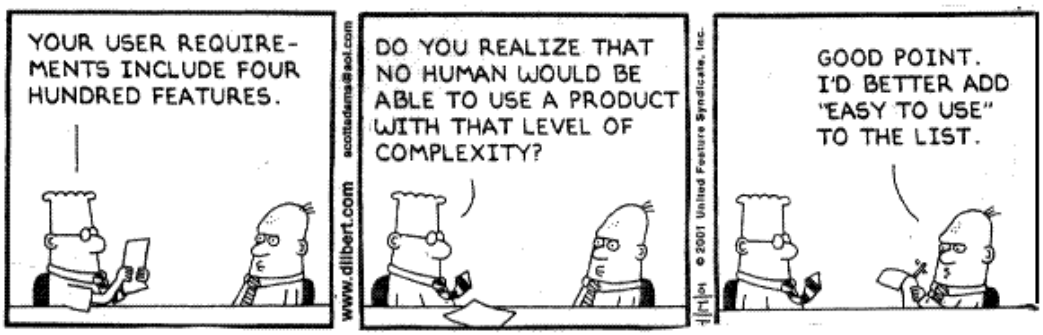
Usabilitymetoder i systemudvikling

- Fokus på brugerne



8. semester humanistisk datalogi

Udarbejdet af: Morten Møller Jacobsen

Vejleder: Tom Nyvang

Censor: Ellen Christiansen

Usabilitymetoder i systemudvikling

- fokus på brugerne

Abstract

Projektet søger at skabe overblik dele de værktøjer og metoder der findes til skabelsen af mere brugervenlige IT-applikationer. Udvælgelsen sker med afsæt i Jakob Nielsens teorier omkring usability, samt den officielle standard for usability: ISO-9241-11 samt dennes guide ISO 13407.

Der diskuteres bl.a. vigtigheden af iterative processor og brugerinddragelse på flere forskellige stadier i en udviklingsproces.

Af:

Morten Møller Jacobsen

8. semester humanistisk datalogi

Vejleder: Tom Nyvang

Censor: Ellen Christiansen

AALBORG UNIVERSITET

Maj 2006

Rapportomfang: 70610 (typeenheder)

Antal sider(norm./word): 29,42/72

Indholdsfortegnelse

Indledning

IT-teknologien er en relativ ny teknologi for de fleste mennesker i dag. Det er kun 25 år siden at IBM, i 1981, viste den første PC frem i New York (WWW [7]).

I 1943 var der ellers ikke meget der tydede på at IBM troede på at der var en fremtid i at sælge computere idet daværende direktør Thomas Watson udtalte: "I think there is a world market for maybe five computers". Senere i 1977 udtalte ligeledes en skeptisk Ken Olson (direktør, formand og grundlægger af Digital Equipment Corporation): "There is no reason why anyone would want a computer in the home" (WWW [4]). I computerens spæde opstart var hardwarepriserne så høje at computeren kun blev brugt i forskningsmiljøer, hvilket nok var en af grundene til denne spådom. Desuden var software ikke-eksisterende i de første år der eksisterede computere. I stedet opererede man maskinerne direkte på hardwareniveau bl.a. ved at flytte ledninger fra et hul i maskinen til et andet. Den høje pris og dermed smalle brugergruppe gjorde at man kunne forvente at brugeren havde en vis ekspertise og uddannelse. Desuden betød den høje pris også at maksimal udnyttelse af computeren blev prioriteret højere end brugerens brugsoplevelse (Nielsen 1993, p. 8).

Dette må siges at være anderledes i dag i forhold til den gang, da man i Danmark købe en ny computer i prislejdet 3-4000kr.

De lave priser på hardware har betydet at der findes en computer i næsten hvert hjem. I 2005 havde 84 % af de danske familier en PC i hjemmet, mens 79 % af befolkning havde adgang til Internettet fra deres hjem via en computer (WWW [3]).

Det høje antal computere i de private hjem, såvel som på arbejdspladserne, bør sætte krav til systemudviklere om at udvikle brugbare og forståelige applikationer.

I det følgende vil jeg skelne mellem to typer af software:

- Et *hyldeprodukt* som distribueres ud til mange kunder, som ikke på forhånd har bestilt et program.
- Et *bestillingsprodukt* som på forhånd har en kendt kunde.

Jeg forventer at denne distinktion vil have en indflydelse på de muligheder der eksisterer for at skaffe oplysninger om brugerne af de forskellige programmer.

Med den øgede mængde IT-firmaer der eksisterer, kan man forvente at en øget fokus på systemets brugere kan blive en konkurrence-parameter (Aykin 2003, p. 424). Dette faktum bør øge vigtigheden af at man investerer tid i at udvikle nogle brugervenlige programmer, da kunderne kan tillade sig at være kritiske i valg af udvikler. Dette forventes især at gælde for de såkaldte bestillingsprodukter, da de som regel har en klar og fastlagt målgruppe.
Lidt sværere formoder jeg det er med hyldeprodukter, da de typisk vil have en bredere og knap så defineret målgruppe. Jeg forventer dog der eksisterer nogle værktøjer til at hæve kvaliteten af begge programtyper.

I enkelte tilfælde har det vist sig, at udviklere er sluppet mindre heldigt af sted med udviklingen af et program.
Et af de helt store skrækeksempler er AFs store satsning i slutningen af 1990'erne og starten af dette årtusinde, AMANDA. Det fyldte meget i medierne og man hørte ofte at systemet var blevet yderligere forsinket og at projektet skulle have tilført yderligere kapital.
Da programmet endelig kørte nogenlunde, viste det sig at være så langsomt og fejlbehæftet at de ansatte på AFs centre begyndte at udvise tegn på stress (WWW [2]) pga. programmets lave brugshastighed.

Ifølge Søren Mikkel Berg (2005a), skyldes over halvdelen af alle fejl i IT-projekter mangelfulde kravspecifikationer. Desuden siger han (2005b) at fejl er relativt omkostningsfrie at opdage i kravfasen, mens det koster op mod 100 gange så meget hvis man først opdager dem i programmeringsfasen, mens det koster hele 1000 gange så meget, hvis de først opdages under testfasen.
Derfor er det vigtigt at have god dialog med sin kunde, men slutbrugerne skal også høres. I visse tilfælde kan kunde og slutbruger dog være den samme, men det er ikke nødvendigvis givet.

Senest har et projekt på KMD også indikeret vigtigheden af at involvere brugerne af et program. I forbindelse med implementering af SAP som udviklingsplatform er KMD ved at re-designe dele (hvis ikke hele) af deres produktportefølje. Dette indebar bl.a. et program skolesekretærer anven-

der til håndteringer af elevoplysninger. Dette projekt var fra start ventet at blive et "1:1-projekt", altså et projekt hvor man blot skulle flytte systemet fra den eksisterende Acces-platform til den nye SAP-platform. Efterhånden som projektet skred frem viste det sig at det ikke var så ligetil og en række sekretærer blev spurgt til råds om hvordan oplysningerne smartest blev organiseret i forhold til hvor ofte de enkelte oplysninger skulle bruges (Foredrag om projektledelse af Elorie Østergaard, KMD; hos InDiMedia 2006).

Kunne dele af disse problemer have været undgået? Det kunne de muligvis. Jokela (Jokela 2003, p. 19) anbefaler at man så tidligt som muligt i en udviklingsproces inddrager usabilityværktøjer, da de kan indflydelse på valg af udviklingsmetode. Usability kan defineres ud fra 4 aspekter:

- Rationalet for brugercentreret design
- Principper for brugercentreret design
- Planlægning af brugercentreret design
- Aktiviteter for brugerdrevet design

Disse fire aspekter vil yderligere beskrevet og diskuteret igennem projektet.

For at et givent IT-projekt kan blive en succes lader det altså umiddelbart til at brugerinddragelse kan være vigtig parameter. Nogle folk kalder det ligefrem en gylden regel at brugerne inddrages (Molich 2003, p. 31). Der findes dog flere forskellige argumenter for at inddrage brugerne under selve udviklingsprocessen. Nogle har lige været kort berørt herover, men argumenterne vil blive yderligere diskuteret senere i projektet.
Ligeledes findes der også mange bud på hvordan brugerne inddrages bedst muligt, både mht. hvornår i processen de skal inddrages og i hvilket omfang de skal inddrages.
I dette projekt forsøges det at skabe et overblik over nogle af de metoder der anvendes til at højne kvaliteten og brugbarheden af IT-produkt. Samtidig forsøges der at opliste nogle af de forbehold der sandsynligvis skal tages når brugerne inddrages.

Formål med projektet

Jeg finder udviklingen af IT-systemer på de fleste platforme, web, almene applikationer, spil, mobiltelefoner m.m. interessant. Det er bl.a. de indledende overvejelser og diskussioner i forbindelse med systemudvikling og design, brugerinddragelse samt en diskussion af hvor meget brugerne skal inddrages og hvor meget værdi/magt deres mening skal tildeles. Mit hovedformål for projektet er som nævnt herover at danne mig et overblik over systemudviklingsmetoder, samt at få et overblik over de værktøjer og metoder til øget usability og brugerinddragelse der findes.
Usability og brugerinddragelse ser jeg lidt som to sider af samme sag. Hvor jeg ser usability som det overordnede mål, eller samling af principper, for en udviklingsproces, mens brugerinddragelse er et værktøj til at komme til målet.
Den viden der dannes her er tiltænkt til brug på de senere semestre med henblik på et speciale indenfor HCI-feltet gerne med øget fokus på brugerinddragelse og håndtering af brugernes kommentarer og input.
Min interesse for dette projekt henvender sig altså primært til det rent metodiske for hvordan processen bliver planlagt og til selve udførelsen. Hovedvægten vil blive lagt på afsnittene omkring udviklingsmetoder og usabilityværktøjer.

Problemformulering

Ovenstående tanker og iagttagelser leder mig frem til følgende problemformulering.

- Hvilke rationaler er der i usability?
  og hvilke værktøjer er der til at frembringe disse?

Hvad er - usability

Usability er blevet et meget brugt buzzword inden for systemudvikling, både når det drejer sig om de mere gængse og traditionelle IT-programmer, men op gennem 90'erne er det især også blevet hæftet på hjemmesider. Ordet betyder *brugbarhed* eller *brugervenlig* og bruges nærmest som et kvalitetsstempel når et program eller hjemmeside er funktionel, let at forstå og/eller anvende (WWW [6]). Under dette projekt vil jeg primært forholde mig til almindelig computerprogrammer, men blot gøre opmærksom på at betegnelse dækker mere end computerprogrammer. Benævnelsen kan også omfatte almindelige dagligdagsprodukter.

Umiddelbart kunne man tro at graden af et produkts brugervenlighed er en meget individuel ting, da vi alle har forskellige præferencer til at opfatte og forstå ting ud fra.
Min egen opfattelse af begrebet i forbindelse med IT-programmer går i retning af at programmet skal:

- være intuitivt at tage i anvendelse.
- have en synlig og anvendelig hjælp.
- i vid udstrækning skal kunne tilpasses den enkelte bruger.
- være driftssikkert.

