

och vis kan man säga att designern kommunikerar med användaren via systembilden.

Systembilden är således designerns materialiserade mentala modell och bildar en konceptuell modell av vilka objekt som finns och hur användaren kan interagera med dem. Som designer handlar det således om att få ordning på systemets konceptuella modell. Det innebär att de objekt som finns i systemet måste motsvara något i användarnas begreppsvärld. Relationerna mellan begreppen måste vara logiska i användarnas sätt att tänka.

Metaforer

Ett sätt att utforma ett systems konceptuella modell, är att bygga på metaforer och genom dem dra nytta av vad användarna vet och kan sedan tidigare. Tanken om att datorn är som ett skrivbord med mappar där dokument placeras är ett exempel på en sådan metafor som gjorde det lättare för nya användare att förstå hur det var tänkt att fungera. Metaforer, liksom mentala modeller, är kulturspecifika (Marcus, 1998). Till exempel hade skrivbordsmetaforen inte inbegripit mappar, om den utvecklats i Sverige. I stället hade dokument sorterats in i pärmar. En del metaforer är emellertid generiska, så tillvida att de kan tillämpas i många domäner. En sådan är en tidslinje för att röra sig genom historisk information (Sutcliffe, 2012). Andra metaforer är mer applikationsspecifika, som exempelvis att en digital marknadsplats skulle kunna fungera som en anslagstavla. Samtidigt är själva idén om en digital marknadsplats också en metafor.

En metafor består av en källdomän och en måldomän. Källdomänen är en för användaren välbekant domän, och mellan den och den obekanta måldomänen finns metaforiska motsvarigheter. Om exempelvis källdomänen är en vanlig fysisk anslagstavla och måldomänen är den nya applikationen, kan de metaforiska motsvarigheterna vara att det i den nya applikationen finns saker som annonser som kan anslås, med information om vem man ska kontakta, information om pris, tid och plats m.m. Andra metaforiska motsvarigheter kan vara att annonser anslås så att de senaste ligger längst fram, och där bakom finns lager efter lager av gamla anslag. Metaforen bär med sig analogisk kunskap om vilka objekt som kan finnas i den nya applikationen, liksom relationen mellan dem och hur man kan tänkas interagera med dem.

Handlingscykeln

En användares interaktion med ett system kan beskrivas i termer av en handlingscykel i sju steg (Norman, 1988):

- 3 Specificera en handlingssekvens
- 4 Utföra handlingssekvensen
- 5 Bli varse vad som sker i världen
- 6 Tolka vad som sker i världen
- 7 Utvärdera konsekvenserna av handlingen.

På det stora hela är detta en förenklad bild av hur människor gör saker, men ofta räcker den för att resonera om vad som sker i interaktionen. En risk med handlingscykeln är att den ger ett intryck av att människan är en seriell varelse som gör en sak efter nästa när så inte är fallet. Samtidigt som vi gör något ser vi också vad som sker. Modellen är alltså inte helt korrekt, även om den genom åren har varit inflytelserik, ungefär som Nils Bohrs atommodell med andra ord.

Två viktiga delar av handlingscykeln är avståndet en aktör måste överbrygga för att utföra en handling (eng. *gulf of execution*) och avståndet den måste överbrygga för att utvärdera resultatet av handlingen (eng. *gulf of evaluation*) (a.a.). Avståndet för att utföra något definieras som skillnaden mellan användarens intentioner och tillgängliga handlingsinviter. Det handlar för användare således att överbrygga det avståndet genom att kunna specificera en handlingssekvens och utföra den. Går det fel i utförandet av handlingssekvensen så säger man att användaren slinter. Går det i stället fel då handlingssekvensen bestäms så sker ett misstag: användare tar fel. Blir det fel då intentionen ska formas så blir felet i stället en underlätenhet att handla över huvud taget, eller att användaren glömde att något skulle göras.

Avståndet för att utvärdera resultatet av en handling definieras av skillnaden mellan användarens förståelse av vad som sker i världen och vad som faktiskt sker i världen. Det handlar för användaren om att överbrygga avståndet genom att ta eller få reda på hur saker förhåller sig. Ett fel som kan ske här handlar att man inte inser vilket läge (eng. *mode*) saker är i. En bilförare kan till exempel felaktigt tro att backen ligger i, titta bakåt, släppa upp kopplingen och köra rakt in i väggen framför bilen.

Avstånden för att utföra och utvärdera en handling kan omsättas i fyra designprinciper (a.a.): Den första principen är att se till att saker syns. Kan användaren med andra ord förstå sakernas tillstånd och vilka handlingar som är tillgängliga? Den andra principen är att sätta upp en lämplig konceptuell modell. Det är en fråga om huruvida designen är konsekvent i hur den presenterar sin funktion. Den tredje principen är att skapa god mappning mellan stimulus och respons. Det innebär att relationerna måste vara

samma sätt är det dålig mappning att behöva klicka på en högerpil för att gå till nästa i en vertikal lista. Det skulle då vara naturligare att klicka på en nedåtpil. Utterligare ett exempel är att det är naturligare att ge respons med höger hand på stimuli som uppträder till höger (Wickens & Hollands, 2000). Den fjärde och sista av Normans designprinciper är att ge bra återkoppling. Får användaren till exempel fullständig och kontinuerlig återkoppling på handlingarnas resultat? Det är ofta bra att använda direktmanipulation, vilket går ut på att ha så responsiva system som möjligt, med kontinuerligt representerade objekt på skärmen, som till exempel en fyrkant och en cirkel på ritytan i ett ritprogram (Shneiderman, 1983). Dessa objekt manipuleras sedan med konkreta handlingar som ger direkt synliga effekter. Genom att dra i hörnen på fyrkanten på ritytan ändras till exempel dess storlek. Att handlingarna också är lätta att ångra ingår också i principen om direktmanipulation, liksom en omedelbar återkoppling.

Återkoppling

Återkoppling (eng. *feedback*) på en användares handling handlar i grund och botten om tydlighet om vilka konsekvenser handlingen fått. Det är emellertid svårt att utforma bra återkoppling. En princip att eftersträva är att uppnå ett visuellt driv (eng. *visual momentum*) (Woods, 1984), så att användaren hänger med från skärmbild till skärmbild eller från ett läge till nästa läge. För att åstadkomma ett visuellt driv behövs till att börja med en översiktlig karta, så att användaren inte behöver hålla den övergripande strukturen i huvudet. En viktig del är att skapa en layout som avspeglar meningsfulla relationer mellan objekt. För att knyta ihop vyer eller sidor som kommer efter varandra behövs någon form av landmärken. Landmärken ger referensramar, så att det blir tydligt hur vyerna eller sidorna relaterar till varandra. Överlapp mellan sidor så att användaren ser lite av den del som nyss lämnades hjälper också användaren att ha koll. I grunden handlar det om en tydlighet om var användaren är och vart han eller hon kan gå härnäst.

Feed-forward

Att ge feed-forward handlar just om vart användaren kan gå härnäst. Användaren behöver i förväg veta vilka handlingar som är möjliga, och behöver kunna bygga realistiska förväntningar om vilka effekter de handlingarna kommer att få. Synliga handlingsinviter spelar en roll i detta, men också att hjälpa och motivera användaren att forma lämpliga mål och intentioner, samt bygga realistiska förväntningar på resultatet. Ett exempel på feed-forward

Uppmärksamhet

En kritisk flaskhals i interaktionen är användarnas uppmärksamhetsförmåga (Wickens & Hollands, 2000). Blir vi distraherade och uppmärksammar något annat så missar vi till exempel vad någon säger. Ibland fokuserar vi på fel saker och ser inte det uppenbara, och det händer att vi försöker dela vår uppmärksamhet mellan flera saker samtidigt, vilket kan leda till att vi misslyckas.

Uppmärksamheten har liknats vid en strålkastare. Dels har ljuskäglan en bredd med ett fokus på det som vi vill bearbeta. Samtidigt faller en del distraherande saker inom ljuskäglan som orsakar delad uppmärksamhet. Ljuskäglan riktning kan ändras, beroende på vad som sker i omgivningen. Riktningen guidas också av våra mentala modeller av hur världen är beskaffad. Vi bygger förväntningar om vad vi ska få se och får vi till exempel en förhandsvisning om vad som komma skall (*feed-forward*) kan vi hålla koll efter just det. Det är dock lätt att distrahera uppmärksamheten genom att presentera saker perifert som drar den till sig. Rörelse eller ens eget namn är exempel på sådant som drar fokus till sig.

