af denne støtte i at kombinere søgning ved forespørgsler med brug af browsing-faciliteterne. Fremfundne tekster placeres således i den samlede tekstsamling ved at blive præsenteret som et udsnit af indholdsfortegnelsen; brugeren har også mulighed for at browse sig til et vist overblik over ordvalget i teksterne, før en forespørgsel formuleres.

TeSS' brugergrænseflade består af et kontrolvindue og en række Text Viewer'e. Alle

søgefaciliteter findes i kontrolvinduet; en Text Viewer kan kun præsentere tekst. Kontrolvinduet er opdelt i fire delvinduer, hvis indbyrdes størrelsesforhold umiddelbart kan ændres. Det første delvindue, "All Blocks in TeSS", udgør en indholdsfortegnelse for de tekster, der er til rådighed i TeSS. Det andet delvindue, "Blocks Selected for Query", er en oversigt over de tekstblokke, der for øjeblikket er udvalgt til søgning. Det tredje delvindue, "Formulation of Queries", giver mulighed for at søge ved hjælp af forespørgsler. Det fjerde delvindue, "Blocks Matching Query", er en liste over de tekstblokke der opfylder den seneste forespørgsel.

De to første og det fjerde delvindue viser en liste med hele eller dele af indholdsfortegnelsen for teksterne i TeSS. Grundlaget for disse listefelter er en opdeling af teksterne i en træstruktur, der beskriver, hvordan manualerne udgøres af kapitler, kapitlerne af afsnit osv. Træstrukturen opdeler således manualerne i tekstblokke, der hver består af en overskrift og den tilhørende tekst. Listefelterne indeholder kun overskrifter, men giver adgang til at åbne tekstvinduer med den tilhørende tekst. Listefelterne giver endvidere mulighed for, at overskrifter foldes ud og foldes sammen efter behov, ligesom der kan foretages enkle søgninger i overskrifterne.

Søgning efter ord i teksterne, fremfor blot i overskrifterne, sker i det tredje delvindue. En søgning angives ved en forespørgsel, der specificerer den kombination af ord, en tekst skal indeholde for at blive fundet frem. Forespørgsler kan angives ved logiske udtryk, hvor kombinationen af søgeord specificeres ved parenteser og boolske operatorer. Denne måde at angive forespørgsler på genfindes i utallige andre søgesystemer. Forespørgsler kan også angives ud fra en grafisk afbildning af et Venn diagram. Her specificeres kombinationen af søgeord dels ved søgeordenes placering i Venn diagrammets grundmængder, dels ved udpegning af den af Venn diagrammets delmængder, der ønskes fremfundet.

Tekster fremfindes ved kald til den underliggende tekstdatabase. I tekstdatabasen er selve teksterne lagret i filer under Unix, mens de datastrukturer, der muliggør effektiv søgning, findes i en relationsdatabase. Denne relationsdatabase indeholder tre tabeller. En tabel fastlægger træstrukturen og knytter dermed de enkelte tekstfiler sammen i tekstblokke. En anden tabel, genereret ved invertering af manualteksterne, indeholder alle de søgbare ord og referencer til, hvor de forekommer. Endelig findes en tabel, der bruges under inverteringen og ved parsning af forespørgsler; denne tabel indeholder de ord, der ikke er søgbare.

Bindeleddet mellem brugergrænsefladen og tekstdatabasen er programmet. Objektmodellen for programmet er udviklet ud fra et næsten færdigt brugergrænseflade-design og er således en repræsentation af brugergrænsefladens objekter og funktioner i en objektorienteret model for programmet. Langt hovedparten af analyse- og designarbejdet i udviklingen af TeSS har udspillet sig i forbindelse med specifikationen af brugergrænsefladen og dens funktionalitet. Tekstdatabasen og specielt objektmodellen har derefter været ret givne.

4.2 Forsøget

Forsøget er tilrettelagt ud fra et generelt og et specifikt sigte. Generelt ønsker vi at tilvejebringe empirisk primærmateriale til studier af menneske-maskine interaktion. Der skal således indsamles detaljerede data om brugernes interaktion med TeSS. Det giver mulighed for, at materialet kan bruges i forskelligartede studier, som vi eller andre - efter aftale med os - ønsker at gennemføre. Specifikt ønsker vi at undersøge og sammenligne, hvorledes teknikker til browsing og søgning ved forespørgsler bruges i og er egnet til forskellige søgninger. Det skal endvidere undersøges, om de to typer teknikker bruges i forskellige dele af en søgning. TeSS omfatter een teknik til browsing og to teknikker til søgning ved forespørgsler. Samlet omfatter forsøget fem konfigurationer: browsing, boolsk søgning ved logiske udtryk, boolsk søgning ved Venn diagrammer, kombinationen af de tre foregående og manuel søgning i papirkopier af manualerne.

