Claus Kirkegaard Clausens selvstændige reflekterende afsnit <20034435>

Vi har i vores projekt Central MONitor System User Interface(CMONUI), udviklet en brugergrænseflade til systemet Centralt MONitoring system(CMON), hvis formål er at overvåge patienter på et sygehus. CMON bliver lavet sideløbende med CMONUI. CMON bliver skrevet i C++, og lavet som en dll fil, hvorved den simpelt kan integreres med CMONUI. Alle data CMONUI skal bruge, kan således hentes igennem et velspecifikaseret interface, og vi kan derfor koncentrere os 100 % om at lave brugergrænsefladen.

CMONUI er lavet så det kan køre på en computer med en touchscreenbaseret skærm.

Udviklingen af CMONUI er forløbet over 2 iterationer. I første iteration lavede vi den del af systemet som giver os oversigt over alle patienterne på sygehuset. Ud fra oversigten udvælges så de enkelte patienter som skal overvåges, og de udvalgte patienter overføres til en specifik overvågningsside. Her kan man klikke på hver enkel patients navn og blive navigeret ind til patientens dataview, hvor patientens informationer bliver præsenteret for brugeren.

Under udviklingen af første iteration havde vi de tekniker vi havde lært i BRGA i baghovedet, med det formål, at mindske fejl ved en afsluttende test. Et eksempel er hele vores første iteration. I stedet for at have en overvågningsside, kunne vi have lavet en side der overvågede alle patienter. Dette konkluderede vi dog ville blive for overvældende, da en alarm ville resultere i, at en sygeplejerske skulle gå hen til overvågningsskærmen, og muligvis scrolle op og ned på skærmen for at finde ud af hvilken patient alarmen tilhørte. I stedet valgte vi, at det var bedst at sygeplejersken i ro og mag kunne udvælge hvilke patienter hun ville overvåge, og hvis en alarm så indtræf, kunne hun hurtigt se hos hvem af de få udvalgte alarmen kom fra. Det ekstra view blev tilføjet for at mindske sygeplejerskens 'memory load' i stressede situationer[Nielsen-93 afsnit 5.3].

Til at evaluere første iteration af brugergrænsefladen benyttede vi os af metoderne Cognitive Walkthrough [Abowd-95] og Heurastic Evaluation[Nielsen-93 s. 20]. Selvom vi havde haft de to metoder i tankerne da vi implementerede programmet, fandt vi dog en del mangler. Blandt andet var der ikke nok tilbagemelding fra programmet, hvilket var et problem, da vores intention var at det skulle være et 'walk up and use' program, hvilket vil sige, at en hvilken som helst sygeplejerske eller læge, skulle kunne bruge det uden introduktion. Cognitive Walkthrough og Heurastic Evaluation er gode metoder til at teste sit program med, og vi følte selv vi fandt størstedelen af fejlene. Problemet var dog, at det var os selv der havde udført testene, og det viste sig senere, at vi var lidt for bekendte med hvordan vores program virkede.

Da programmet skulle køre på et sygehus, og kunne være ansvarlig for patienters liv og død, var det nødvendigt at det var hurtigt at navigere rundt i. For at vise at det var tilfældet, lavede vi en 'keystroke analysis' [MacKenzie afsnit 2.1.2]. Dette viste sig at være en god metode til at bestemme hvor hurtigt man kunne navigere rundt i systemet. Det sted hvor brugeren brugte længst tid, var viewet over samtlige patienter, hvor han/hun skulle udvælge hvilke patienter der skulle overvåges. Det var dog ikke et problem, da det var noget der skulle gøres sjældent.

Vi startede vores anden iteration med en'Thinking aloud' test, hvilket ifølge Nielsen[Nielsen-93 afsnit 6.8], er den mest værdifulde usability metode. Vores testresultat giver ham ret. Vi udførte 2 tests. En på en sygeplejerstuderende i hendes hjem, og en på en færdig uddannet sygeplejerske i hendes vante arbejdsomgivelser på Århus Kommunehospital. Begge forsøg gav os megen feedback at arbejde videre med. Testen på Århus Kommunehospital udviklede sig nærmest til en 'Constructive Interaction', da der var andre sygeplejerske i testmiljøet. De kom hele tiden forbi for at kigge på testen, og for at snakke med testpersonen. Dette medførte, at testpersonen næsten konstant snakkede om programmet, enten med os, eller med andre sygeplejersker.
Forsøgspersonerne fandt en del fejl. Blandt andet havde de problemer med at komme tilbage fra forskellige views, da de følte de manglede en tilbageknap. Vi havde under udviklingen af touchscreenen haft elo's 10 touchscreen applikaition tips[ELO-04] i tankerne, og derfor valgt at gøre brugergrænsefladen så intuitiv så muligt, samtidig med vi guidede brugeren om valgmuligheder. Punkt 7:

```
Test your application on focus groups. If users pause
in confusion—even for a moment—you've identified
the areas than need improvement[ELO-04 punkt 7].
```

viste os dog, at der stadig var plads til forbedringer.
Konklusionen bliver derfor, at testen med rigtige brugere var den der identificerede og fandt flest fejl, samt gav inspiration og ideer til videre udvikling af systemet.