Designmässigt finns det fördelar att använda *multimodal design*, alltså använda flera modaliteter samtidigt i designen (a.a.). En signal kan förstärkas så att den blir tydligare om till exempel ett visuellt verbalt varningsmeddelande också har ett varningsljud och en visuell icke-verbal varningsindikator kopplade till sig. Ljudet kan också styra uppmärksamheten, så att en hög ton kopplas till att man ska titta uppåt efter något, medan en låg ton kopplas till att man ska titta nedåt. Ett alternativ till att använda flera modaliteter för att förstärka signaler är att låta modaliteterna representera olika saker. En del saker kan exempelvis presenteras auditivt, medan andra saker presenteras visuellt, och ytterligare andra saker presenteras *haptiskt* (med känseln). Det finns då en tendens till att synintrycken domineras över hörsel och känsel. Samtidigt har vi lättare att bortse från störande synintryck, men hörsel- och känselintryck kan vi inte stänga ute. Om vi hör ett skarpt ljud, tvingas vi att uppmärksamma det, och likaså om någon petar oss i sidan.

Närhetskompatibilitet

För att användaren ska kunna hålla uppmärksamheten på rätt sak kan en designer dra nytta av något som brukar kallas närläget (Wickens & Carswell, 1995). Det handlar om hur nära varandra saker bör placeras. Närhet handlar huvudsakligen om ett fysiskt avstånd, men närläget kan också förstärkas genom gemensam färgkodning på objekt eller andra

skillnad från om en variabel representeras med en linje medan den andra representeras med en siffra. Principen om närhetskompatibilitet säger att om saker används tillsammans, ska de också vara nära varandra. Men har man saker för nära varandra som inte har med varandra att göra så kommer de i stället att störa varandra.

Arbetsminnet

Om en användare lyckas hålla uppmärksamheten på något beror delvis på dennes arbetsminneskapacitet. Arbetsminnet är det minnessystem där vi tillfälligt lagrar den information som vi bearbetar. Det kan liknas vid en sorts arbetsbänk där vi medvetet undersöker, utvärderar, bearbetar och jämför olika mentala representationer (Wickens & Hollands, 2000). Det är också i arbetsminnet vi håller saker innan vi kan lagra in det i långtidsminnet. Vi har i princip två olika sorters arbetsminne: det *visuospatiala* och det *verbala* (även om ett rörelsemässigt arbetsminne också föreslagits). Designmässigt betyder det att om en krävande uppgift till sin natur är visuospatial, ska vi som designer inte lägga ytterligare belastning i en samtidig uppgift som också är visuospatial. Att köra bil i stadstrafik kräver exempelvis mycket visuell uppmärksamhet, och att samtidigt hålla på med en pekskärm som också kräver visuellt arbete är då en dålig idé. Det är då i stället bättre att lägga en underordnad uppgift på det den verbala delen av arbetsminnet (Liu & Wickens, 1992). Det finns en anledning till att kartläsare i rally läser upp för föraren vad som komma skall, snarare än att visa det på en karta. På samma sätt stör verbal input och output arbetsuppgifter som kräver mycket av verbal bearbetning, som att till exempel att skriva, läsa, räkna, eller använda ett kommandobaserat gränssnitt. Samtidigt är det självfallet effektivare att använda en visuell display för visuospatiala uppgifter, snarare än att använda ljud för att representera visuell och spatial information.

Arbetsminnet har också en central exekutiv del som koordinerar utförandet av samtidiga uppgifter, hämtar och under kortare tid hanterar information från långtidsminnet, samt håller uppmärksamheten på vissa saker i världen. Denna del hanterar sådant som man medvetet måste uppmärksamma och bearbeta. Rutinuppgifter som är automatiska sköter andra subsystem i människans kognitiva apparat. Den centrala exekutiva delen sätter begränsningar på uppmärksamheten, och vi kan inte klara hur mycket som helst på en gång, men allteftersom vi bli skickligare på något kan vi omvandla medvetna krävande processer till automatiska rutinuppgifter, och då belastar de inte heller arbetsminnet.

det och repetera det. Om vi till exempel ska komma ihåg ett telefonnummer då vi går från sovrummet till köket, måste vi hela tiden upprepa det för oss själva. Om någon stör vår uppmärksamhet på denna uppgift tappar vi informationen ur arbetsminnet. Det är därför en god idé att avlasta användaren så att han eller hon inte behöver komma ihåg saker utan kan använda externa minnen som minneslappar eller att ett gränssnitt helt enkelt visar informationen kontinuerligt. Det är exempelvis svårt för en användare av ett hjälpsystem att komma ihåg många steg med instruktioner. Därför läggs oftast hjälpen ovanpå resten av gränssnittet, så att användaren kan ha den framme samtidigt som instruktionerna följs. Om instruktionerna dessutom kan bättas in i användargränssnittet, är det så klart ännu bättre.

Vi kan inte heller hålla hur mycket information som helst i arbetsminnet. Miller (1956) konstaterade i sin artikel *The magical number seven plus or minus two* att arbetsminnets kapacitet är 7 ± 2 informationsenheter. Om vi ska hålla en slumpvis uppsättning siffror i huvudet, så klarar vi mellan fem och nio stycken. Faktum är att det ofta till och med är mindre än så. Är vi till exempel trötta så minskar arbetsminneskapaciteten betänktligt. Det går emellertid att komma runt denna begränsning genom att dela upp det som ska hållas i minnet i större enheter. Det är svårare att hålla "1598322007" i minnet än "159 83 22 007". Det här kallas *chunkning* (eng. *chunking*). Kan vi dessutom göra dessa enheter meningsfulla för oss, så kan vi dra nytta av långtidsminnet vilket ytterligare bidrar till att man minns det enklare. 1598 32 2007 blir för mig en enklare chunkning att minnas, om jag vet att slaget vid Stångbro var 1598 då kung Sigismund var 32 år, och att jag själv var 32 år 2007. En lärdom för design är att underlätta för användaren genom att skapa meningsfulla grupperingar. OCR-nummer på räkningar är exempelvis ett evigt gissel för användare av nätbanker.

Ytterligare en aspekt av arbetsminnet att ta hänsyn till är att vi lätt blandar ihop saker som påminner om varandra. Om man vill vara riktigt elak, kan man rabbla siffror samtidigt som någon försöker komma ihåg ett telefonnummer. Saker som ska hållas isär bör alltså göras olika varandra. Principen om närhetskompatibilitet kommer tillbaka här, och genom utformningen på gränssnittet kan vi alltså göra det mer eller mindre lätt för användarna.

Fitts lag

En annan princip som kan användas för att göra saker enklare för användaren är Fitts lag. Principen säger att tiden det tar att pricka en målyta är beroende av avståndet till målytan och storleken på målytan (Fitts, 1954).

I design betyder det att knappar måste ha en rimlig storlek. Dessutom är kanterna och hörnen på skärmen kraftfulla eftersom att de ger mycket stora objekt. Med muspekaren kan användaren bara snabbt röra den uppåt höger, och den kommer att landa i hörnet. Till sist betyder Fitts lag att komponenter som finns nära det som användaren jobbar på kan nås snabbt. Det är därför kontextmenyer som kommer fram på högerklick är en god idé. Fitts lag pekar också på att det finns en avvägning mellan hastighet och korrekthet. Antingen gör man saker fort och missar lite, eller så gör man saker noggrant och långsamt.

Hick-Hymans lag

Fitts lag kompletteras av Hick-Hymans lag som lite förenklat säger att den tid det tar för mäniskor att göra ett val beror på hur många valmöjligheter de har (Hick, 1952; Hyman, 1953). Dessutom tar det längre tid att välja bland ovanliga och oväntade val än bland vanliga och väntade val. Som designer betyder det att man bör ta bort onödiga valmöjligheter, och om mäniskor kan göra det, delar de upp sina val i kategorier vilket minskar tiden att välja. Om något sticker ut, väljer de det. På det stora hela pekar detta mot att vända ut och in på en sajt, och i stället för att man har en navigationshierarki att ta sig igenom bör man presentera det primära innehållet på förstasidan. Bilder kan fungera som ankare för uppmärksamheten, vilket gör det lättare för användare att dela in valmöjligheterna i kategorier.

Grafisk profil

Föregående del av kapitlet täckte in ett antal principer från mänsk-dator-interaktion och teknisk psykologi som är bra att ha i bakhuvudet då man skissar på gränssnitt. Parallelt med gränssnittskissningen är det inte ovanligt att ta fram den grafiska profilen för projektet.

