Indhold

[Key concerns for en designer er: 3](#_Toc493573605)

[Design: 3](#_Toc493573606)

[Pact: 6](#_Toc493573607)

[People - Menneskers forskellighed: 6](#_Toc493573608)

[Aktiviteter: 7](#_Toc493573609)

[Kontekst: 7](#_Toc493573610)

[Teknologier: 8](#_Toc493573611)

[Input: 8](#_Toc493573612)

[Output: 8](#_Toc493573613)

[Kommunikation: 8](#_Toc493573614)

[Indhold: 8](#_Toc493573615)

[”Scoping a problem with PACT”: 8](#_Toc493573616)

[Forstå kriterier: 9](#_Toc493573617)

[Kriterietyper: 9](#_Toc493573618)

[Dokumentation af kriterier: 9](#_Toc493573619)

[Funktionel: 9](#_Toc493573620)

[Ufunktionel: 9](#_Toc493573621)

[Prioritering af kriterier: 9](#_Toc493573622)

[MoSCoW regler: 10](#_Toc493573623)

[Deltagelsesbaseret design: 10](#_Toc493573624)

[Interviews: 10](#_Toc493573625)

[Strukturerede: 10](#_Toc493573626)

[Semi-strukturerede inteviews: 10](#_Toc493573627)

[Ustrukturerede: 10](#_Toc493573628)

[Contextural inquiry: 10](#_Toc493573629)

[Historier, scenarier og prototyper i interviews: 11](#_Toc493573630)

[Tænk højt kommentarer: 11](#_Toc493573631)

[Praktiske overvejelser i interviews: 11](#_Toc493573632)

[Forberedelse: 11](#_Toc493573633)

[Hold styr på interviewet: 11](#_Toc493573634)

[Fortælle historier: 11](#_Toc493573635)

[Reflektion og undersøgelse: 11](#_Toc493573636)

[Undersøgende spørgsmål i normale situationer: 11](#_Toc493573637)

[Hvornår skal man stoppe?: 12](#_Toc493573638)

[Spørgeskemaer: 12](#_Toc493573639)

[Crowd sourcing: 12](#_Toc493573640)

[”Cultural probes” 12](#_Toc493573641)

[Aktiviteter: 13](#_Toc493573642)

[Historier: 14](#_Toc493573643)

[Konceptuelle scenarier 14](#_Toc493573644)

[Konkrete scenarier 14](#_Toc493573645)

[Use cases 14](#_Toc493573646)

[Brugbarhed: 14](#_Toc493573647)

[Probes 16](#_Toc493573648)

[”Card sorting techniques” 16](#_Toc493573649)

[Arbejde med grupper: 17](#_Toc493573650)

[Brainstorm: 17](#_Toc493573651)

[Benarbejde – observering af aktiviteter i ”situ” 17](#_Toc493573652)

[Arbejdsplads undersøgelser. 17](#_Toc493573653)

[Artefakt indsamling og ’desk work’ 17](#_Toc493573654)

[Suppler tidligere artefakt og dataindsamling med ting fra omgivelserne. Notesbøger, dokumenter, noter, telefon. Læs info on kunden. Se på lignende produkter – god måde at se fordele og ulemper. 17](#_Toc493573655)

[Kontextual design 17](#_Toc493573656)

[”Contextual inquiry” 18](#_Toc493573657)

[Kontekst 18](#_Toc493573658)

[Partnerskab 18](#_Toc493573659)

[Fortolkning 18](#_Toc493573660)

[Fokus 18](#_Toc493573661)

[Praktikaliteter i kontekstuelt interview 18](#_Toc493573662)

[Passende repræsentationer 18](#_Toc493573663)

[Basale teknikker 18](#_Toc493573664)

[Skitser og snapshots 18](#_Toc493573665)

[Storyboard 18](#_Toc493573666)

[Mood boards 19](#_Toc493573667)

[Navigations kort 19](#_Toc493573668)

[Wireframes 20](#_Toc493573669)

[Andre teknikker: 20](#_Toc493573670)

[Prototyper: 20](#_Toc493573671)

[Hi-fi prototyper: 20](#_Toc493573672)

[Lo-fi prototyper: 20](#_Toc493573673)

[Video prototyper: 20](#_Toc493573674)

[Andre typer: 20](#_Toc493573675)

# Key concerns for en designer er:

* Design. Hvad er design og hvordan skal du gøre det?
* Teknologier. De interaktive systemer, produkter, anordninger og komponenter i dem selv.
* Personer. De der vil bruge systemet, og hvis liv ville blive forbedret af dette.
* Aktiviteter og kontekst. Hvad har folk lyst til at gøre. I hvilke sammenhænge finder de aktiviteter sted?

## Design:

Definition: både den kreative proces hvor man specificerer noget nyt, og til repræsentationerne der er produceret i processen.

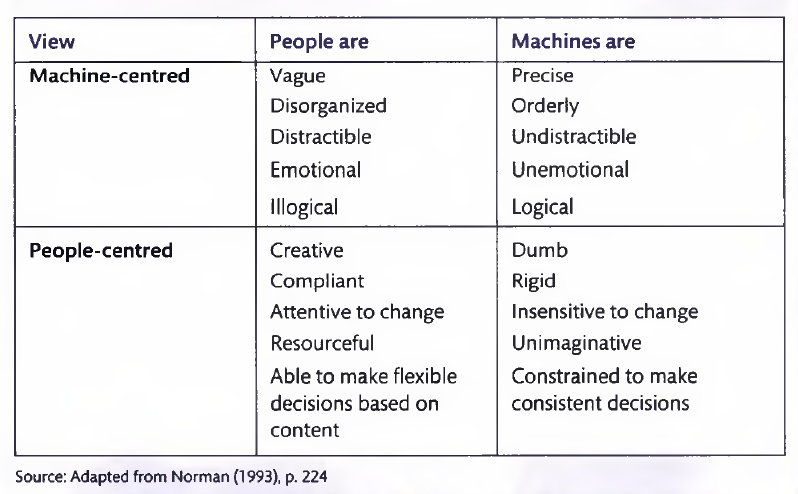
Formaliteter i forbindelse med design:

* Ingeniør design (fx design af bil, hus eller vej). Her bruges videnskabelige fremgangsmåder før konstruktionen begyndes.
* Kreativ og kunstnerisk design. Her bruges innovation, fantasi og konseptuelle ideer som nøgle ingredienser.
* I midten ligger der ’design as craft’. Dette er en blanding af både ingeniør og kreativ fremgangsmåde.

For det meste bruges alle aspekterne i større eller mindre grad. Et design skal være en samtale med materialer - en designer skal altid forstå materialernes natur.

Begrebet ”*Interactive system”* dækker over komponenter, apparater, produkter og software systemer der primært omhandler behandlingen af information.

Interaktive systemer er ting der håndterer transmission, display, storage eller transformation af information mennesker kan opfatte. De skal dynamisk reagere på folks handlinger.



## Interfacet(UI):

Alle dele af systemet folk kan komme i kontakt med, fysisk, perceptuelt , konceptuelt.

Fysisk kan det være at trykke på knapper.

Perceptuelt er det at apparatet laver lyde vi kan høre og displayer noget vi kan se.

Konceptuelt interagerer vi med apparater ved at finde ud af hvad det gør og hvad vi burde gøre. Det giver beskeder som er designet til at hjælpe med at forstå hvad vi skal.

Når man designer interaktive systemer skal man ikke kun designe UI, man tage hele interaktionen mellem mennesker og computere i mente. Man skal tage hele situationen i betragtning.

## At være menneske centreret:

Sæt mennesker først. Design interaktive systemer for at understøtte folk, og som de nyder. Det omhandler at

* Tænke over hvad mennesker har lyst til at gøre, frem for hvad teknologi kan gøre
* Designe nye måder at forbinde mennesker med mennesker
* Inkludere folk i design processen
* Designe for forskellighed

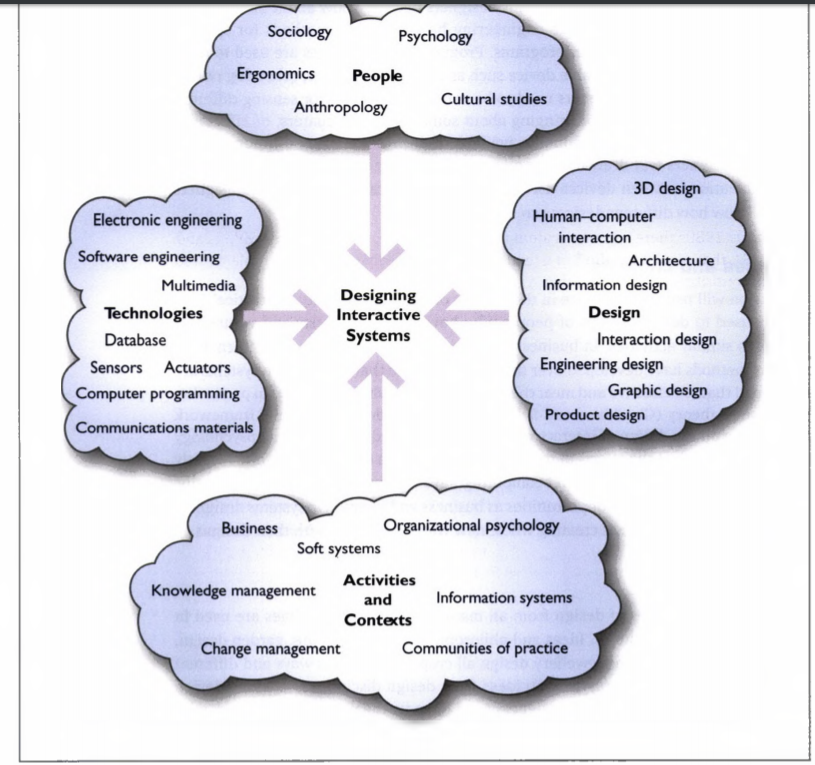
Det er dyrt, man skal observere og tale med folk, og teste ideer på folk. Det er vigtigt af en række grunde:

1. ”Return on investment”: god brugervenlighed resulterer i færre opkald på kunde hjælpelinjer og højere salg. Selskabet ”User Interface Engeneering” påstår at man kan øge salget med 225 procent ved at øge menneskefokuset.
2. ”Sikkerhed”: Det er vigtigt især i systemer der kan være farlige hvis ikke opereret ordentligt, at operatører forstår displayet korrekt. Dette gælder fx atomkraftværk og fly.
3. ”Etikette”: Det er vigtigt at overholde etikken når man designer et system. Dette gælder både deling af brugerdata, brugen af andre menneskers billeder og andet produceret materiale. AMC(Association of Computing Machinery) har gode råd om at overholde god etik i et design.
4. ”Bæredygtighed”: Lokale sprog kan dø ud på grund af at alt information er på engelsk, kinesisk og hindi. Store displays bruger meget strøm osv. Menneske centreret design bør anerkende diversitet og forsøge at forbedre menneskelige værdier.