TeSS er omdrejningspunktet for rapportopgaven om brugergrænseflader på Datalogi 2 i foråret 1993. Forsøget gennemføres som en integreret del af denne rapportopgave. De studerende deltager i forsøget ved at løse rapportopgaven og give tilladelse til, at deres interaktion med TeSS logges. Studerende, der ikke ønsker deres interaktion logget, deltager ikke i forsøget; men de arbejder, i løsningen af rapportopgaven, med de samme opgaver.

Hver studerende skal gennemarbejde fire opgavesæt. Det første opgavesæt, der er udformet som et tænke-højt forsøg, skal gøre de studerende fortrolige med TeSS. Det andet opgavesæt er en række informationssøgningsopgaver. De skal løses i samme rækkefølge af alle studerende; men det varierer, hvilken konfiguration den enkelte studerende har til rådighed under løsningen af de enkelte opgaver. Det tredje opgavesæt består i, at de studerende benytter TeSS under løsning af tre implementationsopgaver. Her undersøges brugen af TeSS i en sammenhæng tæt på den tiltænkte, praktiske brugssituation: i samspil med udvikling af programmer. Det fjerde opgavesæt er et spørgeskema til indhentning af visse oplysninger om de studerende og deres subjektive vurdering af TeSS.

Udover spørgeskemaet består dataindsamlingen fra forsøget i, at de studerendes interaktion med TeSS under løsningen af de tre første opgavesæt logges. Logningen omfatter registrering af, hvilken af forsøgets opgaver, der arbejdes med, og registrering af alle udførte kommandoer, med angivelse af hvor og hvornår de indtræffer. Udover den detaljerede logning af de opgaver, der løses ved brug af de forskellige versioner af TeSS, foretages en grovkornet logning af de opgaver, der løses ved manuel søgning i papirkopier af manualerne.

For at gøre det så let som muligt for de studerende at arbejde med opgaverne indenfor de af forsøget satte rammer har vi implementeret et forsøgsstyringsmodul udenom selve TeSS. Dette modul varetager administrationen af, hvilken konfiguration hver enkelt bruger har til rådighed under løsningen af de enkelte opgaver. Forsøgsstyringsmodulet præsenterer endvidere de opgaver, der skal løses. Endelig er det gennem forsøgsstyringsmodulet, der logges oplysninger om, hvilken

opgave brugeren er igang med, og hvilket resultat han eller hun når til.

TeSS blev, til trods for en meget stram tidsplan, færdig til rapportopgaven på Datalogi 2. Rapportopgaveperioden forløb uden længerevarende perioder med driftproblemer, og de dat2 studerende syntes at kunne løse rapportopgaven ved hjælp af TeSS. En mere udførlig beskrivelse af status for TeSS findes i bilag G.

Referencer

Andersen, K. H., P. Davidsen, M. Foldbjerg, P. Göttsche, M. Hertzum, J. Jensen, L. G. Jensen, K. Lund, M. Rehn & P. Rungø (1992): *Undersøgelse af værktøjer til opbygning af tekstsøgesystemer*. Studenter-rapport 92-4-12. DIKU, København.

Ashford, J. H. (1986): Integrating text and non-text: Why is it important?. I R. Kimberley (red.): *Integrating text with non-text: A picture is worth 1k words. Proceedings of the Institute of Information Scientists Text Retrieval '85 Conference*. Taylor Graham, London. p3-20.

Basch, R. (1989): The seven deadly sins of full-text searching. *Database* **12**(4), 15-23.

Bass, L. & J. Coutaz (1991): *Developing Software for the User Interface*. The SEI Series in Software Engineering. Addison-Wesley, Reading, Massachusetts.

Bates, M. J. (1990): Where should the person stop and the information search interface start? *Information Processing & Management* 26(5), 575-591.

Belkin, N. J. & W. B. Croft (1987): Retrieval Techniques. *Annual Review of Information Science and Technology (ARIST)* **22**, 109-145.

Beta (1992a), *The Mjølner BETA System BETA COMPILER Reference Manual.* Mjølner Informatics Report MIA 90-02(1.0) november 1992.

Beta (1992b), *The Mjølner BETA System Using on UNIX Reference Manual*. Mjølner Informatics Report MIA 90-04(1.0) november 1992.

Beta (1992c), *The Mjølner BETA System Basic Libraries Reference Manual*. Mjølner Informatics Report MIA 90-08(1.0) november 1992.

Beta (1992d), *The Mjølner BETA System X Window System Libraries Reference Manual*. Mjølner Informatics Report MIA 91-16(1.0) november 1992.

Beyer, P., P. Carstensen, A. H. Jørgensen, R. Molich & F. H. Pedersen (1986): *Brugervenlige edb-systemer*. Teknisk Forlag, København.

