Psyke & Logos, 1984, *5*, 302-327

BRUGERVENLIGE EDB-SYSTEMER: EN INTRODUKTION

Anker Helms Jørgensen
I/S Datacentralen af 1959

Edb-teknologien trænger frem overalt. Manuelle redskaber og færdigheder erstattes af formaliserede systemer – som også er arbejdsredskaber. De skal derfor også være brugervenlige. Dette er langt fra tilfældet i dag.

Artiklen opridser først nogle af årsagerne hertil. Dernæst omtales tre forskningsprojekter indenfor brugervenlighed. Det første handler om navngivning af kommandoer; det viser, at navngivning er en utrolig kreativ proces – men alligevel med en underliggende systematik.

Det andet projekt tager hul på området design-psykologi. En række vigtige elementer i psykologien i systemdesign lægges frem på grundlag af interviews med fem systemdesignere. Det er vigtigt at forstå disse elementer, dels fordi systemdesignere næsten enerådende bestemmer udformningen af systemernes brugergrænseflader, og dels fordi de er forbrugere af forskningens resultater.

Det tredie projekt er et traditionelt laboratorieeksperiment om strukturen i den opgave, som brugeren løser. En "objekt-orienteret" struktur sammenlignedes med en "funktions-orienteret". Den første viste sig klart at være den letteste at lære. Dette eksperiment førte til, at jeg som systemdesigner overværede brugernes møde med systemet. Det var utrolig overraskende, hvor stor en kløft der var mellem brugernes forståelse og min egen. Dette leder til artiklens sidste del, nemlig to veje til mere brugervenlige systemer: at skabe "objektiv" viden eller bygge på personlige erfaringer.

Indledning

Scene 1

Udstilling af kontorinformationssystemer. En salgskonsulent beretter engageret, men nøgternt om et edb-systems fortræffeligheder. Ved tastaturet sidder en sekretær, der smilende og indforstået følger salgskonsulentens mindste vink: fodnoter og alternerende sidetal flimrer hen over skærmen. Rundt omkring meddeler store salgsplakater lakonisk:

> "Funktionerne står på skærmen. De skal bare bruge de anviste taster, så kommer der flere oplysninger, der hjælper Dem på vej – og de efterfølgende skridt er ligeså nemme."

Scene 2

Skriverstuen i et større firma. En midaldrende sekretær har netop læst korrektur på et manuskript ved skærmen og vil nu skrive en kopi på papir:

P;*;L34D

Systemet svarer prompte:

INVALID OPERATION

Hun udbryder "Hvorfor?? – det er da rigtigt nok!". Hun ser forvirret ud, tøver et øjeblik og prøver så igen:

P;*;L34D

Ligeså prompte som før svarer systemet:

INVALID OPERATION

Hun ser nu ret desorienteret ud – men styrer pludselig hen til skriveren – og konstaterer, både opgivende og triumferende: "Der er ikke mere papir i skriveren!"

Scene 3

En skoleklasse har time i datalære. Læreren har lavet et program, som eleverne bruger til at lave en mini-udgave af KTAS's "Telefonnøglen". Eleverne skriver efter tur oplysninger om "abonnenter" af fra kartotekskort. Men efter få kort mister eleverne tålmodigheden og begynder at kaste med blyanter. Systemet var intolerant, det tillod ikke eleverne at rette fejl, som de havde begået i tidligere linier.

Mange vil nikke genkendende til disse scener (som er autentiske). Denne artikel handler om det psykologiske samspil mellem mennesker og edb-systemer. Om baggrunden for, at dagens systemer lader så meget tilbage at ønske med hensyn til brugervenlighed, og hvordan denne tingens tilstand kan ændres.

Baggrund

Befolkningen møder typisk datamaskinerne i videospil – og de er jo underholdende! Dette afføder ofte spørgsmålet "Hvad kan vi lære af edb-spil for at gøre edb-systemerne lettere at anvende – og måske mere underholdende?"

Spørgsmålet kan synes relevant, men det overser een ting: At edb-systemer er arbejdsredskaber – hvortil der stilles andre krav, end der stilles til spil. Ar-

bejdsredskaber skal være lette at anvende, og de skal være nyttige – i modsætning til spil, der ifølge sagens natur skal indeholde en vis grad af udfordring, intellektuelt eller motorisk.

I modsætning hertil anvendes arbejdsredskaber til at løse en bestemt opgave, fx at rette et manuskript, at ajourføre en konto eller at finde en bestemt passage i et cirkulære. Løsning af opgaven er det primære – målet – mens arbejdsredskabet – midlet – er det sekundære. Een og samme opgave kan løses ved hjælp af mange forskellige redskaber; valget vil bl.a. afhænge af hvor let det er at anvende redskabet.

Med introduktionen af redskabsbegrebet er vi ved et velkendt område: *ergonomien.* Den består i at tilpasse arbejdsredskaber til de mennesker, der anvender dem. Eksempler fra ergonomien er, at lændestøtten på en kontorstol skal kunne indstilles, og at billedet på en edb-skærm skal stå roligt.

Men dette er ikke nok. Man skal også kunne løse opgaven ved hjælp af edb-systemet. Dette indebærer bl.a., at man skal være i stand til at forstå teksten på billedet, kunne handle derudfra, komme på sporet igen efter en fejl, osv.

Hermed er vi ved den *kognitive ergonomi,* dvs. at tilpasse systemerne til brugeren på det kognitive område. Med andre ord at gøre systemerne lette at bruge, at gøre dem *brugervenlige.*

Hvad ligger der i det? Systemerne skal være nyttige, dvs. de skal kunne udføre reelt arbejde; de skal være lette at lære; det skal være let at huske hvordan man anvender dem; og endelig skal de være behagelige at arbejde med.

Men hvad ligger der så i disse ting? – fx let at huske? Jo, systemet kan fx baseres på begreber, der er velkendte for brugeren. Hvad ligger der så i det? – og hvordan finder man ud af det?

Rækken af spørgsmål kan fortsættes i det uendelige. Det er karakteristisk for udvikling af brugervenlige systemer, at der ikke gives enkle svar. Samspillet mellem bruger og system er så komplekst, at det knap nok kan beskrives fyldestgørende og slet ikke parametriseres. Mere herom i næste afsnit.

Der er imidlertid en anden vej til konkretisering af begrebet brugervenlighed, nemlig at "måle" symptomerne på et systems brugervenlighed. Der er tre hovedelementer i begrebet:

- Brugerens tilfredshed.
- Tid, fx til at lære systemet og løse opgaver.
- Fejl, fx art, hyppighed og konsekvenser.

Brugerens tilfredshed kan være afgørende for anvendelsen af et system, selv om systemet både er nyttigt, let at lære og let at huske. Tilfredsheden afhænger meget af personlige træk, så det er meningsløst at tale om eet bedste system. Mennesker vælger for eksempel også forskellige bilmærker, i øvrigt ofte med et stærkt islæt af følelsesladede argumenter.

Men samtidig findes der givetvis universelle underliggende egenskaber, som er fælles for flertallet af brugere. Og visse af disse kommer til udtryk i indlæringstiden og antallet af fejl.

Viden om brugervenlighed

De interaktive systemer har eksisteret i mere end 20 år, og lige så længe er der givet hundredevis af svar på, hvad der kendetegner et brugervenligt system. Svarene er givet som retningslinier. Nogle er meget generelle:

> "Instruktiv vejledning skal være tilgængelig for brugeren under brug af systemet."

Andre er mere specifikke:

> "Objektet skal stå før verbet i en kommando, fx 'erindringsliste, tryk', idet brugere identificerer et skærmbillede med dets objekt."

Oprindelsen af de fleste af disse retningslinier må betegnes som folkloristisk. Det er oftest erfaringer fra et specifikt system, der generaliseres til andre sammenhænge – hvad der er meget tvivlsomt. Og selv de af dem, der er af mere universiel gyldighed, er meget vanskelige at fortolke i en given sammenhæng. Endelig er en del af dem tvivlsomme, fx strider den ovenfor mod den fundamentale rækkefølge 'verbum-objekt' i sproget.