**Community of practice:** grupper af mennesker der deler interesser og værdier, og deltager i ens aktiviteter

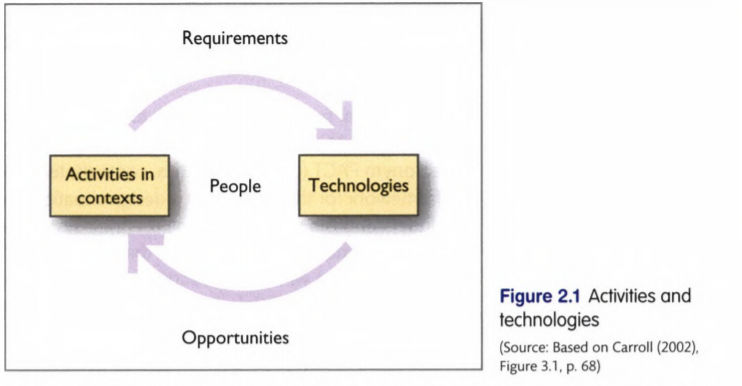
Product design er vigtigt når man designer interaktive systemer

Grafisk design og informations design er vigtigt for informations layout, forståelse og æstetik af et produkt.



# Pact:

Skal gentages mange gange



Teknologi giver muligheder for bedre produkter, mens produkter kræver en hvis teknologi.

Domæne: her sfære af aktivitet

## People - Menneskers forskellighed:

1. Fysik: Fx vægt, højde, sanser osv. Dette kan have en stor effekt på hvor tilgængelige, brugbare og glædesfremkaldende systemer er for folk.
2. Ergonomi: studiet af forholdet mellem mennesker og omgivelser. Se Fitt’s lov (siden 30)
3. Psykologiske forskelle: Folk har forskellige psykiske forcer. Nogle har fx en god stedsans mens andre ikke har. Kulturmæssige forskelle påvirker også hvordan folk opfatter ting.
4. Mentale modeller: Den forståelse og viden vi har. Hvis man ikke rigtig forstår noget på et bestemt område og det går galt, vil man ikke kunne finde tilbage til da det virkede. Et nøgle design princip er at designe ting så mennesker vil forme en korrekt og brugbar mental model af hvad det gør og hvordan det virker. Man skaber en MM ved at bruge systemer, observere forholdet mellem ageren og hvordan systemet opfører sig, og læse manualer eller andre former for forklaring der kommer med et system. Det er vigtigt at have en god nok forklaring af et system.   
   Normans observationer omkring mentale modeller af interaktive systemer(side 32)  
   - Mentale modeller er ufuldstændige. Man vil forstå nogle dele af et system bedre end andre.   
   - Man kan forsøge at bruge den mentale model når man skal, men ofte er der lille præcision.   
   - De er ustabile – man glemmer detaljer  
   - De har ikke faste grænser. Apparater der ligner hinanden og operationer der ligner hinanden bliver forvekslet   
   - De er uvidenskabelige, udviser overtroisk opførsel.  
   - De er påholdende. Mange er villige til at lægge mere fysisk energi i operationer for at minimere mentalt brug. Fx vil mange genstarte et apparat i stedet for at prøve at finde ud af hvad der er galt.   
   Det kan hjælpe at lave associationer til ting folk kender. På den måde har de lettere ved at forstå det.
5. Sociale forskelle: Folk bruger systemer af forskellige grunde. Der er fx forskel på nybegyndere og eksperters viden, og derfor også hvordan designet skal være. Eksperter bruger et system meget, og derfor må det godt være kompliceret, mens nybegyndere bare gerne vil gøre noget bestemt. Nogle mennesker kaldet *directionary users”* bruger ikke systemet, men designeren vil gerne have at de gør. Der er to grupper man kan designe for, homogene og heterogene. Når man designer til de forskellige grupper er det meget forskelligt. Ved en relativt homogen gruppe, kan det være en god ide at få repræsentanter til at hjælpe med designet.

## Aktiviteter:

De følgende ti aspekter er de vigtigste at overveje når man skal designe en aktivitet. Først og fremmest skal designere fokusere på formålet af en aktivitet. Herefter er de vigtigste aspekter at overveje:

* Tidsmæssige aspekter(1-4)
* Samarbejde(5)
* Kompleksitet(6)
* Sikkehed(7-8)
* Naturen af indholdet(9-10)

1. Tidsmæssigt aspekt: hvor ofte bruges en aktivitet. Hvis noget gøres ofte skal det være let at gøre, og hvis det gøres sjældent skal det være let at lære og huske.
2. Tidsmæssigt aspekt: Tidspres, højde og lavpunkter af arbejdet. Noget der virker godt når ting går stille og roligt, kan være rædselsfuldt når der er travlt.
3. Tidsmæssigt aspekt: Noget vil ske som et uafbrudt sæt handlinger mens andre vil være mere sandsynlige at blive afbrudt. Hvis man ofte bliver afbrudt skal man kunne finde ud af hvad man ar i gang med igen. Det er vigtigt at sikre sig at folk ikke laver fejl og glemmer vigtige ting.
4. Tidsmæssigt aspekt: Respons tiden. Folk kan blive frustrerede og sikkerhedsmæssigt kan det være skidt hvis responstiden er for langsom. Folk forventer omkring 100 millisekunders responstid ved hånd/øje koordination, og 1 sekund når man klikker på en knap og noget sker. Mere end fem sekunder vil være frustrerende.
5. Samarbejdsmæssigt aspekt: Skal aktiviteten laves alene eller med andre? Hvis med andre er det vigtigt at kunne kommunikere og koordinerer korrekt.
6. Kompleksitet: Veldefinerede opgaver skal have forskellige designs fra uklare. Med veldefineret opgave kan man bruge simpelt step by step design. En vag aktivitet skal folk have lov til at kigge rundt, se forskellige typer information.
7. Sikkerhed: Nogle aktiviteter er sikkerheds kritiske. Her skal man lægge ekstra mærke til om man kan lave fejl.
8. Sikkerhed: Hvad sker der når folk laver fejl?
9. Natur af indhold: Hvad er data krav af aktiviteten. Hvis meget data skal bruges til input, skal man næsten helt sikkert have et tastatur. Nogle andre ting skal man måske kunne se en film i ordentlig grafik. Nogle kræver ikke så meget.
10. Natur af indhold: Medie aktivitet kræver. To tone display kræver andet design end en farvedisplay.

## Kontekst:

En aktivitet sker altid i en kontekst. Der er tre typer der er gode at skille fra hinanden:

* Organisatorisk kontekst
* Social kontekst
* Fysiske omstændigheder

## Teknologier:

Som designer skal man være opmærksom på mange forskellige muligheder for input, output, kommunikation og indhold. Man skal også være opmærksom på at teknologi hele tiden udvikler sig.

### Input:

Input apparater handler om hvordan man tilfører et system data og instruktioner sikkert. Dette kan være med knapper. Knapper er konstante og fylder. Det er derfor vigtigt at overveje om en bestemt funktion skal have sin egen knap på fx en mobil. Dette handler om hvor ofte funktionen bruges.

Andre input kan være:

* Touch skærm
* Tastatur

### Output:

Output teknologi giver information til brugere. Dette kan fx være skærme og højtalere.

## Kommunikation:

Kommunikation både mellem mennesker og apparater er en vigtig del af at designe et system. Her er båndbredte og hastighed vigtigt. Især også feedback, så brugeren kan forstå hvad der sker i systemet.

## Indhold:

Indholdet omhandler data i systemet, og den form det tager. Det indhold teknologi kan understøtte er kritisk for systemet. Godt indhold er:

* Præcist
* Opdateret
* Relevant
* Godt præsenteret

Nogle teknologier, fx hjemmesider, handler stort set kun om indhold, mens andre, så som en fjernbetjening, mere handler om funktion. For det meste er der et miks.

*Pull technology:* indhold kan blive hentet når nødvendigt, eller det kan blive hentet fra en server. Nyhedsområder på en hjemmeside vil fx automatisk blvie opdateret når der er noget relevant.

## **”Scoping a problem with PACT”:**

Målet med menneske centreret system design er at nå den bedste combination med PACT elementer I forbindelse med udviklingen af systemet. Når man laver en PACT analyse skal man undersøge de forskellige aspekter af elementerne i PACT der er mulige.

Som designer skal man tænke over de fysiske, psykiske og sociale forskelligheder, og hvordan de kan ændres over tid. Man skal overveje alle interessenter i et projekt. Kompleksiteten af aktiviteten, tidsaspektet af det, og de Samarbejdsmæssige aspekter.

# Forstå kriterier:

Definition: noget produktet skal gøre, eller en kvalitet det skal have.

Nye kriterier vil fremkomme under udviklingsprocessen.

Vær kritisk overfor kundens kriterier.

Det er diskuteret hvilket af følgende termer der skal bruges for kriterie aktiviteten:

* Requierments gatering: Dette virker som om at kriterier bare ligger klar til indsamling, ikke så meget behov for udvikling af disse mellem designer og stakeholder
* Requierments generation: Mere kreativ aktivitet, undertrykker den nuværende metode
* Requirements eliciation: Noget iteration mellem designer og stakeholder
* Requirements engineering: Ofte brugt i softwareudvikling. Typisk meget formel fremgangsmåde.

Når kriterier skrives, skal det være i et klart sprog, og det skal formuleres så det er muligt at teste om de er blevet opfyldt til sidst.

## Kriterietyper:

Skal være meget konkrete. Skal kunne testes!!!! Skriv fx i procenter, hvor meget. Det er udviklerens ansvar at formulere kundens kriterier, i samarbejde med denne.

Lav en liste:

Id 🡪prioritet 🡪resume 🡪 kilde 🡪 afhængigheder/konflikter 🡪 Tilfredsstillelses kriterie 🡪 ændrings historie.

Kan laves i tabel, diagram. Noget kan være billeder, videoer.

Hvordan teknologien opfylder disse er ikke specificeret, men kommer senere i design delen. Det er en god ide at supplere listen af kriterier med underbyggelse, dette kan være interview, observationer, rapporter, fotos og artefakter eller videoklip. Dette hjælper læseren med at forstå grunden bag disse.

### Dokumentation af kriterier:

Skal kunne teste. Kan være text, diagram, prototyper. Skal handle om hvordan verden ser ud nu. Forstå problemdomænet først.

### Funktionel:

Hvad systemet skal gøre. Aspekter og funktioner.

### U funktionel (non funktional):

Hvordan systemet gør noget. Kvaliteter designet skal have. Hvordan systemet kommunikerer med brugeren. Ikke bare sige implicit at noget skal være let eller effektivt, men relativt til hvad.

Fx fejlmeddelelser, brugergrænseflader

### Gode kriterier

Kort, præcist, beskrivende, MÅLBART. Ikke relative. Ikke lidt/ meget men målbart. Man skal kunne teste præcist om noget er opnået. Hvordan ville du teste det? Hvad ville opfylde? Kør altid kriterier gennem kunden, og få dem til at acceptere dem (hav beviser). Kunde forstår visuelle kriterier bedst. Brug fx billeder.

## Prioritering af kriterier:

Kriterier bør altid diskuteres med kunder og klienter, og ændres hvis nødvendigt. Beslutninger handler om at prioritere hvor man vil lægge ressourcer

## Hvornår færdig:

Når intet nyt kommer op, gentag samme kriterier

# Envisionment:

Gør ideer synlige. Gør så andre kan forstå den.

Repræsentation. Hvor er du i proces. Undrsøgende (skitser, wireframes)

Accuracy. Har en klar idé (prototyper)

De skal være accurate, ikke forvirrende.