Ovenstående er blot mit personlige syn på usabilitybegrebet. Det kan anklages for at være lidt for specifikt rettet mod IT-produkter og man kan så diskutere hvorvidt det er skidt eller ej.
Der forefindes en officiel definition på usability, som er en isostandard. Første gang jeg hørte at begrebet var blevet standardiseret havde jeg en umiddelbar skepsis overfor hvorvidt dette kunne lade sige gør i særlig stor udstrækning og i hvilken grad definitionen ville være entydig.
Derfor satte jeg mig for at finde ud hvad denne standard sagde om usability, samt undersøge til hvilken grad definitionen var konkret. Her tænker jeg konkret på samme måde som de definitioner der er på f.eks. byggematerialer, hvor der er nogle klare målbare krav og definitioner.
Definitionen på usability lyder ifølge ISO 9241-11:

> *The extent to which a product can be used by specified user to achieve specified*

> *goals with effectiveness, efficiency and satisfaction in a specified context of use.*

(Jokela et al. 2003, p. 54).

Med formuleringen "The extent to which a ***product***…", lader det umiddelbart ikke til at definitionen specifikt er rettet mod computerprogrammer, men på produkter i det hele taget. Dermed kan udviklere af fx fjernsyn eller mikroovne også holde deres produkter op imod denne definition. Jeg vil dog koncentrere mig om definitionen i forhold til IT-produkter.

På dansk bliver denne standard til noget i retningen af: "Det omfang et produkt kan blive brugt af en specifik bruger til at opnå specifikke mål brugbart, effektivt og med tilfredshed i en specifik kontekst".

De første par gange man læser definitionen kan man måske synes den lyder lidt for generelt og intetsigende, men ved nærmere eftertanke giver den alligevel ganske god mening. Som nævnt herover kan man formode at hvad der er brugervenligt for én person, ikke nødvendigvis er brugervenligt for en anden. Dette tilgodeses med formuleringen "en specifik bruger". Desuden antager jeg at den videre formulering "til at opnå specifikke mål" og "i en specifik kontekst" betyder at forskellige brugere kan have forskellige mål og forskellige brugskontekster for det samme system, uden at det dog behøver at være tilfældet. Dette antager jeg er tilfældet med diverse "kontorpakker", som fx Open Office og Microsoft Office, som formodes at have en meget bred brugerskare. I sådanne tilfælde kan det være svært at definere en egentlig kernegruppe af brugere og man skal således forsøge at tilfredsstille mange typer. Værktøjer til dette diskuteres senere andetsteds.
Standarden "tillader" også at man kan skræddersy et produkt til én specifik brugergruppe[1], uden at tage hensyn til brugere uden for denne gruppe. Der kan dog også, som illustreret herover, være situationer hvor flere forskellige brugere anvender samme produkt men på (formodentlig) forskellige vis og til forskellige formål.

[1] Jeg antager at man sjældent udvikler et produkt til kun én person alene.

Umiddelbart synes det svært at skelne de engelske ord effectiveness og efficiency fra hinanden, da begge umiddelbart kan oversættes til "effektivitet" på dansk. Deres uddybende beskrivelser gør det dog en smule lettere:

> *Effectivenes (Es): the accuracy and completeness with which uses achieve specified goals.*
>
> *Efficiency (EY): the resources expended in relation to the accuracy and completeness with which users achieve goals.*

(ibid.)

For at gøre det endnu tydeligere er der opgivet et eksempel for en hæveautomat i en bank:

> *90 % users achieve the goal (Es) in less than 1 minute (Ey) with an average satisfaction rating '6' when users are novice ones, and they want to have a desired sum of cash withdrawn with any bank machine.*

(ibid.)

I et forsøg på at gøre ISO standarden lettere at forstå, bl.a. vha. en uddybende forklaring af begreberne er der udarbejdet en vejledning til forståelse ISO'en: ISO 13407.
I en gennemgang af denne vejledning (Jokela et al. 2003) kommer det frem at den på visse punkter er mangelfuld, da der er nogle begreber fra standarden der ikke er forklaret. Dog indeholder vejledningen en række overordnede betragtninger, som skal overvejes i forbindelse med udarbejdelse af brugervenligt design. En af disse mangler er bl.a. en manglende beskrivelsen af processen med at definere nogle kvantitative målbare krav for *Effectiveness*, *efficiency* og *satisfaction*. Selvom man skulle mene at disse målbare krav burde have en central funktion i kontrollen af hvorvidt ens mål er opnået.

ISO 13407 beskriver bruger-centreret design (BCD) ud fra fire aspekter, der kort blev introduceret indledningsvist:

- Rationalet for BCD

- Principper for BCD
- Planlægning af BCD
- Aktiviteter for BCD

Disse er nærmere beskrevet i afsnittet *hvordan usability*, umiddelbart herunder.

Da dette projekt omhandler usability, føler jeg at det være til gavn at se om der skulle findes nogle andre definitioner eller bud på hvad usability er for en størrelse. Jakob Nielsen som af mange betragtes som en guru inden for usability har selv et bud på en definition, en definition som ifølge bl.a. Timo Jokela (ibid.) er en af de mest kendte definitioner.

Nielsens definition lyder:

”Usability is about learnability, efficiency, memorability, errors and satisfaction” (ibid).

Denne definition opererer altså med fem variabler, hvoraf nogle er gengangere fra ISO standarden:

- Learnability
- Efficiency
- Memorability
- Errors
- Satisfaction

Efficiency og *satisfaction*, har samme betydning som i ISO standarden, mens de øvrige er Nielsens egne, som betyder:

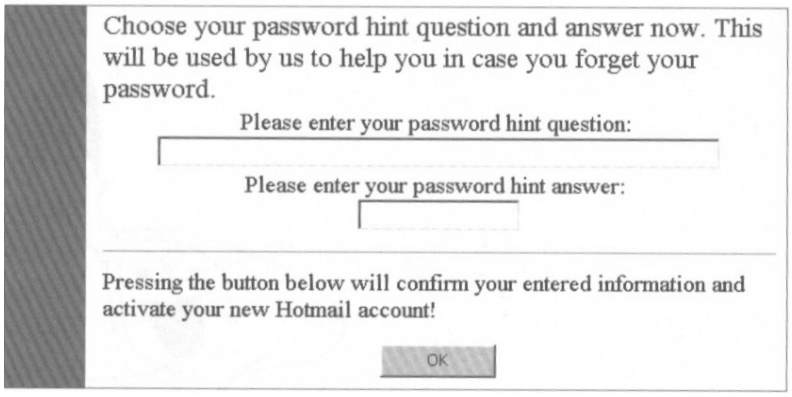
- Hvor let en bruger kan udføre basale opgave første gang vedkommende er på en hjemmeside? (*learnability*).
- Hvor hurtigt en bruger kan opnå samme dygtighed på en side, efter han ikke har besøgt den i et stykke tid? (*memorability*)
- Hvor mange fejl laver brugerne, hvor fatale er de fejl og hvor let kan de komme videre fra disse fejl? (*errors*).

Her synes jeg det er værd at bemærke at Nielsen ikke inddrager hverken brugere eller konteksten, selvom disse kunne forventes at skulle være parametre, da det er i samspil med dem at systemet skal fungere.Til gengæld indeholder Nielsens definition nogle punkter som jeg troede ville være at finde i ISO-standarden, især punktet omkring *learnability* var et jeg savnede.

Nielsens definition af *learnability* lægger sig tæt op af den definition jeg lægger i ordet intuitivt. I mit univers hænger graden af et produkts *usability* til dels sammen med hvor intuitivt det er at gå til.

Et andet punkt hvor de to definitioner adskiller sig fra hinanden er punktet *Errors* i Nielsens definition. Med punktet siger Nielsen direkte at antallet af et produkts fejl har indflydelse på brugbarheden. Et givent program med mangle fejl vil nok også blive bemærket målt imod den officielle definition, da et fejlfyldt produkt med stor sandsynlighed vil en lave score i punktet *satisfaction*. Et produkt kan være nok så logisk at anvende, men hvis det viser sig at det ofte går ned, formoder jeg at brugerne mister tålmodigheden med programmet.
Usability hænger altså dermed både sammen med hvor brugbart produktet er, forstået på den måde hvor let det er at anvende og til dels om det virker rent teknisk.

I "Brugervenlig Webdesign" fortæller Molich (2003, p. 14ff.) at en tidligere undersøgelse af Hotmails[2] registreringsproces i mange tilfælde var uforståelig for brugerne. Under registreringsprocessen blev brugerne mødt af skærmbilledet nedenfor (Figur 1).



Figur 1

Brugerne forstod simpelthen ikke hvad forskellen var på de to inputs de blev bedt om at indtaste. I nogle tilfælde skrev de det samme i begge felter, mens andre gentog deres password i feltet "password hint question".

[2] Et gratis e-mail tjeneste fra Microsoft

Hvorfor usability?

Som det lille ovenstående eksempel viser, så er det ikke nødvendigvis givet at blot et program virker rent teknisk, så er det godt nok. Hvis brugerne ikke forstår de anvisninger eller termer der anvendes.

Foruden at mindske den menneskelige irritation på et program kan der også være en økonomisk gevinst at hente ved at have mere forståelige og brugbare programmer. Et større computerfirma opnåede en besparelse på knap 42000$ første dag at et nyt og hurtigere log-on system blev taget brug. Selve udviklingsforløbet af det nye log-on system havde blot kostet knap 21000$ (Karat 1990 cited Nielsen 1993). Det er ifølge Nielsen dog ikke altid at man kan se nytteværdien af et usabilityprojekt før det er implementeret, hvilket diskuteres i det følgende afsnit, *Hvordan usability?*

Ovenstående eksempel illustrerede nytteværdien af et usabilityprojekt internt i en virksomhed, men der kan også findes eksempler på at det kan betale sig at hæve brugervenligheden af produkter der skal sælges på markedet.

Hvis vi forstiller os en webbutik må det formodes at være vigtigt at de er brugervenlige, da det ellers vil koste potentielle kunder (Nielsen 2001, p. 20). Hvis man som kunde ikke kan finde varerne eller finde ud af at få dem betalt, køber man dem jo heller ikke og en konkurrent er kun få klik væk.

Undersøgelser har desuden vist at folks tålmodighed på Internettet er ringe. Det gør sig især gældende når de venter på at en side bliver indlæst. Undersøgelser har yderligere vist at brugere vil have svartider på under et sekund (Nielsen 2001, p. 48 ff.).

Jeg formoder at noget af det samme gør sig gældende under informationssøgning (og ikke kun navigering). Den ønskede information skal være let at finde, ellers vil folk forlade siden og lede videre et andet sted.

Hvordan usability?

Jeg vil i dette afsnit kort skitsere hvordan usability kan udføres på et overordnet plan og så gå mere i detaljer med metoderne i afsnittet *Operationalisering af systemudvikling* (jf. p. 43).
I forbindelse med øget usability ser jeg umiddelbart 2 tilgange til dette:

1. Øget usability af et eksisterende og kørende produkt
2. Øget usability af et produkt under udvikling

Metoderne og værktøjer til at opnå større usability forventes at være stort set de samme, men i den første tilgang har udviklerne noget konkret og nogle faktiske og ikke mindst kendte brugere at forholde sig til. Derfor formoder jeg at dette forhold vil medføre øget mulighed for mere konkret og autentisk empiriindsamling.

Som nævnt i afsnittet *Hvad er usability* (jf. p. 18), så beskriver ISO 13407 brugercentreret design ud fra fire aspekter:

- Rationalet for BCD
- Principper for BCD
- Planlægning af BCD
- Aktiviteter for BCD

Desuden findes en anden ”opskrift”, som jeg formoder er meget anvendt, nemlig Jakob Nielsens:

1) Opdag behovet for usability i din organisation.
2) Gør det klart at usability har management support (inkluderer bl.a. at promovere en kultur hvor det som noget positivt at udviklere ændrer deres designidéer for at tilpasse foreviste brugerbehov).
3) Afsæt specifikke ressourcer til usabilityudvikling.
4) Integrer systematisk usabilityudviklings-aktiviteter i de forskellige stadier af din udviklingslivscyklus.
5) Sikr dig at alle brugerinterfaces bliver udsat for brugertest.