Brooks, F. P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *Computer* 20(4), 10-19.

Campagnoni, F. R. & K. Ehrlich (1989): Information Retrieval Using a Hypertext-Based Help System. *ACM Transactions on Information Systems* **7**(3), 271-291.

Codd, E. F. (1970): A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM* 16(6), 377-387.

Conklin, J. (1987): Hypertext: An Introduction and Survey. *IEEE Computer* **2**(9), 17-41.

Cooper, W.S. (1988): Getting Beyond Boole. *Information and Management* **24**(3), 243-248

Faloutsos, C. (1985): Access Methods for Text. *Computing Surveys* **17**(1), 49-74.

Fox, C. (1989): A Stop List for General Text. *ACM SIGIR forum* **24**(1-2) (efterår 1989/vinter 1990), 19-35.

Frøkjær, E. & M. Hessellund (red.) (1993): *Datalogi 2 Kursusbog 1992/93. Bind 3: Brugergrænsefladedesign*. DIKU, København.

Frøkjær, E. (1993): *Eksperimentel Undersøgelse af Brugergrænseflader i Tekstsøgesystemet TeSS, Opgaveformulering for den karaktergivende rapportopgave K2, Datalogi 2, DIKU, foråret 1993*. DIKU, København, 6. marts 1993.

Girill, T. R. & C. H. Luk (1992): Hierarchical search support for hypertext on-line documentation. *International Journal of Man-Machine Studies* **36**, 571-585.

Hauptmann, A. G. & B. F. Green (1983): A comparison of menu-selection and natural-language computer programs. *Behaviour and Information Technology* **2**(2), 163-178.

Hertzum, M. & H. Søes (1992): *Fuldtekstsøgesystemer til Fagfolk.* DIKU-rapport 92/10. DIKU, København.

Hertzum, M., H. Søes & E. Frøkjær (1993): Information Retrieval Systems for Professionals: A Case Study of Computer Supported Legal Research. Afsendt til publicering.

Ingres (1991a): *INGRES/SQL Reference Manual for the UNIX and VMS Operating Systems*. Release 6.4. Ingres Corporation, Alameda, CA.

Ingres (1991b): *INGRES Database Administrator's Guide for the UNIX Operating System.* Release 6.4. Ingres Corporation, Alameda, CA.

Ingres (1991c): *Ingres/ESQL Companion Guide for C for the Unix Operating System*. Release 6.4. Ingres Corporation, Alameda, CA.

Kernighan, B W. & D.M. Ritchie (1988): *The C Programming Language*. 2. udgave. Prentice Hall, Englewood Cliffs, NJ.

Mandelkern, D. (1993): GUIs the next generation. Introduction. *Communications of the ACM* 36(4), 36-39. Introduktionen til en special-sektion om næste generation af grafiske brugergrænseflader.

McAlpine, G. & P. Ingwersen (1989): Integrated Information Retrieval in a Knowledge Worker Support System. I N. Belkin & C. J. van Rijsbergen (red.): *Research and Development in Information Retrieval. Proceedings of the 12th Annual International ACM SIGIR Conference*. ACM, New York.

Michard, A. (1982): Graphical presentation of boolean expressions in a database query language: design notes and an ergonomic evaluation. *Behaviour and Information Technology* **1**(3), 279-288.

Myers, B. A. (1989): User-Interface Tools: Introduction and Survey. *IEEE Software*, 15-27.

Nielsen, J. (1990): *Hypertext and Hypermedia*. Academic Press, San Diego, CA.

Radecki, T. (1988): Trends in Research on Information Retrieval - the Potential for Improvements in Conventional Boolean Retrieval Systems. *Information and Management* **24**(3), 219-227

Rasmussen, J. (1992). *Cognitive Engineering Approaches to the Design of Information Systems*. Inviteret indlæg på ACM SIGIR'92, 15th International Conference on Research and Development in Information Retrieval (København, 21.-24. Juni, 1992). Biblioteksskolen, København.

van Rijsbergen, C. J. (1979): *Information retrieval*. 2. udgave. Butterworth, London.

Salton, G. & M. J. McGill (1983): *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

Shneiderman, B. (1992): *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 2. udgave. Addison-Wesley, Reading, Massachusetts.

Smith, S. W. (1976): Venn Diagramming for On-Line Searching. *Special Libraries* **67**(11), 510-517.

Thompson, R. H. & W. B. Croft (1989): Support for browsing in an intelligent text retrieval system. *International Journal of Man-Machine Studies* **30**(6), 639-668.

Whiteside, J., D. Wixon & S. Jones (1988): User Performance with Command, Menu, and Iconic Interfaces. I H. R. Hartson & D. Hix (red.): *Advances in Human-Computer Interaction*, volume 2. Ablex, Norwood, NJ. p287-315.