### MoSCoW regler:

En del af Atern udviklings metoden. Tager et forretningsmæssigt udgangspunkt i prioriteringen af kriterier.

1) Must have – fundementalt kriterie. Uden dette vil systemet være ubrugbart og ubrugeligt. Det effektivt minimum kriterie

2) Should have – Vil være essensielt hvis der var tid, men systemet vil fungere ude det.

3) Could have – Ikke så vigtig, så kan nemt undværes

4) Want to have but Won’t have this time round – Kan vente til senere

# Deltagelsesbaseret design:

Hvis man skal udvikle et produkt er der forskellige muligheder i forhold til at arbejde med deltagerne når et der i en stor vs lille organisation. I en stor er der ikke samme mulighed som i en lille for at arbejde med alle interessenter, og man må derfor finde repræsentanter at arbejde med.

# Interviews:

## Strukturerede:

Bruger præudviklede spørgsmål. Følger ordlyden helt.

For: Lette at udføre.

Imod: Meget smalle svarmuligheder. Svært at finde ud af hvorfor et individ har givet uforudset svar.

Eksempler: Public opinion pools.

Semi-strukturerede inteviews:

Når man har formuleret spørgsmålene før, men kan omformulere eller finde på nye emner hvis det er relevant.

For: Mere besværligt end strukturerede

Imod: mere uddybende svar.

## Ustrukturerede:

Bruges når det er vigtigt at minimere designerens forudfattede meninger, eller når der er en meget lille baggrund for information. Ingen for lavede spørgsmål eller emner, ud over emnet interviewet er omkring. Ofte ved man ikke helt hvad man skal spørge om først. En glidende overgang mellem semi- og u- strukturerede.

Jo mere man ved om emnet og omstændighederne, jo mere struktureret

## Contextural inquiry:

Partnerskab mellem kunden/ brugeren og udvikler

Kontekst – faktisk arbejde. Observer arbejdes proces.

Partnerskab – placering af erfaring, samarbejde mellem kunden eller brugeren, og udvikler. Hvor meget erfaring har udvikler med kundes situation, og omvendt.

Interpretation – abstraktion, fortolkning af hvordan systemet skal være.

Fokus – for detaljeret eller for overfladisk, man kan altid få flere kriterier, men man bliver nødt til at stoppe et sted der ikke er for overfladisk, men hvor man har en god ide om hvordan systemet skal være.

## Historier, scenarier og prototyper i interviews:

Disse kan være gode til at forstå aktiviteter, og få folk til ikke at forestille sig abstrakte situationer. Man kan fx spørge om hvad en normal dag består i, eller hvor personen føler der mangler noget teknologi. Dette kan belyse mange problemer, og navngive individuelle funktioner til den effekt ændringerne vil have i arbejdet.

Prototyper bruges til at vise scenarier.

## Tænk højt kommentarer:

Det kaldes verbal protokol. Handler om at hvis man har brug for at vide nogle lav niveau detaljer om den nuværende teknologi, kan man bede brugeren om at gennemgå de relevante områder, inklusive hvad der foregår i deres hoveder, når de bruger teknologien. Dette kan lave indikationer af nuværende problemer.

## Praktiske overvejelser i interviews:

### Forberedelse:

Kend til baggrunden. Dumme spørgsmål kan hjælpe med at påvise forud antagelser. Dette skal dog være velovervejet. Vær forsigtig med at bruge folks egen jargon før du er sikker på at du har styr på den.

### Hold styr på interviewet:

Der skal gerne være to til at interviewe, en til at tale og en til at tage noter. Noter kan dog undgås hvis det optages. Check op på udstyret jævnligt.

### Fortælle historier:

Husk at folk ikke altid fortæller / husker i proportioner. Tag højde for dette.

### Reflektion og undersøgelse:

Tænk tilbage, og giv den interviewede et resume af interviewet. Så er du sikker på at du har forstået det ordentligt.

### Gruppe interviews:

(s. 156)

Er godt til at få en diskussion i gang hvor man kan få andre detaljer.

### Undersøgende spørgsmål i normale situationer:

‘Tell me about your typical day.’ Tell me three good things about. . . ’ ‘and three bad things.’ ‘What if you had three wishes to make the application better?’ ‘What has gone wrong with the application recently? How did you cope?’ ‘What else should we have asked about?’

### Hvornår skal man stoppe?:

Alle signifikante grupper skal omfattes. Ved generisk udvikling, interview 2-3 personer pr gruppe.

## Spørgeskemaer:

Godt til store mængder data.

### Crowd sourcing:

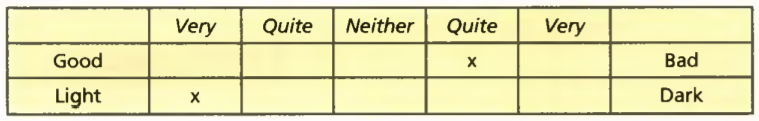
Små specifikke opgaver bliver puttet på nettet, og frivillige tager opgaven for en lille betaling.

Et godt spørgeskema skal opfylde følgende:

* Være forståeligt
* Entydigt
* Samle data der faktisk svarer på evalueringsspørgsmålene
* Let at analysere.

Man kan spørge folk hvor enige de er, bede om at score på 4,7 eller 10 punkts skala. *Probe questionen* er fx ”Systemet var let at bruge”. Her svarer man hvor enig man er.

Bipolar ration scale – semantic differentials.



Det er en god ide at lave det sammen med folk der er i samme fokusgruppe for at se om de forstår/ misforstår spørgsmål. Lav en ”pilot test”

Man kan lave en blanding af interview og spørgeskemaer. De uddybende spørgsmål bør tages i interview.

Bruges gerne sent i processen, men kan bruges til tidligt i processen at få feedback på eksisterende system. Finde ud af hvor problemområder er.

# ”Cultural probes”

En samling af artefakter. Brugeren sender ting til designeren, som sender ting tilbage til brugeren. Kan bruges til at få data uden at forstyrre det naturlige arbejdes flow. Her får man en bedre forståelse af flowet. Det kan være billeder, lydfiler eller ting.

Artefact collection: vi kommer og samler artefakter

Probes: Her bestemmer brugeren hvad det er vigtigt designeren får

Brug en kombination af forskellige metoder for at være sikker på at man får hele billedet.

Forsøg så vidt muligt at finde måder ikke at påvirke arbejdsflow.

Det er vigtigt at kriterier er baseret på data fra brugere, og ikke bare bygget på antagelser.

# Aktiviteter:

Design har tre aktiviteter:

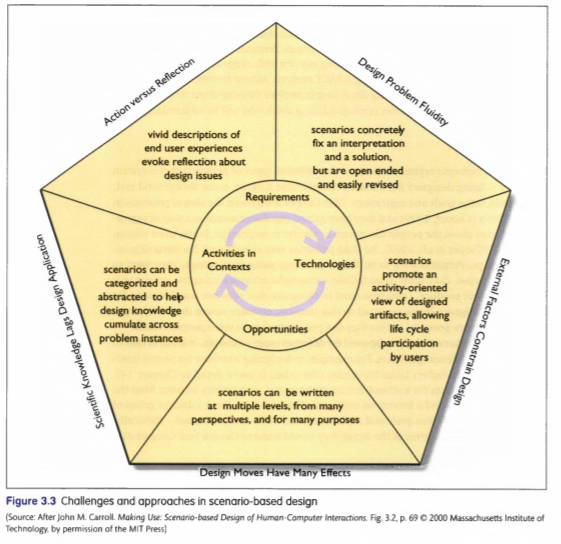
* Forstå
* Observer
* Visualiser

Eller fire

* Evaluere, sker i hvert step.
* Forståelse
* Envisonment (Kan være en del af forståelse, bruger det til at lave forståelse, til at evaluere forståelse)
* Design

Vær åben for at evaluere forståelse og evitionment.

Fem hovedproblemer i design :



Scenarier:

Nøgle teknik i interaktiv design. Kan bruges til at forstå, forestille sig, evaluere og både konsektuelt og fysisk design.

Fire typer: Historier, konseptuelle senarier, konkrete scenarier og use cases.

## Historier:

Tanker, viden og oplevelser fra den virkelige verden. Disse kan være dagbøger, videoe, foto, dokumenter, observertioner og interviews etc.

Fanger detaljer folk finder ligegyldige hvis spurgt mere formelt.

## Konceptuelle scenarier

Mere abstrakte end historier. Godt for design ideer, og for at forstå system kriterier.

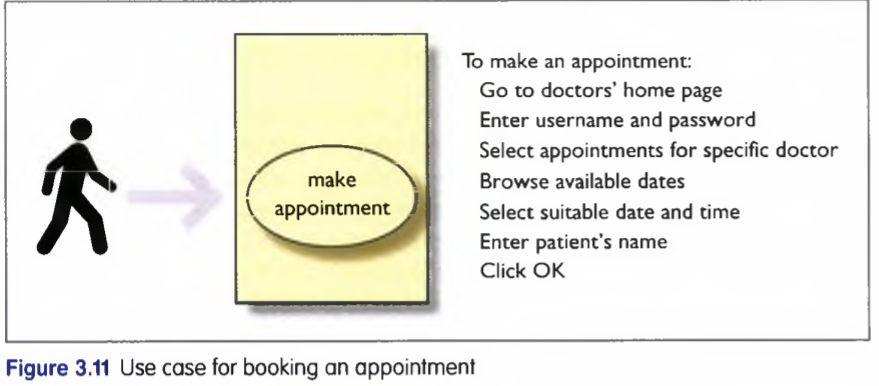
## Konkrete scenarier

Bliver lavet fra konceptuelle scenarier. Ikke en hård grænse mellem disse to. Jo mere specifikt, jo mere konkret.

## Use cases

Forklaring på fremgangsmåde for at nå til specifikt punkt.

Hvordan kan



# Brugbarhed:

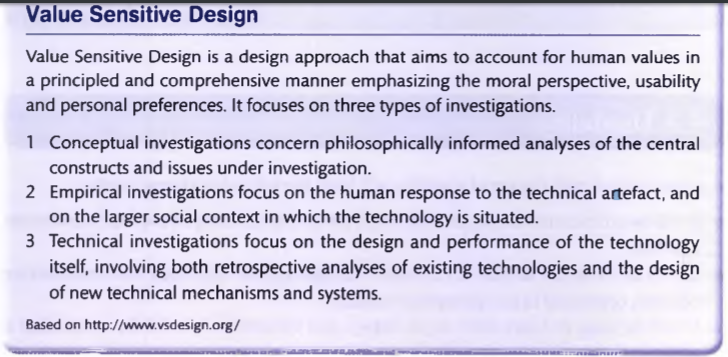
Et meget brugbart system er karakteriseret ved følgende:

* Effektivt da folk er i stand til at gøre ting med en fair mængde energi lagt i
* Effektivt da det har en tilpas mængde funktioner og information, organiseret på en hensigtsmæssig måde.
* Nemt at lære ting og huske hvordan
* Sikkert at bruge i alle henseener
* Høj brugbarhed da det gør hvad folk ønsker

For meget brugbarhed skal man gøre det ”human centred”. En fremgangsmåde hvor evaluation er centralt. Gould et al.(1987), principper

* Tidligt fokus på brugere og opgaver. Gør brugere til del af proces. Forstå kontekst.
* Empirisk måling. Find ud af hvad brugere synes undervejs.
* Iterativt design. Fix problemer i brugertest. Design 🡪 test og mål 🡪 gendesign
* (Integreret brugbarhed, brugbarheds faktorer skal udvikle sig sammen. Ansvar for alle aspekter af brugbarhed skal være under en kontrol. )

Cockton (2009) mener at designers skal forstå værdier deres design strækker sig mod. Mener at overstående er farligt meget forældet.