Rationale for BCD

Det første aspekt, rationalet for BCD går ud på at beskrive de fordele der er at hente ved at iværksætte en proces med skabelse af et brugercentreret design. Det kan være økonomisk gevinst som nævnt ovenfor i afsnittet *Hvorfor usability*, øget brugertilfredshed, lavere support omkostninger eller noget helt andet (Jokela et al. 2003, p. 55).
Dette rationale for usability kunne tænkes at være forbundet med første aktivitet under *Aktiviteter for BCD* herunder, som går på at identificere behovet for (øget) brugercentreret design (jf. figur 2, p. 26). Et argument kan være muligheden for en økonomisk gevinst. For at belyse muligheden for en sådan formoder jeg at en cost-benefit analyse kan være behjælpelig.
Der kan være nogle fordele ikke er umiddelbart synlige inden et evt. redesign eller helt nyt system er implementeret (Nielsen 1993, p. 3). Man kan fx forestille sig at en omorganisering af et programs funktioner eller opsætning, på sigt, efter at brugerne har vænnet sig til det nye design, effektiviserer deres arbejdsgange fordi programmet nu er mere logisk opbygget.

Principper

Principperne for BCD indeholder fire generelle principper der karakteriserer BCD.

- Aktiv involvering af brugere og en klar forståelse af bruger og opgavekrave.
- En passende fordeling af funktioner mellem brugere og teknologi
- Iterativ designproces-
- Multidesign.

Nielsen har sammen med Rolf Molich tilbage i 1990 opsat en lang række principper for usability der ifølge dem bør følges af alle brugerinterfacedesignere. Herunder oplistes et uddrag af disse, for den komplette liste henvises til *Usability Engineering* (jf. litteraturlisten).

- Tal brugerens sprog - brug de termer brugerne forventes at bruge.
- Minimer mængden af information brugeren skal huske gennem interaktionen.
- Bevar konsistens gennem programmet - der skal fx ikke være flere forskellige ord for det samme.
- Tydelig udgang - brugeren skal ikke være i tvivl om hvordan et program eller fejlvalgt funktion lukkes.
- Tillad genvejstaster
- Gode fejlmeddelelser - skulle brugeren lave en fejl, skal han hjælpes videre, ikke noget med ”Error code 123!”.
- Forebyg fejl - endnu bedre end gode fejlmeddelelser.
- Hjælp og dokumentation - hjælpen skal være søgbar og eksempler skal fokuserer på brugerens opgaver.

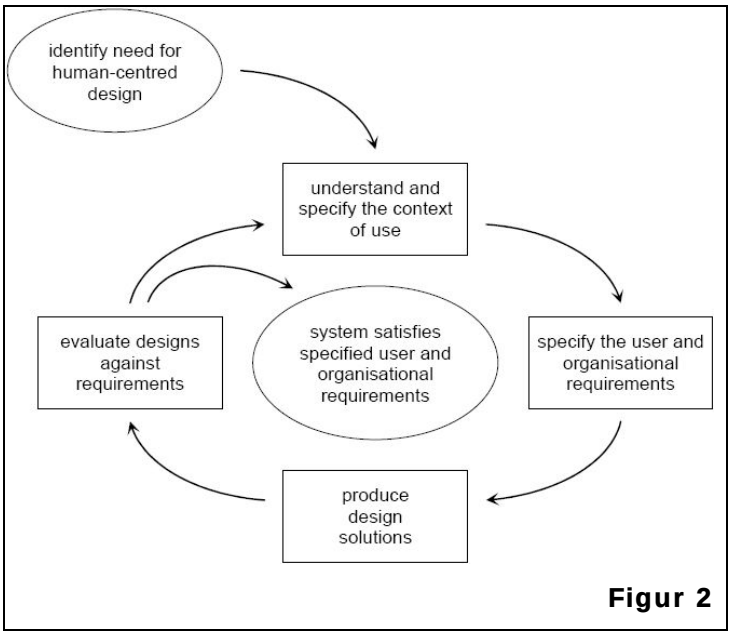
Planlægning

Planlægningen skal sørge for at skabe overblik over usabilityaktiviteterne. Der skal desuden sørges for at aktiviteterne passer ind i den overordnede systemudviklingsmetode.

Usabilityaktiviteter

Som Figur 2 nedenfor viser så er første skridt på vejen til at begynde at overveje at tage usabilityværktøjer i brug at man erkender behovet for øget usability og brugercentreret design i et givent produkt. Denne proces hører til under det punkt der i ISO 13407 blev benævnt aktiviteter for BCD (jf. p. 18).

Efter man har erkendt et behov for brugercentreret design, begynder arbejdet med at forstå og specificerer brugskonteksten. Dette indebærer bl.a. at man skal lære brugerne at kende, brugsmiljøet og de opgaver der skal udføres med produktet. Som det fremgår af illustrationen ovenfor, står der ikke noget om brugere i den første af de firkantede kasser og benævnelserne kan derfor virke en smule misvisende, da tredje fase går ud på at *specify the user and organisational requirements*. Denne fase går mere på kravene til produktet end en egentlig beskrivelse af brugerne, som der ellers lægges op til. I denne fase skal succeskriterierne for produktet bestemmes såsom, hvor hurtigt skal en typisk bruger kunne udføre en bestemt opgave. Desuden bestemmes der guidelines for designet her.



Figur 2

Næste fase *produce design solutions* indeholder hvad den lægger op til, nemlig produktion af designforslag. Efter denne fase skal designforslagene evalueres og holdes op imod brugerkravene. Denne proces kan gennemløbes nogle gange indtil et acceptabelt resultat er nået.

Det helt essentielle i brugercentreret design er den iterative proces, som går igen i mange modeller omkring udarbejdning af brugervenligt design.

Aktiviteterne beskrevet ovenfor minder lidt om de faser man gennemgår under *prototyping* (jf. p. 33).

System-udvikling

Dette kapitel vil indeholde et udvalg - og beskrivelse - af en række af forskellige udviklings-metoder og traditioner. Dette gøres for at skabe et overblik over den overordnede proces. Under *Operationalisering af udviklingsmetode* (jf. p. 43) inddrages metoderne fra usabilityanbefalingerne i et forsøg på at højne outputtet af en udviklingsproces.

Software engineering

Traditionen for Software engineering opstod tilbage i 70'ernes USA, hvor det at udvikle software var forbundet med noget meget hektisk og kaotisk. Det at udvikle software blev sat lig med at kode. Det medførte at udviklingen af software ofte skete uden at der egentlig var en rød tråd for hvordan der skulle udvikles og uden et egentlig overblik over hvem der lavede hvad (Lytje 2000, p. 174-175).
Traditionen indenfor software engineering foreslår at man opdeler hele forløbet i forskellige faser og får sat hele forløbet i system, således at programmering kun er én fase i en større sammenhæng. Tilbage i 1979 kom Yourdon med forslaget om at inddele i forløbet i følgende faser: *Analyse, design, programmering, test og vedligeholdelse* (ibid. p. 176).
Disse faser er siden hen blevet brugt på forskellig vis og er stadig hovedfundamentet i mange udviklingsprocessor. Denne relative stringente opdeling af udviklingsprocessen skal bidrage til at skabe større overblik over forløbet og dermed skabe programmer med færre fejl. Dette vil også betyde en minimering af de samlede udviklingsomkostninger, da der så ikke skal bruges så meget tid på fejlretning til slut (ibid. p. 179).

Udviklingsmodeller

Som nævnt kan ovenstående faser indgå i forskellige udviklingsmodeller på forskellig vis. Udviklingsmodeller kan på mange punkter sammenlignes med en opskrift. De er en illustrativ repræsentation af de aktiviteter der foregår fra idéfase til aflevering og vedligeholdelse og er dermed, som

nævnt tidligere, med til at skabe overblik over hele forløbet (Budde et al. 1992, p. 24).

Herunder vil jeg beskrive en række udviklingsmodeller, som de ser ud i deres "rene" form. Med "rene form" mener jeg metodernes grundform. Det har nemlig ofte vist sig at det kan være vanskeligt at trække en bestemt udviklingsmodel ned over et givent projekt. Derfor vil man ofte tage udgangspunkt i en model og rette den til, så den passer på det aktuelle projekt. Jeg vil i de følgende afsnit beskrive nogle af disse udviklingsmetoder og fremhæve deres forcer og svagheder.

Vandfaldsmodellen

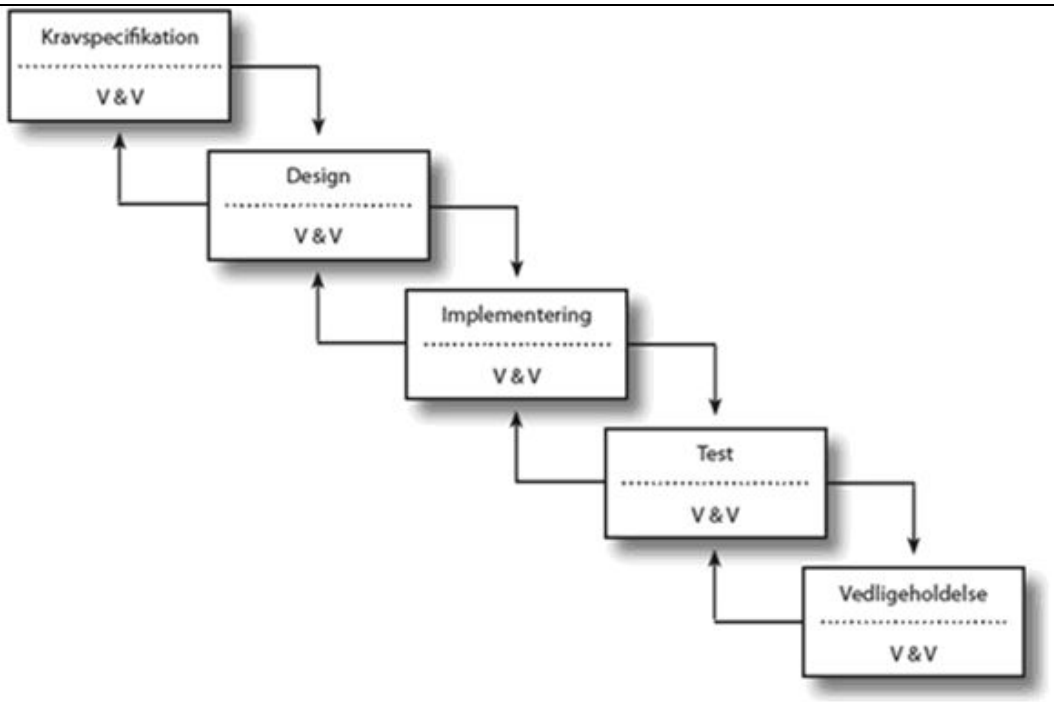
En af de første udviklingsmodeller der blev udviklet var vandfaldsmodellen (Lytje 2000, p. 176).

Modellen var oprindeligt kendetegnet ved at de enkelte faser var skarpt adskilte og når en proces (fase) var afsluttet og den næsten påbegyndt, kunne man i princippet ikke gå tilbage og ændre i noget der hørte til en tidligere proces. Systemudvikling blev altså med dette syn anset som værende af primær teknisk karakter og når først problemet var defineret i kravfasen kunne man straks gå i gang med at designe systemet og derefter kode det, uden på noget tidspunkt at kommunikerer med dem der i sidste ende skal anvende systemet (Budde et al. 1992, p. 25 ff.).

Dette syn er der dog løsnet lidt på, for som figuren herunder (Figur 3) viser, så går pilene mellem faserne i begge retninger. Med denne redigering lægges der nu op til at der at iterative processor kan og bør forekomme.

Figuren viser også at processen strækker sig fra det meget abstrakte, med K*ravspecifikation* og bevæger sig mod det mere konkrete og nære med D*esign*, *Implementering*, *Test* og til sidst (*Aflevering*) og V*edligeholdelse*. Med implementering menes her selve udviklingen af systemet.

V & V står for validering og verifikation, som betyder hhv. "Har vi gjort det, vi sagde vi ville gøre?" og "Har vi gjort det på den rigtige måde?". Disse spørgsmål skal stilles efter hver fase og kunne besvares med et "Ja", inden man kan gå videre til næste fase (Lytje 2000, p. 178).



Figur 3

Den oprindelige mangel på iteration, samt manglende mulighed for brugerinddragelse, har betydet at Vandfaldsmodellen gennem tiden har været udsat for kritik. Det påpeges, bl.a. af Budde, at iteration er tæt på uundgåelig, da systemer ofte er så komplekse at alle problemstillinger ikke kan forudses på forhånd og man kan således heller ikke tage højde for dem (Budde et al. 1992, p. 28).

Der kan f.eks. være forhold der først viser sig under implementeringsfaser, der gør at man bliver nødt til at ændre i enten designet eller kravspecifikationen og muligvis også i begge.

Visse udviklere mener dog at vandfaldsmodellen kan være et godt udgangspunkt hvis der er kort tid til at gennemløbe et projekt. Under mindre projekter med lav kompleksitet vil det givetvis også have en vis berettigelse, da kompleksiteten i så fald vil være lettere at overskue. Desuden formoder jeg at når man ikke bruger tid på at inddrage brugerne her, vil have en relativ effektiv og billig udviklingsmetode. Forstået på den måde at man kan komme relativt hurtigt fra krav til produkt.

Hvis man mener at man er sikre på hvad man skal udvikle, kan man også mene at brugerne ikke behøver at inddrages i samme grad, som hvis man er mere usikkert på hvad der skal udvikles. Problemet er bare hvordan kan man være sikker på hvad der skal udvikles..?

I det indledende eksempel med KMD vidste udviklerne jo hvad der skulle laves, indtil brugerne blev hør, så var målet pludselig ikke klart mere. Det viser der kan nogle problemer der først viser sig når programmer bruges af de tiltænkte bruger.
Den manglende indflydelse fra brugere i vandfaldsmodellen har affødt en række andre udviklingsmodeller hvor brugeren kommer mere i fokus og ses som en samlet del af problemområdet sammen med systemet. Da det bl.a. er brugerinddragelse jeg interesserer mig for i dette projekt vil enkelte af disse modeller, især prototyping blive grundigt beskrevet herunder.
Disse modeller er overordnet set samlet under overskriftet *participatory design*. Under denne tankegang, som bliver beskrevet herunder, er der mere fokus på brugeren og denne bliver inddraget mere aktivt i selve udviklingsprocessen.

Participatory design

Participatory design, eller brugerorienteret design, som er en af de danske benævnelser, er et mærkat som er sat på de udviklingsmetoder der inddrager brugerne mere aktivt i udviklingsprocessen (Lytje 2000, p. 165).
Traditionen har forsøgt at indføre demokratiske processer i systemudvikling.

> *Brugerbehov opfattes som en slags folkelige rationalitet, der står i modsætning til den form for teknokratisk rationalitet, som den stereotypiske softwareingeniør inkarnerer*.(ibid.).

Denne brugerinddragelse er der flere tilgange til og der findes forskellige filosofier til hvordan brugeren inddrages bedst muligt og ikke mindst hvorfor brugeren inddrages.
To af disse filosofier: *Socioteknikken* og den *skandinaviske tradition* beskrives herunder.

Socioteknikken

I den sociotekniske tilgang opereres der med to forskellige typer af målsætninger for systemudvikling.

Den ene sigter imod effektivitetstermer der skal forbedre virksomhedens økonomi og forretningsmuligheder på kort og på lang sigt.
Den anden type målsætning beskrives i arbejdspsykologiske termer og sigter på at inddrage menneskelige og sociale dimensioner i udviklingsprocessen, på en konstruktiv måde. Dette sker ved at analysere arbejdssituationen, ved at kigge på bl.a. jobtilfredshed og arbejdsorganisering.
Disse målsætninger analyseres særskilt og skal medvirke til at skabe klarhed over hvordan arbejdsgivers og arbejdstagers ønsker kan kombineres og skabe et system begge parter er tilfredse med (Lytje 2000, p.166).
Filosofien bag den sociotekniske tankegang er altså at man gennem demokratiske processor kan kombinere de to parters interesser til et fælles produkt.
Umiddelbart er der noget der tyder på at begge parter skal være indstillet på at indgå kompromis i denne tankegang. Men da det er arbejdsgiver der skal betale for systemet, vil det oftest være ham der er mindst tvunget til at gå på kompromis. Arbejdsgiveren må dog også have en vis interesse i at arbejderne er glade for systemet, da glade arbejdere som oftest også er de mest produktive.
Netop denne skepticisme for hvorvidt de to parters ønsker og interesser kan kombineres danner grundlag for den anden tilgang, den skandinaviske tradition.

Den skandinaviske tradition

Den skandinaviske systemudviklingstradition har sine rødder i den norske socialdemokratiske arbejderbevægelse i starten af 70'erne. I denne tradition allierede Det Norske Jern- og Metalarbejderforbund sig i 1970 med forskere ved Norsk Regnesentral i Oslo. De mente ikke at teknologien kunne optræde som en neutral part og tilgodese både arbejdere og arbejdsgiver. De mente at teknologien - i form af udvikleren - altid ville tage arbejdsgivers parti og blive brugt til rationalisering og effektivisering af arbejdskraften (Lytje 2000, p. 167, 213).
Samarbejdet havde til formål at give tillidsmændene på de enkelte arbejdspladser kompetencer til at deltage i udviklingen af de nye systemer og ikke blot se passivt til. Dette skulle fremme en udvikling der passede

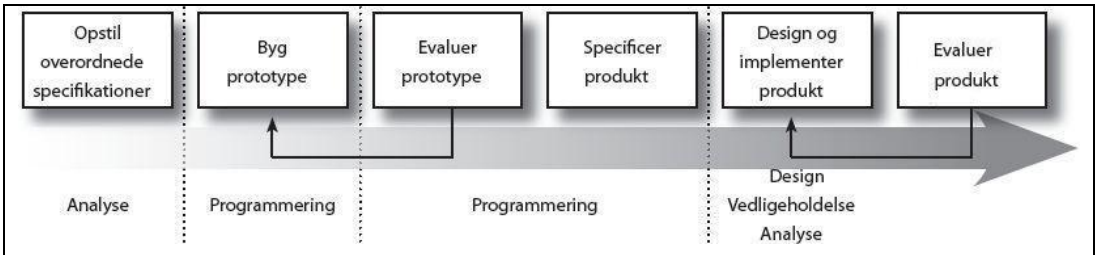
med forbundets og medlemmernes interesser. Med denne tanke ønskede man således at sætte fokus på selve de arbejdsgange der var på arbejdspladserne frem for selve det produkt der var under udarbejdelse (Preece et al. 2002, p. 306).
I begyndelsen af 80'erne begyndte man så at arbejde med deciderede brugerorienterede designstrategier, hvor man via simple mockups i f.eks. karton og papir udarbejdede designforslag. På denne måde kunne de ikketekniske også være med i designfasen (se desuden *Eksplorativ prototyping* p. 36).
Med denne fremgangsmåde ses brugeren ikke kun som sidste led i systemets livscyklus, men også som en aktiv deltager i udviklingsfasen. Bruger og udvikler sættes nu på samme niveau og indgår i et samarbejde (Lytje 2000, p. 220).
Dele af disse arbejdsmetoder blev skabt under et projekt, kaldet UTOPIA-projektet. Det handlede om at undersøge muligheden for at inddrage arbejderne i udviklingsprocessen. En af de værktøjer der blev udarbejdet var bl.a. simple mockup-prototyper og fremtidsværksted (Spinuzzi 2002).

Både den sociotekniske tilgang og den skandinaviske tradition forsøger altså at inddrage brugerne og gøre udviklingsprocessen mere demokratisk. Men folkene bag den skandinaviske tradition vil nok hævde at det sker ud fra forskellige præmisser. Men begge tilgange lægger dog ikke skjul på at brugerinddragelse er en nødvendighed. Inger Lytje (2000) mener i sin bog *Software som tekst* dog ikke at brugerinddragelse alene er vejen frem og at de to tilgange, softwareingineering og participatory design bør kombineres.

I det kommende afsnit vil jeg se nærmere på en udviklingsmetode der har iteration i højsædet og som jeg en fornemmelse af er meget brugt i softwareindustrien.

Prototyping

Prototyper kan bruges mange steder i en udviklingsproces og på mange niveauer. Når man hører eller læser at et firma har udgivet en betaudgave af et produkt, er det faktisk en prototype af et program, der er tæt på at



Figur 4

være færdigt og som lige skal afprøves en sidste gang inden det endelige produkt (eller næste prototype) udgives. Det vil i
dette tilfælde ofte være tale om en relativ avanceret prototype, hvor de hovedparten (hvis ikke alle) af programmets funktioner er implementeret. Denne type prototype vil oftest være mest passende i den sene ende af udviklingsforløbet.
Herover er modellen (Figur 4) vist, som den er blevet illustreret af Christensen og Fischer (2003, p. 34).

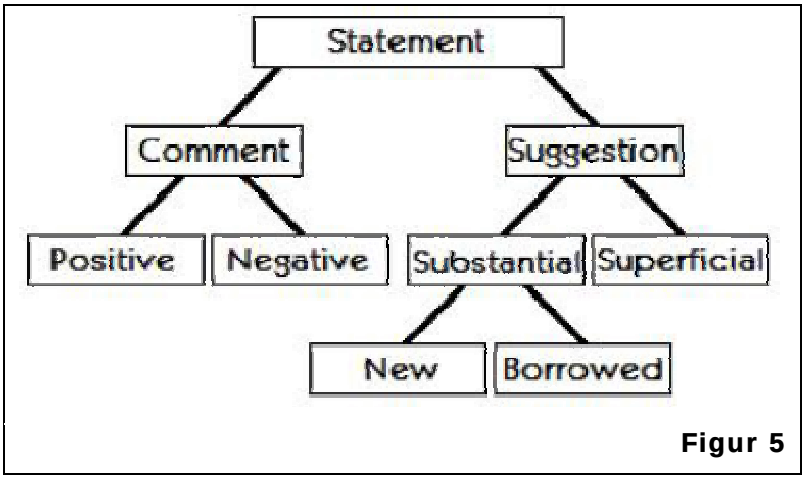
En prototype kan være alt lige fra noget der er lavet i pap/karton, som omtalt før, til mere eller mindre færdige applikationer (Preece et al. 2002, p.241) (se desuden *"Avanceret eller simpel prototype..?"*, p.37.). Prototyper bestående af simple mockups kan i starten af udviklingsprocessen være tilstrækkelige til at give udviklere og især designere, et praj om hvorvidt det tiltænkte design er forståeligt for brugeren. De forskellige prototypers fordele og ulemper vil blive behandlet senere. Først vil jeg kort beskrive hvordan de enkelte faser skal forstås i ovenstående model.

Prototypens faser

Som det fremgår af modellen (jf. Figur 4), består den af seks faser, hvoraf nogle af dem indgår i iterative processer.

Første fase er, som i mange andre udviklingsmodeller, at opstille en række overordnede mål for systemet. I den prototypiske tankegang er disse krav (i modsat til f.eks. vandfaldsmodellen) ikke endelige, men foreløbig og kan ændres undervejs i processen, hvor man gerne skulle blive klogere på brugerens behov, kompetencer og ønsker (Christiansen og Fischer 2003, p. 34 ff.).

I metodens anden fase bygges prototypen, som kan bestå af næsten alt fra simple mockups til power points og halvfærdige systemer. For at nå frem til at bygge prototypen, ser jeg en mulighed for at anvende dele af den føromtalte vandfaldsmodel. Vandfaldsmodellens store force var jo at den var relativ effektiv, så derfor vil man hurtigt kunne få bygget en prototype. Desuden har man de foreløbige krav på plads og bør derfor have en klar idé om hvad det er der skal udvikles. Hvis det så under evaluering viser sig at man tog fejl og dermed fik bygget noget forkert, kan man blive nødt til at gå tilbage og revidere kravene til produktet. Det kan naturligvis også dreje sig om så mindre fejl at man kan nøjes med at redigere prototypen og teste den igen.



Figur 5

Prototypen evalueres sammen med brugeren og derved bliver brugeren (forhåbentlig) i stand til at stille bedre og mere konkrete krav end det var muligt under kravspecifikationen. Disse kommentarer bliver skrevet ind i kravspecifikationen, hvor det er relevant.

Designere fra et projekt omkring udvikling af system til styring af termostater har inddelt brugerens kommentarer i to kategorier: substantielle og overfladiske. En typisk overfladisk kommentar var bl.a. ”Jeg kan ikke lide farvevalget”, mens kommentarer: ”Jeg forstår ikke ordvalgene i menuen” og ”Jeg ville nok ændrer på farven så der kommer større kontrast mellem

baggrunden og teksten", blev kategoriseret som værende substantielle. (Tohidi et al. 2006) (jf. Figur 5 ovenfor).
Denne kategorisering kan på nogle virke lidt subjektive, for hvad hvis farverne primært bestod af rød og grøn og personen der udtrykte din utilfredshed med farvevalget var farveblind, så ville vedkommende nok opfatte det som et substantielt problem.
Denne evaluering foregår (lidt selvsigende) i fasen *Evaluer prototype*. Antallet af iterationer bestemmes bl.a. af projektets budget og hvor mange prototyper udviklerne finder det nødvendigt at lave for at kunne levere et ordentligt produkt. Gennem disse iterationer valideres produktet gang på gang og man øger dermed sandsynligheden for at det endelige system er brugbart.
Når produktet er ved at være færdigt, begynder arbejdet med at specificere det. Der implementeres de sidste kommentarer fra brugerne. Under de sidste to faser *Design og implementer produktet* og *Valider produktet* minder om de foregående iterative processer. Dog med den klare forskel at produktet her er væsentligt tættere på at være færdig.

Denne fremgangsmåde bør desuden give den fordel at hele forløbet bliver veldokumenteret, da man som "restprodukt" kan få en lang række prototyper, brugerkommentarer og reviderede kravspecifikationer, efterhånden som man kommer igennem forløbet.
Som det kort har været nævnt herover så er der forskellige tilgange til hvad en prototype er og forskellige meninger om hvordan man anvender dem. Med det mener jeg at selvom "opskriften" for det meste er den samme, så findes der mange forskellige værktøjer til at nå frem til målet.
Nogle af disse tilgange og brugsmuligheder vil blive beskrevet herunder.

Prototypemetoder

Selvom at prototyping ifølge Lytje (2000, p. 227) opstod som et modstykke til de mere lineære procesmodeller, synes at kan se at den stadig indeholder elementer fra de linere modeller. Men den store forskel er altså at mængden af iterationer er væsentlig højere end i de mere lineære udvik-

lingsstrategier. Der findes forskellige tilgange til prototyping. Hvilken tilgang man vælger er bl.a. styret er hvor langt i processen man er kommet, samt hvad man ønsker at undersøge. Sidstnævnte har bl.a. også indflydelse på valg af prototype (jf. p. 40 ff.) Disse forskellige tilgange der i så fald vil opstå til prototyping har affødt at Floyd, ifølge Lytje (ibid.) har opstillet tre tilgange til – eller formål med– prototyping: *Eksplorativ*, *eksperimentel prototyping* og *evolutionær*.

Eksplorativ prototyping

Hovedfokus for denne tilgang er at klarlægge krav og ønskelige egenskaber for systemet. Her kan alternative løsningsmuligheder diskuteres. Floyd forestiller sig at denne proces foregår i starten af processen hvor systemet skal konkretiseres (ibid. 229). Den har desuden karakter af en gensidig læreproces, hvor bruger og udviklere bliver klogere på hinanden. Typen af protype i den tilgang vil typisk blive et ”byg og smid væk”-produkt, da prototypens formål er opnået når den føromtalte læreproces er opfyldt. Den har altså kun et eksistensgrundlag mens så længe man diskuterer og finder de krav der skal stilles til systemet. Jeg antager at man i denne fase bør overveje at bruge papir/pap-prototyper, da de er hurtige at lave og ikke er for dyre til at de kan smides væk efterfølgende. Lytje er af den opfattelse at denne viden også kan fremskaffes ved at bruger og udvikler i fællesskab besøger arbejdspladser der anvender tilsvarende teknologier eller undersøge systemer, som er designet med samme formål (Lytje 2000, p. 230).

Eksperimentel prototyping

Hovedfokus for denne tilgang er at afgøre hvorvidt et løsningsforslag er passende før man investerer tid i at implementere det fuldt ud. Denne prototype er altså tættere på at være færdig, sammenlignet med eksplorativ prototyping. I denne tilgang høres brugerne ikke så meget til design, men bliver sat til at teste produktet. Designeren observerer brugen af systemet og får derigennem evalueret hvorvidt designet fungerer eller ej. Det kan specielt være givtigt at lægge mærke til om brugeren anvender

systemet på en anden måde end det var tiltænkt. I gennem evaluering skulle designeren gerne få idéer til hvordan designet forbedres (ibid.).

Evolutionær prototyping

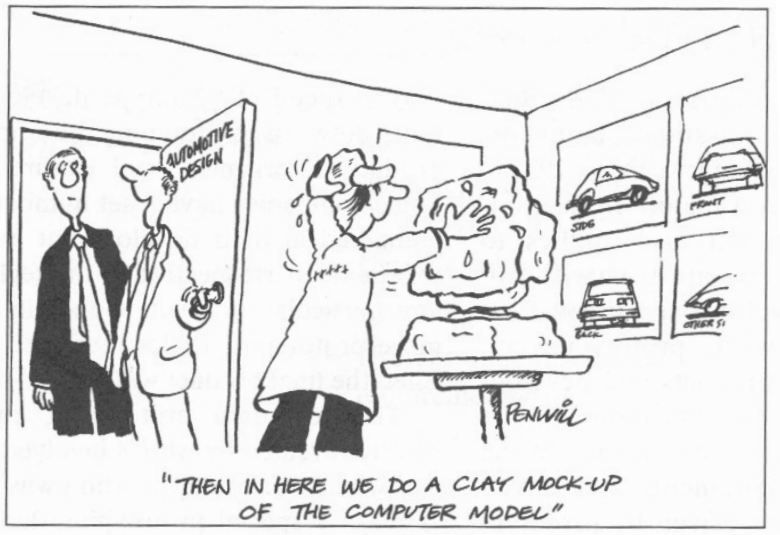
Hovedfokus for denne tilgang går på, gradvist, at tilpasse systemet til ændrede krav, som det ikke ville have været muligt at ændre på et tidligere tidspunkt. Denne tilgang forbinder, ifølge Floyd, systemudvikling med organisationsudvikling (ibid. p 231).
Under denne tilgang afløser den ene version af systemet den forrige og det endelig systems implementering nærmer sig langsom. Den egentlig overgang fra prototype til produkt er meget flydende og det kan derfor være vanskeligt at afgøre hvornår man går fra produktion af ny prototype til "blot" vedligeholdelse af systemet (ibid.). Den store forskel på Eksperimentel- og Evolutionær prototyping består i at sidstnævnte testes i de autentiske omgivelser som det endelige system skal fungere under.

Med disse overvejelser for øje bør man også overveje hvor avanceret ens prototyper bør være alt efter hvor langt i processen man er nået og måske også med henblik på for hvem prototypen skal præsenteres.

Avanceret eller simpel prototype..?

Flere personer inden for HCI kulturen taler om at anvende avancerede og simple prototyper til forskellige formål. Det gør sig bl.a. gældende for Preece et al. (2002, p. 239 ff.) og McCurdy et al. (2006). Desuden refereres der i Budde et al. (1992) til Floyd. Overordnet set snakkes der om *high fidelity* og *low fidelity*, her omtalt som avanceret og simple prototype. McCurdy et al. (2006) opererer desuden med begrebet *mixed fidelity*. Altså en prototype der ikke umiddelbart lader sig definere som entydigt avanceret eller simpel.

Desuden nævnes *vertical prototype* og *horisontal prototype*, som herefter benævnes som dyb og bred prototype.



Figur 6

De har, som det illustreres senere hver især deres fordele og ulemper, samt berettigelse alt efter hvor i processen man befinder sig.

Mathiasen et al. (2001 p. 31) er af den opfattelse at man så vidt muligt (teknisk som økonomisk) bør anvende IT-baserede prototyper, da de ser det som et problem at de mere simple, f.eks. papirbaserede, ikke gør noget af sig selv. Denne holdning er der andre der er uenige i og hilser de papirbaserede prototyper særdeles velkomne, især i den indledende fase hvor designet for systemet skal fastlægges. Det gør sig bl.a. gældende i en nylig undersøgelse af en gruppe HCI-forskere (Tohido et al. 2006). Gruppen undersøgte om antallet af designforslag til en prototype en bruger blev præsenteret for ville påvirke mængden af kvalitativ feedback. Forskerne havde på papir tegnet tre designforslag til et IT-system til styring af termostater. Resultatet var at en gruppe der så alle tre designforslag kom med markant flere brugbare kommentarer end de grupper der kun blev præsenteret for en. Hvis man skulle have lavet 3 halvkørende IT-prototyper ville det have øget udgifterne til denne første prototype betydeligt, uden nødvendigvis at få mere brugbar data som resultat. Desuden skulle programmet kører på en touch screen skærm, så det at brugerne skulle trykke på et stykke lamineret papir for at simulere interaktion med systemet, lå relativt tæt på virkeligheden. En anden undersøgelse har desuden vist at mængden af kvalitativt og kvantitativt data brugeren leverer stort set er den samme hvad enten der anvendes papir- eller IT-baserede prototyper. Brugerne foretrak dog de prototyper der var IT-

baserede (Sefelin et al. 2003, p. 79). På den baggrund kan det umiddelbart anbefales først at gå over til IT-prototyper når ens design er fastlagt. En oversigt over fordelene og ulemper ved hhv. simpel og avanceret prototype er samlet herunder. Tabellen tager udgangspunkt i en tabel af bl.a. Jenny Preece (Preece et al. 2002, p. 46). Teksten i kursiv er egne tilføjelser til punkterne. Listen er dog ikke endelig og der kan sikkert findes flere eksempler på fordele og ulemper.