I litteraturen findes flere oversigter over retningslinier, fx (Smith og Aucella, 1983). Heri beskrives 580 retningslinier for udformning af brugergrænseflader i administrative edb-systemer. Hvem kan rumme alle disse? Hvordan foretager man afvejning i tilfælde af modstrid? Dette er overladt til systemdesignerne.

Et andet eksempel – fra ovennævnte oversigt – viser retningsliniers svaghed.

> "Funktionstaster skal kunne aktiveres ved tryk på een tast, og funktionen skal være den samme ved gentagen aktivering."

(For den ikke-edb-kyndige læser: En funktionstast er en tast med en forud fastlagt funktion fx "skriv en papirkopi af dette skærmbillede".)

Første del lyder rimelig, da det både tager længere tid at betjene to taster end een og giver anledning til flere fejlslag. Anden del lyder umiddelbar også ganske selvfølgelig – og er i øvrigt i overensstemmelse med et meget generelt krav om ensartethed (tasten mærket "a" giver altid bogstavet "a", og ikke nogle gange et "b"). Ikke desto mindre er kravet overtrådt i et af de mest brugervenlige systemer i dag: Xerox Star. Når man i dette system arbejder med en tekst på skærmen, vil eet tryk på funktionstasten på "musen" (der styrer markøren) udvælge det bogstav, som markøren peger på. Her "s" i "skal" vist ved understregning:

> ". . . og funktionen s̲kal være . . ."

Et tryk til udvælger hele ordet:

> ". . . og funktionen skal være . . ."

Endnu et tryk vælger den sætning ordet "skal" står i, og endelig udvælges ved det fjerde tryk – gæt selv – det afsnit, som sætningen står i.

Valget bygger på en iboende hierarkisk struktur i tekster: Bogstav, ord, sætning, afsnit. Dette er velkendt for brugeren, og er derfor let at lære og huske – selv om det er i modstrid med retningslinien.

Proces og produkt

Der er en anden svaghed ved retningslinier, nemlig at de meget sjældent er operationelle, dvs ledsaget af anvisninger på en arbejdsmetode, der leder til et system, der opfylder kravet udtrykt i retningslinien.

En del retningslinier er dog så konkrete, at det ikke er nødvendigt. For eksempel fremgår det umiddelbart, om en funktionstast aktiveres ved eet eller to tryk.

Men hvordan laver man et system, ". . . hvor instruktiv vejledning altid er tilgængelig"? Systemdesignerne kan gøre sig mange velmente tanker om dette undervejs i systemudviklingen, men om kravet er opfyldt eller ej kan kun afgøres ved praktisk brug.

Den vigtige skelnen her er mellem proces og produkt: Processen, der forløber over et vist tidsrum, afsluttet med produktet – edb-systemet:

Næsten alle retningslinier udtaler sig om produktet, ikke om processen. Så der er ikke meget hjælp at hente for en systemdesigner, der gerne vil arbejde målrettet imod et system, der opfylder kravene til brugervenlighed. Systemdesigneren er overladt til at trække på sin erfaring og sunde fornuft.

Men der er en ændring på vej. På konferencen "Human Factors in Computing Systems" (CHI '83), som blev afholdt i Boston i december 1983, var der flere indlæg, der påpegede vigtigheden af at focusere på processen i stedet for på produktet. En systemudviklingsproces, der er rettet mod at gøre systemet brugervenligt, vil lede til et mere brugervenligt system.

Dette indebærer imidlertid, at systemdesignerne som professionelle kommer i focus. Hvilke mål har de i systemudviklingen? Hvilke rammer arbejder de under? Hvilke informationskilder trækker de på? Hvordan opfatter de brugerne og opgaverne? Det bliver nu vigtigt at få klarlagt nogle svar på disse spørgsmål. Dette har givet anledning til et nyt område, benævnt design-psykologi. Det behandles i et senere afsnit.

Baggrunden for denne grundlæggende skelnen mellem proces og produkt karakteriseredes i øvrigt på Boston-konferencen således af Gould og Lewis (1983):

> "Samspillet mellem mennesker og systemer er så komplekst og påvirket af så mange faktorer, der oven i købet vekselvirker, at selv systemdesignere med stor erfaring og indlevelsesevne simpelthen ikke kan forudsige brugervenligheden af et system."

Dette udtrykker en erkendelse af tingenes tilstand i dag: Man kan ikke tænke sig til brugervenligheden. Men hvad gør man så? Svaret er, at man skal lægge mere vægt på det empiriske. Dette beskrives i et senere afsnit.

På Boston-konferencen kom der i øvrigt en anden meget præcis karakterisering af tingenes tilstand:

> "Hidtil har strategien været at udvikle perfekte brugervenlige systemer. En mere realistisk strategi er nok at forsøge at undgå katastrofer!"

Det er måske læseren bekendt, at der i en del år har eksisteret anerkendte systemudviklingsmetoder (fx Jacksons Struktureret Programmering og Yourdons Struktureret Analyse og Design). Om end værdien af dem kan diskuteres, er det et faktum at de har vundet stor udbredelse. Så man kan undre sig over, at brugervenligheden halter så langt bagefter?

En af forklaringerne er, at disse metoder beskæftiger sig med to ting: Funktionaliteten i et system, fx hvilke *funktioner,* der er nødvendige for at kunne fakturere, og *informationsgrundlaget,* fx hvilke oplysninger er nødvendige for at kunne fakturere? Metoderne beskæftiger sig slet ikke med hvordan funktionaliteten og informationen presenteres for brugeren. Det overlades til systemudviklerne.

Lad mig give et eksempel til at illustrere forskellen mellem information og den måde den presenteres på. For nogen tid siden henvendte jeg mig på Hovedbanegården i København for at få oplyst følgende: "Hvornår kører der i morgen tog fra Lyon til Geneve?" En meget konkret opgave til ekspedienten, der er uddannet til at besvare sådanne spørgsmål.

Ti minutter senere havde jeg fået opgivet to tidspunkter. Men jeg turde ikke stole på dem, for ekspedienten havde bladret meget rundt i den franske køreplan, været i tvivl og taget fejl flere gange. (Denne køreplan er "strækningsorienteret" ligesom den danske, dvs. togenes løb vises station for station på strækningen.)

Senere på dagen henvendte jeg mig i det franske turistbureau for selv at prøve at finde ud af det. Ekspedienten stak mig en køreplan, i øvrigt i et andet format end den på Hovedbanegården, og hun gik straks videre i sit arbejde. Så jeg tænkte, at det finder jeg aldrig ud af alene!

Jeg slog op et tilfældigt sted – dernæst et andet. Efter disse to opslag var jeg 100% sikker på, at jeg selv kunne finde 100% korrekt svar på mit spørgsmål! Køreplanen var nemlig "opgave-orienteret": Først skulle jeg finde Lyon (alfabetisk rækkefølge), og dernæst under Lyon finde Geneve (også alfabetisk). Voila: en liste over togene fra Lyon til Geneve – og kun dem. Der var i øvrigt en halv snes!

De to køreplaner indeholder de samme oplysninger, nemlig om toggangen i Frankrig. Men oplysningerne er struktureret forskelligt. Den "opgave-oriente-

rede" er beregnet til at besvare spørgsmål af typen "Hvornår går der tog fra A til B?", som givetvis er det hyppigst stillede spørgsmål. Og struktureringen af oplysningerne stemte så præcis overens med min opgave, at den alene var tilstrækkelig til at forklare brugen af oplysningerne – endda med fuld sikkerhed. Selvsagt har denne strukturering også sine svage sider; fx giver den ikke svar på, om toget standser undervejs.

Parallellen til systemudviklingsmetoderne er, at disse metoder sikrer, at alle oplysninger om toggangen i Frankrig er med, at de er ajour, osv. Men hvordan oplysningerne skal fremtræde overfor brugeren, beskæftiger de sig ikke med.

Teknologi, sund fornuft, teori og metoder

Det er meget karakteristisk for edb-området, at teknologien driver udviklingen. Blot indenfor brugergrænseflader kan nævnes en række nye interaktionsteknikker, der alle er kommet på markedet i de seneste år: "mus", farveskærme, berøringsfølsomme skærme, og tale ind- og uddata.