Brugbarhed kan ses som balance mellem fire aspekter af mennesker centreret…. ”PACT”

* (People) Mennesker
* (Activities) Aktiviteter folk vil udføre
* (Context) Kontekst i hvilken interaktionene skal ske
* (Technologies) Teknologier, hardware og software

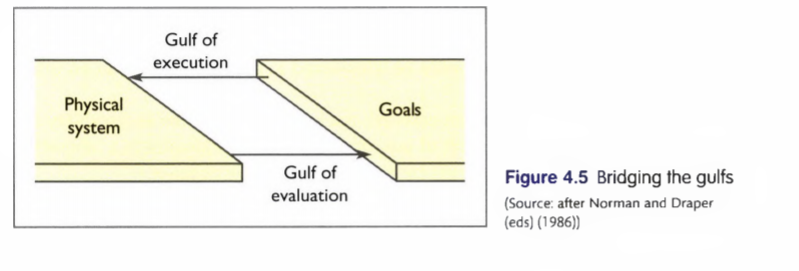
Kombinationen af teknologier er balanceret forskelligt alt efter kontekst og aktiviteter der skal udføres, samt menneskerne der skal udføre dem. Ingen teknologier er bedst i alle omstændigheder.

**Don Norman (Norman, 1988)**

**(s. 83)**

Interface mellem person og teknologi. Problemer folk har i forhold til at oversætte mål til handlinger afkrævet af UI.

* Folk har mål. Byg bro over mål og aktiviteter teknologi skal udføre
* ”The gulf of execution” oversætter mål til handlinger. ”Gulf of evaluation” bestemmer om handlinger er sucessfulde i forhold til personen og dennes mål
* De skal forbindes så person forstår hvad der sker og skal ske, og fysisk, kan person fysisk eller perseptuelt finde ud af hvad denne skal gøre og hvad der skal ske.



Vigtigt aspekt af brugbarhed:

Lav god mental model af systemet. Design skal være klart og godt struktureret konceptuelt.

Let kommunikeret til mennekser. Komplekst design gør dette svært. Klar, simpel og konsistent konceptuel model gør system mere brugbart.

# Probes

Artefakter til at samle ideer eller holdninger i bestemt sammenhæng.

**Kulturelle probes:** skal konfrontere, samle inspiration til designere i stedet for at fremkalde kriterier. Samles med ting som dagbøger og billeder.

**Teknologi probes:** Samle kriterier.

Gør at designere kan fokusere på hverdagen og ikke bare det generelle.

# ”Card sorting techniques”

Bruges til at forstå hvordan folk klasificerer og kategoriserer ting. Meget vigtig ved website design, da stukturen her er meget vigtig.

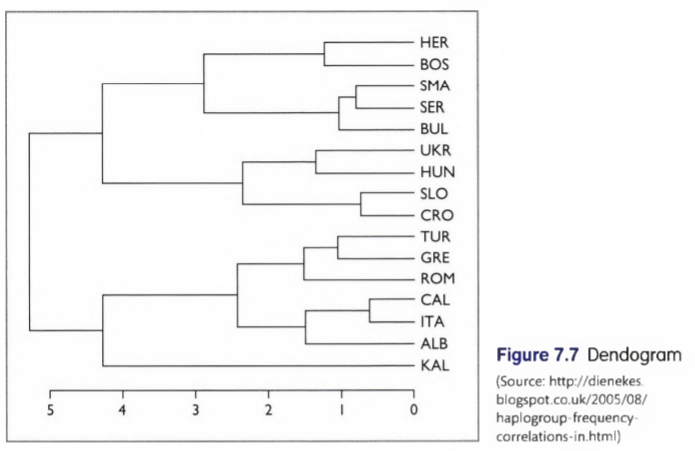
Fx folk kan få kort og grupere dem. Resultatet er en taxonomy (klasification), og nogle high level koncepter kaldet ontology.

Ved resultater for mange mennesker, kan forskellige matematiske grupperings teknikker bruges. Kan gøres face- to -face eller online.

To typer:

* Åben, blanke sider, deltagere skriver handlinger og objekter de synes er vigtige i domæne. Samles efterfølgende og puttes i kategorier.
* Lukket, forud definerede kategorier. Deltagere putter objekter i kategorier.

Lukket bruges når man allerede har kategori.



Dendogram viser hierarkiet af grupperinger af objekter.

# Arbejde med grupper:

Fokus grupper. En gruppe testpersoner i dialog med hinanden. Medlemmer kan stimulere hinandens hukommelse, og dialog flyder mere naturligt end i et en mands interview. **Joint application development method (JAD).**

CARD(s. 156). Bruger fysiske spillekort som gruppen kan lægge ud, modificere og diskutere flowet i aktivitet.

## Brainstorm:

Gruppeaktivitet. Burde være sjove at deltage i. Skal bruge stimuli, fx billede, tekst, video, for at få ideer til at flyde. Der skal være en måde at optage tanker og ideer. Måske tavle, papir eller lignende. Post-it notes i forskellige farver. Kan være brugbart hvis der efterfølgende er en ”**affinity analyse”** hvor ideer bliver gruperet ved brug af forskellige kriterier. Giv gerne gruppemedlemmer forskellige roller, fx ide generator, kritiker, skeptiker osv.

# Benarbejde – observering af aktiviteter i ”situ”

At observere folk lave aktiviet er en god måde at forså og generere kriterier. Det er bedst at gøre efter et start interview for at forstå hvan man skal observere. Stop når du ikke får mere info. Tag noter, og optag gerne video for at huske og dele. I tilfælde af inaktivt arbejde eller meget lidt menneskelig kontakt, er det bedre at bede folk om at demonstrere .

## Arbejdsplads undersøgelser.

Observer arbejde i rigtige omstændigheder. **”Computer Supported Cooperative Working (CSCW)”**. Ethnographic.

**Simonsen og Kensing(s. 159)**, fire brtingelser for brugen af etnografi i kommerciel sammenhæng:

* Både analytiker og organitiation skal have positiv atitude og investere en signifikant mængde rescourser.
* Stakeholders skal forstå formålet med det nye system, fremgangsmåden og hvorfor analytikeren arbejder som denne gør.
* Analytiker skal være klar til at håndtere politiske issues forbundet med undersøgelsen
* Fokusområdet skal identifiseres.

Software værktøjer der hjælper med at analysere tekst noter; Adas.ti and Ethnograph.

**Vignette**: kort beskrivelse af typiske scener. Ligner et scenarie, men er mindre struktureret. Mere som teksten i et film script, med scene directions.

# Artefakt indsamling og ’desk work’

# Suppler tidligere artefakt og dataindsamling med ting fra omgivelserne. Notesbøger, dokumenter, noter, telefon. Læs info on kunden. Se på lignende produkter – god måde at se fordele og ulemper.

# Kontextual design

## ”Contextual inquiry”

Første step i kontextuel design. Kombination af interview og observation. Fire guidene principper: kontekst, partnerskab, fortolkning og fokus.

### Kontekst

Observer arbejdsstedet. Giver detaljer. Kan blive suppleret med reflektioner fra observatør.

### Partnerskab

Analytiker og kunde er hver især eksperter i deres eget aspekt. Analytiker kigger efter mønstre, kunde tilføjer viden. ”master-apprentice model”. Analytiker får forklaringer, og ser dem i kontekst for at få fuldt billede.

### Fortolkning

Analytiker skal fortolke hvorfor kunde gør som den gør, ikke bare hvad delle gør. Herefter skal analytiker dele fortolkning med kunde og lytte til respons.

### Fokus

Fokuser hver især på et aspekt af konteksten når du observerer aktiviteten. Dette hjælper med detaljer.

## Praktikaliteter i kontekstuelt interview

Hav notesbog og optagelsesudstyr.

# Passende repræsentationer

Alt efter hvem man skal diskutere med og have respons fra, skal man bruge forskellige repræsentationer. Fx hvis man skal vise noget til en kollega er det fint nok at tegne det på et stykke papir, mens der til en kunde skulle en helt anden fremgangsmåde til.

## Basale teknikker

Hvis meget materiale, indeks kriterier og referer til dem af kode.

### Skitser og snapshots

Skitser er en god måde hurtigt at skrive ideer ned. Designere burde altid have en notesbog i nærheden til at skrive og tegne tanker og ideer. Når man skal se mange data er der fremgangsmåden (Ben Shneiderman) Oversigt 🡪 Zoom og filter 🡪 Detaljer hvis ønsket.   
Lav en skitse over funktionalitet på forskellige kriterier. Brug disse til at vise kriterier til brugere. Så er det også nemt at få idéer fra dem.   
Signalerer at noget kan ændres, ikke er fastsat, flere idéer fra kunde/ brugere.

Hvis du simulerer interaktion med skitse er det en prototype.



### 

### Storyboard

Tegn situationer design skal gøre. Hjælpsomt at basere på konkret scenarie. God måde at komme igennem design ideer med kunder.

#### Traditionel storyboarding

Nogle noter der forklarer situation. I interaktive systemer er noter typisk steps fra et scenarie. Skitser er annoterede for at vise interaktiv opførsel.  
Mest normal hvis der ikke er meget multimedie.

#### Scored storyboards

Bruges hvis der er meget bevægende grafik. Her er der fx noter om type, farver, billeder, lyd

#### Text only

Hvis der er mange komplekse sekvenser. Fortæl hvilke billeder der kommer og tekst ved. Alt er beskrevet i noter.