Om vi återvänder till vårt projekt Datorförstärkta landskap, så utgick vi där från brukskvalitetsanalysen såsom den uttrycktes i ordmolnet (figur 4.4) och gestaltade den visuellt i en *moodboard*, alltså en i form av ett collage med bilder som vi associerade med nyckelorden. Det gjorde att vi kunde ta fram ett färgschema, en känsla och en ton att anslå i designen. Det är i detta skede viktigt att hitta ett bildspråk som kan rikta in designarbetet, särskilt på grund av att ord kan vara så tvetydiga. Jag lyssnade på ett föredrag av den norske designern Einar Hareide, där han gav följande exempel: Låt oss säga att en kund argumenterar för att designen ska vara typiskt skandinavisk. Då kan jag få folkdräkter, kurbitsmålning och lusekoftor i åtanke, eller så tänker jag



Figur 4.22 Moodboard (foton av Thea Dahlqvist och Mattias Arvola).

bli konkret på ett helt annat sätt om den riktning arbetet ska ta. Figur 4.22 visar ett exempel på en moodboard som huvudsakligen är uppbyggd av foton tagna på Astrid Lindgrens Näs.

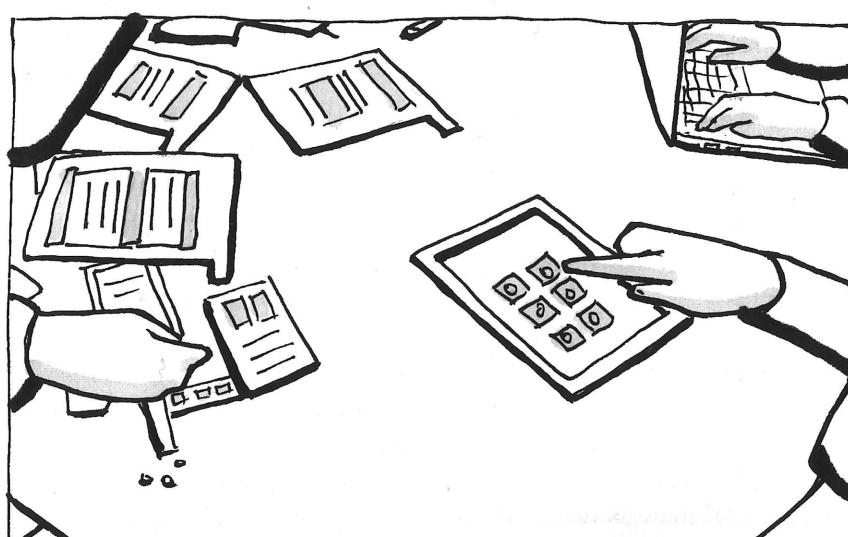
Pappersprototyper

När designgruppen har en uppsättning skissade skärbildsritningar och en bild av hur de ska ordnas i webbkort och flöden är det dags att sätta samman en pappersprototyp. I gränssnittskissen handlar det fortfarande om att vara öppen och sökande, men i pappersprototypen går man över till att bli konkret i sin lösning som ska beskrivas och testas. En pappersprototyp består av papper och kartong och statiska skärmkomponenter. Den blir interaktiv genom att en mänsklig "spelar dator" och ger användaren den återkoppling som datorn i verkligheten skulle göra. Det sker genom att användaren pekar på skärmkomponenter och säger vad den gör, varvid den som spelar dator lägger ut nya dialoger och skärbilder framför användaren.

(Saffer 2010, s. 174). Under bearbetningsfasen arbetar en designer gärna med papper och penna innan det är dags att gå över till datorn. Faktum är att så många som 80 procent av alla interaktionsdesigner arbetar med pappersprototyper (Warfel, 2009). Det gör det till det överlägset populäraste prototypningsverktyget.

En pappersprototyp ska vara grovhuggen och snabbt hopsatt. Det ska synas att designgruppen inte lagt så mycket jobb på den. Då har användaren lättare att komma med kritiska kommentarer, utan att känna att den riskerar att förlämpa designern. Även om utformningen ska vara grovhuggen, måste den vara skalenlig för att man inte ska råka göra en gränssnittsdesign som sedan inte blir bra på den apparat och med den upplösning som slutprodukten har. Figur 4.23 visar en pappersprototyp som testas med användare.

Det som huvudsakligen testas i en pappersprototyp är flödet, begripligheten och det övergripande konceptet. Eftersom de görs snabbt och ingen kod skrivs så är de också det billigaste prototypningssättet. Det är också lätt för användare och andra intressenter att vara med och bygga pappersprototyper och de är enkla att modifiera tillsammans med användare. Genom att de är fysiska är de lätt att samlas och diskutera runt, både under skapandet i designgruppen och i tester med användare. Det blir inte samma dynamik när man jobbar i datorbaserade verktyg som till exempel Balsamiq (ett datorverktyg för att bygga skissartade klickbara prototyper). Runt de fysiska materialen kan alla mötas på ett mer jämställt sätt (Bødker & Grønbæk, 1991).



En annan fördel med pappersprototyper är att kreativiteten inte begränsas av existerande mjuk- och hårdvara, och redan färdiga och förutbestämda komponenter.

Vad behövs då för material för att bygga en pappersprototyp? Warfel (2009) rekommenderar följande:

- Vanligt vitt papper som bakgrund
- Genomskinliga plastark (sådana som man brukade ha till overheadprojektorer) för att skapa effekter som aktiveras när muspekaren hålls över en gränssnittskomponent, och för skugga ut komponenter som inte är valbara
- Index-kort för dialoger och menyer
- Klisterlappar för att visa förändrade tillstånd och markering av val
- Färgpennor av olika tjocklek och färger
- Tejp, limstift eller häftmassa för att hålla gränssnittskomponenter på plats
- Tandtråd för att skapa animationer.

Warfel betonar att det är en prototyp och inte Mona Lisa som ska göras. Det betyder att designgruppen inte ska överarbeta den. Eftersom att det är fullt möjligt att fortsätta och finslipa i oändlighet är det viktigt att sätta upp ett slutt datum och endast bygga en prototyp för det som man vill testa. Som nämntes i första kapitlet talar man ofta om T-prototyper. Det betyder att man börjar med att definiera vilka uppgiftsflöden som man ska bygga prototypen för. Sedan ritas bara skärmbilderna och dialogerna som behövs för de viktigaste flödena. På ytan ser det ut som om hela prototypen skulle vara byggd, men användaren kommer bara att kunna göra ett fåtal uppgifter i den. Det är alltså inte meningsfullt att prototypa hela systemet om det är ett stort system, utan det finns bara ett djup på ett fåtal funktioner.

En god start är således att börja med att definiera vilka uppgifter som ska testas med användarna innan prototypen byggs. Välj ett rimligt antal av de viktigaste uppgifterna. Inte för många, för då blir det ohanterligt, men inte så få att testet blir meningslöst. En bra uppgift är ytterst specifik och beskriver en hel aktivitet som ska testas. Det ska vara tydligt var uppgiften börjar och var den slutar. Helst ska uppgiften som användaren ska göra sträcka sig över flera funktioner, så att man testar hur gränssnittsdelen samspelear. Ett exempel på en tydlig uppgift i en musikapplikation skulle kunna vara: "Hitta och starta låten Uprising av Muse i den version som är på skivan Live At Rome Olympic Stadium". För denna uppgift finns en tydlig start, så att

sina användningsscenarion för att hitta lämpliga kandidater. Det kan vara en god idé att testa sina testuppgifter med någon, för att säkerställa att de är realistiska och att de data som används i uppgiften också är realistiska.

När uppgifterna är klara är det bara att rita upp alla skärmbilder och dialoger som behövs för att genomföra uppgiften, men innan testerna med användare startar är det alltid en god idé att köra en eller ett par pilottester med kollegor, släkt eller vänner, för att säkerställa att man inte gjort några uppenbara misstag i upplägget och i formuleringarna.

Bearbetningsfasens värderingar

Det är i testerna av pappersprototyper som resultaten av bearbetningsfasen värderas. När man testar dem används tänka-högt-protokoll, vilket är en teknik som också togs upp i konceptfasens insiktsdel (se kapitel 2). Det går ut på att de som testar ombeds att tänka högt medan de löser de uppgifter de getts. Efter det att pilottesterna är kördta, eller redan tidigare, måste testanvändare rekryteras. Det tar nämligen alltid längre tid än man tror att få tag på deltagare till ens tester. Nedan beskrivs hur tester av pappersprototyper görs. Beskrivningen bygger i mångt och mycket på Rettigs procedur (Rettig, 1994).

Deltagare

De deltagare som ska testa prototypen bör så långt som möjligt vara representativa för den tänkta målgruppen, särskilt med avseende på utbildning, datorkunnande, domänkunskap och vilka uppgifter som användarna typiskt sett utför. För att få med så många potentiella designproblem som möjligt ska så mycket som möjligt av variationen bland användarna täckas in. Att få tag på representativa användare kan ibland vara svårt, och då vill man ha så kallade surrogat användare, som är så lika som möjligt på de mest kritiska aspekterna. Om det, till exempel, är svårt att få tag på yrkesverksamma lärare, skulle lärarstudenter på sitt sista år kunna fungera som surrogat. Endast i sista hand ska man testa med kollegor, släkt och vänner.