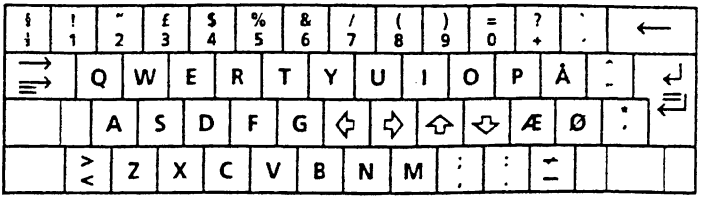
Men nye teknikker giver nye problemer: Farveblinde brugere vil have besvær med farveskærme; en "lyttende" datamat kan ikke forstå "sin" brugers talte instruktioner, hvis han eller hun er forkølet!

Tiden må vise, hvordan brugerne arbejder sammen med de nye teknikker. Systemdesignerne må efter bedste evne overføre deres viden fra "gamle" teknikker til de nye, suppleret med intuition og sund fornuft. Da vil der efterhånden udvikles en håndværksmæssig tradition, støttet på erfaring, typisk udtrykt som tommelfingerregler.

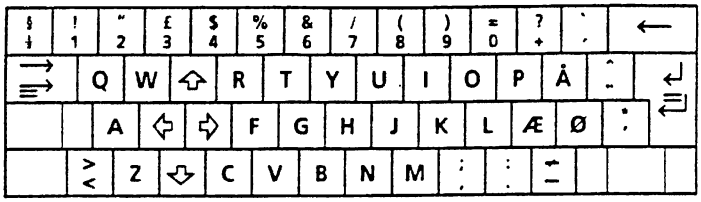
Længst tilbage i udviklingen står metoder og teorier. Først nu, mere end 20 år efter fremkomsten af interaktive systemer, er der tilløb til et metodisk fundament – med den iterative tilgang til systemudvikling: anvendelse af prototyper. Indenfor teorien er der sket et markant skridt med udgivelsen af bogen "The Psychology of Human-Computer Interaction" (Card et al., 1983). Den er baseret på en halv snes års arbejde ved Xerox Palo Alto Research Center, et af foregangslaboratorierne indenfor feltet.

Man kan imidlertid filosofere over, hvor meget teori der er nødvendig for at udvikle brugervenlige systemer? Det følgende eksempel illustrerer spørgsmålet.

I et tekstbehandlingssystem anvendes fire taster på række til at styre markøren (med samtidig nedtrykning af en kontrol-tast):



I et andet system anvendes også fire taster:



Er der nogen tvivl om, hvilket arrangement der er lettest at huske?

Her er teori næppe nødvendigt på grund af den klare overensstemmelse med den iboende rumlige struktur i opgaven.

Men andre spørgsmål kan ikke klares så let: Skal objektet eller verbet stå først i en kommando? Hvilke forkortelsesformer leder til sikrest gendannelse af ord? Hvilke strategier anvender brugere i menu-styrede systemer? Hvordan skal indbygget "hjælp" udformes? Hvilke dele af systemerne skal kunne individualiseres?

Række af spørgsmål kan fortsættes i det uendelige – der er nok at tage fat på for forskningen. I et senere kapitel gives eksempler på forskningen indenfor den kognitive ergonomi.

Disse spørgsmål må løses i samarbejde mellem de involverede parter: Brugerne, systemudviklerne og forskerne. Dette er ikke let, ikke mindst fordi så forskellige fagområder som psykologi, lingvistik, ergonomi, datalogi og ingeniørvidenskab er involveret.

Edb-verdenen har været længe om at akceptere, at edb er andet end teknik, logik og formalisme. Og omvendt har de humanistiske videnskaber været længe om at indse, at anvendelse af teknik udmærket kan indeholde interressante teoretiske problemstillinger. Men i de seneste år er der sket en hidtil uset udvikling indenfor feltet i retning af at finde et ståsted, i takt med opblødningen af faggrænserne.

Forskning i brugervenlighed

I dette afsnit beskrives tre eksempler på forskningsprojekter indenfor brugervenlighed. De tre projekter er fra mit licentiatarbejde (Jørgensen, 1983a), og de er lavet i samarbejde med en gruppe af kognitive psykologer på Applied Psychology Unit i Cambridge, England.

Navngivning af kommandoer

Ved brug af interaktive systemer er der næsten altid et sprogligt "lag" mellem brugeren og systemet – i modsætning til for eksempel en bil, hvor man styrer, skifter gear, og bremser ved direkte manipulation. Kun i visse nyere systemer

har man en ikke-verbal kommunikationsform, for eksempel at pege på objekter.

Dette sproglige "lag" har erfaringsmæssigt betydning for brugervenligheden, blandt andet fordi systemdesignernes edb-specialistsprog her kommer direkte til udtryk for brugerne. Der er allerede foretaget en del undersøgelser af dette spørgsmål. De falder i to grupper:

Man har ladet brugere selv vælge navne på funktioner – de kender jo arbejdsopgaven bedst! Derefter har man undersøgt, om brugerne var hurtigere til at lære systemerne med deres egne navne, lavede færre fejl, osv. Resultaterne er ikke entydige; men meget tyder på, at brugerne ikke er gode til at designe navne. Disse undersøgelser viste i øvrigt, at der var en uhyre spredning i de valgte navne (Landauer et al, 1983).

Man har også foretaget systematiske undersøgelser af underliggende faktorer i navnenes struktur, for eksempel verbum-objekt form og modsvarende par af kommandoer (fx "åbne" og "lukke"). Her var resultaterne helt klare: Enhver form for underliggende struktur i navnene gør dem lettere at lære og huske.

Spørgsmålet er så, hvordan systemdesignere navngiver funktioner? Er de så "teknik-orienterede", som brugerer ofte påstår? Er der ligeså stor spredning i deres navne som i brugernes? Er der nogen struktur i deres navne?

Disse spørgsmål blev undersøgt i det første projekt (Jørgensen et al, 1983b, c) ved at udsende et "interaktivt spørgeskema" til et antal systemdesignere. Skemaet udsendtes som en serie "før" og "efter" billeder på designernes skærmterminaler, svarende til billedet før og efter udførelsen af en funktion. Designerne skulle give et passende navn til transformationen fra "før" billedet til "efter" billedet – svarende til det navn, som brugeren skal angive. Skemaet blev besvaret af 110 systemdesignere, der hver navngav 12 funktioner. En af funktionerne er vist skematisk i figur 1.

"Før": Contact Agent Albert z7qip at Kennedy Airport "Efter": Contact Agent Albert at Kennedy Airport
Figur 1. En funktion fra "spørgeskemaet"

Systemdesignernes valg af navne er vist i figur 2.

20 delete	ade	disc	purge
12 remove	arrange	drop-element	regap
11 del	clean	dropelem	removeelement
9 drop	cleanup	dropitem	ridwaste
4 discard	compress	ecode	select
4 ignore	decode3	edit	strip
2 elim	delelem	exclude	suppress
2 delel	delelement	f	unwant
2 elide	delelm	lose	xerase
2 erase	delete?	noise	
2 omit	delete-e	omit-elem	
2 tidy	delete-element	omitelmnt	
2 zap	delnois	pickelement	

Figur 2.

110 systemdesigneres valg af navne
for funktionen vist i figur 1.
Tallene er frekvenser.

Der er 48 forskellige navne! Dvs. cirka et nyt navn for hver anden designer. Hvori består forskellene mellem navnene? En del af spredningen ligger i det syntaktiske, for eksempel forkortelser: delelem, delelm, delelement, delete-e, delete-element; men også i hvor meget, der inddrages i navnene: delete og delete-element, og hvordan dette benævnes: delete-element og delnois. Bemærk, at ordet "element" var systemets betegnelse for "ordene" i teksten.

På trods af spredningen er der også fælles træk. For det første er praktisk taget alle navnene verbum-former, dvs. et verbum alene eller et verbum med det tilhørende objekt. Faktisk forekommer kun een ren objektform: noise. Dette mønster var karakteristisk for de andre funktioner i skemaet, der ligesom funktionen i figur 1 ændrede ved en tekstlinie: Navnene refererede til en handling.