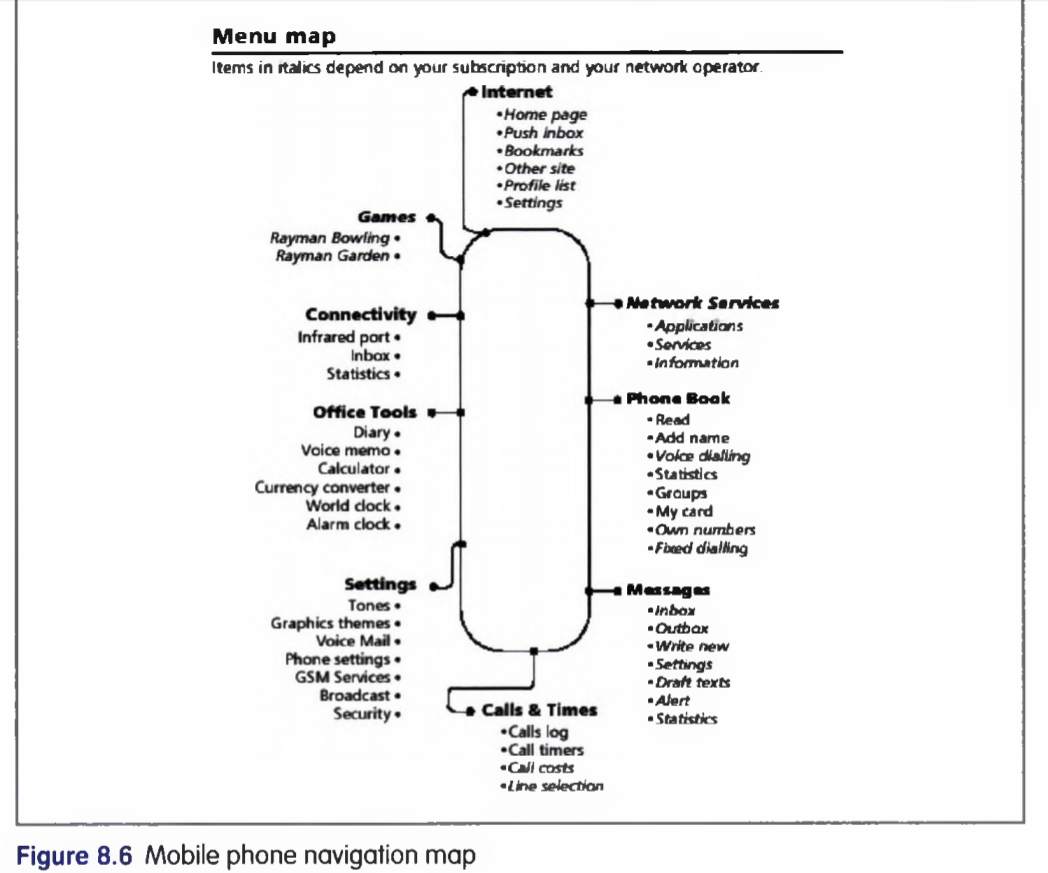
### Mood boards

Bruges meget i reklame og indendørs design. Samler visuel stimuli der fortæller hvad du føler omkring design. Det sættes på opslagstavle. Anything goes. Er kun inspiration. En variation er at skrive tillægsord der beskriver et aspekt. Få gerne klient til at lave et.

## Navigations kort

Hvordan går folk gennem applikation. Hvordan oplever folk siden. Hver side i programmet laves som en boks. Ting man kan navigere til skal flyde fra det. Vis begge veje. Kan vise hvor folk strander.

***Orphain pages:*** *Sider der ikke kan tilgås*

### Wireframes

Kontur af struktur af softwaresystem. Fokuserer på struktur af bestemte typer sider. Fokuserer på generelt design og ikke detaljer. Den bedst kendte software for wireframes er [www.axure.com](http://www.axure.com). Der er flere software packages der laver fungerende prototyper fra wireframes.

Lav detalje skitser. Man bruge alt muligt som fx PowerPoint. Kan bruge farver til at fremhæve, fx rød som noget dårligt. Hav den farveløs ellers hvis muligt, så farver er synlige.

### Andre teknikker:

Mind maps, flow diagrams, transition diagrams, flip chart, collaborative writing (en gruppe arbejder på samme dokument)

### Værktøj

Ulempe er at man bare kan tilføje fordi man kan. Gør ting mere komplekst. Måske ikke en del af design idé. Gør simpel.

Balsamiq (skitse)

Axure.com (wireframes) – svært at lære, tager lang tid.

# Prototyper:

(s 175)

LAV PROTOTYPER TIDLIGT. Ellers kan du ende med et dårligt design der ville være meget omkostningsfuldt at ændre.

God til at kommunikere og opdage muligheder og visions. Skitser og simple prototyper kører sammen. Prototype er interaktiv. To typer: lo-fi og hi-fi

Ikke kun tal med kunder – tal med burgere hvis muligt.

## Hi-fi prototyper:

(s 176)

Når det er lavet med Software – altid hi-fi.

Føles ens og ser ens ud med produkt. Lavet i software. Det er vigtigt at lave den gennemført, da folk ikke forstår de detaljer der ikke passer ind, og derfor vil dømme produkt ud fra det.   
Mange detaljer, layout og grafik vigtig. Kan være få detaljer, men er ikke typisk.

* Dyr
* Realistisk
* Høj validitet test
* Test senere
* For tidlig fastsættelse – ikke villig til store og små ændring
* Kunder kan tro at de er ægte

## Lo-fi prototyper:

(s 176)

Ofte kaldet ”paper prototypes”. Fokusere på generel underliggende design idéer, fx indhold og struktur samt design tema. Lad være med at være for detaljeret. Det er smart at have en måde brugeren kan vise sit input, fx ved at gøre at man kan rykke rundt. Fokuser på indhold. Bredt.

* Billig
* Ikke mange detaljer
* Ikke realistisk
* Ikke meget validitet når man tester
* Tidlig

## Video prototyper:

(s 178)

En slags simuleret demonstration af produktet, med måske en skuespiller hvor man får det til at se ud som om systemet gør noget.

”**Wetherman technique”**

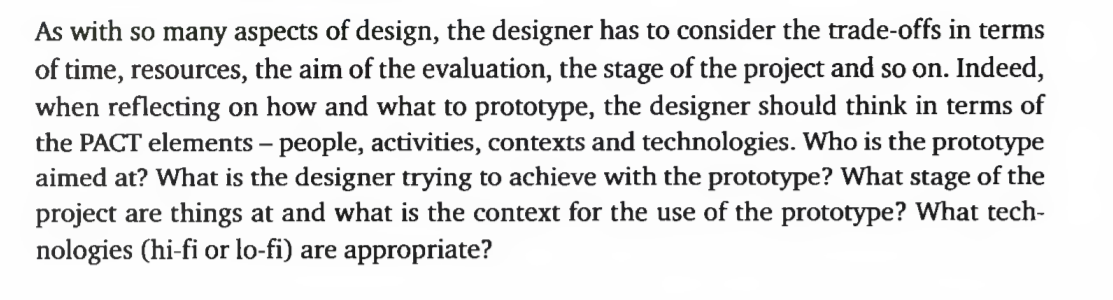
## Andre typer:

Full prototype:Fuld funktionalitet, lavere performance

Horizontal prototype:Vil gå gennem hele systemet, men kun vise top level funktioner.

Vertical prototype:Viser alle features fra top til bund, men kun ved nogle få funktioner i systemet.

s. 180



## Bruger skritser:

Lad bruger være en del af at skitsere. Lav dem med brugere. Giver forståelse for hvordan de tænker.

# Evaluering:

## Ekspert evaluering:

(s. 217)

En hurtig og effektiv måde at evaluere et system på er, at få en professionel evaluerings ekspert ud og se på projektet. De er gode tidligt i designet. De finder ofte opståede problemer i designet.

### ”Heuristic evaluation”:

Dette er en gruppe af metoder øvede folk bruger til at se om designet lever op til en række principper, guidelines eller ’heuristics’ for at det er et godt design.

Design principperne (heuristics) er blandt andet følgende:

## 

Pro:

* Nem og let
* Ingen brugere
* 3-5 inspektioner finder 70% af problemer

Von:

* Mange kosmetiske problemer
* Falske brugbarheds problemer.

Man kan aldrig finde 100% af problemer i brugbarhed – alle er forskellige.

## ”Discount usability engineering”:

Listen ovenover kan kort genfortælles som fire overordnede principper, ”learnability (1-4), effektivitet (5-9) og accomodation (10-12). Hvis man har travlt kan man kigge på disse tre. Dette kaldes ”Discount heuristic evaluation”.

Man bør ikke evaluere egne designs med mindre det er strengt nødvendigt. Når man bruger disse metoder skal man huske at finde ud af hvad virkelige personers viden er.

### ”Ecological validity”

Mennesker bliver ofte afbrudt når de udfører en opgave. Dette er vigtigt at tage hensyn tik (bund s. 218, top s. 219).

### Cognitive walkthrough:

(s. 219)

En analytiker går igennem de kognetive opgaver der skal udføres når man interagerer med teknologi. Dette er en systematisk fremgangsmåde baseret på godt etableret teori, og ikke på ”trial and error” eller en heuristisk baseret fremgangsmåde.

Input i processen er:

* En forståelse af de mennesker der forventes at bruge systemet.
* Konkrete scenarier der viser både meget almindelige og ualmindelige kritiske sekvenser af aktiviteter
* En komplet beskrivelse af systemets interface. Dette skal indeholde en repræsentation a hvordan det præsenteres og en korrekt sekvens af handlinger for at opnå scenariets opgaver, normalt en **hierarchial task analysis (HTA)**

Når dette er gjort skal man spørge sig selv:

* Vil brugere forsøge at opnå den rette effekt?
* Vil de opdage at den rette handling er tilgængelig?
* Vil de associere den korrekte handling med den de forsøger at opnå?
* Hvis den rigtige handling foretages, vil de så se at der er fremskridt i forhold til at opnå målet med aktiviteten?

Hvis nogen af disse spørgsmål kan besvares med nej, vil man se at der er et problem. Dette skal ikke løses endnu.

Analysen laves optimalt af designere og analytikere sammen. De skal beskrive hvordan brugeren ville identificere, udføre og overvåge den korrekte sekvens af handlinger.

(s 220)

Der er flere forskellige ”cut-down” versioner af teknikken:

* ”The cognitive jogthrough”. Video optager. Man må gerne give design forslag.
* ”The streamlined cognitive walkthrough”. Her vises ikke problemfrie steps. Kun to spørgsmål.

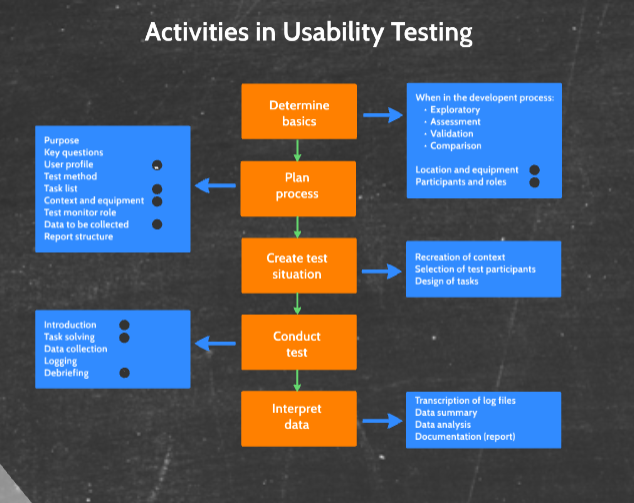
Her er detaljer tabt, men man ser mere af systemet som helhed.

### Mangler:

Eksperter finder problemer der ikke eksisterer. Vil ikke finde problemer der kommer fra en kæde af forkerte handlinger, eller stammer fra gennerelle mistolkninger.