Förberedelser

Vid själva testet, som ofta tar mellan en halvtimme och en timme per testanvändare, behövs fyra olika roller. Till att börja med behövs en värd som tar emot testanvändaren och får denne att känna sig bekväm, ger en introduktion och delar ut eventuella förtestenkäter. Nästa roll är facilita-

flera observatörer som tar anteckningar på index-kort. Endast en observation skrivas ned på ett kort, men idéer på lösningar kan också skrivas ned på samma kort. Man blir ofta trött av att koncentrera sig på en sak hela tiden, och ska många deltagare testa i följd är det bra att rotera mellan rollerna. Ibland kan en person fylla flera roller, även om det inte är att föredra.

Inför testet, när testanvändaren kommer, är det viktigt att få till en god stämning, och introduktionen kan med fördel ske under en liten fikastund. Deltagaren behöver få rätt förväntningar på det som ska ske och ge uttryck för *informerat samtycke*, antingen muntligt eller skriftligt, precis som i konceptfasens insiktsdel (se kapitel 2). Det betyder att värden måste förklara hur pappersprototyper fungerar, vad syftet är och hur insamlade data kommer att användas och av vem, samt hur tänka-högtprotokoll fungerar. Värden säkerställer att deltagaren förstår att deltagandet är helt anonymt och helt frivilligt, vilket betyder att han eller hon kan avbryta när som helst. Testanvändaren måste också förstå att det är designen som testas, och inte testanvändaren. Om deltagarna är minderåriga, måste informerat samtycke inhämtas från vårdnadshavare. Sammanfattningsvis innebär informerat samtycke att deltagarna förstår:

- att de går med på att delta i studien
- att det är frivilligt att delta
- att de går med på att omedelbart säga till om de undrar över något eller känner obehag
- att deras interaktion observeras och eventuellt spelas in
- hur resultat och inspelningar kommer att användas
- att de kommer att anonymiseras.

Inför testet ställs också eventuell inspelningsutrustning upp. Om testet spelas in på video, kan kameran ställas så att den spelar in deltagarens interaktion med prototypen, men inte dennes ansikte: därigenom säkerställs anonymiteten. Ibland vill man emellertid spela in deltagarnas reaktioner och då måste även ansiktet vara med. Om deltagandet inte kan anonymiseras, måste detta framgå tydligt. Före testet är det också brukligt att samla in en del data om användaren, genom en enkät eller intervju. Förtestfrågor bör vara nedskrivna och väl förberedda. Annars kan testledaren råka ställa ledande frågor. Saker som är vanliga att ta upp inkluderar:

- Kön
- Ålder

- Kännedom om konkurrenter
- Datorvanor (plattformar, typiska aktiviteter, webbläsare, webbplatser och applikationer som ofta används)
- Språkkunskaper
- Domänkunskap
- Teknisk kunskap
- Kontaktinformation (om man vill återkomma med fler frågor).

Under testet

När själva testet startar kan designgruppen ta reda på deltagarens initiala intryck och förståelse för designen genom att fråga deltagaren vad dennes första intryck är och vad den tror att det är för sorts produkt, system eller tjänst. Därefter kan testledaren kort introducera designen till den nivå som designgruppen tror att en faktisk användare kommer att ha. En god idé är att sedan be deltagaren att beskriva vilka han tror att alla från början synliga komponenter i gränssnittet är. Det ger in inblick i användarens initiala mentala modell av systemet. Om dennes beskrivning inte motsvarar systemets konceptuella modell, är det en brist i gränssnittets utformning.

Efter dessa första frågor kan testet av uppgifterna ta vid. Det bästa är om varje uppgift är nedskriven på ett papper som kan räckas över till testdeltagaren, så att uppgiften presenteras på samma sätt till alla. Uppgiften ska vara formulerad så att den fokuserar på vad användaren vill uppnå utan att beskriva hur användaren ska göra det. Om uppgiften ger en vink om hur användaren ska göra, testas inte längre designen. Då testas i stället om användaren kan läsa instruktioner, och det är ju inte meningen. Om det är möjligt, bör ordningen på uppgifterna varieras. Annars kommer det att verka som om det är fler designproblem i första uppgiften än i sista uppgiften, eftersom användarna lär sig hur systemet är uppbyggt under testets gång. Om två olika designalternativ testas (s.k. AB-test) och ska jämföras, måste hälften av deltagarna börja med den ena designen, medan andra hälften får börja med den andra, för att jämna ut inlärningseffekter.

Under testens gång måste testledaren ibland uppmuntra deltagaren att tänka högt. Det är annars lätt hänt att tystna när det blir svårt. Ibland kan även observatörerna skjuta in en fråga, men ingen får skratta, säga "aha", titta menande på varandra eller visa andra reaktioner på det användaren gör. Det är viktigt att inte påverka användaren på något sätt under testningen, och absolut inte få användaren att känna sig obekvämt. Skulle det hända så erhålls inte några pålitliga resultat. Det är också viktigt att varken testledaren eller

försöker och uppmana dem att tänka högt. Först när de verkligen har kört fast kan testledaren berätta hur det är tänkt att göras.

Pappersprototypen ska ordnas så att testanvändaren inte ser de nästkommande skärmarna. Annars får de bättre överblick än det faktiska gränsnittet kommer att kunna erbjuda dem, och testet blir inte tillförlitligt. Den som spelar dator måste ha koll på exakt när vilken återkoppling ska ges och det kan vara bra att köra träningsomgångar innan, så att det inte strular i testsituationen.

Vad är det då som ska observeras i testet? Det viktigaste är att skriva ned vad användarna gör och säger, för att i slutändan kunna dra slutsatser om hur de tänker och känner. Det gör att man antecknar följande saker under observationen och tittar efter i videoinspelningar:

- Vilka vägar användaren tar genom prototypen
- Framgång på uppgiften (lösar inte uppgiften, löser den med svårigheter, löser den med lätthet)
- Situationer där användaren hamnar i svårigheter
- Beteenden som visar att användaren inte förstår gränssnittet.

Det handlar i princip om att identifiera problem som användarna stöter på. Som problem räknas saker som hindrar användaren att nå sina mål, som leder användaren fel, skapar förvirring, eller ger upphov till felhandlingar, som att användare slinter, gör misstag, eller underläter att gör något. Det kan vara att användarna missar saker som de borde lägga märke till, att de antar att något är korrekt när det faktiskt är fel, eller att de tror att de är klara när de inte är det. Andra exempel på problem är att de väljer fel funktion, missförstår innehållet eller inte förstår navigationen.

Efter testet

Efter testet kan testledaren och observatörer lyfta upp intressanta episoder för diskussion. Eftertestfrågor bör, precis som förtestfrågor, vara nedskrivna och väl förberedda. Efter testet är det viktigt att samla in användarens intryck och tankar kring designen. Lämpliga frågor att ställa efter testen täcker in följande punkter:

- Övergripande intryck
- Vad som var bra
- Vad som inte var bra
- Vad som kan förbättras
- Vad som saknas (innehåll och funktioner)

Analys

Anteckningarna bör samlas på index-kort. Anledningen till det är att det blir lättare att analysera resultaten. Rettig (1994) föreslår att designgruppen efter testningen lägger ut hela pappersprototypen på ett stort bord och sedan lägger varje kort vid de gränssnittskomponenter som kortet berör. Sedan kan gruppen dela upp arbetet med att gå igenom högarna med kort för att sammanställa och prioritera problem, innan man tillsammans går igenom dem och försöker hitta lösningar på identifierade problem.

Lösningarna kan skissas ned och med ett gem sättas fast på den aktuella delen av pappersprototypen. Detta blir sedan grunden för att gå över i nästa iteration med en mer detaljerad datorprototyp eller designspecifikationer.

Granskning som alternativ till testning

Ett alternativ till att testa sin prototyp med användare i bearbetningsfasen är att göra en granskning. Ett exempel på en vanlig granskningssmetod är *heuristisk utvärdering* (Nielsen, 1993). Heuristik är det samma som en uppsättning tumregler, och det är därifrån metoden fått sitt namn. Metoden går till så att man samlar 3–5 granskare och utgår från en utprovad uppsättning av tumregler eller principer att utvärdera prototypen utifrån. Till att börja med bekantar sig granskarna med tumreglerna och diskuterar ihop sig om vad de betyder i detta specifika fall. Sedan sätter de sig enskilt och går igenom prototypen ett första varv för att få en överblick. Därefter går varje granskare igenom prototypen mer noggrant, skärbild för skärbild, och identifierar problem utifrån var och en av tumreglerna (fortfarande individuellt). Detta ger en lista med potentiella problem vilken tas tillbaka till granskningsgruppen där alla individers input sammantälls. Dessutom bedömer granskningsgruppen hur allvarliga de identifierade problemen är. Problem som uppträder ofta för användarna är allvarligare än de som uppträder sällan. Problem som har stor effekt och som är svåra att åtgärda är allvarligare än de som har liten effekt och är enkla att åtgärda. Problem som uppträder varje gång användaren använder en vanlig funktion är allvarligare än problem som bara uppträder en gång för användare som inte känner till det. Hur allvarliga problemen är kan bedömas på en femgradig skala från 0 till 4:

- 0 Jag håller inte med om att detta är ett problem över huvud taget
- 1 Kosmetiskt problem – behöver inte åtgärdas om det inte finns tid över
- 2 Mindre problem – att åtgärda det bör ges låg prioritet

Vilken uppsättning av tumregler som används styr självfallet vilka problem som kommer att observeras. Det finns olika uppsättningar med tumregler, men i gränssnittsdesign är det vanligt att man använder Nielsens (1993) uppsättning av tumregler (baserat på en översättning av Ottersten & Berndtsson [2002]):

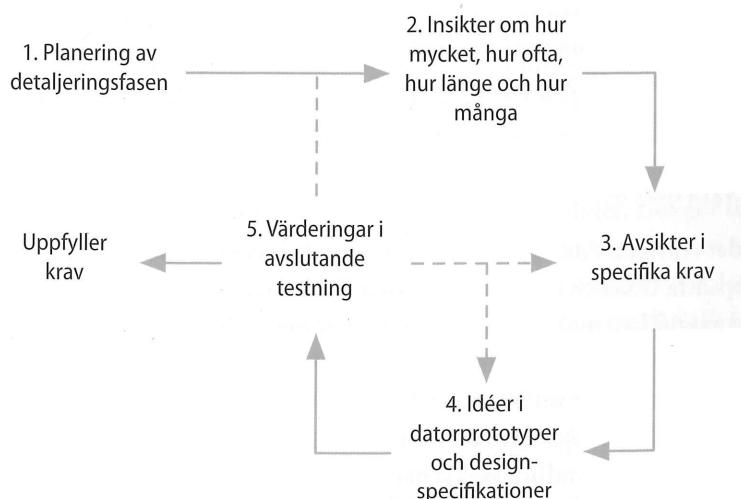
- 1 *Enkel och naturlig dialog*: Ingen irrelevant eller sällan använd information; relevant info ska vara synlig, och komma i naturlig och logisk ordning.
- 2 *Använd ett naturligt språk*: Använd ord som är bekanta för användaren.
- 3 *Minimera användarens minnesbelastning*: Gör valbara objekt och funktioner synliga, så att användaren inte behöver komma ihåg saker från en del av systemet till en annan. Instruktioner ska vara synliga eller lätt att få fram.
- 4 *Enhetlighet*: Användare ska inte behöva fundera på huruvida olika ord, situationer eller handlingar i systemet betyder samma sak. Följ plattformsriktlinjer.
- 5 *Förse användaren med återkoppling*: Systemet ska informera om vad som sker i det.
- 6 *Förse användaren med klart markerade funktioner för att avbryta dialogen*: Det är lätt att välja fel av misstag; gör det därför möjligt att ångra och reparera. En tydligt markerad nödutgång behövs för att hitta tillbaka.
- 7 *Effektiv användning*: Kortkommandon snabbar upp för experten. Stöd både erfarna och oerfarna användare.
- 8 *Tydliga felmeddelanden*: Använd ett enkelt språk, visa vad som är fel och föreslå en lösning.
- 9 *Förhindra fel*: Bättre än ett bra felmeddelande är att utforma produkten så att problemet inte uppstår.
- 10 *Hjälp och dokumentation*: Hjälp och dokumentation ska vara lätt att söka i, fokuserad på användarens uppgift, och inte för omfattande. Lista konkreta arbetssteg.

Nästa fas

Efter att värderingen i bearbetningsfasen har omsatts i omdesign är det dags att bli ännu mer detaljerad och fokuserad. Designarbetet går då in i den iteration som kan kallas detaljeringsfasen. Det är där designspecifikationer tillverkas och dokumenteras.

DETALJERINGSFASEN, ÖVERLÄMNING OCH AVSLUTNING

Den tredje iterationssnurran i designprojektet är detaljeringsfasen (figur 5.1). Där kan man vid behov börja med att komplettera sin undersökning med användare och intressenter med frågor om hur mycket, hur ofta, och hur länge användarna eller intressenterna behöver göra något. Resultaten av den formativa utvärderingen av bearbetningsfasens design och den kompletterande undersökningen omsätts i mer specifika kvantitativt mätbara användar- och verksamhetskrav. Därtill görs en omdesign som kan resultera i interaktiva datorbaserade prototyper med hög detaljeringsgrad (Arnowitz, Arent & Berger, 2007; Warfel, 2009), på vilka en mer formell avslutande användbarhetstestning kan göras gentemot ens kvantitativa krav (Tullis & Albert, 2013) eller genom fälttestning (Sharp, Rogers & Preece, 2011). En sådan datorprototyp kan fungera som specifikation i sig själv, men det är vanligt att också ta fram särskilda designspecifikationer (Goodwin, 2009).



Figur 5.1

Detaljeringsfasen
en del av design
människan i centrum
enligt ISO 9241-210

Detaljeringsfasens insikter

Kompletterande datainsamling i detaljeringsfasen handlar om att specificera de abstrakta kvalitativa kraven från bearbetningsfasen, genom att ta reda på mättsatta och kvantitativa krav om mängd, frekvens och längd.

Det kan till exempel vara värt att ta reda på hur ofta en användare gör något, hur lång tid det brukar ta, eller hur många objekt som hanteras samtidigt. Det kan också handla om hur väl användarna upplever att de kan lösa en uppgift, eller hur mycket konkurrenter uppskattas. Enkäter brukar i detta skede kunna fungera som underlag, men även arbetsmöten med intressenter kan vara en lämplig datainsamlingsmetod.

I detta skede tar man reda på så specifika saker att det blir möjligt att sätta siffror på designmålen för att sedan ta reda på om man når dem. Det handlar alltså om att hitta kritiska värden att jämföra den egna designen mot.

Detaljeringsfasens avsikter

När data är kompletterade är det dags att definiera mätbara användar- och verksamhetskrav. Exempel på mått skulle kunna vara:

- 100 % av användarna ska klara de viktigaste uppgifterna (vilka måste specificeras).
- 90 % av kunderna ska vara nöjda med väntetiden.
- 90 % av studenterna ska uppleva innehållet som trovärdigt, lärorikt och intressant.
- 80 % av ärendena ska ta mindre än tre minuter.
- 90 % av uppgifterna ska kunna utföras utan några fel.
- Tiden för att utföra uppgifterna ska vara kortare än hos konkurrenterna.
- Upplevd användbarhet, mätt med System usability scale (SUS), ska vara över 85.

Här är det viktigt att de uppsatta mätten grundas i effektmålen för projektet och i uppsatta UX- och brukskvalitetsmål. För att mäta hur väl någon lyckas med en uppgift kan man antingen ha ett binärt mått (lyckades eller lyckades inte) eller så kan man använda bedömningsnivåer om hur väl de lyckades. Effektivitet mäts ofta med hur lång tid det tar att utföra en uppgift, men det är också vanligt att registrera fel som användarna gör, det vill säga avsteg från den optimala handlingssekvensen. Hur lätt systemet är att lära sig – lärbart – mäts som förändringar i effektivitet över tid. Ju längre tid det tar

användaren stöter på i olika uppgifterna. Som mål, eller krav, kan man då sätta upp en gräns för hur många problem som kan accepteras. För de viktigaste uppgifterna kanske inga problem kan accepteras, medan enstaka problem för en del användare kan accepteras för de mindre viktiga uppgifterna.

Detaljeringsfasens idéer

I designprocessens sena skede handlar idéarbetet primärt om utseende och känsla. Interaktion och struktur revideras vid behov, och ibland kan funktionalitet och innehåll läggas till eller tas bort. Den följande texten tar upp just utseende och känsla. Därefter beskrivs hur datorprototyper kan användas och designspecifikationer sättas upp.

Utseende på gränssnittet

Tidigare i processen har de visuella aspekterna varit skissartade. I detaljeringsfasen måste varumärket kommuniceras och det är dags att besluta vad produkten ska likna och vilka associationer den ska ge upphov till. Tidigare framarbetad moodboard måste bakas samman med skissade skärmbildsritningar och pappersprototyper. Därutöver måste den visuella kommunikationen och hierarkin mellan olika delar i gränssnittet tydliggöras.

Genom att ordna saker visuellt i grupperingar och genom att synliggöra relationer och skillnader mellan objekt stöder man uppmärksamheten och arbetsminnet. Till att börja med bör den visuella designen styra ögats flöde, så att arbetsuppgiftens flöde framträder. Användaren ska se och tänka: ”Först gör man det, sedan det och sedan det”. I vår västerländska kultur tenderar vi att börja uppe till vänster och arbeta oss ned till höger.

Komponenterna på skärmbilderna riktas också in, så att de är justerade i förhållande till varandra. Det underlättar för användaren att snabbt skanna bilden. Till stöd för att göra denna justering bör designern använda ett rutnät (eng. *grid*) som återkommer i alla skärmbilder. Det gör designen vilsammare, och det blir lättare att märka ut sådant som användaren bör uppmärksamma. Om designen följer ett mönster och sedan bryter det, märks det mönsterbrottet. Om designen är en visuell geggamoja där allt ser olika ut, blir det omöjligt att urskilja något alls. Det är alltså bättre att sätta upp ett rutnät, justera alla element, styra ögat uppifrån vänster ned mot höger och sedan följa ett mönster som bryts på strategiska ställen, för att dra till sig uppmärksamheten.

För att förenkla en skärmbild tar man bort saker som inte behövs. Det

kombineras och slås ihop till sammansatta element, särskilt om de också ska användas tillsammans.

En visuell hierarki skapas genom att den viktigaste informationen och gränssnittselementen betonas och de mindre viktiga tonas ned. Det görs genom att sätta upp en kontrast mellan objekt i termer av visuella egenskaper som storlek, position, form, färg och mättnad. Ju större kontraster, desto tydligare blir hierarkin.

Viktiga saker placeras högt upp eller centralt; de bör vara stora, urskilja sig i form och färg, och kan vara mer mättade. De får då en större visuell tyngd och syns i det perifera seendet. I text handlar det om rubriknivåer och skillnaden mellan dem, men också om hur punktlistor poppar ut från resten av texten. Bilder drar till sig ögat och fungerar som ankarpunkter för att segmentera avläsningen av en skärmbild.

Segmenteringen av skärmbilden handlar i mångt och mycket om just visuella ankare eller landmärken där ögat kan stanna, men också hur objekten grupperas på ytan. Olika sorters objekt ska ha sina platser. Vitytan runt en grupp objekt måste vara så stor att gruppen inte blandas samman med en annan grupp. Inom gruppen måste den vara så stor att de enskilda objekten kan skiljas åt, men inte så stor att de inte längre bildar en grupp. Del-helhets-relationer måste framträda tydligt. Saker som har samma färg eller samma form upplevs också höra ihop, liksom saker som följer samma rörelsemönster.

Det finns avsevärt mer att säga om visuell kommunikation i användargränssnitt, men det går bortom denna introducerande bok. Läs mer i Mullet och Sanos mycket läsvärda bok *Designing visual interfaces* (Mullet & Sano, 1995).

Känsla i interaktionen

Om den ena delen av den detaljerade designen handlar om utseende, så handlar den andra om känslan hos produkten eller tjänsten. Lim, Lee och Kim (2011) arbetar med något de kallar *interaktivitetsattribut* för att beskriva känslan. Det är egenskaper som handlar om hur interaktiviteten är, eller ska vara. Det är en uppsättning osynliga och dynamiska egenskaper som en designer kan tänka igenom för att föreställa sig uttrycksfulla och nyskapande interaktiva produkter och tjänster, och för att få med såväl inmatnings- som utmatningsbeteende hos den interaktiva produkten eller tjänsten. Interaktivitetsattributen kan användas för att förfina designen och mejsla ut tydliga karaktärer.

De sju attribut som Lim m.fl. introducerar skiljer sig åt på dimensionerna

Samtidighet:

- Samtidig: Cirklarna på skärmen rör sig alla samtidigt när översta cirkeln klickas (relaterade egenskaper: tung, intetsägande, enkel, artificiell, vardaglig, osympatisk).
- Sekventiell: Cirklarna rör sig sekventiellt när översta cirkeln klickas (relaterade egenskaper: lätt, mustig, komplicerad, naturlig, exotisk, sympatisk).

Kontinuitet:

- Diskret: En liten cirkel rör sig med diskreta steg längs en cirkulär bana när knapparna trycks in med diskreta tryck (relaterade egenskaper: tung, intetsägande, hård, enkel, tydlig, artificiell, vardaglig, osympatisk, digital).
- Kontinuerlig: En liten cirkel rör sig kontinuerligt längs en cirkulär bana när ett dragreglage dras kontinuerligt (relaterade egenskaper: lätt, mustig, mjuk, komplicerad, tvetydig, naturlig, exotisk, sympatisk, analog).

Förutsägbarhet:

- Förutsägbar: Varje cirkel rör sig till närmaste position när den klickas (relaterade egenskaper: tung, intetsägande, hård, enkel, ytlig, tydlig, vardaglig).
- Oförutsägbar: Varje cirkel rör sig till en slumpvis vald position när den klickas (relaterade egenskaper: lätt, mustig, mjuk, komplicerad, djup, tvetydig, exotisk).

Rörelse:

- Små rörelser: När pekaren är nära cirklarna rör de sig med små rörelser (relaterade egenskaper: tung, intetsägande, hård, enkel, djup, artificiell).
- Stora rörelser: När pekaren är nära cirklarna rör de sig med stora rörelser (relaterade egenskaper: lätt, mustig, mjuk, komplicerad, ytlig, naturlig).

Hastighet:

- Långsam: När pekaren är nära en cirkel rör den sig sakta (relaterade egenskaper: tung, intetsägande, mjuk, djup, tvetydig, vardaglig, sympatisk, analog).
- Snabb: När pekaren är nära en cirkel rör den sig snabbt (relaterade egenskaper: lätt, mustig, hård, ytlig, tydlig, exotisk, osympatisk).

Exakthet:

- **Exakt:** En siffra visar exakt storlek på cirkeln när positionen på dragreglaget ändras (relaterade egenskaper: tung, hård, enkel, ytlig, tydlig, vardaglig, osympatisk, digital).
- **Ungfärlik:** Det finns ingen exakt siffra som visar cirkelns storlek när positionen på dragreglaget ändras (relaterade egenskaper: lätt, mjuk, komplicerad, djup, tvetydig, exotisk, sympatisk, analog).

Responsivitet:

- **Fördröjd respons:** flera klick på varje cirkel får den att röra sig (relaterade egenskaper: tung, intetsägande, mjuk, komplicerad, djup, tvetydig, exotisk, sympatisk, analog).
- **Omedelbar respons:** Ett enda klick får den att röra sig (relaterade egenskaper: lätt, mustig, hård, enkel, ytlig, tydlig, vardaglig, osympatisk, digital).

Dessa interaktivitetsattribut kan som sagts användas för att på ett detaljerat sätt utforska hur interaktiviteten med produkten eller tjänsten ska vara. Det är en god idé att skissa ned olika alternativ och väga fördelar mot nackdelar (skriva enkla plus-minus-listor). Ska designen till exempel byggas på diskreta steg eller kontinuerlig förändring, och vilka fördelar och nackdelar har de alternativen? De relaterade egenskaperna hjälper designern att fokusera på känsla, uttryck och identitet (Lim m.fl., 2011).

Datorprototyper

Till skillnad från pappersprototyper är datorprototyper detaljerade vad gäller det visuella och det interaktiva. Det gör det möjligt att testa designens utseende och känsla. För de uppgifter som prototypen byggs för går det också att testa effektivitet, genom att till exempel mäta hur lång tid det tar för användaren att lösa en uppgift, givet att den inkluderar realistiska data.

I bland talar man om integrerade prototyper. Då åsyftas sådana som dels är implementationsprototyper som testar hur det tilltänkta systemet ska konstrueras och implementeras, dels testar systemets roll och funktionalitet, samt dess utseende och känsla.

I bland frågar mina studenter mig hur mycket de ska implementera, varvid jag svarar att de ska bygga det som de kan bygga givet deras tidsbudget och fejka det som de inte kan bygga. Om de ändå har med lösningar som ingen kan bygga, är det fel på deras design. Det är också viktigt att inte bygga mer än nödvändigt för att kommunicera, specificera och testa designen. Det

programmeringsmiljö som slutprodukten. Det kan då göras i en GUI-byggare, alltså ett verktyg för att bygga grafiska användargränssnitt (GUI = *graphical user interface*). GUI-byggare erbjuder normalt en uppsättning färdiga gränssnittskomponenter som placeras ut, vilket självfallet gör det effektivt att bygga. Samtidigt finns det då en risk att man som gränssnittsdesigner lockas att använda just de färdiga komponenterna, trots att det skulle bli en bättre design med en specialbyggd komponent. Exempel på GUI-byggare är Interface Builder, Visual Studio Express och Eclipse. LiveCode är ett annat exempel på en användbar grafisk programmeringsmiljö med ett enkelt programmeringsspråk (som dock professionella programmerare kan uppleva en aning pratigt). Min högst personliga erfarenhet är att det åtminstone tar fyra gånger längre tid att bygga en evolutionär prototyp än en som ska kastas bort.

Ett sätt att fejka när man bygger gränssnittsprototyper är att hårdkoda alla variabler och sökresultat så att varje skärmbild i testscenariot alltid ser likadan ut, samtidigt att tända och släcka synligheten på gränssnittskomponenter, vilket gör att användaren upplever det som att saker händer i systemet trots att det endast händer något på ytan i gränssnittet. Faktum är ju att det inte finns något där bakom.

Ett annat sätt att fejka är att göra en Trollkarlen från Oz (Dahlbäck, Jönsson & Ahrenberg, 1993). Det kan till exempel vara så att det är svårt att snabbt sätta ihop ett program som kan tolka tal eller gester från användaren. Då kan prototypens respons på användarens handlingar styras av en trollkarl som ser användarens gester och lyssnar på vad denne säger och därefter styr den respons som systemet skulle ge. En trollkarl gör det möjligt att testa vilket beteende en AI (artificiell intelligens) borde ha, utan att spendera enorma resurser på att utveckla och träna den.

Om den detaljerade designen inte ska testas med användare kan en video-prototyp (i form av en *screencast*) på användargränssnittet också göras. En sådan linjär prototyp kan göras i en presentationsmjukvara som Keynote eller Powerpoint, där designern sedan bara spelar in en film av en kommenterad genomstegning av gränssnittet.

Powerpoint och Keynote eller liknande presentationsverktyg kan också användas för att göra interaktiva prototyper. Varje bild i presentationen blir då ett tillstånd i gränssnittet och länkar till andra bilder läggs på objekten på bilderna. När de klickas, hoppar alltså användaren till en annan bild i presentationen (Warfel, 2009; Kelly, 2007).

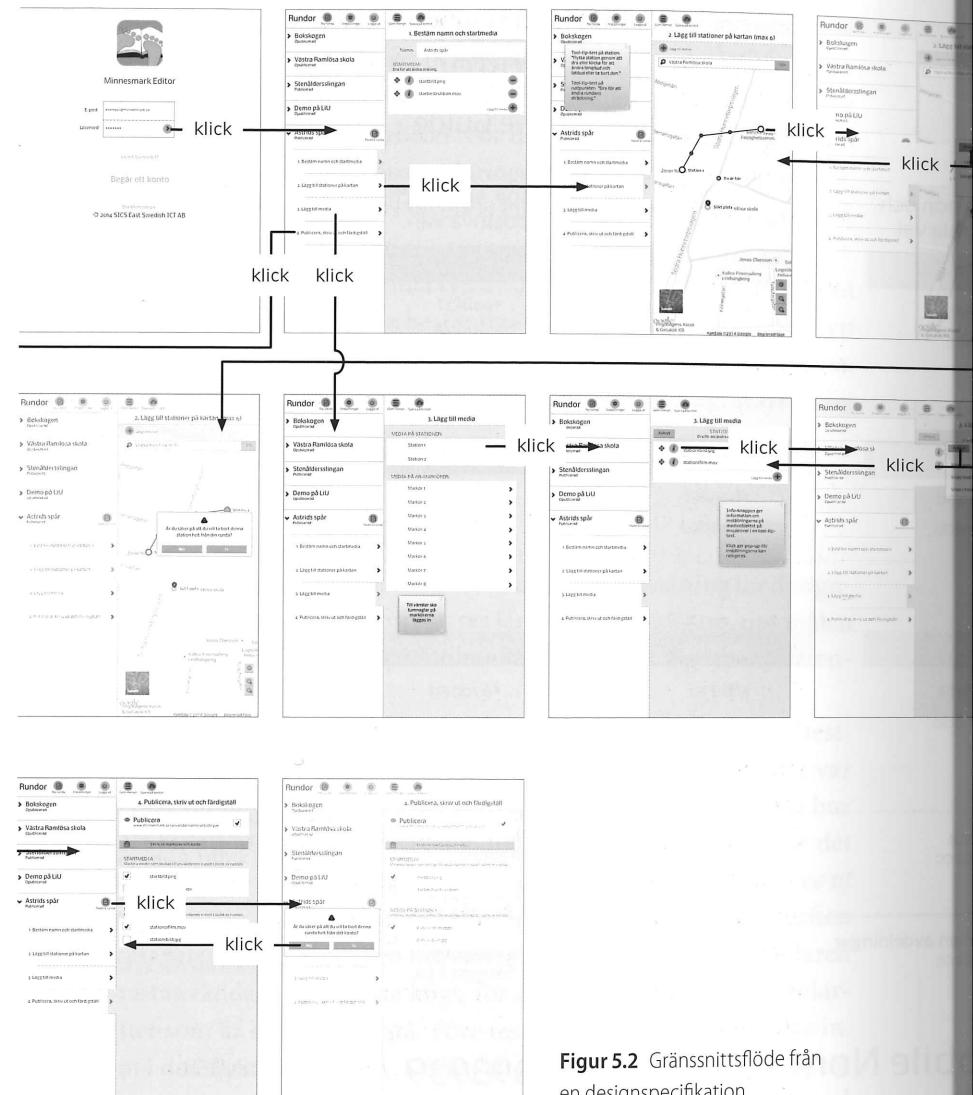
Det är också enkelt att göra animationer i presentationsprogram.

Det finns också verktyg som är specifikt gjorda för att prototypla användargränssnitt med hög detaljeringsgrad. Verktyg som är aktuella i skrivande stund inkluderar Axure RP, Flint, Fluid UI, Indigo Studio, Flash Catalyst och Expression Blend, tillsammans med SketchFlow. Många bygger också sina prototyper i HTML (Warfel, 2009; Ramsay & Buley, 2008). Det dyker hela tiden upp nya verktyg, så det här är en lista som snart är inaktuellt. Det bästa är att göra en sökning på webben med sökord såsom "user interface prototype tools".

Designspecifikationer

En annan möjlig leveransartikel från den detaljerade designfasen är pixelperfekta statiska skärmbilder. Med "pixelperfekta" menas att varje pixel i skärmbilderna ligger där de ska. Goodwin (2009) beskriver vad en designspecifikation för en framtida produkt eller tjänst brukar innehålla på designföretaget Cooper. Där används följande disposition:

- *Sammanfattning* med en översikt över produkten eller tjänsten, de viktigaste idéerna och ovanliga element i designen, samt en förklaring av värdena som designen ska bidra till.
- *Personor och kritiska krav* där de förklarar vad personor är, vilka personor denna design bygger på och vilka de viktigaste kraven är för varje persona.
- *Översikt över produkten eller tjänsten* där de unika idéerna i tjänsten lyfts fram, liksom UX- och brukskvaliteter, formfaktorer, samt vilka gränssnitt som riktar sig till vilka personor, och varför.
- *Interaktionsramverk* med översikt över fysiska interaktionskomponenter, de viktigaste skärmbildernas funktioner, övergripande element som återkommer på en hel uppsättning skärmbilder, navigation och relationer mellan vanliga element, samt dataobjekt som är relevanta för gränssnittet.
- *Scenarion* för varje gränssnitt beskrivna som flödesdiagram eller storyboard på gränssnittsnivå. Figur 5.2 är ett exempel på ett detaljerat gränssnittsförflytt (eng. wireflow) ur designspecifikation för Minnesmark-editorn som används för att sätta ihop datorförstärkta rundvandringar.
- *Skärmbilder och funktioner*, där både översikt och detaljer beskrivs för varje skärmbild eller funktion. Det inkluderar syfte, skärmbildens anatomi, innehåll för specifika komponenter, och exakta



Figur 5.2 Gränssnittsförflytt från designspecifikation.

- *Visuellt formspråk* beskriver kopplingar till varumärke och identitet, färgscheman, typsnitt, riktlinjer för fotografier och bildspråk, iconer, och layout. Även här bör exakta mått på storlekar och avstånd angis. Det är viktigt att visa med exempel hur det visuella formspråket ska se ut. Figur 5.3 är ett exempel från Minnesmark-projektet på formspråket i editorn.

Minnesmark
Webbeditor

Design Style Guide

Navigation och neutrala val

Ta bort och negativa val	Skapa, spara och positiva val
Aktiv	Du är här Fyllning #ffffff
#1dc5f5 #179fc5	Ändringsdatum: 2014-03-15 Författare: Mattias Arvola
Aktiv	Sökt plats Ram 4 px #ff8e03 Fyllning #ffffff
#ff8e03 #cc7202	

Vald	Vald	Vald
Aktiv	Aktiv	Aktiv
#148bad #1abodb	#b26201 #e48002	#3c7221 #57a530

Inaktiv	Inaktiv	Inaktiv
Aktiv	Sökt plats Ram 4 px #ff8e03 Fyllning #ffffff	Du är här Fyllning #ffffff
#7edef9 #7bc8dd	#fffb70 #e1aeef	#a7db8d #98bd85

Linje i lista 1 px #cccccc

Linje mellan avdelningar 2 px #cccccc

i Runda knappar (ingen skugglinje):
30 px i diameter

Klar Rektangulära knappar:
24 px höjd. Skugga 2 px.

Kvadratiska knappar:
30 px sida. Skugga 2 px.

Tool-tip-text
Nobile Normal 400 11 pt #393939.
Bakgrund #ffffff. Ram 1 px #cccccc

Ikoner från Typicons.com har 75% opacitet.
Chevron i vald flik har 45% opacitet.

H1 Nobile Normal 400 24 pt #393939

H2 Nobile Normal 400 18 pt #393939

H3 NOBILE NORMAL 400 14 PT VERSALER #393939

Brödtext Nobile Normal 400 13 pt #222222

Liten brödtext Nobile Normal 400 11 pt #393939

Ettikett Nobile Normal 400 9 pt #222222

Knapp- och karttext Nobile Bold 200 11 pt #222222

och professionell och inte använda pretentiösa formuleringar. Utvecklingsgruppen som ska realisera designen och implementera den ska i detta skede redan vara bekant med alla idéer i den, så att de redan har kommit i gång att bygga parallellt med designarbetet. För dem ska designspecifikationen inte innehålla några obehagliga överraskningar som kulla kastar deras arbete. För att det ska vara möjligt krävs givetvis en god kommunikation mellan designer och utvecklare.

I stället för att man skriver denna typ av designspecificationsdokument kan också den slutgiltiga designen bestämmas i en interaktiv datorprototyp av den sort som beskrev ovan. Om designgruppen arbetar nära utvecklarna, är det kanske att föredra, men om man ska göra en överlämning, är en tydlig specifikation bättre.

Detaljeringsfasens värderingar

I en avslutande – även kallad *summativ* – användbarhetstestning tar designgruppen reda på om produkten förbättras jämfört med tidigare, om målen har uppnåtts, och/eller hur ens produkt eller tjänst står sig i konkurrensen med andra.

Precis som i den formativa testningen i bearbetningsfasen måste testuppgifter förberedas, men här är det ännu viktigare att det är tydligt var de börjar och var de slutar, så att det blir möjligt att till exempel mäta hur lång tid det tar att utföra uppgiften (eng. *time on task*). Förutom tid är det vanligt att mäta hur väl användaren lyckas med uppgiften (eng. *degree of success*) och hur stor andel av testanvändarna som lyckades med uppgiften (eng. *success rate*). Precis som i den formativa utvärderingen ber testledaren normalt testanvändarna att tänka högt, för att identifiera vad i användargränssnittet som är svårt att förstå. Före testet samlas bakgrundsdata in, precis som i den formativa testen. I delen efter testningen är det emellertid också vanligt att utvärdera hur nöjd användaren är med en standardiserad enkät. Ett exempel på en sådan enkät som är både vanlig, pålitlig och enkel att använda är SUS (*system usability scale*), som ger ett sammantaget värde på upplevd användbarhet utifrån tio frågor. Tullis och Albert (2013) ger en mycket bra genomgång av SUS, men även andra enkäter. Även Sauro (2011) har en bra beskrivning av hur man utvärderar med SUS. Såväl Tullis och Albert (2013) som Sauro och Lewis (2012) ger ordentliga genomgångar av avslutande användbarhetstestning och hur man dessutom rapporterar dem med korrekt statistik.

kort som representerar dennes upplevelse av systemet och sedan motivera varför just dessa kort valdes. Resultatet av en test med reaktionskort kan sedan ställas mot de avsedda UX- och brukskvaliteterna som satts upp för projektet under konceptfasen och reviderats under bearbetningsfasen.

I ett senare skede när designgruppen har en fullt fungerande evolutionär prototyp eller beta-version går det att göra fälttest (Sharp, Rogers & Preece, 2011). Då testas hur systemet verkligen fungerar ute i fält i verkligheten snarare än i en experimentell situation. Exempel på det är när ett nytt trafikledningssystem körs parallellt med det gamla på riktiga data, för att utvärdera hur det fungerar. Det är ett säkerhetskritiskt system, så därför kan inte det gamla systemet tas ur drift innan man är helt säker på att det nya systemet fungerar felfritt.

Efter det att värderingen i detaljeringsfasen har omsatts i revideringar av designspecifikationen eller av den slutgiltiga gränssnittsprototypen är det huvudsakliga designarbetet avklarat. Men det betyder inte att det är helt slut. Därefter följer överlämning och uppföljning, och en hel del arbete sker också i dessa följande faser, även om det inte är lika intensivt.

Överlämning

När designen till sist har godkänts av de olika intressenterna och uppfyller kraven sker en överlämning till implementation och en uppföljning av hur den fortskrider. Vanligen måste dock designförändringar göras även under implementationens gång. För en designer som sitter på samma kontor som utvecklarna kan det innebära en lång period av fem-minuters-konsultande, där en utvecklare kommer in på ens kontor och frågar hur en gränssnittsfråga bör lösas, och för en konsult ska man räkna med tid för regelbundna designmöten genom hela utvecklingsprojektet. Samtidigt startar ofta en planering av nästa version av en produkt och nya designprojekt. Under utvecklingens gång kan nya saker hända i omvärlden som gör att designen måste revideras. Det kan till exempel dyka upp en ny konkurrent på marknaden, eller så kommer ett operativsystem i en ny version. Sådana saker händer hela tiden, och det måste man som beställare och som projektledare ta med i beräkningarna.

En designspecifikation eller interaktiv prototyp kan omöjligt täcka upp allt som behöver göras under utvecklingen och frågor kommer att dyka upp. Regelbundna granskningar och utvärderingar löpande under utvecklingens gång är därför nödvändiga. I en modern agil systemutvecklingsprocess är de dessutom en förutsättning.

Innehållsproduktionen är ofta något som förbises men många gånger står och faller en produkt eller tjänst med kvaliteten på dess innehåll. Det gäller särskilt medieintensiva produkter som spel och interaktiva media.

Uppföljning och reflektion

Både den tidiga designprocessen som denna bok har som sitt primära fokus på och den utvecklingsprocess som tar vid där denna bok nu slutar, är lärandeprocesser. Allteftersom projektet fortgår kan designer, utvecklare, kunder och användare ställa mer och mer specifika frågor, och svaret på de frågorna är en mer och mer specificerad design.

Genom att design är en lärandeprocess, är det också viktigt att avsluta varje projekt med en reflektion och diskussion kring vilka lärdomar man gjort. En strukturerad debriefing kan användas för att reflektionen inte ska bli ytlig. Nedanstående frågor kan med fördel gås igenom då erfarenheterna från ett projekt ska tas tillvara (Gibbs, 1988):

- *Beskrivning:* Beskriv enskilt vad som hänt, utan värderingar och slutsatser.
- *Tankar och känslor:* Beskriv och diskutera enskilt eller i par vilka reaktioner, tankar och känslor du upplevde.
- *Utvärdering:* Värdera i par eller enskilt vad som var bra och vad hade kunnat vara bättre med erfarenheten.
- *Analys:* Diskutera i mindre grupper hur kan du förstå situationen och händelserna, och vad som egentligen som hänt. Jämför olika personers erfarenheter.
- *Generella slutsatser:* Diskutera i större grupper vad du i allmänhet kan ta med dig från de erfarenheter och den analys du gjort.
- *Specifika slutsatser:* Analysera enskilt och presentera i en större grupp vad du kan ta med från dina egna specifika, unika, och personliga situation eller arbetssätt.

Erfarenheterna att ta med sig till framtida projekt blir mer konkreta och lättare att tillämpa, om reflektionen börjar med att beskriva vad som hänt och hur man erfor det, innan värdering och analys görs och innan några slutsatser dras.

Avslutande ord