Imidlertid var der også fire funktioner af en anden karakter i skemaet. De "skaffede" ny information på skærmen, fx. fyldte værdier i en tabel. Og i de funktioner var mønsteret ligeså tydeligt: Navnene refererede til den nye information på skærmen, dvs. objektformer.

Der er også en anden forskel mellem navnene, nemlig semantisk. For eksempel mellem de to oftest forekommende verber: delete og remove. Delete betyder at slette, dvs. objektet forsvinder, mens remove betyder at flytte objektet fra et sted til et andet. Dette illustreres ved to arketypiske sætninger:

Cykler fjernes uden ansvar

Tavlen må ikke slettes

At de to ord er semantisk forskellige, fremgår tydeligt, hvis de byttes om i de to sætninger!

Men de anvendte verber har alligevel et fælles semantisk præg, idet de alle på en eller anden måde beskriver det "at tage noget ud af sin sammenhæng". For eksempel betyder "purge" at rense ud, som det jævnligt foregår i Kreml, mens "elide" betyder, at et bogstav er stumt, dvs. forsvinder fra den skrevne til den talte form.

Det er vel næppe overraskende, at designerne ikke valgte navne som "create" eller "insert", dvs. det modsatte af funktionen. Men det overraskende er, at så mange sammenhænge er representeret. Og det frister til en parallel til brugernes situation under anvendelsen af systemerne: De skal altid angive eet navn for at få udført en funktion – endog ofte i een bestemt forkortelse (med undtagelse af enkelte systemer, der tillader forskellige forkortelser eller endog synonymer).

Analysen af hver enkelt designers valg af de 12 navne viste, at der ikke var meget systematik i navnene, hverken med hensyn til forkortelsesform, verbum-objekt-form eller lign. Men hertil skal siges, at designerne ikke havde lejlighed til at kigge tilbage under besvarelsen af "skemaet", så de var afskåret fra at anlægge en strategi, baseret på et indtryk af alle funktionerne.

Resultatet stemmer godt overens med undersøgelserne af brugernes navngivning med hensyn til syntaktiske og semantiske aspekter. Så på trods af at edb-folket sprogligt lever i en specialist-verden, synes det altså ikke nævneværdigt at have begrænset deres kreativitet i navngivning.

Designpsykologi

Navngivning af kommandoer er et eksempel på en af de mange aktiviteter i systemudvikling, hvor systemdesignerne er nøglepersoner. Deres opfattelse er bestemmende for udformningen af systemerne og deres brugervenlighed. Det andet projekts formål var at undersøge disse ting. Hvordan opfatter designerne opgaven, som brugerne skal løse? Hvordan opfatter de brugernes færdigheder og motivation? Hvilke informationskilder trækker de på? Hvilke mål går de efter?

Vi udvalgte tre interaktive systemer, der alle var beregnet til brugere uden særlige forudsætninger indenfor edb: et tekstbehandlingssystem, et generelt databehandlingssystem for mindre virksomheder, og et grafisk system. Vi anvendte selv systemerne for at sætte os ind i deres virkemåde. Derefter interviewede vi fem designere, der havde designet systemernes brugergrænseflader. Interviewene forløb ret frit, idet vi kun anvendte en liste over bemærkelsesværdige ting i systemerne som checkliste, ikke som styrende for forløbet.

Transkription af interviewene resulterede i cirka hundrede siders materiale, som analyseredes og klassificeredes. Analysen omfatter bla. en model for systemdesign, mangel på viden i design af brugergrænseflader, begrænsninger, feedback fra brugere og designernes syn på brugervenlighed, brugere og "Human Factors" (som er den amerikanske betegnelse for disciplinen ergonomi). I denne korte fremstilling, der er baseret på (Jørgensen et al, 1983d) og (Ham-

mond et al, 1983), gives udpluk fra analysen.

Der var en del forskel mellem designerne med hensyn til interesse for at lave brugervenlige systemer – men var alle blandt de mere interesserede. De erkendte alle at have behov for mere viden om hvad brugervenlighed er, og hvordan man udvikler brugervenlige systemer. En af dem udtalte:

> "I'd like to have something that would help me make up a set of fundamentals that I could use to construct a good model of the system before I ever start thinking about specific details. That's certainly the area where I'd like the most help."

De havde alle henvendt sig til ergonomer (Human Factors-folk) for at få råd og vejledning. I et tilfælde – design af et kommandosprog – lød svaret sådan:

> "We'll go and think about it, and perhaps run a few experiments."

Dette svar er ikke meget bevendt for en systemudvikler, der skal have svar her og nu. Set fra ergonomernes synspunkt kan svaret være rimeligt nok, for de har nu engang heller ikke viden om alt, men må skaffe sig den. En anden designer ville gerne have hjælp til design af et skærmbillede:

> "I asked the Human Factors representative how I could make this display panel look better. He made little nitty-gritty comments, like 'Don't use dashes here, use dots'."

Et sådan svar er selvsagt ikke befordrende for systemdesignernes tillid til ergonomerne, for det kunne de lige så godt have sagt selv. Så en af dem konkluderede:

> "We eventually came to the conclusion that not only are we not Human Factors experts, but no one is either. Nevertheless anyone claims to be an expert or at least will give you an expert opinion."

Spørgsmålet er nu, hvad designerne så gjorde – for systemerne skulle jo udvikles. De trak på mange informationskilder; en af de væsentligste var erfaring:

> "You tend to build upon the experience you take for granted, either your own experience or others."

Som et eksempel nævnes tilordning af funktioner til funktionstasterne:

> "It was obvious from our experience which functions should be assigned to which function keys!"

I dette tilfælde stammede erfaringen fra designet af en editor, beregnet til edb-folk (som er langt mere tolerante overfor bruger-u-venlige systemer). Og visse ting i brugergrænsefladen førte til vanskeligheder for brugerne, for ek-

sempel at overskrive linienumre med et-bogstavskommandoer: "d20067", hvor "d2" skrives oveni "000067". Det betyder slet to linier fra linie 67.

Hvor erfaringen slap op, trådte intuitionen til. I beslutningen om, hvordan man i tekstbehandlingssystemet bedst markerede en blok af tekst, der skulle slettes eller flyttes, havde designerne valget mellem følgende tekniske muligheder: omvendt baggrund, understregning, høj lysstyrke, og første og sidste bogstav blinkende:

> "We did the least obtrusive technique – having the first and last character blinking. But we have no idea whether that's right or wrong from a Human Factors point of view."

Designerne var i tvivl om mangt og meget undervejs. Som en af dem sagde, de havde "a good many discussions" – ofte om konkrete ting, som for eksempel om længden af breve. Spørgsmålet "Hvor langt er et brev?" er meget konkret, men ingen er i stand til at give et fyldestgørende svar! Så designerne måtte ofte støtte sig til deres intuition. En af dem karakteriserede det således:

> "We don't actually flip a coin, but we do have a lot of uncertainty on the decision we finally come up with. We pick what we think is best, not necessarily based on experimental evidence or anything like that."

I lyset af manglen på viden om udformning af brugergrænsefladen kom funktionaliteten oftest først – den er lettere at håndtere, mere substantiel. Designerne ville være sikre på, at der var tilstrækkelig funktionalitet i systemerne. I det følgende eksempel betegner "the copier pit" den situation, der opstod med nogle af de første fotokopimaskiner, hvor man ikke kunne kopiere bøger:

> "The point is that the copier pit is a disaster. You may only want to copy books five percent of the time, but when you want to, you want to. And so, even if we had some clumsy interfaces, we tried to make sure that we weren't prohibitive in doing things."

En anden designer udtrykker prioriteringen således:

> "We tended to look for the most powerful and sophisticated functions and tried to figure out how to make them simply presented to the novice."

Denne prioritering hænger givetvis sammen med den feedback, som systemdesignerne får fra brugerne om systemerne og deres funktionalitet:

> "I remember one customer saying: 'If you're going to make everything simple, then do it with your other products. We need the function on the document processor, we've got to do these things, for Heavens sake give us the capability to do it!"