Type	Fordele	Ulemper
Simpel prototype	o Lavere udviklingsomkostninger o Hurtige at udvikle o Evaluering af flere designforslag o Brugbart kommunikations-redskab o Løser problemer med skærm layout o Brugbart til at identificere markedsbehov o Proof-of-concept	o Begrænset mulighed for at opdage fejl *(primært tekniske)* o Få detaljer/specifikationer at kode ud fra o Facilitator-styret o Begrænset brugbarhed efter at krav er opstillet o Begrænset brugbarhed til brugbarhedstest – *på hvilket plan?* o Begrænsninger på navigering
Avanceret prototype	o Fuldt ud funktionelle *(næsten)* o Fuldt ud interaktive *(næsten)* o Bruger-dreven o Klar navigeringsstruktur o Brugbar til test o Ligner det endelig produkt	o Højere udviklingsomkostninger o Mere tidskrævende at udvikle o Ineffektiv til proof-of-concept designs o Ikke brugbare til kravbestemmelse

	o Tjener som realistisk specifikation o Marketings- og salgs-værktøj	

Figur 7

Man kan hævde at der listes en meget skarp adskillelse mellem avanceret og svag prototype.

McCurdy (2006) fremhæver i en artikel han er medforfatter på, at graden af en prototypes kompleksitet bør være styret af hvad man har til formål at undersøge. Dette må siges at give en hel del mening især med undersøgelser herover for øje. I artiklen af McCurdy et al. består en prototype af fem faktorer som hver især kan være simpel eller avanceret. Disse faktorer er:

- Level of Visual Refinement
  - Hvor tæt er det visuelle på det endelige produkt?
- Breadth of Functionality
  - I hvilken grad er en eller flere funktioner fuldt ud implementeret?
- Depth of Functionality
  - I hvilken grad er alle systemets endelige funktioner repræsenteret
- Richness of Interactivity
  - Hvor meget interaktion tilbyder prototypen?
- Richness of Data Model
  - Hvor meget data er autentisk/virkelighedsnært og hvor meget er fiktivt?

En prototype kan ifølge denne tankegang være simpel eller avanceret indenfor hver enkelt af disse faktorer.

Hvis man f.eks. ønsker at teste en ny type hæveautomater kan man i første omgang ønske at teste en del af systemet. Man kan så lave en bred prototype, der viser de tiltænkte funktioner og så evt. nøjes med at lave få eller ingen af disse dybe. Dermed har man mulighed for at teste om brugeren forstår systemet og brugeren har mulighed for at sige: "Jeg savner personligt en funktion der tillader mig at…".

Alle disse faktorer skal der tages hensyn til når man designer/udvikler en prototype og det lægger i sagens natur op til at man som udvikler og designer må gå på kompromis med nogle af systemets elementer alt efter hvor i processen man befinder sig. Det er f.eks. ikke sikkert at det er nødvendigt at have designet helt på plads ned til sidste pixel under første pro-

totype, men at man her kan nøjes med nogle grove linjer der viser i hvad retning man har tænkt sig at går og så i stedet koncentrerer sig om f.eks. elementernes (menuer og lign.) placering i forhold til hinanden.

Jeg vil i det følgende afsnit forsøge at samle ovenstående betragtninger til et samlet hele og vise hvordan de er indbyrdes forbundne og afhængige af hinanden.

Operationali-sering af sy-stemudvikling

I de præsenterede udviklingsmodeller herover var første fase at udarbejde en kravspecifikation. Det lyder egentlig umiddelbart fornuftigt at starter der. Men som eksempler i indledningen antydede, kan der være en vist rationale i at starte med at få sin målgruppe på plads. Det kan ikke udelukkes at målgruppen kan have en vis indflydelse på udarbejdelsen af kravspecifikationen.
Men inden processen med brugerinddragelse begynder kan det være man har gjort sig nogle dybere tanker om hvorfor man vil have øget fokus på brugerne.
Det kan f.eks. være at en cost/benefit-analyse har vist at man kan spare nogle penge ved at målrette et system mere mod brugernes krav og ønsker. Hvis beslutningen for brugerinddragelse er taget, er spørgsmålene så, hvordan gør vi det?

Udvælgelse af brugere

Når behovet for et brugercentreret design er opdaget er første opgave at definere sine brugere og inddele dem i brugergrupper, såfremt der findes flere segmenter til programmet. Selvom man ikke skulle ønske at inddrage brugerne aktivt i udviklingsprocessen antager jeg at det er en god idé at man har eller laver en beskrivelse af dem. Dette for at øge muligheden for at skræddersy produktet mest muligt til de tiltænkte brugere.

Både ISO-standarden og Jakob Nielsen anbefaler at såfremt man har flere målgrupper og/eller kontekster skal der laves analyser for hver af disse. Det pointeres dog også af bl.a. Nielsen (1993) såfremt budgettet er stramt, bør man koncentrere sig om kernemålgruppen. tidspunktet for inddragelsen af brugerne bør være afspejlet hvad man ønsker at få ud af brugerne. Hvis man fx ønsker at de skal være med til at definere kravene for systemet er det logisk at inddrage dem meget tidligt i processen. Hvis man derimod blot ønsker at bruge dem som testpersoner og fortælle om det der udviklet er godt nok eller ej, skal de inddrages noget senere.

Kravspecifikation

I forbindelse med udvikling af en kravspecifikation er der en række metoder der kan hjælpe med denne proces. Dels kan man blot nøjes med at mødes med kunden. Dette kan dog vise sig ikke at være tilstrækkeligt, hvis vedkommende ikke har erfaring i at købe IT-produkter og/eller ikke har et godt kendskab til systemets brugere og den kontekst det skal fungere i. Såfremt dette er tilfældet formoder jeg at det kan være svært at opliste en række sigende og tilstrækkelige krav til programmet. Der kan muligvis kompenseres for dette hvis udviklerfirmaet har kendskab til den kontekst programmet skal fungere under. Til hjælp til at skabe en bedre kravspecifikation ser jeg bl.a. tre nedenstående metoder.

Scenarier

Scenarier er fiktive brugssituationer med det kommende system. Disse kan hjælpe til med at opdage mindre heldige situationer i det endnu ikke udviklede system. Lausen (Lausen 2005, p. 161 ff.) oplister to typer af scenarier:

1) Den ene type er en lille historie omkring den kontekst systemet skal fungere under.
2) Den anden type beskriver en mere konkret arbejdsopgave der foretages med systemet.

Disse to typer kan hver især have deres forcer.
Hvis vi forestiller os at produktet er et bookingsystem til en hotellobby kan vi måske få følgende ud af at anvende scenaier.

Der har været hektisk aktivitet hele dagen og Peter har nu været på arbejde i 10 timer. Der har været masser at lave og heldigvis har Camilla også været på arbejde så de har kunnet supplere og støtte hinanden. Han har i løbet af dagen bl.a. skulle organisering en turistrundfart den følgende dag, samt stå for organiseringen af aftenens underholdning i en af hotellets barer.

Da Peter endelig får fri efter 12 timer kommer direktøren og spørger hvor mange værelser Peter personligt har lejet ud i løbet af hans vagt.

Ovenstående eksempel indikerer at hoteldirektøren er interesseret i at vide hvor mange værelser hans ansatte hver især lejer ud. Derfor er det vigtigt at det i forbindelse med booking af et værelse kan tilføjes hvem der har lejet det ud.
Et konkret scenarie kan så evt. vise hvordan denne registrering foregår.
Disse små historier kan være med til at skabe diskussioner i løbet af udviklingsprocessen.

Disse scenarier kan udarbejdes på forskellig vis. De bliver typisk udarbejdet af enkeltpersoner (Tedjasaputre et al., 2004). En undersøgelse af bl.a. Tedjasaputre (ibid.) har dog vist at der var en fordel i at lade folk skrive dem i par. Der blev dannet fem par hvor de to personer havde hvert deres ekspertområde, fx en person fra usabilityområdet og en software designer. Gennem skriveprocessen blev deres bidrag til scenariet mere nuanceret og de fik skabt større forståelse for hinandens områder, til gavn i den senere proces. Scenarierne blev derfor mere brugbare, da situationerne blev beskrevet fra mere end én persons synsvinkel. En interessant undersøgelse kunne så være at lade samme to personer fortsætte deres makkerskab gennem længere tid, for at se om deres scenarier bliver ved med at være lige nuancerede, eller om de gradvist ensretter deres tankegange i forbindelse med udarbejdelsen af scenarier.
Jeg kan dog også se noget fornuft i at disse scenarier bliver skrevet af en tekstskribent, fx en journalist el.lign. der har adgang til de faktiske brugere og lader dem fortælle historier om interaktionen med et evt. eksisterende produkt, eller blot historier fra en typisk hverdag. Det kan dog også være at der i udviklingshuset findes personer med kendskab til brugskonteksten, enten i det aktuelle implementeringsmiljø eller fra et tilsvarende projekt. Man kan desuden altid argumenter for, at man kan få dobbelt så mange scenarier for de samme penge, ved at sætte enkeltpersoner til at skrive dem, så der skal ske en afvejning af nytteværdien af at bruge parskribenter.

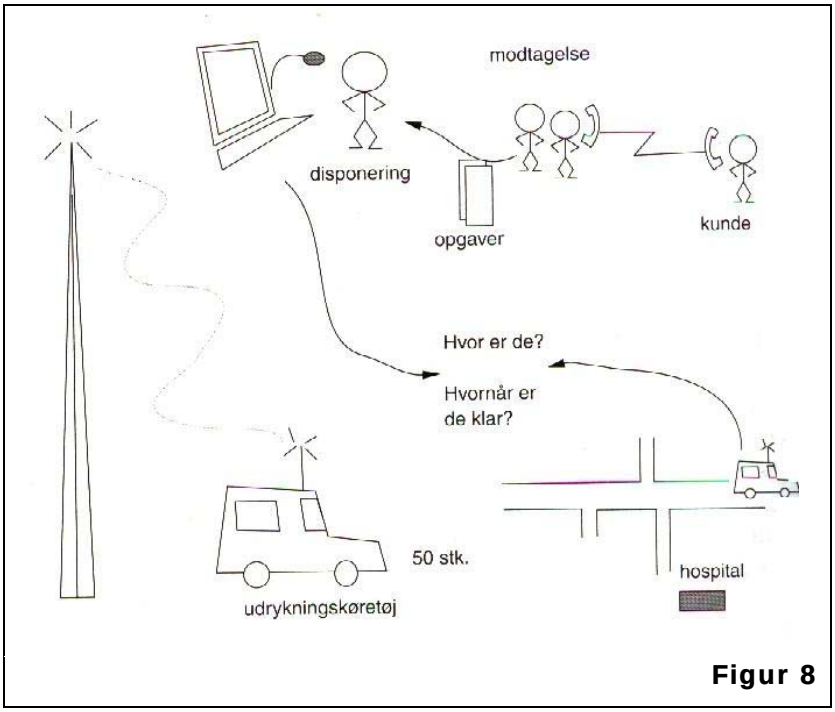
Rige billeder

En metode der kan anvendes både udelukkende internt i udviklingsgruppen og sammen med kunde og/eller de kommende brugere er rige billeder. Rige billeder er små tegnede illustrationer der viser en persons forståelse af en situation. Systemet kan fx vise den proces der sættes i gang når en person flytter fra ét amt til et andet. Der er bl.a. inkluderet en opgave med at få sendt vedkommendes journal fra sygehuset til det nye sygehus. Hvem skal sende den, hvem skal modtage den og hvordan skal den videredistribueres til en bestemt afdeling på det nye sygehus? Dette kan måske være med til at give brugbar information i forbindelse med udvikling af den elektroniske patientjournal.
Alt efter hvad man ønsker at finde ud af, kan man vælge at lave et billede der fokuserer på stabilitet eller forandring (Mathiassen et al. 2001, p. 25). Der oplistes tillige en række råd omkring tegning af rige billeder (ibid. p. 29).

Rige billeder skal bl.a:

1) indeholde meget information og være åbne for fortolkning
2) præsentere processer og strukturer på en sammenhængende og afbalanceret måde
3) indeholde mindst et problematisk område
4) ikke kun fokusere på data og databehandling.

Jeg kan se et muligt problem med de rige i billeder idet der ikke findes én måde at tegne dem på, men tæt på ligeså mange som der er folk der tegner dem. Dermed kan der skabes tvivl om hvordan et rigt billede skal forstås. Dermed ser jeg intet problem i at overholde første råd herover. Denne subjektive måde at afbilde en situation på, kan dog også have den modsatte effekt, at starte en diskussion, der fører til større forståelse af problemet og/eller brugskonteksten. Man kan dog forestille sig at de udviklingshuse der ofte anvender rige billeder har en guide til hvordan de bør tegnes, således at det kun er den afbillede situation der diskuteres og ikke valg af ikoner. Hvis dette er tilfældet kan man sige at de internt i firmaet til en vis grad selv overholder Jakob Nielsens råd om at bevare konsistensen gennem et produkt (jf. p. 22 ff.). Figur 8 ovenfor er et rigt billede der illustrerer en situation for en redningsstation (Mathiasen et al. 2001, p. 396).



Figur 8

Opstartsmøde

I forbindelse med større IT-projekter hvor budgettet tillader det og hvor systemets indvirkning på de kommende brugeres hverdag ønskes undersøg, kan man afholde et opstartsmøde med et repræsentativt udvalg af brugere fra de enkelte brugergrupper. Overvejelser omkring hvordan og hvornår de udvælges diskuteres i afsnittet *Udvælgelse af brugere*.

Under sådanne typiske opstartsmøder laves bl.a. diverse analyser af bl.a. faldgrupper for systemet. Jeg har den opfattelse at disse møder typisk primært består af medlemmer i udviklingsteamet. Jeg ser dog en mulighed for at få brugerne på banen allerede her. Det kan bl.a. ske i form af rollespil eller fremtidsværksted. Hvis vi vender tilbage til eksemplet med den elektroniske patientjournal (EPJ), så kunne man indbyde en række læger, sygeplejersker, hospitalsdirektører og hvem der ellers kunne tænkes at skulle bruge systemet. Jeg mener at have hørt at man på enkelte sygehuse kan have forskellige betegnelser for de samme ting i journalerne. Hvis dette er korrekt antager jeg at man bør inddrage personale fra forskellige hospitaler for at undersøge om dette har nogen indflydelse når en patient overdrages fra et sygehus til et andet.

Efter at brugeren og kravspecifikationen er fastlagt, er man forhåbentlig nogenlunde klædt på til at påbegynde selve udviklingen af programmet.

Udvikling

Hvis man ønsker at følge de principper der er fastlagt for usability skal design (og udviklingsprocessen) være iterative. For at tilgodese dette kan man inddrage prototyping som metode.

Test

Når man har udviklet en prototype eller det endelige produkt skal det som nævnt i *Hvordan prototype?* (jf. p. 21 ff.).
Inden man iværksætter en test er der nogle ting man skal have på plads. Fx skal man

- Beslutte, hvad ønsker vi at undersøge med testen?
- Hvilke(n) test understøtter dette?
- Hvad er vores bruger segment?
- Hvor mange brugere skal vi bruge?
- Hvor skal vi teste henne?

Ovenstående er blot et udvalg af de ting jeg ser man skal have styr på inden man kan teste et produkt. Disse punkter er yderligt uddybet i det følgende afsnit.

Forbedring af test

Inden en test sættes i gang, skal det besluttes hvad man ønsker at undersøge med den (Nielsen 1993, p. 170). Jeg formoder at undersøgelsens fokus har indflydelse på valg af testmetode, ligesom målet har indflydelse på f.eks. valg af prototype (jf. p. 37 ff.). Når fokus så er defineret bør man være bedre rustet til at vælge testmetode.
Inden man kan afholde testen skal der naturligvis udvælges nogle brugere. En fremgangsmåde til dette er nærmere beskrevet under *Udvælgelse af brugere* herover (jf. p. 45).
Samtidig med at brugeren bliver indbudt til testen, skal han have at vide hvordan testen kommer til at foregå. Denne information anbefales gives igen umiddelbart inden testen starter.
En sidste ting man skal overveje er hvor testen skal finde sted. Der er forskellige muligheder og jeg vil blot nævnte nogle stykker.

- i et kontor/lokale i den pågældende virksomhed[3].
- hjemme hos testpersonerne selv.
- i et testlaboratorium eller andet lokale hos udviklerfirmaet.

(Molich 2003, p. 139) og (Lausen 2005, p. 418).

Stederne har hver deres fordele og ulemper, som man bør veje op imod det fokus man har for testen.
De to første steder har begge den fordel at testpersonen ikke behøver at tage et bestemt sted hen for at deltage i testen. Den kan udføres enten på hans arbejdsplads eller i hans eget hjem. Det kan dog have den ulempe at udviklerne skal slæbe udstyr med for at testen kan gennemføres.
Det tredje eksempel herover har den fordel at den person der skal foretage testen har det nødvendige udstyr stående. Et decideret testlaboratorium giver desuden den fordel at udviklerne kan følge med i et tilstødende lokale og dermed selv opleve brugernes interaktion med systemet. Det

[3] Dette kan selvfølgelig kun lade sig gøre såfremt der er tale om et bestillingsprodukt.

har så til gengæld den ulempe, at brugerne skal bruge tid på at rejse hen til det sted hvor testen skal foregå.

Når antallet af testpersoner skal bestemmes er relativ bred enighed om at 4-6 deltagere er et passende antal. Jakob Nielsen mener at det er nok med 5 brugere, da de vil kunne opdage ca. 85 % af usability problemerne, mens det kræves ca. 15 personer hvis alle problemer skal afdækkes (WWW [5]). Molich (Molich 2003, p. 149) mener at der skal være mindst fire fra hver brugergruppe, mens seks personer er det optimale.

Herunder er listet to typer af tests, som hver har deres force som jeg ser det.

Tænke-højt

En kendt test er den såkaldte tænke-højt-test, hvor brugere bliver bedt om at udfører nogle opgaver med systemet. Mens brugeren løser opgaven skal han fortælle **hvorfor** han gør det og ikke hvordan. Metoden er anderkendt af bl.a. Nielsen (Nielsen 2003, p. 195ff) og Molich (Molich 2003, p. 134ff). Nielsen vurderer oven i købet, at hvis man kun har råd til at foretage én type test, så er denne test den bedste at anvende til det formål.
Testen har dog en ulempe, idet det for de fleste personer virker unaturligt konstant at fortælle, hvorfor man gør som man gør (Nielsen 2003, p. 196). Derfor anbefales det også at testdeltageren får nogle lette opgaver til at starte med for at få nogle succesoplevelser inden de evt. lidt svære opgaver introduceres. Jeg har selv oplevet at udføre sådan en test og syntes i starten at det var unaturligt at sige mine tanker højt. Denne følelse blev dog svækket efterhånden som testen skred frem.
For at testdeltageren ikke skal koncentrerer sig om at huske opgaven anbefales det at han får opgaverne på skrift (Molich 2993, p. 143). Desuden anbefales det testdeltageren ikke kender det planlagte antal opgaver, således han ikke føler sig dum, hvis han ikke når alle opgaverne igennem inden for den afsatte tid.
Denne test er altså anvendelig til at undersøge brugerens adfærd og tankestrategi. Dette kan f.eks. give nogle oplysninger om hvor brugeren for-

venter at finde nogle funktioner, eller hvad han forventer en bestemt funktion vil gøre.

Observation

Observation kan bruges på forskellige tidspunkter i processen. Den kan f.eks. bruges i forbindelse med dataindsamling til kravspecifikationen. Her kan man observere brugen af et system magen til det man selv skal udvikle. Det kræver naturligvis at man har adgang til sådanne brugere. Hvis man skal re-designe et eksisterende program kan man observere i den pågældende virksomhed og undersøge hvor det er brugerne oplever problemer, specielt hvis det er grundlaget for et re-design.
Denne test kan fortælle hvorledes brugerne anvender systemet i dets naturlige omgivelser om man vil og ikke kun hvordan brugerne antager de vil bruge det, når de sidder til en test. Jeg tror dog ikke på at testen kan stå alene hvis observatørerne blot observerer. Jeg tror på at brugerne kan have nogle arbejdsgange eller fortager nogle valg som ikke kommer tydeligt og klart frem, hvis ikke de suppleres af nogle kommentarer fra brugerne. Som det skal vise sig i næste afsnit, er det dog ikke altid at brugerne svarer korrekt på opfølgende spørgsmål.
Jeg ser dog den fordel ved observation at brugerne kan bidrage til processen uden at skulle forlade arbejdet og arbejdsgiveren vil muligvis derfor være mere tilbøjelig til at lade brugerne deltage i arbejdstiden, hvilket de sikkert vil sætte pris på.
Jeg ser dog også den fare at dem der bliver observeret føler at der kommer nogle mennesker udefra og kikker dem over skulderne og/eller føle ubehag derved. Derfor antager jeg at man er sikre på at de observerede har det fint med man er der.

Forbehold

I forbindelse med brugertest er der en række forbehold man bliver nødt til at forholde sig til. I forbindelse med observation har det bl.a. vist sig, at når der spørges nærmere indtil hvorfor brugerne gør som de gør, eller hvordan de vil løse en bestemt opgave, så er det ikke altid at de giver det

korrekte svar (Lausen 2005, p. 480ff). Dette kan skyldes at de ikke lige kan svare på det af forskellige årsager og ikke vil tabe ansigt ved at sige: ”Det ved jeg ikke”.
En undersøgelse af bl.a. Shinyoung Park har vist at såfremt brugere har præference for et bestemt brand (i undersøgelsen to forsk. mobiltelefon-producenter) vil det smitte af på de kommentarer brugeren vil komme med i forbindelse med en test af et produkt fra samme firma (Park et al. 2006). Denne undersøgelse indikerer altså at der eksisterer et vist rationale i at undersøge brugernes umiddelbare holdninger til en producent inden de indkaldes til en test.

Hvis produktet efter en eller flere af disse test accepteres af brugerne kan produktet enten gives frit, eller næste prototype kan begynde at blive produceret. Første test kunne jo udelukkende være tiltænkt det design-mæssige, således at de tekniske dele først implementeres i en senere prototype.