I anden iteration videreudviklede vi også CMONUI med et Voice User Interface(VUI). Ideen bag funktionen med et VUI var, at en sygeplejerske skulle kunne udføre andet arbejde samtidigt med at hun betjente CMONUI. For ikke at tilføje for meget forvirrende information til touchscreenen besluttede vi i stedet, at en bruger skulle igennem et fem minutters kursus, hvor man kunne tilegne sig de forskellige voice kommandoer.
Vi testede selv VUI med HE og CW, hvilket ikke gav nogen fejl. Forudsætningen for HE og CW testene, var det førnævnte fem minutters introduktionskursus. Vi mener, at de fem minutter burde være nok til at kunne navigere rundt i programmet uden besvær. Men siden vi ikke har haft tid til at udforme og undersøge et sådan introduktionskursus, kan dette naturligvis variere.
Det ærgrer mig meget at vi ikke havde tid til at teste vores VUI på testpersoner. Erfaringerne fra vores første brugertest er, at det er her den bedste feedback på programmet kan findes. Især med stemmegenkendelse kan det være et problem at det er udviklerne der tester, da de ifølge Yankelovichs har en højere genkendelsesrate end almindelige testpersoner [Yankelovich-95 side 3 tabel 2][1].
Vores VUI virkede på den måde, at hver gang den havde forstået en kommando, gav den et svar tilbage om, hvilken handling den havde udført. Hvis en bruger f.eks. sagde 'Select 1', ville vores VUI svare tilbage med 'Patientens navn seleced'. Den syntetiske stemme er i forhold til almindelig

[1] [Yankelovich-95] er den ekstra artikel jeg bruger i mit selvstændige afsnit. Artiklen er fundet på http://citeseer.ist.psu.edu/ og omhandler udvikling og test af et speech user interface. Artiklen virker meget troværdig, og referer til en del materiale som er opgivet som pensum i faget BRGA. Heriblandt Jakob Nielsen.

stemmeføring langsom. Fordi det er en handling der kan udføres gentagende gange og fordi patientnavnene kan være lange, vil det virke som lang tid for brugeren at benytte VUI'en. At den syntetiske stemme virker langsom for brugeren viser en udtalelse fra en af Yankelovichs testpersoner:

```
One user commented: "I had to get adjusted to
it in the beginning...I had to slow down my reactions." [Yankelovich-95 side
4].
```

Computerens svar tilbage til brugeren, er dog med til at forsikre brugeren om at han har valgt rigtigt, og de kan derfor ikke udelades fra programmet. En løsning kunne være kortere, men mere præcise output.

Et af de største problemer med stemmegenkendelse er, at der kan opstå fejl når computeren forsøger at genkende hvad der blev sagt. Der er tre typer af fejl, som vi alle tager hånd om i vores projekt. De tre typer er som følger; rejection, substitution, and insertion[Yankelovich -95 side 6].

Rejection er, når computeren ikke forstår hvad der er blevet sagt. I vores tilfælde skal brugeren være klar over, at computeren reagerer på alt hvad den hører, så hvis den ikke reagerer ved brugeren derfor, at den ikke har forstået hvad der er blevet sagt.

Substituition er, når et ord misforstås som et andet ord. Vi har prøvet at undgå dette i vores program, ved at lytte efter så få ord som muligt. Skulle det alligevel ske, vil brugeren blive gjort opmærksom om hans valg, hvorefter han kan rette det. Denne fejlkilde er også en af grundene, at det er en god ide at bibeholde den syntetiske computerstemme.

Insertion er, når baggrundsstøj bliver opfattet som en kommando. Dette var noget der ofte skete for os under de første test. Vi løste problemet ved at lave en kommando, hvorved vi kunne slå stemmestyring til og fra, på samme måde som lægerne styrer Hermes på Hvidovre hospital[Pol-04].

Begejstringen i gruppen var meget spredt når det angik speech recognition. Vi var ikke lige begejstrede for hvor mange ord den genkendte, eller hvor værdifuldt det var i en applikation. Min egen holdning er, at det er ufattelig interessant. Jeg føler, at genkendelsesraten var rigtig god, og jeg kan forestille mig mange nye applikationer eller eksisterende der kan laves/forbedres med speech recognition, hvilket jeg ikke er ene om[Yankelovich-95 side 1].

Vi overvejede længe at udvikle et mobilt userinterface, men vi konkluderede, at der ikke var et reelt behov. Systemet virker på den måde, at når en sygeplejerske/læge ser/hører en alarm, begiver han sig straks ind til patienten, hvor patientens vitale data bliver vist.

Claus Kirkegaard Clausen