Sådanne udtalelser kan ikke netop siges at medvirke til at motivere systemde-

signerne i retning af at udvikle mere brugervenlige systemer. Det er en ret generel erfaring, at også brugerne skal opdrages i at systemerne godt kan være lettere at bruge, og ikke som denne bruger tro, at 'let at bruge' står i modsætning til funktionalitet.

Men systemudviklerne havde også ganske klare opfattelser af brugerne og mulighederne for at tilfredsstille dem:

> "You never seem to be able to satisfy any large group of users anyway because they're too disparate."

Dette kan så tolkes på to måder: Enten at opgive det hele, eller at gøre systemerne så fleksible, at de kan opfylde flere brugeres behov. Ovenstående udtalelse er på et meget generelt niveau. Jo mere konkrete niveauer, desto klarere opfattelser havde systemudviklerne. Her er en mere konkret:

> ". . . you just get very frustrated by inadvertent deletes."

og en meget konkret vedrørende formatet for udskrivning af tal:

> "If people double 1.20 dollars the expect to get 2.40 dollars, not 2.4 dollars."

Så at udvikle systemdesignernes opfattelser af brugerne og deres måde at opfatte tingene på, er måske en mere farbar vej til mere brugervenlige systemer end at producere retningslinier om egenskaber ved systemerne. Det har i hvert tilfælde den fordel, at systemdesignerne kan relatere det til deres egne erfaringer som brugere af edb-systemer.

Men selv om systemudviklerne gerne vil arbejde for brugervenligheden, er der hindringer. I mange tilfælde har et nyt system en forgænger, som det afløser. Skal man så bryde totalt med brugergrænseflade-designet i det gamle system, og derved forkaste de færdigheder, som brugerne – måske møjsommeligt – har oparbejdet? Eller skal man bevare overensstemmelsen med det gamle system, vel vidende at det kan gøres langt mere brugervenligt:

> "A design trade-off you're constantly confronted with is: where do you become revolutionary and when do you cave to conservatism in terms of constantly maintaining compatibility? When does compability mean mediocrity?"

En anden hindring kan være mangel på støtte fra ledelsen – de smukke ord om brugervenlighed skal jo også gerne føres ud i livet:

> "It takes perhaps three weeks to sort out the fundamentals of a system. Everybody agrees about taking that time until it really comes down to it. Then somebody in management realises that we're challenged from this product over there and it's a race who can get it out first!"

Til slut et par ord om designernes egen opfattelse af sig selv og deres mål. En

designer udtrykker helt klart, at de tekniske aspekter af et system ikke må kompromitteres til fordel for brugervenligheden:

> "There are some things that designers should not compromise on, as for example achieving a clean internal structure."

I dette system betød det, at den indre struktur afspejledes i brugergrænsefladen i den måde, som man "bevægede" sig rundt i systemet. Det var rent hierarkisk, og det var derfor meget besværligt at komme rundt til systemets dele. Det er blevet ændret i de senere udgaver af systemet – efter massivt pres fra brugerside! Denne designer kunne have lært af at lytte til en af de andre, der udtaler følgende om designeres situation:

> "There's a point in time when a designer ought to ask himself if he's doing things because of some user requirement or because he's used to it. A designer has a burden in understanding his own presuppositions."

Konkluderende kan siges, at der skal en del til for at rokke systemdesigneres holdning til brugervenlighed; de tekniske aspekter er dybt rodfæstede. Og selv velmenende designere kan have svært ved at føre deres ideer ud i livet, både på grund af mangel på viden, men også på grund af de organisatoriske rammer, som lægges udefra.

Struktur i opgaven: Et eksperiment

I interaktionen mellem bruger og system kan flere niveauer identificeres, bla. opgave-niveauet, det semantiske niveau og det syntaktiske niveau. Antag for eksempel, at man skal slette medlem nr. 3207 i en forenings medlemsfortegnelse – det er opgaveniveauet. Her kan deltrin være at checke eventuelle kontingentrestancer, slette medlemmet og meddele dette til medlemmet. Til selve sletningen vælges et verbum, for eksempel "fjern" eller "slet" – det semantiske niveau. Endelige udtrykkes selve sletningen overfor systemet ved at skrive for eksempel "fjern 3207" eller "fj,3207" – det syntaktiske niveau.

Ledgard et al (1981) undersøgte betydningen af det syntaktiske niveau på brugervenligheden. De sammenlignede to editorer, der kun afveg i deres kommandosyntaks. Det viste sig at den naturligt-sprog-lignende form

> replace "tooth" with "truth"

var lettere for brugerne end formen

> rs: /tooth/,/truth/

hvor rs står for "replace string".

På omtrent samme måde fandt Barnard et al (1982) på det semantiske niveau, at semantisk specifikke kommandonavne, som for eksempel "rubout" for en

slette-operation, var lettere at lære og huske end semantisk generelle, som for eksempel "edit".

Det tredie niveau i interaktionen er opgavens struktur, som givetvis også har betydning for brugervenligheden. Dette undersøgte vi i et klassisk psykologisk laboratorieeksperiment med grupper af forsøgspersoner, variable og statistiske analyser (Jørgensen et al, 1983e). Men eksperimentet gav samtidig en fantastisk indsigt i forholdet mellem bruger og systemdesigner; herom senere.

Eksperimentet – som var et piloteksperiment – gik ud på at tage hul på betydningen af struktureringen af opgaven. Det blev gjort ved, at forsøgspersonerne skulle lære at anvende et lille edb-system, der behandlede telex-lignende meddelelser. For eksempel skulle prisen for udgående meddelelser beregnes og debiteres afsenderen, mens ankomsttidspunktet for indgående meddelelser skulle anføres på meddelelsen.

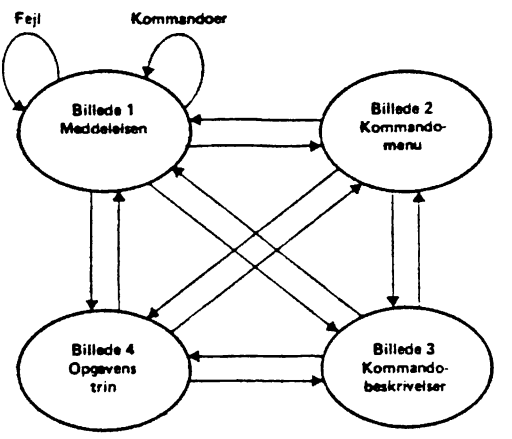
Figur 3 viser skærmbilledet i systemets grundtilstand: i midten en meddelelse, foroven plads til visse nøgleoplysninger, og forneden kommunikationsfeltet.

```
originator ref:

time:                        register:

mailpoint:

=================================INCOMING===
   To: Mr A. Tomkins
       Works Department
       Matric Ltd
       Oxbridge

Your proposed plan to build four new
assembly units near our premises at
Duckacre Park is unacceptable to us.

             From: Mrs E. King
                   Apex Publications
                   Corby
=============================================
Type a command or a number:

1:                    2:See commands
3:See definitions     4:See task details
```

Figur 3

Skærmbilledet i systemets grundtilstand (1).

I kommunikationsfeltet kan enten skrives en kommando eller et af tallene 2, 3 og 4. Herved fås hjælp fra systemet, se figur 4, der giver en oversigt over systemets skærmbilleder.



Figur 4

Oversigt over systemets skærmbilleder:

1 : grundbillede
2 : liste over kommandoer
3 : beskrivelse af kommandoer
4 : beskrivelse af opgavens trin

Valget "2" giver billede 2, der er en alfabetisk ordnet liste over kommandoerne i systemet. Valget "3" giver billede 3, der er en alfabetisk ordnet liste med beskrivelse af kommandoerne. Valget "4" giver billede 4 (figur 5), der beskriver den rækkefølge, som opgaven skal udføres i.

Hver meddelelse behandles i 8 deltrin (kommandoer), der parvis er sammenhørende. Den første kommando i hvert par tilvejebringer information, mens den anden anvender informationen. For eksempel parret CLOCK og STAMP, hvor den første tilvejebringer tiden og skriver den i øverste felt efter "time:" i det øverste felt på billede 1 (se figur 3), mens STAMP anvender den ved at kopiere tidspunktet ned i meddelelsens øverste højre hjørne.

Den primære experimentelle variabel var rækkefølgen af deltrinnene i opgaven. Det ene niveau var den parvise opdeling:

skaf information 1
anvend – –

skaf information 2
anvend – –

skaf information 3
anvend – –

skaf information 4
anvend – –

Denne struktur er "objekt-orienteret", da objekterne er det primære strukturereringskriterie. I det andet niveau er deltrinnnene i opgaven "funktions-orienteret", idet funktionerne er det primære:

skaf information 1
skaf information 2
skaf information 3
skaf information 4

anvend information 1
anvend information 2
anvend information 3
anvend information 4

```
Task details

STAGE 1:  * Establish the time at which
            the message arrived
          * Mark the message with its
            time of arrival

STAGE 2:  * Establish the "originator reference"
            of the person who sent the message
          * Acknowledge the receipt of the
            message

STAGE 3:  * Establish a "register number" where
            a copy of the message is to be kept
          * Keep a copy of the message in the
            register

STAGE 4:  * Establish the "mail point" of the
            person getting the message
          * Forward the message to its
            "mail point"
```

Figur 5

Billede 4: Beskrivelsen af opgavens deltrin

I figur 5 ses, hvordan dette fremtrådte overfor forsøgspersonerne. Figuren viser billede 4, der beskriver deltrinnenes rækkefølge, her struktureret objekt-orienteret. Forsøgspersonernes strategi vil naturligvis først være at se hvilket opgavetrin, der skal udføres (se figur 5). Antag det er det første: "Establish the time at which the message arrived". Dernæst skal forsøgspersonen finde den kommando, der udfører netop dette opgavetrin, det fremgår af billede 3 (ikke vist her) – det er CLOCK. Forsøgspersonen går derefter til billede 1 (figur 3) og skriver kommandoen CLOCK. Herved vises ankomsttidspunktet øverst på skærmen (figur 3), og næste trin i opgaven kan nu findes ved igen at se på billede 4 (figur 5); sådan fortsættes. Efterhånden vil forsøgspersonen lære opgavetrinnene og kommandoerne, og kan skrive dem uden hjælp eller blot ved at konsultere billede 2 med listen over kommandoerne.

Fjorten mennesker uden kendskab til edb deltog som forsøgspersoner. De var opdelt i to lige store grupper. Hver forsøgsperson prøvede under vejledning 2 meddelelser, hvorefter de fortsatte på egen hånd med 12 meddelelser. Forsøgspersonerne var mellem 1/2 og 2 timer om at fuldføre opgaven. Som mål for deres indlæring og hukommelse benyttedes tid (på 10 msec niveau), antal forespørgsler om hjælp og antallet af fejl.

For en datalog var det spændende at lave et psykologisk eksperiment, og umådeligt udbytterigt at se forsøgspersonernes arbejde med systemet. For eksempel var der nogle meget slående forskelle mellem forsøgspersonernes strategier. Nogle fulgte slavisk en "fejl-sikker" strategi: billede 4, billede 3, billede 1, skriv kommando, osv. Dette tog selvsagt lang tid, men resulterede i stort set fejlfrit arbejde.

Andre forsøgspersoner kastede sig straks ud i at prøve sig frem: "Går den, så går den!" De lavede selvsagt mange fejl, især i begyndelsen, men kom ofte hurtigt efter det.

Men på trods af store individuelle forskelle var der også systematik i forsøgspersonernes arbejde. Deres "tænketid" før de skulle skrive en kommando, kom tydeligt frem. I de tilfælde, hvor forsøgspersonen skriver en kommando spontant (uden at have anvendt hjælp), tog det 2.2 sek. (sd = 2.3) før den blev skrevet; forholdsvis lang tid og en ret stor spredning. (Tiderne gælder for den sidste af de 14 meddelelser.)

Hvis forsøgspersonen derimod havde hentet hjælp fra systemet (billede 2, 3 eller 4), var tiden 1.0 sek. (sd = 0.7) før den blev skrevet: kort tid og lille spredning. Dette er blot udtryk for reaktionstiden fra systemet er vendt tilbage til billede 1 og er klar til at modtage en kommando, indtil forsøgspersonen skriver det første tegn i kommandoen.

Selve resultatet viste klart – på trods af det beskedne antal forsøgspersoner og den store individuelle variation – at den objekt-orienterede struktur var lettest, og det endda med hensyn til alle målene.

Dette må sikkert forklares ved den stærkere semantiske binding i den objekt-orienterede struktur end i den funktions-orienterede struktur. Der var i den objekt-orienterede struktur stærke semantiske bindinger indenfor hvert par, for eksempel mellem operationerne COST, der beregner prisen for at sende meddelelsen, og CHARGE, der debiterer den afsendende afdeling beløbet.

For det første er det oplagt, at man skal beregne prisen før man kan debitere nogen, og for det andet at debitering følger naturligt efter en prisberegning. I den objekt-orienterede struktur er der kun tre 'skift' mellem objekter, i den anden er der to gange tre 'skift'. Dette må sammenholdes med, at der i den funktions-orienterede struktur ikke er nogen særlige romantiske bindinger mellem de fire operationer indenfor hver halvdel. Man kan skaffe oplysninger i vilkårlig rækkefølge, for eksempel adressen på modtageren før arkiv-nummeret. Så forsøgspersonerne har været henvist til mere leksikalsk orienterede strukturer i 2*4 strukturen.

At iagttage forsøgspersonernes møde med systemet

Det var lærerigt at udføre et eksperiment; men det var mindst lige så lærerigt at sidde ved siden af forsøgspersonerne og hjælpe dem med de to første meddelelser. Jeg – som havde designet systemet – mente absolut, at det var meget let at bruge. Det indeholdt al den nødvendige information, og var lille og overskueligt. Alligevel løb forsøgspersonerne gang på gang ind i vanskeligheder. Systemdesigneres standard-reaktion er her: "Brugerne er dumme!"

Men brugerne er ikke dumme – i hvert tilfælde ikke dummere end andre mennesker. En smule eftertænksomhed var altid tilstrækkelig til at finde årsagen til brugernes "dumhed". I alle tilfælde var det såre oplagt; jeg havde ikke forudset det under designet. Men hvordan jeg i designet kunne have undgået det, er en anden sag. I det følgende giver jeg nogle eksempler på forsøgspersonernes vanskeligheder.

Der opstod ofte tvivl hos forsøgspersonerne, når de havde lavet en fejl. Hvis de for eksempel havde anvendt en kommando "for tidligt" (systemet skal have kommandoerne i en "rigtig" rækkefølge), og kommandoen derfor var blevet afvist, kom de i tvivl, når de skulle skrive kommandoen på det "rigtige tidspunkt" i forløbet. Spørgsmål som "Skal jeg skrive den igen – jeg har jo allerede skrevet den!" forekom ofte. Det vil sige, at de opfattede ikke fejlen som "rigtig", de troede mere på deres egen oplevelse af at have skrevet den. Når forsøgspersonerne lavede en fejl, reagerede systemet med en fejlmeddelelse, for eksempel "CLOCK can't be used now". Forsøgspersonerne var åbenlyst i tvivl om, hvordan de kom videre. En spurgte hvordan man fjernede fejlmeddelelsen, mens en anden sagde "Kan jeg gå til billede 4 nu og derved få fejlen væk?" Forklaringen er den, at når de normalt kom til billede 1 fra billede 4, var der ingen fejlmeddelelse. En af grundene til tvivlrådigheden var klart nok, at fejlmeddelelsen ikke var handlingsorienteret – den påpegede kun, hvad man ikke måtte.

Hvis forsøgspersonerne havde stavet forkert, afviste systemet også kommandoen, for eksempel "AUTHIR is not a command". En af forsøgspersonerne peger på I'et i AUTHIR i fejlmeddelelsen og spørger "Hvordan retter jeg stavefejlen?". Det synes at forekomme dem fjernt, at de bare kan skrive kommandoen igen, som om intet var hændt. De skaber med andre ord en slags fejltilstand som en parallel til grundtilstanden (billede 1, se figur 4). De tror så, at de eksplicit skal forlade fejltilstanden for at komme i grundtilstanden.

Endelig forvekslede forsøgspersonerne ofte de sæt af numre, som er indbygget i systemet. Hele fem forskellige sæt af numre kan identificeres:

1) Billederne 1, 2, 3 og 4 (figur 4)
2) Deltrin-parrene 1, 2, 3 og 4 i opgavebeskrivelsen (figur 5)
3) Deltrin 1 og 2 i hvert deltrin-par
4) Den alfabetiske rækkefølge af kommandoerne på billede 2 og 3
5) Læserækkefølgen af informationerne i øverste felt i billede 1 (figur 3)

Disse sæt af numre forveksledes ofte. Følgende udsagn var typisk:

"Nu har jeg udført det første deltrin i opgaven, så nu vil jeg gå til billede 2!"

En bruger hæfter sig på billede 1 ved de sidste to ord i instruktionen: "Type a command or a number: ". Vedkommende ved godt, at kommandoen CHARGE skal anvendes nu. For at omsætte denne viden til et tal går brugeren – ubevidst om sin fejltagelse – til billede 2 og tæller sig frem i den alfabetiske rækkefølge: "APPEND, AUTHOR, CHARGE – jeg skal altså skrive 3!"

Forsøgspersonerne stiller information sammen, som på ingen måde er ment sådan. Og det var utroligt at se den mangfoldighed af måder, som systemets enkeltdele kunne kombineres til meningsfyldte strukturer. Det havde jeg slet ikke forudset – det var utroligt overraskende – og lærerigt. Det var en levende illustration af Gould og Lewis's påstand om, at selv systemdesignere med stor erfaring og indlevelsesevne ikke kan forudse brugervenligheden af et system.

Det er klart den mest effektive måde at illustrere afgrunden mellem systemdesignernes verden og brugernes. At sætte sig ved siden af en bruger – og undlade at lege bedrevidende – er for det første en meget enkel og billig metode. Og den kan – hvis designeren er motiveret for at lære – virkelig flytte hegnspæle, idet den konfronterer designeren med konsekvenserne af sine egne beslutninger.

Kan man udvikle brugervenlige systemer?

På baggrund af det foregående fristes man måske til at spørge, om det overhovedet kan lade sig gøre at udvikle et brugervenligt system? Det synes at være meget komplekst! Svaret er ja – heldigvis. Men det kræver en anden måde at forholde sig på til systemudvikling end den traditionelle.

Løsningen ligger gemt i ordet: prototyper. Det vil sige at udvikle en anvendelig model af systemet og prøve den under rigtige omstændigheder. Andre ingeniørmæssige discipliner har baseret udvikling på prototyper i talrige år. Det anses for at være en nødvendig del af en udvikling, fordi man ikke på forhånd kan tænke sig til alt, forudse alt. På den tidligere nævnte konference i Boston forelagde Gould og Lewis (1983) fire udviklingsprincipper baseret på prototype-filosofien:

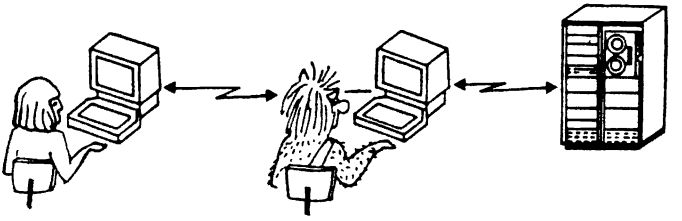
(1) Systemdesignerne må forstå, hvem de kommende brugere er. Dette gøres ved at undersøge deres kognitive, adfærdsmæssige og holdningsmæssige egenskaber, og ved at undersøge arten af deres arbejde.
(2) En gruppe kommende brugere skal arbejde tæt sammen med systemudviklerne lige fra de tidligste formuleringsfaser.
(3) Tidligt i udviklingsfasen skal kommende brugere anvende prototyper til at udføre rigtigt arbejde, og deres præstationer og reaktioner skal måles.
(4) Opdagede uhensigtsmæssigheder skal rettes, dvs. designet skal itereres: afprøvning, rettelse, osv.

Disse regler er udviklet gennem lang tid. Bemærk, at de udelukkende udtaler sig om systemudviklingsprocessen, ikke om produktet. Kelley (1983a og b) har vist princippernes duelighed i praksis ved udvikling af et kalendersystem. Kelley havde et meget ambitiøst mål: At anbringe en bruger ved en terminal og få brugeren til at udføre meningsfyldt arbejde med kalendersystemet uden instruktion, uden hjælp fra systemet, uden menuer og uden dokumentation. Dette er i strid med alle de fremherskende opfattelser af, hvad der kendetegner brugervenlighed. Men det lykkedes for Kelley på følgende måde:

Først interviewede han 23 forretningsfolk ("professionals"), der i deres arbejde var afhængige af en kalender. Interviewene var meget dybtgående, og herved fandt han adskillige overraskende ting, bla. at flere af de interviewede anvendte seks kalendere for at administrere deres tid.

Dernæst konstruerede han et sæt basale datastrukturer og funktioner i en database, der kunne håndtere den indsamlede type af information og operationerne på dem.

Det tredie trin, som er genialt, bestod i at lade brugere arbejde med et simuleret kalendersystem, hvori en "trold" simulerede systemet, se figur 6.



Figur 6

Simulation med en "trold"

For brugerne så det ud som om de arbejdede med et rigtigt system, men rent faktisk omsatte "trolden" deres spontant afgivne kommandoer til det primitive systems syntaks – men brugerne opdagede det ikke. Et menneske simulerede en maskine.

Alt hvad brugerne spontant skrev til "systemet" blev logget og derpå analy-

seret. På grundlag heraf opbyggede Kelley et program, der kunne håndtere disse ting.

Næste trin bestod da i en lignende opstilling, men hvor trolden kun greb ind, hvor det var nødvendigt, dvs. når det primitive system ikke kunne forstå brugerens kommandoer. De indvundne erfaringer blev derpå ligeledes indbygget i systemet.

Til sidst blev en afprøvning uden "trolden" foretaget. Som brugere havde man "professionals", og de blev ikke instrueret, der var hverken hjælp eller menuer, ej heller dokumentation. Men ikke desto mindre var de i stand til spontant at udføre meningsfyldt arbejde ved hjælp af systemet. Afhængig af måden at opfatte fejl på forstod systemet mellem 86 og 97 procent af brugernes spontane kommandoer. Herunder nogle smagsprøver på, hvad systemet kunne:

Weds, July 14, 1982 meet Susan at 2 p.m.

Bemærk forkortelsen "Weds" og redundansen i angivelse af både dato og ugedag.

tommorrow, call john at 7:30am

Bemærk stavefejlen i "tommorrow" og det manglende blanktegn mellem "7:30" og "am".

wed meet in room 2-3 from 4 to 5

Bemærk at systemet opfatter rum- og tidsangivelse rigtigt!

Man fristes til sammenligning at spørge læseren, om hun eller han nogensinde har anvendt et system, der tolerede fri anvendelse af blanktegn, stavefejl og redundans?

Nu gror træerne ikke ind i himlen, og systemet måtte da også give op overfor visse af brugernes ønsker. Her er nogle eksempler, som glimrende illustrerer banale ting, som ingen havde forudset:

Tues 13 10:AM Bob Jones

Systemet kunne ikke klare det forkert placerede kolon – om end det er ganse oplagt for den menneskelige læser, hvad meningen er. På tilsvarende måde med

July 15 ii:30 'handball Smith'

hvor ingen havde forudset anvendelsen af bogstavet i som cifferet 1. Endelig var visse af brugernes formuleringer for generelle for systemets sprogforståelse:

change handball to no known ending time

Systemet er baseret på en ordbog; man kan nu spørge, om ikke den bliver ved med at vokse i takt med, at flere og flere brugere kommer til med hver deres ud-

tryksformer. Svaret er nej, idet antallet af nye ord aftager og bliver efterhånden ganske marginalt.

Kelley slår meget på, at dette ikke er et nyt ekspertsystem (som har ry for at være meget tunge, bla. med hensyn til svartider). Systemet klarer næsten alle forespørgsler på få sekunder. Til gengæld er den indbyggede sprogmodel primitiv og specifikt rettet mod anvendelsen i kalendere, idet den er empirisk afledt, ikke teoretisk.

Kelley anfører selv nogle grunde til det vellykkede resultat. For det første baseredes designet på empiri og en grundig analyse af opgaven. Dernæst anvendelsen af "trolden", der gjorde det muligt at få brugere til at udføre rigtigt arbejde og derved registrere deres spontane kommandoer. Og endelig den holdning, der lå bag systemudviklingen: Hvis systemet ikke kan klare en kommando, hvis fejl er det så? Systemets . . . eller brugerens?

Dette eksempel viser meget klart, at udvikling af brugervenlige systemer sagtens kan lade sig gøre. Men det tager tid og det kræver nye metoder og tænkemåder. Men resultatet er et system, som brugerne er meget tilfredse med, og som så sparer dem meget tid. Prototype-filosofien har også vist sin duelighed i andre systemer, således ved udviklingen af Xerox Star (Smith et al, 1982), forgængeren for den nye klasse af meget brugervenlige systemer.

Perspektiver for brugervenlighed

Edb-folk sukker efter metoder og værktøjer til at designe systemer, ikke mindst til at designe brugergrænseflader. Men her er det meget svært at give håndfaste metoder, dels fordi det psykologiske samspil mellem menneske og systemer er utrolig komplekst, og dels fordi sagen drejer sig mere om 'blød' psykologi end om 'hård' edb.

For mig at se er der to veje til mere brugervenlige systemer: den objektive og den subjektive. Den objektive består i ved traditionelle forskningsmetoder at skaffe 'objektiv' viden om de psykologiske faktorer, der ligger til grund for brugervenlighed.

Men de objektive resultater skal forstås, akcepteres og anvendes af systemdesignere. Deres motivation har således stor betydning for, om resultaterne anvendes, og hvordan. Det er derfor vigtigt at udvikle systemdesignernes erfaringer, holdninger og forståelse af dette – den subjektive vej.

Jeg ser tre muligheder for en sådan udvikling, der alle bunder i designernes personlige erfaringer som frustrerede brugere – hvad selv de mest hårdnakkede systemdesignere må indrømme at have prøvet.

For det første er det afgørende, hvordan oplevelsen af at være frustreret og forvirret bruger behandles: Om man lader sig undertrykke af systemet eller om man stiller spørgsmålstegn ved systemets brugervenlighed. Edb-folk – og andre brugere – har i alt for lang tid akcepteret utrolig bruger-u-venlige systemer.

Stilles spørgsmålstegn ved systemernes brugervenlighed, vil det lede til at man bliver i stand til at identificere de uheldige sider i dialogen, og efterhånden bliver klar over, hvordan de kan forbedres. En konstruktiv måde til at ud-

vikle designeres opfattelse af, hvad der er brugervenligt, er at lade dem arbejde med brugervenlige systemer. Dette er vigtigt, da de dagligt anvendte systemer er normdannende.

For det andet er det karakteristisk, at systemdesignere opretholder en skarp skelnen mellem deres to roller: Som designere og som brugere (af andre designeres produkter). Dette virker hindrende for forståelsen af, at ens egne beslutninger i designet af en brugergrænseflade på tilsvarende måde har konsekvenser for brugerne, nemlig at de også får problemer. Nedbrydes dette skel mellem de to roller er vejen banet for, at man konstruktivt kan begynde at anvende de elementer i brugervenlighed, som man tidligere blev i stand til at identificere. En måde at nedbryde skellet er at lade systemdesignere overvære en brugers første møde med systemet. Herved konfronteres designeren med konsekvenserne af sine egne beslutninger, og afstanden mellem de to roller som bruger og designer bliver minimal.

Det tredie forhold drejer sig om at sammenkæde disse elementer i ens indsigt til helheder og forstå dem i større sammenhænge. Det kan for eksempel gøres ved at opdage paralleller til brugervenlighed af andre former for redskaber, der også kræver indsigt, og hvor designeren ofte har overset dette. Som eksempel kan nævnes en køreplan og en fotokopimaskine. Det er præcist de samme underliggende faktorer, der gør sig gældende her, når man har vanskeligheder med at anvende sådanne redskaber – og det har alle prøvet.

Den objektive vej og den subjektive vej til mere brugervenlige systemer er ikke alternativer. De kan tværtimod komplettere hinanden. Men så længe den objektive vej dyrkes med så snævre metoder som tilfældet er i dag, tror jeg at den subjektive vej er den mest frugtbare.

REFERENCER

BARNARD, P. J., N. V. HAMMOND, A., MACLEAN & J. MORTON: Learning and remembering interactive commands in a text-editing task. *Behaviour and Information Technology*, vol. 1, nr. 4, pp. 347-358, 1982.

CARD, S. K., T. P. MORAN & A. NEWELL: *The psychology of human-computer interaction.* Hillsdale, New Jersey: Lawrence Erlbaum, 1983.

CHI '83: *Proc. CHI '83 Human Factors in Computing Systems* (Boston, Dec. 12-15, 1983). New York: ACM, 1983.

JØRGENSEN, A. H.: The psychology of developing and using computer systems: five contributions. *Rapport 83/14,* Datalogisk Institut, Københavns Universitet, 1983a.

JØRGENSEN, A. H., P. BARNARD, N. HAMMOND & I. CLARK: Naming commands: an analysis of designers' naming behaviour. In: T. R. Green S. J. Payne & G. v. d. Veer: *The Psychology of Computer Use.* London: Academic Press, 1983, pp. 69-88 b.

JØRGENSEN, A. H., P. BARNARD, N. HAMMOND & I. CLARK: Naming commands: an analysis of designers' naming behaviour. *Rapport 83/3,* Datalogisk Institut, Københavns Universitet, 1983c.

JØRGENSEN, A. H., N. HAMMOND, A, MACLEAN, P. BARNARD & J. LONG: Design practice and interface usability: evidence from interviews with designers. *Rapport 83/10* Datalogisk Institut, Københavns Universitet 1983d.

JØRGENSEN, A. H., P. BARNARD, N. HAMMOND & A. MACLEAN: The effect of task structure in interactive systems: a pilot experiment. *Rapport 83/13,* Datalogisk Institut, Københavns Universitet, 1983e.

HAMMOND, N., A. H. JØRGENSEN, A. MACLEAN, P. BARNARD & J. LONG: Design practice and interface usability. Evidence from interviews with designers. *Proc. CHI '83 Human Factors in Computing Systems,* Boston, Dec. 12-15, 1983. ACM, New York, pp. 40-44.

GOULD, J. D. & C. LEWIS: Designing for usability – key principles and what designers think. *Proc. CHI '83 Human Factors in Computing Systems,* Boston, Dec. 12-15, 1983. ACM, New York, pp. 50-53.

KELLEY, J. F.: An empirical methodogy for writing user-friendly natural language computer applications. *Proc. CHI '83 Human Factors in Computing Systems,* Boston, Dec. 12-15, 1983. ACM, New York, pp. 193-196 a.

KELLEY, J. F.: An iterative design methodology for user-friendly natural language office information applications. *ACM Trans. on Office Informations Systems,* vol. 2, nr. 1, pp. 26-41, 1984b.

LANDAUER, T. K., K. M. GALOTTI & S. HARTWELL: Natural command names and initial learning: a study of text editing terms. *Comm. ACM,* vol. 26, no. 7, pp. 495-503, Juli 1983.

LEDGARD, H., J. A. WHITESIDE, A. SINGER & W. SEYMOR: The natural language of interactive systems. *Comm. ACM,* vol. 23, nr. 10, pp. 556-561, Okt. 1981.

SMITH, D. C., C. IRBY, R. KIMBALL, B. VERPLANK: Designing the Star user interface. *BYTE,* apr. 1982, pp. 42 ff.

SMITH, S. L. & A. F. AUCELLA: Design guidelines for the user interface to computer-based information systems. *Report ESD-TR-83-122,* The MITRE Corporation, Bedford, Mass., Mar. 1983.