## Deltagerbaseret evaluering:

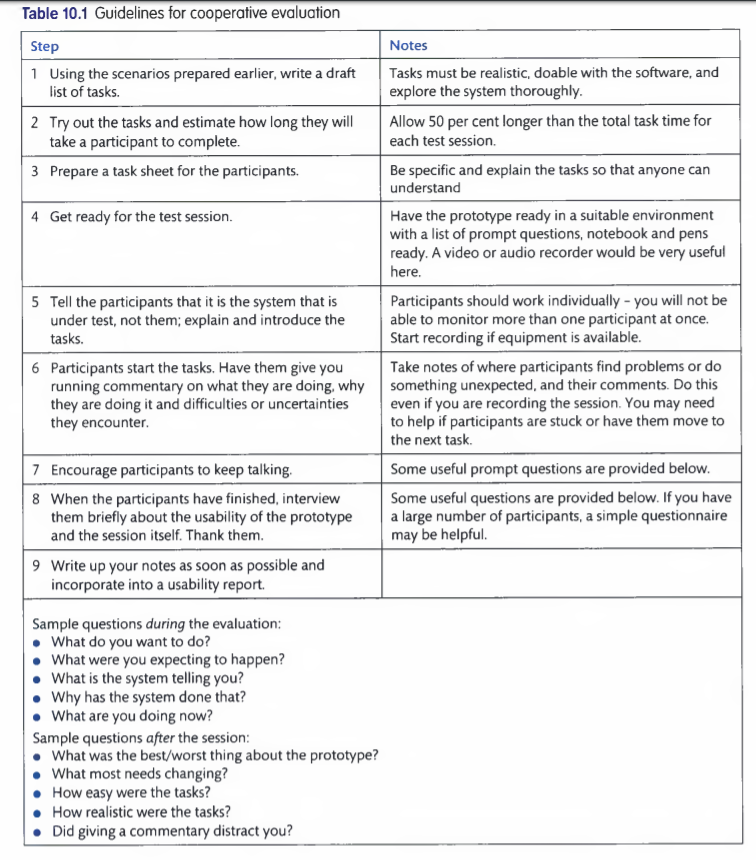
### Aktiviteter i usability testing:



### ”Cooperative evaluation”:

(s. 221)

Billig og pålidelig metode. Maksimerer data samlet. Deltagere er ikke passive subjekter, men er ”co-evaluators”.



### Participatory heuristic evaluation

(s. 222)

Samme fremgangsmåde som med ekspert, men burgere er “work-domain” eksperter.

### Co-descovery:

(s. 222)

Godt sent I design. Godt til at se førstehåndsindtryk. I stedet for en enkelt person der skal bruge programmet og observeres, bliver dette gjort i par. Mere naturligt flow af kommentarer. Brug folk der kender hinanden. Realistiske opgaver.

Evaluator kan stille spørgsmål eller forslå aktiviteter. Spørgsmål drejer output i unaturlig retnign, men hjælper med at få vigtige vinkler med.

### Living labs vs lab:

(s. 222)

Tag undersøgelserne i produktets omgivelser. Giv fx prototyper til folk for at se hvordan de vil reagere. Brugeroplevelse kan være meget anderledes i de rigtige omgivelser end i lab. Her er der dog ingen kontrol i modsætning til i lab hvor der er helt kontrol.

Fordele ved lab:

* Godt til at skelne mellem årsager!?!?
* Mange kan følge med
* Kan kommunikere bag scenen
* Man kan lettere få brugeres tanker om brugen af systemet. Hvorfor synes de det er besværligt?

Lab svaghed:

* Boost måske resultat kunstigt
* Testpersoner kan komme til at føle sig som forsøgsdyr
* Svært at vurdere realisme
* Bruger meget tid.
* Lidt sterilt miljø. Unaturligt.

### ”Controlled experiments” - sammenlign

(s. 222 bund)

Når man vil undersøge specifikke features af et design.

Fremgangsmåde:

1. Fastsæt hvad du ser på. Find ud af hvad du vil sammenligne. *Independend variables*.
2. Fastsæt hvordan forskellen måles. *Dependend variables*.
3. Design eksperiment. Sørg for at intet kommer i vejen for de andre to. Ting der måske kommer i vejen er *confunding variables.*

Sørg for balance mellem 1 og 2 så du er sikker på at det er det du måler på, og at der ikke kommer noget 3 i vejen.

1. Bestem om deltagere skal deltage i en comdition (within- subject design) eller kun en (between- subject design)

## Evaluering i praksis:

(s. 224)

Hovedpunkter i at lave et simpelt og effektivt evalueringsprojekt er:

1. Fastsæt hvad du vil opnå. Deltagere, kontekst af system, find eller lav scenarier der viser brug af system.
2. Vælg evalueringsmetoder. Burde være kombination af ekspertbaseret review og deltager metoder.
3. Lav ekspert review.
4. Planlæg deltager testing. Brug resultat fra eksper review til fokus.
5. Find mennesker. Organiser testomgivelser
6. Lav evaluering.
7. Analyser resultater, dokumenter og raporter tilbage til designer.

**Kontekst i planlægning: (Slides forelæsning d. 21/09)**

Hvor skal det være? Konteksten, både fysisk og social.

Hvem skal deltage? – forstå hvem der skal bruge det.

Hvad skal de bruge det til?

Hvordan er det brugt? Typiske handlinger, husk at fokuser på at gøre data så realistiske og relevante som muligt. Hvis det er i laboratorie skal man forsøge at simulere den rigtige situation så meget som muligt.

**Testdeltagere:**

Skal være repræsentative. Hvilken demographic. Hvor meget erfaring.

Hvor mange testpersoner skal der være? Man skal kunne generalisere ud fra antallet af testpersoner. Afhænger også af hvor mange problemer der er fundet.

Andre studerende? Meget farligt hvis de ikke er repræsentative. Man skal stadig tænke på erfaring og motivation og tage forbehold for det.

**At bestemme opgaverne:**

Hvilke basale opgave skal brugere have i forhold til systemet?

Man skal gerne lave testen på højest 45 minutter så de ikke bliver trætte

Meget klare opgaveformuleringer. Kom med et scenarie. Forklar hvorfor skal de gøre det, sæt dem ind i konteksten. Gør situationen virkelig.

Hvor lang tid tager hver opgave? Hvilke er vigtigst? Sørg for at alle opgaver passer ind i tidrammen. Hvis de bliver trætte kan resultater ikke bruges.

**Hvad skal man måle, og hvordan?**

Hvilke dele er relevante?

Hvordan samles data?

Skal du forsøge at få brugere til at tænke højt? Godt for research. Folk er langsommere hvis de skal tænke højt. Nogle studier viser dog at man er hurtigere når man tænker højt. Folk strukturerer hvordan de bruger systemet bedre når de tænker højt. Når man måler fysiske reaktioner på folk, kan tale forstyrre udstyr. Dog giver det stort set altid mening at få folk til at tænke højt.

**Let at lære vs. Effektivt at bruge:**

Mange gange er noget der er lettere at bruge for nybegyndere kan være hurtigere/ mere effektivt i længden. Når man sammenligner to systemer på dette måler man på tid. Tid første gang man bruger systemet vs. tid efter øvelse?

**Usability metrics:**

Objctive metrics:

Effectiveness: hvor mange opgaver er gennemført

Efficiency: hvor hurtigt er de gennemført

**Subjektiv(perceived):**

Interview data

Sprørgeskema - Mangler hvad hvorfor man svarer noget. Folk lyver.

**Videooptagelse:**

Hvor meget skal optages? Hele skærmen, omgivelserne? Det er forskelligt om dette er relevant.

## Hvilken type test?:

Hvor er du i processen?

Typer: exploratory, assessment, validation, comparason.

Ansigtsudtryk er vigtigt for det viser hvad folk føler omkring noget.

**Deltagere og roller:**

Subjekt og monitor. Man kan have en monitor siddende ved siden af subjekt. Kan gøre subjekt mere afslappet, få dem til at føle sig mindre som forsøgsdyr. Man kan få et andet indblik i situation. Husk noter.

**Monitor:**

Problemer:

Fortæl ikke brugere hvad de skal gøre for at løse opgaver. Kan guide dem med spørgsmål. Hvad tænker du? Hvorfor gjorde du det?

Kan gå for meget op i testplanen. Når man er tæt på opgavens ”deadline” kan man spørge fx om hvor tæt bruger er på at løse opgave.

**I løbet af testen:**

Start med at samle data.

Forklar situation til deltager. DET ER IKKE EN TEST AF DIG, MEN SYSTEMET.

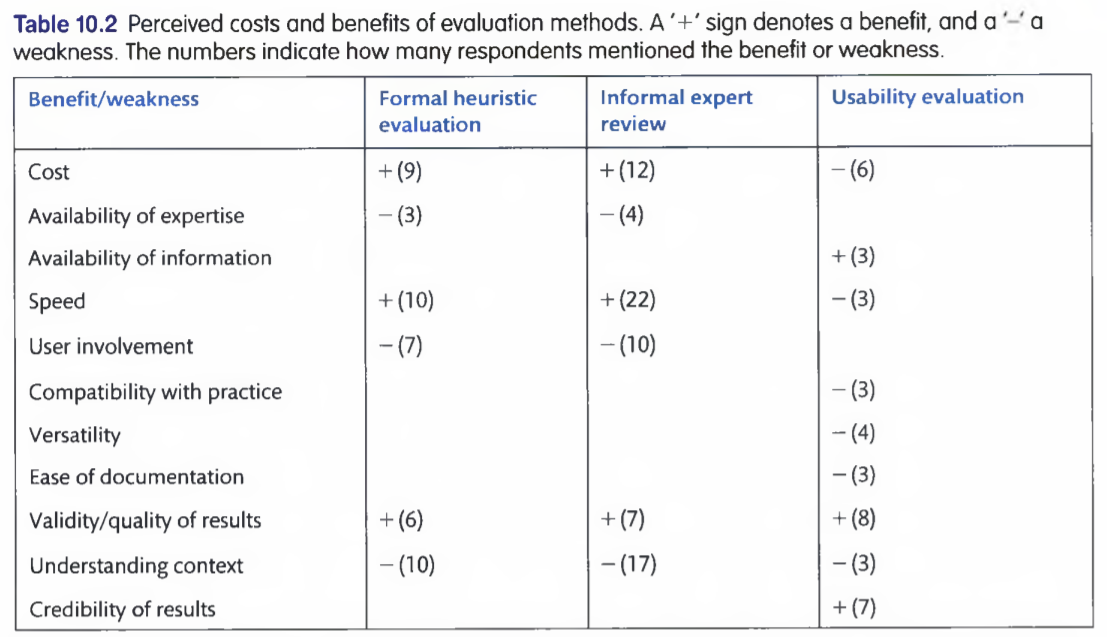
Løs opgaven. De skal kunne stille dig klargørende spørgsmål.

Logging

Defriefing. Fortæl hvordan de klarede sig.

**Introduktion af test:**

Nedskriv en introduktion. Sørg for at alle får samme introduktion. Læs højt hver gang.



### Mål for evaluering:

(side 225)

qualitative (non-numerical)

quantitative (numerical)

### Mennesker:

(side 227)

3-5 mennesker i testing ifølge Nielsen. Nogle mener flere. Få mennesker giver mening i homogene brugergrupper. Tag 3-5 mennesker fra hver gruppe hvis flere. Hvis du ikke kan finde nogen der repræsenterer ordentligt, så find nogle andre. Det er vigtigt at få andre øjne på.