Perspektive-ring

Da vi allerede én har fået konstateret at der er en hård konkurrence på softwaremarkedet, må vi antage at firmaerne ønsker at skille sig ud fra mængden. Et af de områder hvor jeg tror dette kan gøres er gennem brugerinddragelse, kundepleje og i det hele taget skabelsen af større helhedsfornemmelse ved køb af software. Dette gælder primært i situationer hvor man har med en specifik kunde at gøre, da man der har stor mulighed for involverede de kommende brugere.

Jeg har på en messe snakket med et udviklingsfirma, som helt seriøst mente at usability var det samme som at tilpasse brugerne systemet og ikke omvendt. Jeg foreslog dem om det ikke kunne være en idé, at indgå en dialog med brugerne, når alt deres arbejde bestod af bestillingsarbejde, hvor kunden kendes og brugerne dermed relativt let kan identificeres. Det blev der blot rystet på hovedet af og sagt at det var der ikke økonomi til. Desuden var de gået over til at udvikle i et system der består af standardkomponenter (for at rationalisere og effektivisere deres udviklingsproces), så evt. tilpasning ville alligevel kun kunne blive relativ. Der lå lidt en ”Vi ved alligevel bedst”-holdning mellem linjerne i samtalen.

Da udviklingsbranchen er en hård branche bliver man altså nødt til at skille sig ud fra mængden og det er ikke sikkert at et godt produkt er nok i sig selv. Jeg har indtryk af at flere og flere også sætter pris på de lidt mere bløde værdier, såsom følger der god service og support med når vi har modtaget produktet, eller er vi (kunden) ”glemt” når produktet er købt og betalt.

Konklusion

Projektet har betragtet en lille del af det enorme og righoldige teorimateriale der findes om usability og brugercentreret design. Dette blev gjort gennem refereret viden i diverse bøger og videnskabelige artikler, primært baseret på empiriske undersøgelser. Min undren inden projektet gik på om der var et rationale i (øget) usability eller om det bare var et modefænomen. Jeg havde dog en formodning om at der ville eksistere en række rationelle begrundelse for øget usability. Denne formodning viste sig at holde stik, da der igennem historien har været flere eksempler på hvor firmaer har sparet mange ressourcer ved at investere lidt tid og penge i usability – også i allerede implementerede produkter. Et rationale kunne således være i form af en målbar økonomisk gevinst, som kan ske på forskellige måder. Nogle projekter affødte at medarbejdere fik effektiviseret deres arbejdsprocesser, mens andre oplevede øget brugertilfredshed.

For at nå frem til dette eksistere der en række metoder, som i varierende grad søger at inddrage brugerne direkte. Dette gælder bl.a. for fremtidsværksted og brugertest. Til hjælp til at holde resultaterne fra brugertestene op imod noget, anbefales det at bl.a. ISO-standarden og Jakob Nielsen at der defineres nogle klare og kvantitative mål, hvilket i ISO-standarden blev ekspliciteret med tre parametre: *Effectiveness*, *efficiency* og *satisfaction*.
Det kan så altid diskuteres hvornår et system er effektivt, men det må være en variabel parametre fra projekt til projekt.
At brugerne skal inddrages aktivt er der bred enighed om, men hvornår de skal inddrages kan der ikke svares entydigt på. Det må ske ved en vurdering hvad man ønsker at bruge brugerne til.
Helt overordnet set kan det siges at det råd som stort set alle de tekster jeg har beskæftiget mig med giver videre og trækker meget på er den iterative proces. Derfor må det forventes at én brugersession næppe er tilstrækkelig hvis man for alvor vil sætte fokus på usability og brugercentreret design.

Hvis yderligere forhold skulle undersøges har jeg en forventning om at det vil være interessant gå mere i dybden med at under de deltagere der eksisterer i en udviklingsproces med henblik på at afdække hvilke magtfak-

torer de hver i sær har. Skal programmøren fx holde sig udelukkende til kunden eller brugerne siger, eller skal han betragtes som en rådgivende konsulent, der har ekspertviden inden for programmering, men til gengæld måske ikke ved så meget om implementeringskonteksten… Og hvad med kunden? Har kunden altid ret fordi det er ham der skal betale produkt?

Proces-beskrivelse

Dette projekt tog afsæt i en interesse for usability og øget brugervenlighed af diverse gadgets og IT-programmer. Dette med en formodning om at et øget fokus på brugerinddragelse og brugervenlighed vil kunne skabe en differentiering i forhold til ens konkurrenter. De fordele der blev oplistet gennem artikler og bøger om området var svære at modargumenter og mængden af ulemper var i undertal. Det kan så altid diskuteres hvorvidt dette skyldes at størstedelen af de inddragede artikler stammer fra diverse Human Computer Interaction-konferencer.
I forbindelse med udarbejdelse af dette projekt læste jeg at læse en række artikler og bøger inden for området. Det viste sig desværre at jeg havde sat for meget tid af til at læse i forhold til at skrive. Dette betød at jeg havde notater fra en masse kilder uden at have et klart overblik over hvorledes de var indbyrdes enige og uenige. Det har betydet at der visse steder er foretaget en lettere forhastet sammenligning mellem diverse kilder under udarbejdelse af projektet.

Foruden en udvidelse af min viden inden for usability-området, har jeg altså også fundet ud af at jeg skal arbejde en del med at få planlagt den tid jeg har til rådighed i kommende projekter. I forbindelse hermed tror jeg, at jeg for fremtiden vil kunne drage nytte af et eller andet system, der kan assistere med mig at holde styr på langt jeg var nået i forhold til en planlagt tidsplan. Dette kan både være analogt og digitalt.

Litteraturliste

AYKIN, N., 2003. User-Centered Software Design and Development: Ensuring Customer Satisfaction. *In:* J. Jacko, C.
Stephanidis, eds. Human-Computer Interaction: Theory and Practice, part 1 volume 1.
Mahwah, USA: Lawrence Erlbaum Associates, Incorporated page, 424-428.

BERG, S. M., 2005a, Kravspecifikation betaler sig. *ComON*, 21. april [online].
Tilgængelig fra: http://www.comon.dk/index.php/news/show/id=21931
[Tilgået 13. maj 2005].

BERG, S. M., 2005b, Kravspecifikation betaler sig. *ComON*, 21. april [online].
Tilgængelig fra: http://www.comon.dk/index.php/news/show/id=21930
[Tilgået 13. maj 2005].

BUDDE, R., KAUTZ, K., KUHLENKAMP, K. og ZÜLLINGHOVEN, H., 1992. *Prototyping – An Approach to Evolutionary System Development.* Berlin: Springer-
Verlag.

CHRISTENSEN, M., OG FISCHER, L.H., 2003. *Udvikling af multimedier – en*
helhedsorienteret metode. 2.udg. København: Ingeniøren-Bøger.

JOKELA, T., 2003. Beyond Usability Methods: Usability Engineering Through Processes and Outcomes. *CutterITjournal*, 16 (10), 13-20.

JOKELA, T., IIVARI, N., MATERO, J. OG KARUKKA, M., 2003. The Standard of User-Centered Design and the Standard Definition of Usability: Analyzing ISO 13407 against ISO 9241-11. *ACM International Conference Proceeding Series; Vol. 46, August 17-20 2003 Rio de Janeiro.*
New York: ACM Press, 53-60.
LAUSEN, S., 2005. *User Interface Design.* A Software Engineering Perspective. Harlow: Pearson education limited.

LYTJE, I., 2000. *Software som tekst – en teori om systemudvikling*. Aalborg: Aalborg Universitetsforlag.

McCURDY, M., CONNORS, C., PYRZAK, G., KANEFSKY, B. OG VERA, A., 2006. Breaking the Fidelity Barrier: An examination of our Current Characterization of Prototypes and an Example of a Mixed-Fidelity Success. *Conference on Human Factors in Computing Systems, April 22-27 2006 Montréal, Québec.*
New York: ACM Press, 1233-1242.

MOLICH, R., 2003. *Brugervenligt webdesign*. 2. udg.
København: Ingeniøren|bøger.

NIELSEN, J. 1993. *Usability Engineering*.
London: Academic Press

NIELSEN, J., 2001. *Godt webdesign*.
Valby: IDG Forlag.

PARK, S., HARADA, A., IGARASHI, H., 2006. Influence of Personal Preference on Product Usability. *Conference on Human Factors in Computing Systems, April 22-27 2006 Montréal, Québec.*
New York: ACM Press, 87-92.

PREECE, J., ROGERS, Y. OG SHARP, H., 2002. *Interaction Design – beyond human-computer interaction*. John Wiley & Sons, Inc.

SEFELIN, R, TSCHELIGI, M., OG GILLER, V., 2003. Paper Prototyping – What is it good for? A Comparison of Pater- and Computer-based Low-fidelity Prototyping. *Conference on Human Factors in Computing Systems*, April 05-10 2003 Ft. Lauderdale.
New York: ACM Press, 778-779.
SPINUZZI, C., 2002. A Scandinavian challenge, a US Response: Methodological Assumptions in Scandinavian and US Prototyping Approaches. *ACM*

Special Interest Group for Design of Communications, Oktober 20-23 2002 Toronto, Ontario.
New York: ACM Press, 208-215.

TEDJASAPUTRE, A., B., SARI, E., R., STROM, G., 2005. Sharing and Learning through Pair Writing of Scenarios. *ACM International Conference Proceeding Series; Vol. 82, October 23-27 2004 Tampere, Finland.*
New York: ACM Press, 229-232.

TOHIDO, M., BUXTON, W., BAECKER, R. OG SELLEN, A., 2006. Getting the Right Design and the Design Right: Testing Many Is Better Than One. *Conference on Human Factors in Computing Systems, April 22-27 2006 Montréal, Québec.*
New York: ACM Press, 1243-1252.

WWW

[1] ANON, 2006. *Worldstats.*
Tilgængelig fra: http://www.internetworldstats.com/stats.htm
[Tilgået 12.maj 2006].

[2] DALL, S., 2000. *Amanda stresser AF-personale.*
Tilgængelig fra: http://ing.dk/article/20000913/ARKIV/109130880
[Tilgået 12. maj 2006]

[3] Danmarks Statistik., 2006. *Nøgletal om informationssamfundet Danmark – 2006 Danske tal* [online]. København, Danmarks Statistiks trykkeri.
Tilgængelig fra:
http://www.dst.dk/Statistik/ags/IT/NinfoDK2006.aspx
[Tilgået 9. maj 2006].

[4] DILIDDO, A., MILLER, S., MAYER, J., MOY, R., WALLANCE, D., HALLEN, E. OG MONGE, T., 2005. *Our Favorite Technology Quotations* [online].
Tilgængelige fra:
http://whatis.techtarget.com/definition/0,,sid9_gci534467,00.html
[Tilgået 14. maj 2006].

[5] NIELSEN, J., 2000. *Why You Only Need to Test With 5 Users*. Tilgængelig fra: http://www.useit.com/alertbox/20000319.html [Tilgået 18.maj 2006].

[6] NIELSEN, J., 2000. *Usability 101: Introduction to Usability*. Tilgængelig fra: http://www.useit.com/alertbox/20030825.html [Tilgået 19.maj 2006].

[7] TEIMANSEN, E., 2001. *Pc-en fyller 20* [online]. Oslo, Dagbladet.no Tilgængelig fra: http://www.dagbladet.no/dinside/2001/08/13/274563.html. [Tilgået 14. maj 2006].