### Testplan og opgavespecifikationer: !!!!

(side 228)

### Rapportering af ue resultater til designteam!!!

(side 228)

Husk at fx måle fra et til fem (mindre iritation til fatal mangel)

## Andre problemer i forhold til evaluering:!!!

# Usabillity testing!!:

Formål: find problemer i forhold til brugbarhed

Udfald: Liste af problemområder

Hvordan evalueres det:

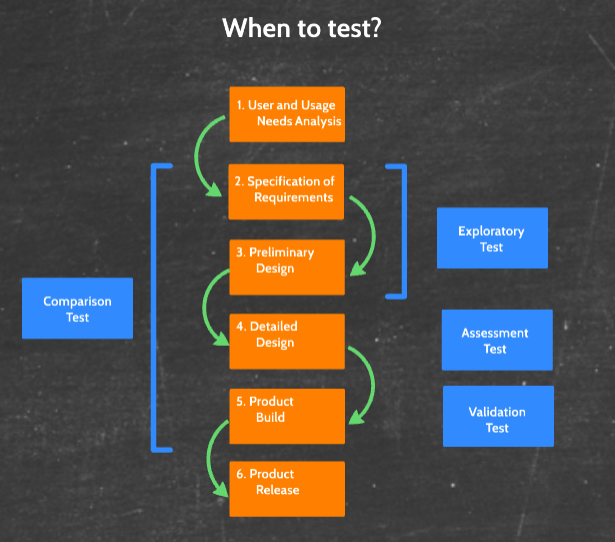
Inquiry: forstå brugere.

Testing: brugertest og produkt design

Inspektion: testing med eksperter.

**Hvornår test:**

Effektivitet er lavere i papir prototype.



Lad være med at sammenligne papirprototype med konkurrerende produkt.

# Analysering af data og observertioner

(kapitel 11, side 245 i Rubin & Chisnell - Handbook of Usability Testing)

Dataanalyse foregår typisk gennem to processer, med forskelligt resultat.

## Indledende analyse (preliminary analysis):

Den indledende analyse. Her finder man hurtigt de værste problemer. Disse kan designere så arbejde på uden at skulle vente på de endelige test rapporter. Så hurtigt som muligt efter testning er udført. Resultatet er enten en lille skrevet rapport, eller en verbal præsentation af hvad man har fundet ud af, og hvad man foreslår. Fremgangsmåde er som følge:

Compile data:

Opsummer data:

Analyser data:

Pointen her er at kunne se det store billede og mønstre. Skal være hurtig men grundig. Man kan ikke bare komme senere og ændre i det, der bliver designet ud fra denne analyse. Hellere for lidt end for meget. Omfat kun de åbenlyse problemer, og husk at fortælle hvis du er usikker på anbefalede løsninger. Efter hver dag skal du samle data, og sørge for at dette er brugbart.

### Compile data:

(side 249 Rubin & Chisnell - Handbook of Usability Testing)

Det har en tendens til at være iterativt, da nogle svar stiller flere spørgsmål.

Compiling involverer at sætte alle data i en form der gør at du kan se mønstre. Bør gøres over hele test sessioner. Dette gør det hurtigere, og derudover er det også en god måde at se om du samler de rigtige data, og om de matcher problemudtrykkene i test planen. Det viser også om du har glemt noget vigtigt, og få overblik over hvad du har samlet.

#### Organiser rå data:

(side 248 Rubin & Chisnell - Handbook of Usability Testing)

Brug de værktøjer der kan hjælpe med at overskue hvordan data ser ud, fx overstregnings touch eller sticky notes.

Du kan bruge et ”spreadsheet progeam”.

#### Opsummer rå data:

(side 249 Rubin & Chisnell - Handbook of Usability Testing)

Når du er færdig med compiling så skal du samle alt data i ”summary sheets”. Hermed får du et bedre overblik over hvad der skete i løbet af testen. Hermed kan du også få overblik over hvordan de forskellige grupper af typer testpersoner klarede sig (fx nybegyndere vs. eksperter), eller i forskellige versioner af produktet.

#### Opsummer performance data:

##### Opgave præcision:

(side 249 Rubin & Chisnell - Handbook of Usability Testing)

Muligheder:

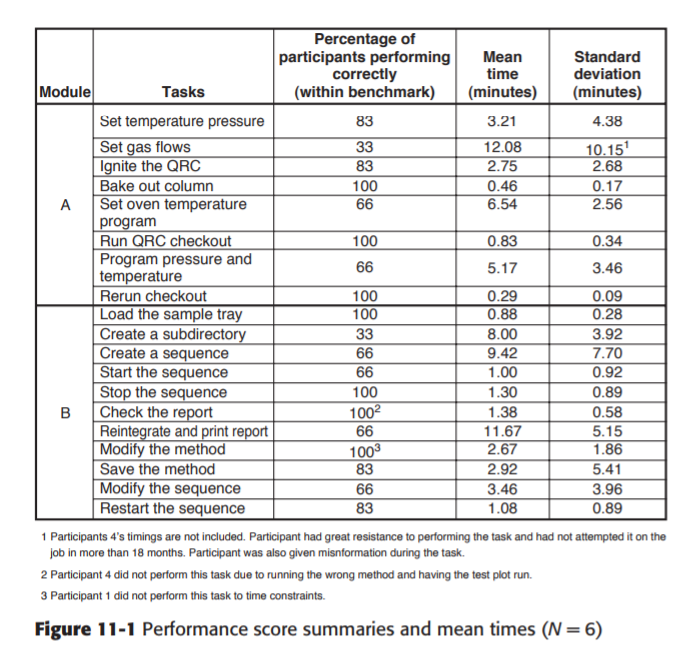
* Tæl antallet af errors og del dem op i typer. Fx noget er udeladt(omission), noget unødvendigt (commission) osv.
* Tæl antallet af folk der klarede opgaven rigtigt, både indenfor og udenfor tidsrammen.
* Se på dem der klarede opgaven med hjælp. Denne virker dog kun hvis du har samme type hjælp hver gang.

Her er tre statistikker at gå med:

Procent testpersoner der klarede opgaven, inklusive dem med hjælp

Procent testpersoner der klarede opgaven

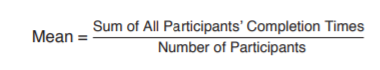
Procent testpersoner der klarede opgaven indenfor tidsrammen



##### Task Timings

Hvor længe er testpersoner om at klare opgaver.

###### Mean time to complete



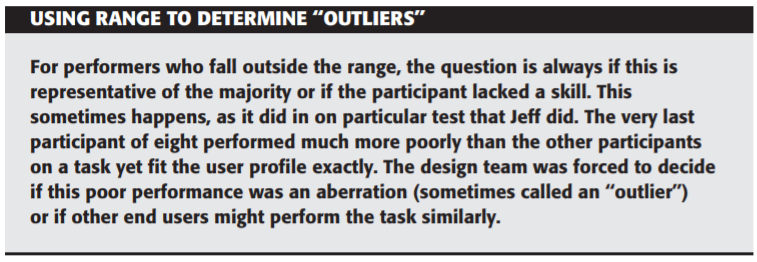
Hvis de højeste og laveste tider er langt fra hinanden, så brug median.

###### Median time to complete:

Find medianen af alle tider.

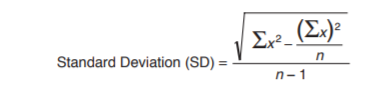
###### Range (høj og lav) of completion times:

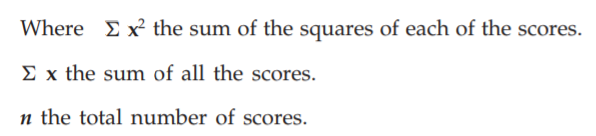
Finder den højeste og laveste tid for hver opgave. Denne kan sige meget hvis der er meget stor forskel. Især hvis der er få deltagere.



###### Standard deviation (SD) og completion times:

Måler variation. Hvor forskellige er tiderne fra hinanden? Hvor tæt ligger tiderne på middel tiden?





SD er altid i den originale måleenhed. Dette er godt til at måle hvor længe tiderne generelt ligger fra hinanden. Hvis værdien er stor skal man undersøge hvorfor. Kan en årsag identificeres?

#### Summarize Preference Data

Kan være alt muligt, spørgeskema…

##### Spørgesmål med få valgmuligheder:

Summer svar, så du kan se hvordan disse fordeler sig. Hvis der er mange kan du lave procenter.

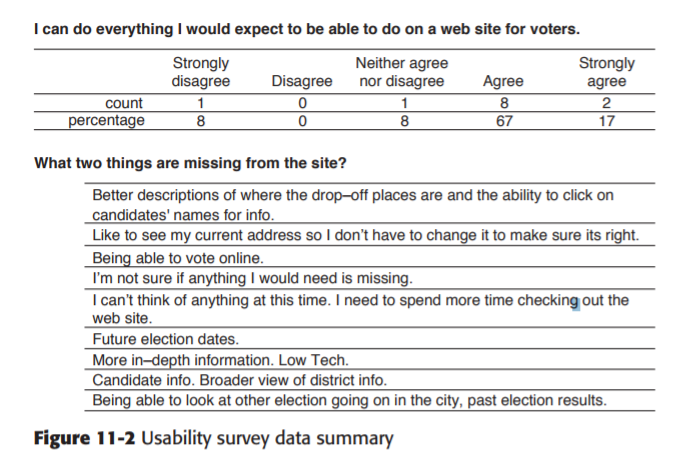
##### Frie spørgsmål og kommentarer:

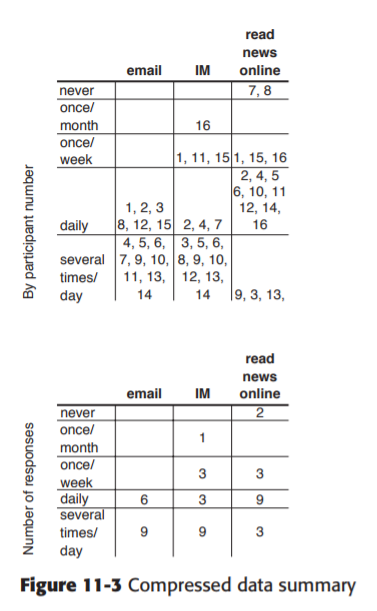
Læg alt i kategorier.

##### Debriefing:

Lav et transcript.







#### Kompiler og opsummer andre måleenheder:

Ting man kunne spørge sig selv om:

* Antal gange bruger vender tilbage til hovednavigation unødvendigt
* Antal og typer af hints og prompts
* Antal gange site map blev tilgået
* Punkter for tøven, hvor længe.

#### Opsummer resultat ved grupper eller versioner:

Opsummer data for hver gruppe hvis der er flere.

## Analyser data:

(kapitel 11, side 258 i Rubin & Chisnell - Handbook of Usability Testing)

Fokuser på opgaver. Disse viser brugers synspunkt eller mål.

### Identificer opgaver der ikke levede op til succeskriterierne:

Se på de prædefinerede succeskriterier til hver opgave. 70% er et standard valg som kriterie. Især i tidlige versioner af systemet. I slutningen er det 95%.

Non-criterion tasks: opgaver der ikke lever op til kriterier.

Ved få folk vil man ofte selv lægge mærke til besværlige opgaver. Man skal finde en måde at kunne skelne niveauer performance. Så er det lettere at se problemområder

### Identificer brugerproblemer og besværligheder:

Når du har fundet en ikke opfyldt opgave, så find ud af hvad der er galt. Et problem er alt brugeren gør der ikke er forventet. Det kan være godt at definere hvad en ”error” er før studiets begyndelse.

### Lav en errorkilde analyse:

Find kilden til hvert enkelt problem. Her skifter man fra opgaveorienteret til produkt orienteret. Find en produktbaseret grund til problemer. Gør det sammen med andre der har været med til at udføre testen. Nogle problemer er åbenlyse mens andre er komplekse. Der er dog oftest en primær og sekundær grund. Når man ordner den primære del vil det gøre de næste mere overskuelige. Husk at overvej den bruger der fandt problemets baggrund. Lad være med at finde løsninger før alle problemer er fundet.

### Prioriter problemer:

Criticality – se på problemet, hvor seriøst er det, og hvor ofte vil det ske.

Criticality = Severity + Probability of Occurrence

Rank dem efter sevirity:

1. Irriterende
2. Mindre hindringer, mulige problemer, men vil ikke hindre bruger
3. Seriøst problem, kan hindre bruger
4. Opgave fejler, bruger kan ikke komme videre

Rank dem efter frekvens (estimeret):

1. 51-89%

Heri indgår to faktorer:

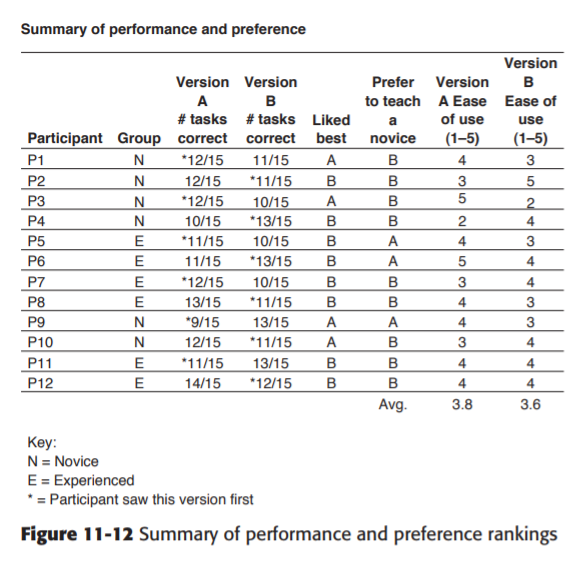
* Procent af brugere påvirket
* Sandsynligheden for at en bruger fra den påvirkede gruppe vil opleve problemet.

Fx hvis 10% vil få problemet 50% af tiden er der kun 5 procent frekvens.

I små tests kan du bare spørge brugere hvad der er mest problematisk.

### Analyser forskelle mellem grupper og produktversioner:

Analyser mængden, typen og alvorligheder af fejl brugeren laver i to eller flere versioner eller grupper, og gruppers ”preference” ratings, rankings og generelle kommentarer.

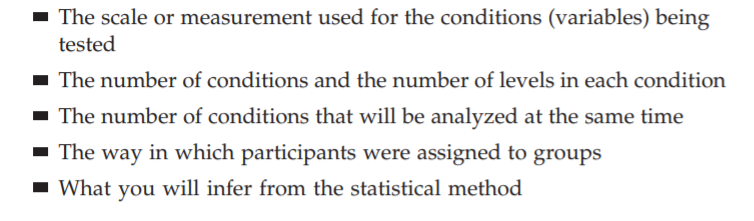


”source of error” analyse er godt til at sammenligne, da man så kan se på fordele og ulemper. Dette kan også ved prototyper hjælpe med at lave en version med det bedste fra alle.

### Using inferential statistics:

Nogle gange, især når to systemer bliver sammenlignet, skal man give statistisk signifikante resultater. Her skal man bruge ”inferental” statistik. Her udleder man noget om den brede befolkning fra en smal befolkningsgruppe.

Hvilken statistisk teknik der benyttes, kommer an på følgende:



Resultat af den måde du udfører test på vil variere på baggrund af om du vil samle statistiske data.

Før du skal overveje dette, skal du have mindst 10-12 deltagere pr. opgave.

## Omfattende analyse (comprehensive analysis):

Tager 2-4 uger efter test. Giver en mere endelig og omfattende rapport end indledende. Rapporten burde indeholde alle observationer fra den indledende rapport, muligvis opdaterede, og andre analyser og observationer der ikke blev fundet sidst.

# Noter forelæsning d. 26/09/2017

## Userbility test:

### Mulige faldgrupper:

Sørg for alt er klar, undersøg mulige faldgrupper

* Gør opgaver så realistiske som muligt
* Gør dem actionable
* Undgå clues og at beskrive steps. Sørg for at de selv løser opgaven, og ikke bare kører efter hvad du siger

Skriv opgaven ned, og læs den op til alle på samme måde, så alle får samme informationer. Husk at fortælle at det er for at alle kan få samme information, så testpersonen ikke føler sig dårligt tilpas.

Instruktioner omkring test

Få dem til at skrive under på en ”Consent form”, så du kan være sikker på at have tilladelse til at bruge det til had der er vigtigt.

Spørgeskema(demographic), du skal vide nogle basale, vigtige ting om dem

### Tænk højt:

Sørg for at finde ud af hvordan brugeren tænker højt. Hermed kan du også forklare dem hvordan du vil have at de tænker højt. Sørg for at få informationer om hvordan brugeren løser et problem, processen.

Information du vil have fra at brugeren tænker højt:

* I want to do…
* Looking at the UI, and I think it does..
* Hmm, that’s not what I expected; i thought…
* Giv dem eksempler på hvad du vil høre.

Hvis de går i stå og stopper med at tale højt, så stil spørgsmål, hvad tænker du nu…

### Gode opgaver:

* Motiver (hvorfor skal de lave opgaverne?)
* Beskriv enderesultatet
* Repræsenter de rigtige brugere
* Inkluder relevante data
* Små underopgaver grupperes sammen

### Dårlige opgaver:

* Vage, uklare og generelle opgaver
* Giver for meget hjælp
* Indeholder jargon og ukendte termer
* Tvinger brugeren ind i en specifik sekvens. Det er godt når du skal vide om de kan forstå med dit design, men ellers giver det ikke god mulighed for at teste.

Observer relevante data og saml dem

### Debrif:

Husk at debrief.

* Kan gøres med spørgeskema
* Spørg om ting
* Bed om kritik.

## Analyser data 🡨 i dag

Problemliste

Log fil:

Vis forløbet, hvad tid noget sker, hvad der bliver sagt og hvordan, og så problem.

|  |  |  |
| --- | --- | --- |
| Tidspunkt (i forhold til testen) | Transkript | Problem |
| 03:55 | Dette sker, det siger denne. Lidt som et manuskript. | P1 Dette er det første problem fundet  P2 |

Lav en oversigt over hvor mange og hvilke brugere har hvilke problemer.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Problembeskrivelse | Kategori | Testperson  1 | 2 | 3 | 4| |
| P1 |  | Kritisk/seriøst/kosmetisk | X| | | X |

### Problemtyper:

#### Kritisk:

* Når bruger ikke kan komme videre
* Det er en ”katastrofe” når flere har dette problem.

Hvis det stopper brugen.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Delay | Irritation | Forventning vs. faktisk |
| Kosmetisk | <1 | Lav | Lille forskel |
| Seriøst | Flere minutter | Medium | Signifikant forskel |
| Kritisk | Stopper | Høj | Kritisk forskel |

### Unikke problemer:

Et problem kun en af testpersonerne har.

Få flere til at analysere data – en kan misfortolke, mange giver mere stabilt resultat. Omkring 3.

### I forbindelse med en tænk højt session:

#### Hvordan identificeres problem - clues

##### Userbase:

* Et problem der gør at de ikke kan udføre en opgave. ¨
* Bliver frustreret.
* Laver fejl
* Stopper med at tale – de er for koncentreret og forstår ikke hvad de skal gøre
* Overser vigtig info
* Forvirret og overrasket, tager lang tid, kigger andre steder først. Så jeg trykker bare her… Viser at personen bare gætter sig frem.
* Ændrer strategi og fremgangsmåde
* Spørger om hjælp – så har du et problem, hvis uløst problem gør det umuligt at komme videre, så hjælp, vær altid sikker på hvad problemet er før du hjælper
* Laver negative kommentarer

Find ud af hvorfor de overstående er et problem

Spørg lige efter opgave hvad problemer er. Husk ikke at give for meget respons hvis du er moderator. Skal simulere at det er den rigtige situation.

Sørg for at opgavebeskrivelse ikke kan misforstås. Sig hvad de skal gøre specifikt, og hele historien. Det skal være tydeligt om opgaven er løst, hvornår den er løst.

#### Expert based evaluation:

Her er det POTENTIELLE problemer der er identificeret til at gå imod en specifik heuristic eller guideline. Er ikke definitivt et problem, da det bare er et forslag for hvad der nok er et problem.

## Roller:

### Test monitor (moderator):

#### Vigtige egenskaber:

* Lærer hurtigt, skal hurtigt kunne tilpasse sig brugeren og stemningen
* Skal vide meget om området. Skal vide hvad der er problematisk, og hvad man ikke skal fokusere på.
* Skal være god til at få et godt forhold til testperson, få dem til at føle sig trygge og tilpasse.
* Skal være godt til at lytte og strukturere samtalen, så man holder sig til det relevante
* Kan håndtere usikkerhed.
* God hukommelse
* Skal kunne improvisere. Er meget vigtigt. Det er fx vigtigt hvis brugeren rammer en spændende men ny vej som gerne skal udforskes.

#### Typiske problemer:

* Vil kontrollere i stedet for at støtte
* Bør ikke være så fokuseret på at samle data, men holde flow i samtalen
* Skal ikke holde sig for tæt på testplanen. Prøv at styre flowet lidt, uden at påvirke resultatet for meget.
* Virker bedrevidende. Får ikke brugeren til at føle sig tilpas.
* Hopper til konklusioner.

Må give hjælp hvis det er nødvendigt. Sørg for at finde ud af hvad problemet er. Når du har fundet ud af præcis hvad problemet er, kan du gå videre.

* Hvorfor er bruger forvirret, hvad gør at de er forvirret

Bør ikke tage noter, det er ikke hvor fokus skal ligge. Han skal observer og data logger.

Husk at spørge om mening om systemet efter hver opgave.

Husk at give respons på at du har hørt testpersonen.

Husk at sætte dig i testpersonens sted, og ikke tro at ting automatisk er implicit, selvom det virker sådan.

Husk at der er en hårfin grænse mellem at bryde isen og gøre testperson utilpas. Lad være med at grine af fejl. Lav i stedet kommentarer om dine egne fejl.

Hvis noget ikke er interaktivt i prototypen, så bed personen om at fortælle hvad denne ville gøre hvis det var muligt. Hvorfor gør du det? Hvad forventer du der sker ved det?

Vær neutral. Sørg for at deltagere ikke tror at de fornærmer dig når de ved at det er dig der udvikler det. Forklar at alt kritik er godt og vigtigt at få.

Lad være med at tale på nogen måde om testpersons præstation. Fokuser kun på hvordan programmet er. Det er om programmet, ikke personen. Spørg hvad de ville ændre efter hver opgave.

Få bruger til at læse opgave op. Spørg om de forstår hvad det er de skal gøre. Giv i denne sammenhæng ikke nogen hints, men sørg for at testpersonen ikke er i tvivl om hvad opgaven går ud på, og hvorfor de skal udføre denne. Bed dem om at forklare i egne ord hvad de regner med opgaven handler om.

Spørg først om de forstår hvad systemet egentlig gør. Find ud af om dette er klart.

Sørg for at de aldrig siger ”hvad vil du have…” eller ”jeg gætter på at det er hvad du vil have”

Sørg for at man aldrig kan se hvilke områder der er interaktive, så vil brugere lede efter dem.

### Test subjekt:

### Observer:

### Data logger og video udstyrs operatør: