Gränssnitt för små och mobila klienter

Institutionen
Handelshögskolan vid
Göteborgs Universitet

Av: David Johanson
Handledare: Sandra Magnusson
Magisteruppsats 20p
HT 2001

Abstrakt

Avhandlingen visar på vad en konstruktör av användargränssnitt bör ha i åtanke vid konstruktion av ett gränssnitt för små och mobila klienter. Den ger även en bild av vad ett användargränssnitt är, samt förklarar inte bara *hur* något bör göras utan även *varför*. Detta åstadkoms genom att utgångspunkten för avhandlingen är användaren och de mentala processer och förmågor som föreligger ett visst handlande från denne. Litteraturstudier inom ämnet psykologi ger detta. Dessa litteraturstudier har i sin tur resulterat i ett antal olika gränssnitts rekommendationer som kan användas till både analys av befintliga gränssnitt såsom vid konstruktion. Avhandlingen har även resulterat i ett program för små och mobila klienter. Programmets syfte är att illustrera ett alternativt gränssnitt samt att visa hur rekommendationerna kan användas i praktiken, det vill säga som ett verktyg att mäta rekommendationerna mot. Det slutliga resultatet i denna avhandling består i att skillnaderna mellan gränssnitt för små och mobila klienter och stationära datorer inte är särskilt stora men att de främst skiljer sig åt på två punkter. *In- och utmatningsverktygens storlek* och *den miljö som de används i*. Detta är något som kräver särskild uppmärksamhet vid gränssnitts konstruktion.

Tack

Ett stort tack till samtliga handledare som jag har förbrukat under mitt arbete på denna uppsats, för stöd och uppmuntran; Jens Bergqvist, Birgitta Ahlbom och slutligen Sandra Magnusson. Jag vill även rikta ett särskilt tack till de *försökspersoner* som har hjälpt mig med att testa applikationen och som även har gett förslag till förbättringar; Daniel Martinison och Louice Johanson.

Innehållsförteckning

1 Introduktion 6
1.1 Bakgrund 6
1.2 Definition av gränssnitt 7
1.3 Problemställning 8
1.4 Syfte och mål 9
1.5 Målgrupp 9
1.6 Tidigare forskning 10
1.7 Översikt 10
2 Metod 11
2.1 Kvalitativa metoder 11
2.2 Tillvägagångssätt 16
3 Teoretisk grund 19
3.1 Likheter mellan Användargränssnitt och Språk 19
3.2 Användarvänlighet 22
3.3 Användarens mentala förmågor 24
4 Gränssnittsrekommendationer 39
4.1 Definition: små och mobila klienter 39
4.2 Stöd för arbetsminnet 40
4.3 Stöd för långtidsminnet 40
4.4 Fokusering av uppmärksamhet 41
4.5 Avlasta tänkandet 43
4.6 Underlätta för perceptionen 44
5 Applikationen 47
5.1 Beskrivning av applikationen 47
5.2 Funktionalitet 47
6 Analys av applikation 53
6.1 Arbetsminnet 53
6.2 Långtidsminnet 53
6.3 Uppmärksamhet 54
6.4 Tänkandet 54

6.5 Mentala Modeller 55
6.6 Perception 56
7 Slutsats 58
8 Referenser 60
8.1 Internetlänkar 61

1 Introduktion

I denna introducerande del introduceras läsaren i problematiken som ligger till grund för denna uppsats. En definition på gränssnitt ges, uppsatsens syfte och mål klarläggs samt en målgrupp för avhandlingen anges.

1.1 Bakgrund

Att använda datorer kan vara mycket frustrerande, särskilt när de inte fungerar på ett sätt som användaren vill eller har tänkt sig. Datorer har dock relativt stora skärmar och behändiga inmatningsverktyg jämfört med små mobiltelefoner och handdatorer. Små bärbara mobila klienter har dessutom en extra distraktionsfaktor som inte existerar i samma utsträckning när vid arbete vid en vanlig stationär dator - det är den runtomliggande ”bullriga” och distraherande miljön eller omgivningen. Dessutom har dessa små klienter ett mer frustrerande och *mindre* in- och utmatningssystem. Man kan fundera över om det är möjligt att få se liknande scener med frustrerade handdatoranvändare som har observerats med vanliga datoranvändare. Enligt en undersökning utförd av den brittiska tidningen Computeractive (26 februari, 1998) skriker hälften av alla användare åt sin dator när den inte gör som de vill. Var fjärde användare låter sin ilska gå ut över människor i deras närhet. Hela sex procent har så svårt att kontrollera sina känslor att de utsätter sin dator för misshandel. Det är dock inte det värsta en dator kan råka ut för; även mord förekommer. En man i USA arresterades i juli 1997 för att ha skjutit sin dator. Han vägrade att kommentera händelsen, men enligt frun var orsaken förmodligen att datorn inte startade tillräckligt snabbt. En av de poliser som tog hand om mannen sade att *han kom ut ur sitt hus med ett leende på läpparna*. Relationen mellan människa och dator kan vara nog så problemfylld, *vad kan göras för att underlätta denna relation*? Eller kommer vi att få bevittna liknande scener med människor som frustrerat skäller ut och vandaliserar sin stackars Palm eller PocketPC maskin?

Kukulska-Hulme (1999) framhöll dessutom när hon diskuterade språkets användning i användargränssnitt att vi människor är vana vid att kunna kommunicera enkelt, främst med hjälp av vårt språk. Även dator-experter förundras ofta över de förklaringar som ges i så kallade hjälp-avsnitt. Hur de blir förvirrade över innebörden i de ord som förkommer i menyer och knappar samt hindrade vid informations åtkomst genom att behöva använda termer som inte på ett klart sätt uttrycker deras behov. Kukulska-Hulme (1999) fortsätter att skriva att de ord som används på dator-skärmen kan skapa en barriär för kommunikation, och ändå är de användare som vänder sig till hjälp-funktioner och dokumentation ofta besvikna. Hon skriver vidare att Lynne Truss (1996), en kolumnist för Times, skriver, efter ännu en fruktlös session framför datorn:

> *Jag har till och med slutat läsa dom där små 'readme'-filerna eftersom det är tråkigt men sant men: jag har aldrig ännu öppnat en sådan vars innehåll jag kunnat förstå*.

Kukulska-Hulme (1999) skriver vidare att ett frustrerat utrop av att *jag förstår inte menyerna på skärmen* ofta leder till att användarna begagnar sig av ett *pröva och se vad som händer*-beteende, vilket kan få ödesdigra konsekvenser.

Gränssnittsdesign är inte något nytt. Datorer med grafiska gränssnitt har funnits ett tag och det börjar finnas en allmän vana att både använda och designa gränssnitt för dem. Små mobiltelefoner och handhållna datorer, som den här uppsatsen kommer att fokusera på, blir allt mer avancerade, kraftfulla och får fler och fler funktioner och tillämpningsområden. Men det får helt enkelt inte plats lika mycket information på skärmarna, det finns helt enkelt inte lika många bildpunkter eller bildpunkter som på en vanlig datorskärm. Hade det funnits lika många bildpunkter hade den information som visats på skärmen ändå varit oläslig utan förstoringsglas på grund av skärmens litenhet till ytan. Är det då möjligt att göra ett bra användargränssnitt för dessa små skärmar?

1.2 Definition av gränssnitt

Ett gränssnitt är det medium genom vilket användaren och datorn kommunicerar med varandra. Det kan liknas vad att användargränssnittet är en tolk vid viktiga förhandlingar i till exempel FN. De personer som förhandlar kan inte förstå varandra, men tolken kan förstå och vidareförmedla vad respektive part säger och menar till den andre. Den perfekta tolken (gränssnittet) skulle därmed kunna exakt överföra varje liten nyans och skiftning i det språk som den ene parten talar till den andre partens språk, och tvärtom. Men detta är tyvärr, som bekant, omöjligt i dagsläget med dagens språk. Läget är liknande för användargränssnitt. Användargränssnittet har samma problem som tolken; att på ett exakt sätt förmedla användarens uttrycks- och begreppstermer till ett språk som datorn förstår och vice versa.

En annan bra och lite varierad syn är att:

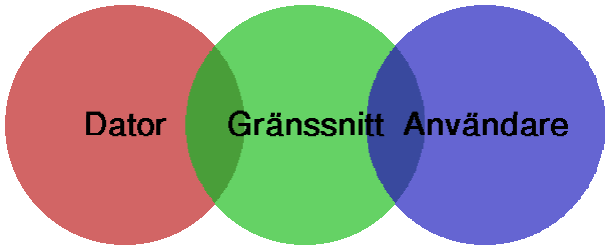
> *”Ett gränssnitt delger medlet för kommunikation mellan två entiteter. T ex skulle gränssnittet mellan en förare och en bil vara ratten, fartmätaren, gaspedalen och andra reglage och mätare på instrumentpanelen. Detta gränssnitt tillåter föraren att överföra sina önskemål till något som bilen kan förstå. Föraren ger input: genom att röra på gaspedalen, bromsa eller svänga på ratten. Bilen svarar med output: vilken hastighet som bilen håller, hur mycket bensin som är kvar eller genom att ändra färdriktning. Föraren använder bilens output, i tillägg till sin egen motivation, för att bestämma vilken input som skall ges till bilen. Liknande gränssnitt finns mellan en användare och ett program. Specifikt kan man säga att ett människa-datorgränssnitt är den del av ett datorsystem med vilket en användare interagerar för att åstadkomma någon speciell uppgift. Det inkluderar både de metoder som krävs för att tillåta en användare att utföra olika handlingar och även de medel genom vilka datorn rapporterar information till användaren*.”
> (Koelle, 1996)

Så länge som det är fråga om att användaren skall styra ett mekaniskt objekt, som en bil, är gränssnittets funktion ofta mycket mer lättfattlig och förståelig. Men när datorteknik kommer med i bilden blir objekten gärna mer komplexa och funktionaliteten mer

svårförståelig. Det går att argumentera för att det är fråga om en vanesak. En människa som har växt upp med datorteknik, som tidigare generationer har växt upp med mekanik, kan lättare skapa sig någon form av mental modell av hur objekten fungerar som därmed underlättar förståelsen och användningen.

I och med att objekten som skall styras av användaren blir alltmer komplexa och svårfattliga ökar användargränssnittets betydelse allt mer. Det finns ingen som helst mening med att lägga till en massa ny funktionalitet om den är svårbegriplig och svåranvänd, resultatet blir att den inte alls kommer att användas.

Det intressanta med små, mobila och handhållna klienter är att till skillnad från stora och stationära datorer har funktionaliteten blivit mindre. Hittills har ofta de små handhållna klienterna haft mindre datorkraft (något som håller på att ändras) och därmed mindre tillämpningsområden och mjukvarumässig lägre komplexitetsgrad. Men användningsområdena har även dessa varierat, de används helt enkelt inte till samma uppgifter som stora stationära datorer. En författare som skall skiva sin senaste bästsäljare sätter sig inte ner och pletar med en liten handhållen dator bara för att det är möjligt att sitta i parken och skriva. Utan han sitter hellre med sin ordbehandlare på sin PC, en bärbar portföljdator skulle vara ett bättre alternativ än en liten Palm eller PocketPC om det finns en önskan att vara mobil. Det är med andra ord inte bara komplexiteten som har ändrats utan även tillämpningsområdena.



Gränssnittet har begrepp från både användare och dator, det har snitt från båda. Det kan tala bådas språk.

Gränssnittet skall vara en delmängd av båda de delar eller entiteter som det delger kommunikation mellan. För att kunna *tolka* mellan dessa två delar måste det ha och kunna behärska relevanta begrepp från båda. Resultatet blir att två i sig skilda *mängder* (för att använda mängdläran) kan komm-unicera som genom en tolk, en tolk som behärskar bådas *språk* och begreppsvärld. Gränssnittet blir en egen mängd mellan användare och människa som har ett snitt med båda.

1.3 Problemställning

Problemet är inte att skapa ett gränssnitt som kan tolka och förstå en dators tankebanor och uttryck. Inte heller är det ett större problem att skapa ett gränssnitt som klarar att interpretera och klarlägga användarens uttryckssätt, även om denna uppgift är betydligt mer komplex och varierande. Det verkliga problemet ligger faktiskt i hur gränssnittet skall förmedla respektives signaler till den andre; hur gränssnittet skall klara att överföra sina begreppstermer till mottagarens begreppstermer, datorn och människan. Problemet med att göra användbara gränssnitt ligger inte i kommunikationen mellan dator och gränssnitt, den kommunikationen löses genom olika programmeringstekniker. Datorn är trots allt en maskin byggd av människor för att användas av människor. Datorns syfte

med sina själva existens är att avhjälpa människan, användaren, att utföra någon specifik uppgift; det var inte människan som konstruerades för att passa ihop med och komplettera datorn. I begreppstermen dator menas inte i detta fall endast stationära PC-datorer utan även små, mobila och handhållna klienter som handdatorer (Palm och PocketPC) men även mobiltelefoner.

De frågor som detta arbete kommer att fokusera på är följande:

- Vilka faktorer gör att användaren uppfattar ett datorgränssnitt som användarvänligt, funktionellt och användbart?
- Hur kan dessa faktorer tillämpas på ett *litet* användargränssnitt för små och mobila klienter?

Vilka egenskaper utmärker ett *bra* gränssnitt? Gränssnitt som förmedlar information mellan dator och människa på ett sätt så att människan, det vill säga användaren, kan arbeta på ett effektivt och obehindrat sätt utan att denne upplever att användandet av applikationen ifråga är klumpigt och omständligt. I alla fall inte klumpigare och mer omständligt än hur det är att arbeta med en vanlig motsvarande applikation på en vanlig dator.

1.4 Syfte och mål

Syftet med denna uppsats är att ur ett användarperspektiv ta reda på hur ett användarvänligt och funktionellt gränssnitt byggs upp. Särskilt med tanke på de särskilda problem som små och mobila klienter har. Jag ville även ytterligare fördjupa mitt kunnande om det mänskliga psyket och tänkandet.

Uppsatsen har tre delmål.

1. Göra en mental profil av användaren.
2. Utfärda ett antal gränssnittsrekommendationer baserade på den psykologiska profilen. Dessa rekommendationer skall sedan även kunna användas vid - konstruktion och gränssnittsanalys
3. Skapa en exempelapplikation med ett gränssnitt som skall användas för att analyseras av de gränssnittsrekommendationer som blivit utfärdade.

Uppsatsens syfte är, med andra ord, konstruktion och analys av en applikation för små och mobila klienter, men också att ge en större insikt i det mänskliga tänkandet och hur denna insikt kan tillämpas på användargränssnitt.

1.5 Målgrupp

Den här uppsatsen riktar sig till personer med visst datorkunnande men med intresse att analysera eller konstruera gränssnitt för små och mobila klienter. Några direkta förkunskaper inom ämnet människa-datorinteraktion är däremot inte nödvändigt eftersom utgångspunkten blir från själva människan. Men även eftersom de flesta begrepp och termer förklaras. Det kommer inte bara fram att det är bra att göra på det

ena eller andra sättet utan även varför det är så. Vilket betyder att denna förkunskap inte är nödvändig.

1.6 Tidigare forskning

Exempel på redan utförd forskning och framtagna experimentella gränssnitt för små skärmar är Staffan Björks (2000) avhandling betitlad *Flip Zooming – the development of an information visualization technique*, och de olika applikationslösningar som blivit framtagna i samband med den.

1.7 Översikt

Nedan följer en kort beskrivning av varje kapitel och dess innehåll:

1. **Inledning.** Ger en introduktion till problematiken och vissa kortare inledande definitioner och beskrivningar. Det huvudsakliga syftet är att ge en försmak, men även mersmak.
2. **Metod.** Ger en allmän beskrivning av de metoder som har använts och hur dessa har använts, samt varför just dessa har valts framför andra metoder och tillvägagångssätt.
3. **Teoretisk grund.** Ger den nödvändiga teoretiska bakgrund som det vidare arbetet bygger på, bland annat en utförlig beskrivning av användarens mentala förmågor. Innehåller även en mer grundlig definition av gränssnitt samt hur andra gränssnittsexperter definierar användarvänlighet.
4. **Gränssnittsrekommendationer.** Innehåller diverse olika rekommendationer som i första hand bygger på den i föregående kapitel framtagna beskrivningen av användarens mentala förmågor.
5. **Applikationen.** Tillhandahåller en beskrivning av den framtagna applikationen, *File Quest*, då främst dess olika funktioner och dess användargränssnitt.
6. **Analys av applikationen.** I detta kapitel utförs en grundlig analys av *File Quest* med utgångspunkt från de i föregående kapitel framtagna gränssnittsrekommendationerna.
7. **Slutsats.** En kort slutdiskussion där kärnan av vad som tidigare har framkommit läggs fram.

2 Metod

Vad en person än vill företa sig krävs ett tillvägagångssätt eller en metod för att det skall vara möjligt att nå det uppsatta målet. Exempelvis om en person skulle gå vilse i en skog (utan mobiltelefon märk väl). Men om hon inte har tillgång till eller kan komma på en tillförlitlig metod för att hitta ut ur skogen kan detta scenario komma att bli ödesdigert. På ett liknande sätt måste en vetenskaplig uppsats ha en bra metod vid genomförandet för att det önskade resultatet skall nås på ett effektivt sätt. Utan en bra, konsekvent och för uppgiften lämplig metod kan målet bli svårt och tidskrävande att nå. Det kan även bli så att resultatet vid närmare granskning inte håller. De metoder och det tillvägagångssätt som används vid utförandet av en vetenskaplig uppsats måste noga dokumenteras tillsammans med de källförteckningar som använts. Detta måste göras för att det skall vara möjligt för en oberoende person att kontrollera att de slutsatser som uppsatsen framhåller är tillräkneliga med bakgrund mot de använda metoderna och källorna. Det räcker inte att enbart räkna upp vilka metoder som använts och i vilken ordning, utan en beskrivning av metoderna i sig och vad de innebär måste även finns med. Anledningen är att det skall stå klart att författaren av uppsatsen har förstått hur metoderna skall tillämpas och hur de bör genomföras.

2.1 Kvalitativa metoder

Inledningsvis antar uppsatsen ett kvalitativt perspektiv eftersom utgångspunkten ligger i hur människan, användaren, uppfattar och tolkar den omgivande verkligheten. Detta leder även till att uppsatsen antar en subjektiv utgångspunkt, det är inte fråga om objektivt mätandet av en företeelse eller verklighet. Intresset riktas istället mot hur individen tolkar och utformar omvärlden i relation till de redan erhållna erfarenheter och kunskaper som denne innehar. (Backman, 1998)

Enligt Backman (1998) karakteriseras de kvalitativa delarna, av en rapport, av att de berör skeenden, förlopp eller processer snarare än produkter och resultat. Människan, ofta i kommunikation med andra människor, är det instrument som den kvalitativa rapporten är byggd med. Den är inte alltid enbart fråga om objektivt iakttagande från den studerandes sida utan det kan även vara så att det sker en interaktion mellan den person som utför studien och den eller de personer som utgör studien.

Det kvalitativa rapport förfarandet börjar med empirin, insamlandet av data, varefter, eller ofta samtidigt, hypoteser eller teorier börjar utformas. Detta kallas *induktion*. Motsatsen kallas *deduktion* där tyngdpunkten initialt läggs mer på de begreppsliga formuleringarna av teorier, frågeställningar och hypoteser, antaganden. (Backman, 1998)

Kvalitativa forskningsmetoder är lämpliga för att ge en djupare insikt än den mer fragmenterade insikt som erhålls vid kvantitativa metoder, men även för att skapa en förståelse för helhetsbilden. Kvantitativa metoder som finns att tillgå har en mera direkt mätande karaktär, till exempel genom olika statistiska analyser av observerat material. Detta leder även till att de kvantitativa metoderna antar ett mera objektivt synsätt.

Kvalitativa metoder använder verbala analysmetoder medan *kvantitativ*a metoder mer använder statistiska förfaringssätt för bearbetning och analys. (Patel, 1994)

2.1.1 Den kvalitativa forskningsprocessen

Backman (1998) framhåller vidare att den kvalitativa forskningsprocessen är mycket flexibel och innehar stort utrymme för variationer, den är därmed inte särskilt standardiserad och sekventiell, ibland samkörs vissa moment i stark interaktion mellan varandra. Det betyder att ett moment i den kvalitativa forskningsprocessen inte måste vara avklarad för att nästa del skall kunna börja utföras utan vissa delarn kan vid behov utföras samtidigt.

Enligt Backman (1998) kan den kvalitativa forskningsprocessen beskrivas i följande steg:

1. **Fråga.** Arbetet inleds med en fråga eller en frågeställning, de handlar ofta om *hur*- och *varför*-frågor.
2. **Litteraturgranskning.** Den här delens syfte är bland annat att klarlägga vilken kunskap som redan finns inom det område som berör den ställda frågan, belysa problemet, hjälpa vid problemformulering etc.
3. **Val av analysenhet.** Avgränsning, val av fallstudie och infalls- och angreppsvinkel.
4. **Problem-/frågeformulering.** Den här delen av processen sker ofta simultant med insamling av data, det vill säga litteratursamling. De första problemformuleringarna är ofta diffusa och generella men under arbetets gång preciseras dessa mer och mer.
5. **Observation.** Den kvalitativa observationsmetoden utgörs inte enbart av en objektiv observation av verkligheten utan den är beroende av observatören som ett tolkande objekt. Leder fram till en hypotes, ett antagande, om verkligheten.
6. **Analys.** Sker ofta kontinuerligt under datainsamlingsmomentet. Strävar efter att avge en helhetsbild och ibland dess viktiga underliggande orsaksmekanismer.
7. **Tolkning.** Sker simultant med *analys* och *observationsmomenten*. Skall ge mening och innebörd åt de gjorda observationerna. *Tolkningsfasen* kräver därför kunskap, sensitivitet, insikt och ibland även en gnutta intuition från observatören. Strävar efter att ge en holistisk beskrivning av den observerade verkligheten.
8. **Rapportering.** Den kvalitativa rapporten har ingen specifik mall för sitt utförande, till skillnad från den traditionella rapporten som kan skrivas i sin helhet i efterhand. Den kvalitativa rapportören bör däremot ha rapporten och dess utförande i åtanke under hela forskningsprocessen. Viktiga aspekter som framhålls är dock att val av *fokus* – rapportens tema eller inriktning – *målgrupp* – hur insatta de personer som ingår i målgruppen är i det ämne som rapporten behandlar är direkt avgörande för dess utformning – *disposition* – rapportens

struktur – *revision* – en granskning av rapporten av medlemmar av målgruppen vilket kan ge ytterligare information som bör vara med i den slutliga rapporten.

2.1.2 Den traditionella forskningsprocessen

Några detaljer som skiljer traditionell forskningsprocess åt från den kvalitativa är att i den traditionella processen sker inte några moment simultant i samma utsträckning som i den kvalitativa. I den traditionella forskningsprocessen arbetas det fram en hypotes *innan* observationsfasen, en hypotes som sedan prövas mot verkligheten på ett strikt objektivt sätt. Varefter en analys görs på resultatet. Den traditionella forskningsprocessen antar därmed ett *deduktivt* angreppssätt. (Backman, 1998)

2.1.3 Prototyping

Prototyping innebär att en provversion utvecklas först innan den skarpa versionen släpps. Den utvecklade prototypen måste därmed inte vara identisk med den färdiga produkten men den bör dock vara så lik som möjligt. Prototyping är en iterativ process. Det innebär att vissa delar av utvecklingsprocessen kommer att upprepas tills det att prototypen uppfyller vissa krav varefter den skarpa versionen kan släppas. Prototyping ger användaren möjlighet att påverka utvecklingen redan på ett tidigt stadium och ge kommentarer och synpunkter på hur väl produkten stämmer överens med de önskemål som denne har. Felaktigheter och oönskade element kan därför tidigt upptäckas och sållas bort. (Andersen, 1994)

Den utvecklingsprocess som tillämpas vi prototyping kan beskrivas i fem antal enkla steg:

1. Identifiera centrala behov
2. Utveckla en första prototyp
3. Demonstrera och diskutera sätt att förbättra applikationen
4. Realisera förbättringar
5. Täcker prototypen behoven?

Nedan beskrivs mera ingående hur Andersen (1994) beskriver de fem stegen

1. **Identifiera centrala behov**
   Det här steget innebär att de centrala behoven som användaren har identifieras. Vilken typ av applikation skall byggas? Vilka funktioner behöver implementeras? Vad vill användaren ha ut från applikationen? Den första prototypen bör klara handha cirka 60% av de funktioner som användaren har satt upp på sin önskelista därför att om den inte liknar det som användaren har önskat minskar entusiasmen att diskutera prototypen.
2. **Utveckla en första prototyp**
   Det är av största vikt att den första prototyp som tas fram innehåller de viktigaste delarna för att användarens intresse skall stimuleras och en diskussion om

funktionalitet kan komman igång. Den första prototypen bör även komma snabbt för att användarens motivation skall hållas uppe.

3. **Demonstrera och diskutera sätt att förbättra applikationen**
   Syftet med den här fasen är att få fram ny eller reviderade krav på programmet. Det sker genom att några av de framtida användarna får, i lugn och ro, använda, observera, experimentera med och kritisera prototypen. Utvecklaren får inte personligen ta illa upp och försvara prototypen om den skulle bli kritiserad utan syftet är att komma på idéer för förbättringar.
4. **Realisera förbättringar**
   I det här steget realiserar utvecklaren de förbättringar som har framkommit i föregående fas och bygger en ny prototyp.
5. **Täcker prototypen behoven?**
   I det här steget handlar det om att analysera om den prototyp som senast har framtagits täcker de behov som den färdiga produkten avsetts kunna täcka**.** Om så inte är fallet krävs ytterligare en prototyp tills dess att användarna är nöjda. Därav en iterativ process.

2.1.4 Metoder för att utvärdera användargränssnitt

Enligt Holyer (1993) är *utvärdering* en övergripande term som innefattar ett brett spektrum av olika aktiviteter med olika syften. Exempelvis kan utvärderingen ifråga vara *formativ* eller *summativ*. Den formativa utvärderingen utförs under själva utvecklingsfasen och kan därmed påverka applikationens utformning, därav namnet. Medan den summativa utvärderingen används när programmen är färdiga, exempelvis för att avgöra vilket office-paket som skall införkaffas. Ett begrepp som användarvänlighet få olika innebörd beroende på hur applikationen används. Om den inte används ofta blir faktorer som intuitivitet och tydlighet viktiga medan om användningen är mera frekvent blir faktorer som produktivitet och effektivitet viktigare.

På grund av att de personer som har utvecklat metoder för utvärdering av användargränssnitt har olika bakgrund, datorvetare, psykologer, lingvister med flera, finns det ett antal olika metoder som bygger på olika principer. (Holyer, 1993) Det är fråga om en slags *funktionell fixering*, där de överför sin befintliga kunskap på ett nytt område. (Allwood, 1991)

Det går inte att påstå att en metod för användargränssnittsutvärdering är bättre än en annan. Så gott som alla är ytterst krävande vad gäller tid, resurser och kunskap. Resultaten är även ofta svåra att överföra till konkreta gränssnittsrekommendationer. De olika metoderna täcker olika delar. Det kan därför vara en bra idé att kombinera olika metoder i olika stadier i programmets utveckling för att kunna täcka in så mycket som möjligt. Trots att det finns brister i metoderna för gränssnittsutvärdering bör en sådan utvärdering göras för att på så sätt kunna undvika många brister i gränssnittet. (Holyer, 1993)

2.1.4.1 Kognitiv psykologi

Metoder som bygger på kognitiv psykologi antar att användarens interaktioner med systemet består av ett antal *högnivåmål* som kan delas upp i *operatorer* och *metoder*. Operatorer är oberoende delar som tillsammans uppfyller ett mål. Dessa operatorer kan i sin tur delas upp i metoder. Flera olika metoder kan ofta användas för att uppnå ett mål. Utvärderingen innebär att man tar en linjär beskrivning av hur en användare interagerar med systemet. Varefter den omvandlas till en rikt strukturerad beskrivning av de kognitiva processer som pågår under interaktionen. Denna omvandling är mycket krävande i tid, kunskap och arbete. (Hoyler, 1993)

Ett exempel på en metod inom kognitiv psykologi tas fram av May et al. (1995) i deras avhandling. Deras metod ger dock inte konkreta förslag på utformandet av själva gränssnittet utan resultatet efter en analys av ett gränssnitt med deras metod ger enbart ett svar på om gränssnittet är bra eller inte.

2.1.4.2 Socialpsykologi

Målet med de socialpsykologiska metoderna är att få reda på vad användaren tycker om programmet. Detta görs med hjälp av intervjuer och frågeformulär. Antalet frågor varierar; det finns metoder med tio frågor medan andra har upp till femtio. Kvaliteten på frågorna är minst lika viktig som kvantiteten eftersom undersökningar visar att resultaten från en metod med tio frågor jämfört med en med femtio har en korrelation på 0,86. De här metoderna säger dock inte så mycket om vad som är dåligt med gränssnittet medan det är relativt lätt att luska ut om programmets gränssnitt är bra eller dåligt. Man får, med andra ord, inga direkta rekommendationer om vad som kan förbättras på gränssnittet och hur detta skall göras. (Hoyler, 1993)

2.1.4.3 Social vetenskap

Socialvetenskapliga metoder innebär att frivilliga får prova systemet eller se en demonstration av det. (Holyer, 1993)

Om försökspersonerna får prova systemet, gärna för att lösa vissa förutbestämda uppgifter, försöker man på olika sätt att spela in deras interaktion. De frivilliga uppmanas att tänk högt om vad de gör och vad de tycker om systemet. Det är lämpligt att undersökningsledaren tar rikligt med anteckningar under tiden och kan även ha en aktiv dialog med användaren om hur gränssnittet kan förbättras; ofta sker det i praktiken även om det inte alltid är meningen. (Holyer, 1993)

Vid demonstrationer visar utvärderingsledaren programmet för en grupp frivilliga. Efter demonstrationen leder utvärderaren en halvstrukturerad diskussion om vad gruppen tyckte om programmet. (Hoyler, 1993)

Dessa metoder är relativt enkla och naturliga. Användbara resultat kan fås ganska enkelt. Dock är det tekniskt svårt att spela in vad som händer på ett användbart sätt, som inte stör utvärderingen. Användbar information kan vara svår att hitta eftersom inspelningen ofta genererar stora mängder data. Det största problemet är att resultatet i hög grad beror på hur duktig utvärderaren är.(Holyer, 1993)

Demonstrationsvarianten har fördelen att utvärderaren är i minoritet, varför det är troligare att gruppen inte bara säger det som utvärderaren vill höra utan yttrar sina sanna åsikter. Med relativt liten ansträngning kan ganska mycket användbar information fås och gruppen pekar ofta på saker som utvärderarna inte hade tänkt på. (Hoyler, 1993)

2.1.4.4 Systemutveckling

Dessa metoder grundas på metoder som är vanliga inom systemutveckling. En metod, som heter *heuristisk* utvärdering, genomförs genom att låta en gränssnittsexpert undersöka gränssnittet i fråga, eller en mer eller mindre exakt specifikation av det. Med sina erfarenheter som grund, möjligen hjälpt av olika riktlinjer, påpekar han sedan olika saker som kan tänkas orsaka problem. En annan metod är att man gör en *papperskörning*; en grupp går igenom gränssnittet steg för steg och undersöker vid varje punkt alla händelser som tänkas kan. (Hoyler, 1993)

Det är ofta både betydligt snabbare och minst lika effektivt att låta en expert gå igenom gränssnittet. Metoden beror dock mycket på hur duktig utvärderaren är. Effektivitet uppnås först när flera utvärderare används eftersom en utvärderare tenderar att hitta runt 35% av alla fel. Olika utvärderare hittar ofta olika fel. Tre till fem personer anses därför vara det mest kostnadseffektiva antalet. (Nielsen, 1994 [1])

Papperskörningen är mycket arbetskrävande och många formulär måste fyllas i under arbetets gång. Den fungerar bäst för program som används ibland, som har ett begränsat användningsområde och där programmet går igenom ett mindre antal lägen. Det är i princip omöjligt att tillämpa den på mer komplexa program, eller program som inte har väl definierade lägen; vilket ofta är fallet i grafiska miljöer med direkt manipulation. (Hoyler, 1993)

2.2 Tillvägagångssätt

Vid utformandet av den här uppsatsen har, i stora drag, en kvalitativ forskningsprocess tillämpats. Uppsatsen antar en kvalitativ karaktär eftersom den främst handlar om hur människan uppfattar sin omvärld, där datorgränssnitt ingår, vilket gör att den antar en icke-kvantifierbar karaktär. (Backman, 1998)

Efter att ha etablerat ett generellt problemområde - datorgränssnitt - startade jag litteraturstudierna. Källor blir främst Internet-dokument och böcker lånade från biblioteket samt diverse artiklar. Litteraturstudierna var till en viss del *explorativa* i sin natur eftersom jag ville komplettera mina kunskaper om hur människan uppfattar sin omgivning. Det var dock främst fråga om *deskriptiva* litteraturstudier eftersom jag redan hade viss kunskap i området och därför att jag även begränsade mina litteraturstudier inom mänskligt tänkande till aspekter som var av relevans för gränssnitt. Arbetet med uppsatsen gick därefter in i en *normativ* eller *hypotesskapande* fas, där jag försökte hitta samband mellan den utarbetade teorin, från den deskriptiva fasen, och gränssnittsutformning. (Patel & Davidson, 1994)

Efter att jag hade läst Staffan Björks (2000) avhandling om *Flip Zooming* ringde jag honom för att få förslag på någon lämplig applikation att utveckla för handhållna datorer. Det av de förslag som han gav som jag fastnade mest för var en filhanterare. Det

kändes genomförbart och intressant. Jag har använt en hel del olika filhanterare och kände därför att jag även hade en bra bild av vad som krävdes. Utvecklingen av applikationen pågick parallellt med de redan beskrivna litteraturstudierna, jag väntade inte tills den normativa fasen var klar. Den utvecklingsmetod som jag tillämpade vid byggandet av applikationen var *prototyping*. (Andersen, 1994)

Efter att den normativa fasen var avklarad gjordes en *beskrivning* av programmet och dess funktioner i sin dåvarande form. Eftersom avhandlingen handlar om användarvänlighet antog beskrivningen formen av en användarmanual och inte vilka programmeringstekniker eller programmeringsspråk som har används.

När programbeskrivningen var klar startade analys eller utvärderingsfasen. Den utvärderingsmetod av de beskrivna metoderna för gränssnitt som jag valde var en *heuristisk*-metod. Den största anledningen var dess enkelhet. Eftersom en enda utvärderare, även om denne är expert, i genomsnitt upptäcker 35% av alla fel hade det givit enormt mycket om fler hade varit med om utvärderingen. (Nielsen, 1994 [1]) Utgångskriterierna för utvärderingen blev inte redan fastslagna tumregler för ett bra gränssnutt utan de egenframtagna, det fanns helt enkelt inte någon annan expert på området som kunde tillfrågas om att utföra en analys med dessa normer som mall.

Sist i avhandlingen följer därefter en slutdiskussion där tyngdpunkten ligger på vad som är viktigt att tänka på vid utformning av gränssnitt för små och mobila klienter.

2.2.1 Tillämpning av prototyping

Som redan har blivit klarlagt och beskrivit är prototyping en iterativ process i fem steg. (Andersen, 1994) Nedan beskriver jag i korthet hur jag har tillämpat dem:

1. **Identifiera centrala behov**
   Först och främst bestämde jag mig för att jag skulle göra en filhanterare. Jag hade redan från start en bra bild av vilka grundläggande funktioner som en *bra* filhanterare har, funktioner som att kunna förflytta sig i filstrukturen, skapa kataloger och flytta, kopiera och byta namn på filer. Det var detta som jag utgick ifrån.

2. **Utveckla en första prototyp**
   Den första prototypen av filhanteraren framställdes väldigt snabbt, den hade dock ingen funktionalitet alls och visade enbart ungefär hur jag hade tänkt att navigationen i filstrukturen skulle te sig. Vid det här tillfället var även filträdet simulerat. Programmet vara bara ett skal utan motor. Andersen (1994) skriver att i det här skedet bör prototypen hellre ha de grundläggande funktionerna implementerade, vilket i mitt fall skulle ha inneburit olika grundläggande filhanterings operatorer som filkopiering, och inte avancerade skärmdumpar. Men eftersom tyngdpunkten i detta fall ligger på själva gränssnittet valde jag att istället att låta den första prototypen enbart vara en *avancerad skärmdump* med begränsad funktionalitet, för att ge en känsla av hur det hela skulle se ut.

3. **Demonstrera och diskutera sätt att förbättra applikationen**
   Applikationen har vid ett flertal tillfällen visats för ett litet antal testpersoner

som har fått ge sina synpunkter. Dock har de flesta idéer om förändringar kommit från mig självt. Dessa testpersoner har varit två till antalet. Två erfarna datoranvändare varav en som har stor erfarenhet med att använda olika filhanterare och en som inte har annan erfarenhet än med Utforskaren. Försökspersonerna var en man och en kvinna för att även på det sättet kunna få lite olika perspektiv.

4. **Realisera förbättringar**
   Efter att nya idéer om förbättringar har framkommit har dessa även realiserats i programmet.

5. **Täcker prototypen behoven?**
   Applikationen har genomgått ganska drastiska förändringar och förbättringar i olika steg, från att ha varit ett skal som opererat i ett simulerat filträd till att faktiskt kunna hantera de mest grundläggande funktioner som en filhanterare har. Applikationen är ännu inte färdig och kan därmed ännu inte lämna prototypstadiet och iterationsprocessen får fortsätta även efter den här uppsatsen. Med andra ord, prototypen täcker ännu ej de krav som en filhanterare skall uppfylla.

3 Teoretisk grund

Det här kapitlet är själva stommen i uppsatsen. Den del som alla de andra efterföljande delarna lutar sig emot. Dess syfte är att ge läsaren en ökad insikt i de bakomliggande teorierna vid människa- datorinteraktion. Inte bara att en användare uppfattar en företeelse som bra eller dålig och att det är bra att göra en sak på det ena eller det andra sättet utan även *varför* det är så.

3.1 *Likheter mellan Användargränssnitt och Språk*

Ett sätt att belysa vad ett gränssnitt är och även ge en utförligare definition är genom att likna det med ett språk. Precis som språk är ett medel för kommunikation, fungerar även ett gränssnitt mellan datorer och människa som ett kommunikationsmedel. Om man studerar det talade språkets problematik kanske det är möjligt att i samma veva komma en bit på vägen att konstruera ett bättre gränssnitt mellan datorer och människor. Enligt Kukulska-Hulme (1999) är det dessutom lämpligt att se på alla datoranvändare som människor som är i färd med att *lära sig ett språk.* Detta eftersom inlärandet av varje ny applikationsmiljö innebär att termer och koncept måste inhämtas tillsammans med att redan kända termer antar nya innebörder.

Kommunicerandet med det talade språket har reella problem. Inte bara att det råder en förbistring mellan olika språk, utan även människor som talar samma språk kan ha problem med att förstå varandra. Det gäller att se vad som ligger bakom eller bortom själva orden, själva andemeningen av vad den kommunicerande parten vill framföra. Umberto Eco tar upp människans sökande efter det *fullkomliga* språket i sin bok *The Dream of a Perfect Language*. (Eco, 1995)

3.1.1 Sökandet efter det fullkomliga språket

Människan har i många århundraden sökt efter det fullkomliga språket som talades i Edens lustgård av Adam och Eva, innan språkförbistringen vid Babels torn, d*et fullkomliga kommunikationsmedlet*. Åsikterna om hur detta språk var utformat har varit lika många som de som har forskat i ämnet. Teorier har gått från tro på allt från att något av de befintliga språken varit det rätta, vanligtvis det språk som förespråkaren själv talar. Till exempel trodde Martin Luther att tyska var det språk som stod närmat Gud och därför var det fullkomliga språket. Vissa aspirationer på att svenska var det fullkomliga språket gjordes även av bland andra Olaus Rudbeck. Han ansåg även att Sverige var det mytiska Atlantis från vilket civilisationen har spridit sig utöver Jorden. Åter andra pekade på hebreiskan. Man menade bland annat att alla andra språk kan härledas från hebreiskan, som var det språk som Guds utvalda folk, israeliterna, talade och talar fortfarande för den delen. (Eco, 1995)

3.1.1.1 Konstruerade språk

Det var inte enbart de befintliga språken som undersöktes. Det fanns de som ansåg att dessa språk var långt ifrån perfekta. Av de som trodde detta försökte många själva konstruera fullkomliga språket. Försök gjordes där man sneglade på matematiken och där man försökte bygga språk med hjälp av komplicerade matematiska modeller. Eco (1995) tar upp ett exempel på ett matematiskt språk som Raymond Lully plockade ihop (800- eller 900-talet). Lully såg på sitt matematiskt perfekta språk som ett instrument för att omvända de otrogna. Det sägs att han åkte för att visa sitt språk för muslimerna, för att omvända dem, och, oövertygande som hans språk var, blev dödad. Många andra mer eller mindre lyckade försök att skapa eller återskapa *det* fullkomliga språket har gjorts. Det har gjorts försök på allt från skumma bokstavskombinationer till speciella musiksekvenser. Men misslyckandet har varit totalt. Enligt Eco (1995) fanns det en man vid namn John Wilkins som på 1600-talet gjorde de första försöken med hypertext i sin strävan att skapa det fullkomliga språket. Men även han misslyckades, som alla andra. Han tog sig vatten över huvudet. Andra spin-offs från denna jakt har till exempel varit den moderna logiken, helt fruktlöst har det dock inte varit. (Eco, 1995)

3.1.1.2 Parameterspråk

Eco (1995) tar upp ett sätt att försöka lösa problemet med ett så kallat parameterspråk. Vid översättning mellan de naturliga språken får man oundvikligen problem. Ett uttryck på ett språk kanske inte alls har en *exakt* motsvarighet i det andra språket som man vill översätta till. Ett parameterspråk är ett slags medium genom vilket man går för att översätta mellan språk A och B. När man vill översätta ett uttryck från språk A till språk B bestämmer eller kommer man fram till att båda uttrycken, i språk A och B, är ekvivalenta med just ett uttryck i språk C, vårt parameterspråk. Man översätter alltså först uttrycket på språk A till parameterspråket C. Man har på förhand kommit fram till att uttrycket i språk A är ekvivalent med ett visst uttryck i språk C, för att sedan översätta uttrycket från språk C till språk B, på samma basis. Men uttrycken på språk A och B var inte direkt ekvivalenta med varandra, man måste gå över språk C för att uppnå ekvivalens och förståelse enligt Eco (1995).

Detta parameterspråk kan ses som ett gränssnitt, tolk, mellan de två andra språken. Ett medium genom vilket utövare av de respektive språken kan få en exakt bild av vad den andre tänker och menar. Ett språk som exakt kan uttrycka innebörden av olika uttryck från andra språk, och som kan förmedla vidare denna innebörd oförändrad till ett tredje språk. Om vi överför termerna till datorer. Vi talar här om ett språk eller ett gränssnitt som kan förmedla till datorn exakt vad människan menar på ett sätt som datorn förstår. Och som kan åt andra hållet exakt förmedla datorns tankebanor på ett sätt som människan kan förstå, utan friktion, irritation och frustration. Det fullkomliga gränssnittet.

En intressant parantes är det översättningsprojekt som pågår för att ta bort språkbarriärerna på Internet. Det inbegriper användandet av ett universalt parameterspråk. Det blir därmed inte behövligt att alla språk översätts till och från alla

andra språk, utan det enda som behövs är att språket översätts till och från parameterspråket. Systemet fungerar på det här sättet:

1. Skribenten skriver text på sitt eget språk.
2. Datorn översätter meningarna till en rad logiska påståenden på det nya datorparameterspråket.
3. Datorn översätter sedan tillbaka till originalspråket. Nu kan skribenten rätta till eventuella tvetydigheter.
4. Nu är det möjligt för varje användare att läsa texten på sitt eget språk. Datorn ser till att översättningen är till 100 procent korrekt.

Den enda tänkbara nackdelen är att ett sådant system kommer att styra hur man formulerar sig. Men å andra sidan kan luddiga och tvetydiga formuleringar filtreras bort. (Illustrerad Vetenskap, 2000)

3.1.1.3 Aymara

Finns det ett sådant naturligt språk som kan användas som parameterspråk eller måste det konstrueras? Om det är frågan om ett naturligt språk krävs det att det är så pass perfekt, flexibelt och kraftfullt att det kan utgöra detta tredje språk som alla andra språk kan identifiera sig med. Enligt Eco (1995) beskrev jesuiten Ludovico Bertonio språket Aymara år 1603. Ett språk som till viss del fortfarande talas av indianer mellan Chile, Bolivia och Peru. Aymara är ett språk som är utrustat med en väldig flexibilitet, förmågan att beskriva neologismer och det är särskilt anpassat för att uttrycka abstrakta koncept. Neologism är förmågan att ta upp nya ord och uttryck eller förmågan att ta gamla ord och ge dem en ny mening eller innebörd. Nyligen har man även visat att Aymara inte baseras på den Aristotiska enkla tvådelade logiken, antigen sant eller falskt, utan på en tredelad logik. På grund av detta kan Aymara uttrycka modala finesser och hårfina detaljer som andra språk endast kan omfånga genom komplexa omskrivningar. Enligt Flem-Ath (1996) stod det i en artikel i Times att:

> *Aymara har en syntax som är väldigt simpel och väl definierad, det betyder att dess syntaxiska regler alltid gäller, och att det kan kortfattat uttrycka den typ av algebraiska språk som datorer förstår. Dess renhet är så total att somliga forskare hävdar att det inte utvecklades, som andra språk har gjort, utan att det har blivit konstruerat från grunden.*

Aymara är också mycket ordrikt, enligt Pedraza (2001) har det mer är 363 394 720 ord!. Det här ställer intressanta frågor om varifrån och vem som konstruerat språket, **om** det nu har blivit konstruerat, men det skall vi inte gå in på här. Det finns dock de som menar att det är överlevande från Atlantis som står bakom språket, men det är en annan historia. Flem-Ath (1996)

Men hur skulle då Aymara fungera som gränssnitt mellan olika andra språk? Duger det som parameterspråk? Om Aymara skulle fungera som ett gränssnitt skulle det alla översättningsproblem som finns. Det skulle enkelt gå att datorisera översättning genom att alltid först översätter från ett språk till Aymara för att sedan översätta från Aymara

till vilket språk det nu vara månne som det önskas översättas till. Men tyvärr, så fungerar det inte. Aymara klarar helt utmärkt av att fånga upp andra språks uttryck och fånga upp dem i sina egna termer, men processen fungerar inte alls åt det andra hållet. På grund av att Aymara är så perfekt kan det mycket väl tolka alla tankar i andra sinsemellan oöversättbara språk. Men priset för denna flexibilitet, när väl det perfekta språket har överfört tanken eller uttrycket till sina termer, är att det inte går att översätta tillbaka till våra naturliga språk. Aymara är som ett stort svart hål. (Eco, 1995)

3.1.1.4 Gränssnitt och språk

Betyder det att även det perfekta gränssnittet kommer att sluka alla intryck från datorn eller från användaren? Det vill säga att gränssnittet kommer att vara fullt kapabel att själv förstå och tolka intryck från både användare och dator men inte kommer att klara att överföra intrycket till den respektive andre. Aymara, trots sin perfektion, klarar inte av att lösa problemet, kanske helt enkelt på grund av att det inte är ett fullkomligt språk. (Eco, 1995) Sökandet efter det perfekta gränssnittet kan man säga inbegriper skapandet av ett nytt vetenskapligt Aymara, ett perfekt kommunikationsmedium. Ett kommunikationsmedium som klarar av att tolka båda sidor *och* som klarar att återge för den ene vad den andre uttrycker genom detta perfekta gränssnitt. Problemet ligger med andra ord i hur man skapar ett gränssnitt som kan överföra respektives signaler till den andre. Alltså hur gränssnittet skall kunna överföra *sina* begreppstermer till mottagarens begreppstermer, datorn eller människan. Problemet ligger inte i att skapa ett gränssnitt som kan tolka och förstå datorn eller människan.

Att lära sig använda en dator eller ett visst datorprogram kan liknas vid att lära sig ett språk. Datorn kommunicerar på ett visst sätt och användaren måste lära sig att tolka signalerna och kommunicera på ett för datorn begripligt sätt.

3.2 Användarvänlighet

Nedan följer ett par av gränssnittsexperter redan framtagna tumregler. Först de vad Cox & Walker (1996) anser karaktäriserar ett fullgott användargränssnitt mixat med Coopers (1995) idéer om att ett gränssnitt skall använda sig av, för användaren, närliggande modeller. Sist tas även de av Nielsen (1994 [2]) framtagna tumreglerna. Detta för att illustrera vad tidigare forskning inom användarvänlighet har kommit fram till. Vilket kan vara bra att ha i bakhuvudet vid den senare analysen och de egna rekommendationerna.

De egenskaper som karaktäriserar en användarvänlig och funktionell datorapplikation är enligt Cox & Walker (1993) att det är användaren som styr. Men även att verktyget eller applikationen är genomskinligt, flexibelt och att det finns möjligheter för användaren att lära sig applikationen, vad den kan användas till och hur den används. Genomskinlighet innebär att när användaren väl har lärt sig att handskas med applikationen *försvinner* gränssnittet och användaren kan koncentrera sig på uppgiften och inte applikationen. Genom att vara flexibel kan applikationen användas av olika personer på olika sätt som passar just dem samt att applikationen kan användas till andra saker än vad den ursprungligen var tänkt till.

För att användaren skall kunna lära sig hur applikationen fungerar måste denne skapa sig en *konceptuell modell* av hur applikationen och systemet fungerar. För att möjliggöra detta bör ett gränssnitt fungera efter modeller som ligger nära användaren som denne med lätthet kan ta till sig och lära sig. Om däremot systemet eller applikationen tydligt uppvisar sin *implementationsmodell*, hur programmet är implementerat, menar Cooper (1995) att applikationen blir svår att förstå och använda. Applikationen måste vara konsistent för att användaren skall kunna skapa sig denna; liknande objekt måste uppföra sig på liknande sätt. Annars får användaren problem med att skapa modellen och måste inkludera undantag som kan vara svåra att memorera. Systemet är inte längre lättlärt och användarvänligt.

För att sammanfatta det hela bör ett användarvänligt gränssnitt:

- låta användaren styra händelseförloppet
- vara genomskinligt
- vara flexibelt
- vara lättlärt
- fungera enligt modeller som ligger när användaren
- vara konsistent

Nielsen & Mack (1994) tar upp en lista på tio tumregler vid användargränssnitts design som påminner om ovanstående sex regler. Dock är denna beskrivning lite mer utförlig, inte på en lika hög abstraktionsnivå och de ger praktiska tips om vad och hur de kan tillämpas. (Nielsen, 1994 [2])

- **Systemstatus skall vara synlig**
  Systemet bör informera användaren, utan för stor fördröjning, vad som pågår.
- **Likheter mellan systemet och den verkliga världen**
  Systemet bör tala användarens språk med ord, fraser och koncept som användaren känner igen snarare än systemorienterade termer. Information bör framträda på ett naturligt och logiskt sätt på ett sätt som påminner om den verkliga världen.
- **Användarkontroll och användarfrihet**
  Användare använder ofta systemfunktioner av misstag och behöver därför en klart markerad *nödutgång* för att kunna lämna det oönskade tillståendet utan att för dens skull behöva gå igenom en utdragen dialogprocess. *Ånger-* och *upprepningsfunktioner*.
- **Konsistens och standarder**
  Användare skall inte behöva fundera över om olika ord, handlingar och situationer betyder samma sak i olika system. Följ därför plattformsstandarder i möjligaste mån.

- **Felundvikande**
  Något som är till och med bättre än bra felmeddelanden är ett användargränssnitt designat på ett sådant sätt att fel helt enkelt förhindras från att inträffa.

- **Igenkännande snarare än ihågkommande**
  Synliggör objekt, handlingar och val. Användaren skall inte behöva komma ihåg information från en del av *dialogen*, systemet, till en annan. Instruktioner för användning av systemet skall vara väl synliga eller lätt åtkomliga närhelst det är önskvärt.

- **Användningseffektivitet och flexibilitet**
  Olika *acceleratorer* som är osynliga för den oerfarne användaren kan ofta snabba upp användandet för den erfarne användaren och kan därvid tillfredsställa både den erfarne som den oerfarne. Tillåt användaren att *spela in*, skripta, ofta förekommande handlingar.

- **Estetisk och minimalistisk design**
  Dialoger inom systemet bör inte innehålla irrelevant eller sällan behövd information. Varje extra vit information som förekommer tävlar med de relevanta informationsbitarna och förminskar deras relativa synlighet.

- **Hjälp användaren att känna igen och återhämta sig från fel**
  Felmeddelanden bör uttryckas i vanligt språk och inte i koder, precist indikera vad som är fel och föreslå en lösning på problemet.

- **Hjälpfunktioner och dokumentation**
  Även om det är bättre om systemet kan användas utan hjälp och dokumentation, vara intuitivt, kan det vara nödvändigt att tillhandahålla sådan. All sådan information bör vara lätt att söka igenom, fokuserad på användarens uppgifter, uppvisa en lista på konkreta steg över vad som skall utföras och inte vara för lång.

3.3 Användarens mentala förmågor

Antropologen Henry F Osborn har sagt att hjärnan är *det mest förunderliga och hemlighetsfulla i universum.* (Meldau, 1968) I samma bok står det att fysiologen Charles Sherrington lär ha ställt frågan: *Hur frambringar hjärnan tankar? Det är den centrala frågan och vi har ännu inget svar på den.* Hur människan exakt fungerar mentalt är något som vi egentligen inte vet och det är detta som är det stora problemet inom gränssnittsutveckling. Det senaste om hur våra tankeprocesser och hur vår hjärna anses fungera påvisar jag med hjälp av de framlagda teorierna. Men det är viktigt att framhålla att de psykologiska teorier som läggs fram är just bara teorier. Det är i allra högsta grad som biologen Francis Crick säger:

> *Trots den stadigt ökande mängden detaljerad kunskap om hur människohjärnan fungerar är den fortfarande i hög grad ett mysterium.* (Scientific American, september 1979)

Det skall ändå inte ses som omöjligt och därför onödigt arbete att göra interaktionen med datorer enklare och effektivare trots den kunskapsbrist som råder. Det är precis det

som är syftet med det här kapitlet, att framlägga de befintliga teorier om människor mentala processer och därmed försöka ge bild av hur människor fungerar och vad som driver oss.

För att kunna bygga ett användarvänligt gränssnitt måste hänsyn tas till just användaren och de mentala processer som ligger bakom denne. Utan kunskap om användaren är det omöjligt att kunna bygga ett gränssnitt enligt modeller som ligger nära denne. Värt att ha i åtanke är att de processer som styr en människas handlande och tänkande är ytterst personligt och kan variera på grund av olika uppväxtförhållanden, erfarenheter och genetiska betingelser bland annat. Vad som kommer att tas upp i detta stycke blir därmed allmänna regler och principer.

3.3.1 Arbetsminnet

Enligt Haberlandt (1994) är *arbetsminnet tänkandets centrum*. Både lagring och manipulation av information är uppgifter som det hanterar. Han fortsätter med att skriva att arbetsminnet eller korttidsminnet har olika processer för sökning och jämförelse, numerisk omvandling, språkförståelse och problemlösning. Arbetsminnet lagrar temporär information under ovanstående operationer samtidigt som det koordinerar dem. Han skriver vidare att arbetsminnet är en effektiv operatör som effektivt och snabbt hämtar viktig information som är vital för utförandet av tankekrävande uppgifter. Arbetsminnet är mycket snabbt men det har tyvärr vissa lagringsmässiga begränsningar. Som tur är kan denna lagringskapacitet utökas med hjälp av träning. Innehållet i arbetsminnet ändras dynamiskt efter behovet av de pågående kognitiva mentala processerna (tankeprocesserna). Arbetsminnet representerar den mest aktiva delen av vår kunskap och det är där som tankeverksamheten är aktiv. (Haberlandt, 1994)

3.3.1.1 Problemlösning

Haberlandt (1994) skriver sedan att arbetsminnet innehåller den plan eller det tillvägagångssätt som används för att lösa ett specifikt och aktuellt problem. Den problemlösande delen av arbetsminnet (problemlösaren) representerar problemstrukturen, med dess initial-, mellan- och sluttillstånd, ungefär som man memorerar en lista med ord. Transformationer och mellanstillstånd hålls därmed också i arbetsminnet för att göra åtkomsten till dessa snabbare. Svårigheter med problemlösning uppstår när planen eller tillvägagångssättet inte får plats i arbetsminnet, antigen på grund av att den är för komplex eller på grund av att arbetsminnet är begränsat eller skadat. Problemlösaren måste hålla varje delmål, i vilken ordning (sekvens) som delmålen kommer och vad som skall utföras i arbetsminnet. Om antalet delmål ökar kommer komplexiteten av tillvägagångssättet att öka och efter ett tag kommer problemlösaren att ha svårt att hålla ordning på och komma ihåg målen och den nuvarande positionen bland delmålen. (Haberlandt, 1994)

3.3.1.2 Lagringskapacitet vs arbetsprestation

Arbetsbördan eller arbetsprestationen slåss om utrymme i arbetsminnet med lagringskapacitet på ett mycket direkt och påtagligt sätt. Det finns ett direkt samband

mellan det faktum att ju mer arbete som problemlösaren lägger på omvandling och beräkning av innehållet av arbetsminnet desto mindre kapacitet kvarstår till lagringskapacitet. (Haberlandt, 1994)

3.3.1.3 Språkförståelse

Arbetsminnet spelar även en nyckelroll när det gäller språkförståelse. Vi skulle aldrig kunna förstå en komplett mening om vi inte kunde komma ihåg ord från dess första delar. (Haberlandt, 1994)

> *Ett bra exempel för att illustrera hur arbetsminnet fungerar är lagringen av ett telefonnummer mellan den tid som det letas fram i telefonboken och den tid det tar att ringa det.*

Den sista förekomsten av ordet *det* i den föregående satsen refererar på ett föregående ord, i det här fallet *telefonnummer*, som nämns 17 ord tidigare. För att det skall vara möjligt att förstå det pronomen som används i satsen måste lyssnaren också komma ihåg vilket substantiv som det syftar på. Lyssnaren måste med andra ord känna igen ordet *det*, förstå att det syftar på ett föregående ord, genomsöka arbetsminnet efter det föregående ord som åsyftas, och sedan förstå det koncept som ordet *telefonnummer* innehar. Till en viss del är samtliga dessa processer samtidiga och pekar därmed på att arbetsminnet fungerar både som operatör och lagrare av information. (Haberlandt, 1994)

3.3.1.4 Sökning i arbetsminnet

En av de bäst undersökta egenskaperna hos arbetsminnet är sökning i det. I sökning ingår att leta efter ett mål eller ett specifikt objekt bland att visst antal olika objekt. Haberlandt (1994) tar det exemplet att när en person letar efter en socka i en sockläda. Då har hon skapat en inre mental bild av den socka som hon letar efter och jämför de sockor som hittas i lådan med den mentala bild av en socka som hon har. Rätt socka har hittats när den mentala bilden och en socka matchar varandra. Generellt kan sägas att en person, när denne utför en sökning, skapar en mental representation av målet, det som letas efter, och sedan jämförs målet med de objekt som letandet sker bland. (Haberlandt, 1994)

Experiment pekar på det faktum att den tid det tar att utföra en uppgift, till exempel en sökning, ökar om belastningen på arbetsminnet ökar på grund av arbetsminnets kapacitets begränsning. Om den mängd med objekt som det skall letas bland ökas kommer det därmed att ta längre tid att hitta målet. (Haberlandt, 1994)

Vid en sökning sker ett igenkännande av målet snabbare om det objekt som föregick det objekt som man letar efter liknar det objekt som är eftersökt. (Haberlandt, 1994)

3.3.2 Långtidsminnet

Långtidsminnets funktion och förmåga att prestera är av yttersta vikt för att vi skall kunna åta oss att på ett godtagbart sätt utföra de mesta av vad som krävs av oss dagligen. Vi kommer ihåg många olika saker, under olika omständigheter och över olika tidsintervall. (Haberlandt, 1994)

3.3.2.1 Minnets dimensioner

Minnesforskare talar om tre olika stadier eller faser för minnesrepresentationer: inhämtning eller tillgodogörande, lagring och återskapande eller återhämtande. Med inhämtning menas själva tillgodogörandet av information. Inhämtningen sker under diverse omständigheter när en person får fakta sagda till sig, läser en historia, ser en händelse utspela sig eller vid explicit memorering av information. Information som är inhämtad och tillgodogjord kommer att finnas kvar och den kommer att lagras som en representation av en viss *styrka* och *aktiveringstillstånd*. Vid en viss tidpunkt är de flesta av våra lagrade representationer inaktiverade. Återskapande innebär att man återhämtar informationen från långtidsminnets djupaste vrår och återför den till ett mera aktivt tillstånd. (Haberlandt, 1994)

Ihågkommen information har två olika attribut; dess styrka och dess aktiveringstillstånd. Med minnesrepresentationens styrka menas hållbarheten över tiden för den givna representationen. Information som är ordentligt tillgodogjord har hög styrka, till exempel namnet på dina föräldrar eller ditt telefonnummer. Dessa minnesrepresentationer sitter väldigt envist kvar i långtidsminnet utan att det alltid finns ett behov att tänka på dem. Å andra sidan har fakta som är inlärda för länge sedan och som inte har använts på länge en mycket låg styrka. Aktiveringstillståndet av en minnesrepresentation, till skillnad från dess styrka, är temporärt. En person kanske får reda på namnet på en annan person på en fest, informationen är då tillfälligt aktiv. Men informationen kan glömmas snabbt och aldrig få någon högre grad av styrka. Aktiveringstillståndet är med andra ord en bild av hur tillgänglig informationen är, hur lätt det är att återhämta den. (Haberlandt, 1994)

Haberlandt (1994) skriver att information och kunskap existerar i många olika former. Skillnad görs mellan procedurmässig- och deklarativkunskap och mellan semantiska- och episodiskaminnen. I procedurmässiga kunskaper ingår motoriska- och kognitivaförmågor som vi inte lätt kan uttrycka med ord. Deklarativa kunskaper är kunskaper som innehåller information om den fysiska-, sociala- och språkligamiljön, inklusive semantiskakunskaper och kunskaper om ords betydelse. Semantiska minnen bevaras över en lång tid, även om vi inte längre kommer ihåg under vilka specifika episoder som vi inhämtade och tillgodogjorde oss minnena. Medan i episodiska minnen kommer vi fortfarande ihåg under vilken episod som minnena uppstod som tillika minnena själva.

Om man får i uppgift att memorera en lista, till exempel en lista med ord, är det generellt sett lättast att återge de första och de sista orden från listan. Det finns ett antal olika faktorer som direkt kan påverka minnesförmågan. (Haberlandt, 1994)

- **Längden på den lista som skall memoreras.** En längre lista är svårare att komma ihåg.
- **Den tid som man lägger på att lära sig listan.** Desto mer tid som läggs på att memorera en lista desto bättre koms den ihåg. Det är också oftast bättre att sprida ut inlärandet i intervall snarare än att ta in allt på en gång.

- **Informationen tas bättre upp om testmiljön påminner om läromiljön.** Det är med andra ord lättare att återkalla ett minne om den miljö som en person befinner sig i om den påminner om den miljö som hon befann sig i när hon en gång lärde sig det som hon vill komma ihåg.
- **Information som är distinktiv koms lättare ihåg.** Olika information som inte är distinktiv utan liknar annan information som också skall memoreras grupperas lätt tillsamman. Det kan vara bra när information är relaterad men risken blir å andra sidan att den blandas ihop med varandra. Medan information som på ett markant sätt särskiljer sig lättare blir ihågkommet.
- **Det är oftare lättare att *känna igen* något än att *komma ihåg* det.** Ta bara ett främmande språk som vi behärskar någorlunda som exempel. Vi kanske förstår alla ord som människor säger till oss eftersom vi känner igen dem, men vi kanske inte kommer ihåg alla dessa ord när vi skälva skall säga något.

3.3.3 Uppmärksamhet

Den här delen av medvetandet berör vad som vårt medvetande är fokuserat på. Det vill säga, vad som just nu är aktivt i arbetsminnet. Men, enligt Wickens (1992), handlar det också om vilka mentala processer som underligger att vi som människor väljer att fokusera på rätt saker för att vi på ett så effektivt sätt som möjligt skall kunna utföra en uppgift. Det kan handla om ett visst tillvägagångssätt eller uppförande som på ett optimalt sätt kan maximera ett förväntat värde eller minimera en kostnad. Det kan beröra sådant som att det finns ett tvång eller behov av att regelbundet kontrollera vissa mätvärden, som exempel kan tas en pilot som flyger ett flygplan och skall landa. Om piloten endast regelbundet kontrollerar flygplanets hastighet men struntar i att kontrollera höjdmätaren uppträder denne inte alls optimalt för att på ett bra sätt utföra landningen. En pilot som kontrollerar både hastighets- och höjdmätare men som inte bryr sig om att ta in den visuella världen genom fönstret beter sig på ett sätt optimalare men riskerar att missa viktiga saker som inte instrumenten visar på, exempelvis en fiskmås. (Wickens, 1992)

Tillgodogörande av information från omgivningen eller de informationskanaler som finns är guidat av den förväntade kostnad som uppträder om information missas. Den förväntade kostnaden är direkt relaterad till den faktiska kostnaden och i vilken frekvens som information uppdateras, det vill säga hur ofta kanalen bör kontrolleras.. Om de kanaler som uppdateras ofta inte kontrolleras tillräckligt ofta kommer sannolikheten att öka att informationen missas. Till exempel om en pilot kontrollerar sin omgivning utanför planet vid en tidpunkt och ser att allt är lugnt, det finns inga synliga hinder, kommer ännu en kontroll genom fönstret två sekunder senare att med största sannolikhet visa samma bild. Men om piloten väntar två minuter med att se ut genom fönstret kommer sannolikheten att han skall ha missat vital information, som ett annat flygplan, att öka betydligt. (Wickens, 1992)

Enligt undersökningar lär sig människor att oftare kontrollera informationskanaler där information uppdateras mer frekvent och att mer sällan kontrollera informationskanaler där informationen uppdateras mindre ofta. Men att trots detta tenderar frekvensen av

kontrollerna av de kanaler som uppdateras mer sällan att ske för ofta och kontrollen av de kanaler som uppdateras oftare att ske för sällan. (Wickens, 1992)

3.3.3.1 Åkallande av uppmärksamhet

Vidare tar Wickens (1992) upp frågan om det finns någon skillnad i effektivitet, när det gäller att åkalla uppmärksamhet till något specifikt, mellan hörsel-, syn- och känselsignaler. Både hörsel- och känselsignaler är vida överlägsna i effektivitet för att åkalla uppmärksamhet till något som uppmärksamheten inte är fokuserat på. Känsel är dessutom mer effektiv än hörsel. Till exempel, trots att det blinkar vansinnigt av varningssignaler på datorskärmen kommer vi ändå att vända bort vår uppmärksamhet därifrån om någon skulle knacka oss på axeln.

***3.3.3.2** Delad uppmärksamhet*

Vi människor är inte begränsade till att endast kunna utföra en uppgift åt gången. Vi kan köra bil, hålla utkik efter vägskyltar, tugga tuggummi och lyssna på musik samtidigt (dock med en viss prestations förlust för varje, till skillnad från om vi endast hade utfört en uppgift). Det finns fyra egenskaper hos uppgiften och hos individen som påverkar hur bra man kan utföra parallell informations behandling, nämligen (Wickens, 1992):

- spatial separation
- objekts integrerbarhet
- uppgiftens strukturella likhet
- uppgiftens resurskrav

***3.3.3.3** Spatialseparation*

Människans synfält är begränsat. Av den anledningen är ofta viktiga instrument koncentrerade fysiskt nära varandra för att öka den parallella informationsinhämtningen. Kontrollers närhet är dock inte alltid en garanti för att parallell informationsinhämtning sker på ett bra sätt. Kontrollers närhet kan även orsaka problem med fokusering av uppmärksamheten. Problem kan komma att uppstå när operatören endast vill ta in information från en enda informationskälla. Om många olika informationskällor ligger för nära varandra kan problem relaterade till både perceptions- och gensvarskonflikter uppstå. (Wickens, 1992)

Perceptionskonflikt relaterar till vad som kan kallas för: *virrvarr*. Det handlar både om svårigheten med att lokalisera en specifik informationskanal som kanske ligger tätt intill andra liknande informationskällor, och resurskonflikt i det visuella systemet på grund av att de andra närliggande informationskanalerna tar upp resurser. Kort sagt: man ser inte träden för hela skogen. (Wickens, 1992)

Gensvarskonflikt framkallas från automatsik behandling av närliggande irrelevant information. En gensvarskonflikt uppstår när närliggande icke-relevanta informationskällor framhäver ett gensvar som motsäger det gensvar som den relevanta informationskällan pekar mot. Ett exempel där vi ombes fundera på det gensvar som ges

på ordet ***upp***. Men om på båda sidorna av ordet upp läggs pilar som pekar nedåt, dessa pilars missledande information kommer att göra gensvarstiden längre. (Wickens, 1992)

Nära visuell spatialseparation kallas för ett tvåeggat svärd. Om separationen minskas kan parallell informationsbehandling bli uppmuntrad, men detta kan dock leda till distraktion. (Wickens, 1992)

3.3.3.4 Objekts integrerbarhet

Det finns en del bevis för att det är möjligt att öka den parallella informationsbehandlingen om den informationskälla som användes har två eller fler stimuli eller attribut. Det har blivit påvisat att om en person fokuserar sin uppmärksamhet på ett attribut hos ett objekt, till exempel dess färg, kommer hon även att parallellt inhämta dess storlek, form och orientering. (Wickens, 1992)

Vid ett experiment utfört av Lappin (1967), som Wickens (1992) nämner i sin bok, lät han ett visst antal försökspersoner rapportera tre olika attribut från en mycket kort exponering till objekten. När de tre attributen var färgen, storleken och formen av ett enda objekt av en viss geometrisk form blev rapporten mycket mer korrekt än när försökspersonerna ombads rapportera storleken av ett objekt, färgen av ett annat och ett tredje objekts form.

Detta har störst betydelse för statisk symbolik där två eller fler olika data, som alla berör samma element, visas (Till exempel, storleken, politisk åskådning och medelålder av medborgare på en politisk karta). Den här objektmodellen verkar vara mycket användbar för att visa integrerad information. (Wickens, 1992)

Men studier visar att den integrerade objektmodellen inte alltid är den bästa. Vid ett experiment vid ett kärnkraftverk där tränade operatörer fick avgöra om någon parameter hade ett felaktigt värde visade sig den integrerade objektmodellen vara bättre. Men när operatörerna skulle avgöra hur många parametrar och vilka som var ur balans var det displaysystem som de hade haft från början som var bättre. Det displaysystem som de hade haft från början visade de olika parametrarna separat och inte som ett integrerat objekt.

Detta pekar på att den integrerade objektmodellen möjliggör lättare upptäckt av abnormaliteter. Eftersom den låter operatören integrera samtliga data till ett mönster, där hon lätt kan avgöra om mönstret antar ett normalt eller ett onormalt utseende. Medan en separat visning av parametrarna är mer lämplig när det behövs specifikt identifiera abnormaliteter och där varje parameter som har antagit ett felaktigt värde behöver behandlas separat. (Wickens, 1992)

3.3.3.5 Strukturell likhet och resurshantering

Det är lättare att utföra flera parallella uppgifter om uppgifternas art skiljer sig från varandra än om de är alltför lika. Som exempel kan tas att det är mycket svårare att lyssna på två olika samtal samtidigt än att lyssna på ett samtal och köra bil samtidigt. Det är ett väl etablerat faktum att en serie objekt som ser lika ut, har ett liknande ljud eller har liknande innebörd ofta kommer att blandas ihop när dessa skall bli ihågkomna.

Minnet kommer oundvikligt att bli påverkat vid inlärning av ett visst material om en person utsätts för liknande material precis före och efter det att hon har lärt sig det material som hon skall lära sig. (Wickens, 1992)

Det kan kort sägas om resurshantering att ju komplexare varje uppgift är desto svårare är det att parallellt utföra flera uppgifter åt gången. (Wickens, 1992)

3.3.4 Kognition

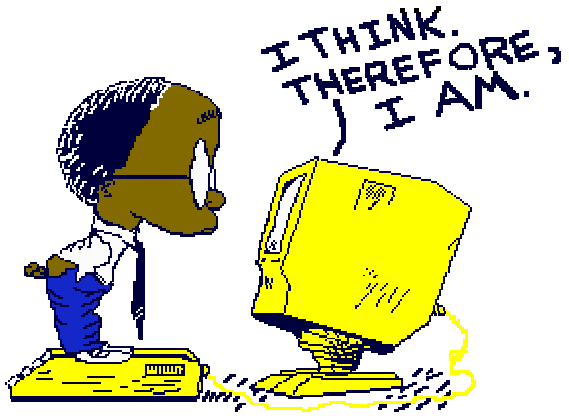
Det var den franske renässansfilosofen Descartes (Gaarder, 1995) som med sin första tes, *cogito ergo sum - jag tänker, alltså är jag*, använde sitt mänskliga tänkande för att bevisa sin egen existens. Användarens tanke-processer antar en avgörande roll vad gäller datorinteraktion. Det beteende och de reaktioner som hon uppvisar på olika förutsättningar i ett datorprogram har till mycket stor del att göra med individens förförståelse av programmet i synnerhet och datorer i allmänhet. (Allwood, 1991)

3.3.4.1 Förförståelse

Allwood (1991) skriver att en avgörande faktor för individens, det vill säga användarens, tänkande är den förförståelse som hon har med sig till varje ny situation. Med förförståelse menar Allwood summan av de bakgrundserfarenheter som användaren har med sig. Förförståelse är ett omfattande begrepp som inkluderar både kognitiva, motivationella och emotionella delar.

De kognitiva delarna behandlar individens förståelse av verkligheten och individens olika färdigheter, vilket ofta betecknas som dennes *förkunskaper*. Dessa förkunskaper motsvara vad som i ett föregående kapitel (3.3.2.1 Minnets dimensioner) betecknades som *deklarativa* och *procedurella* kunskaper. Allwood (1991) fortsätter med att säga att helt uppenbart är inte all förståelse av olika begrepp eller färdigheter över huvudtaget relevanta i datorsammanhang. Dock kan ibland förkunskaper som inte alls ansetts vara av betydelse visa sig påverka hur en användare beter sig vid datorinteraktion.

Vad gäller de motivationella och emotionella delarna berör de hur användaren förhåller sig till situationen och dennes känslomässiga förväntningar. Det handlar om huruvida användaren känner sig engagerad och villig att anstränga sig. Om användaren känner att det skall bli stimulerande och roligt att ta sig an uppgiften. (Allwood, 1991)



Jag tänker, alltså är jag

3.3.4.2 Allmänna och

specifika mentala strategier

Allwood (1991) skriver att uppgiftslösandet tar karaktären av problemlösning när en individ har ofullständiga kunskaper inom det område som hon ger sig i kast med. En individ blir tvungen att tillgripa och förlita sig till *allmänna* strategier i sitt tänkande om hon inte innehar tillräcklig kunskap om antigen uppgiftsområdet eller det aktuella datorsystemet. Allmänna strategier utmärks av att de inte kräver några speciella förkunskaper inom det aktuella området men att de är tillämpbara i många olika situationer. De allmänna strategierna är dock inte särskilt effektiva vad gäller att ta individen närmare målet, med andra ord är de inte särskilt kraftfulla. (Allwood, 1991)

Exempel på allmänna strategier som Allwood (1991) tar upp är att ordentligt klargöra vad målet är med den uppgift som skall utföras. Ett annat exempel på en allmän strategi är att försöka dela upp uppgiften i mindre och enklare delmål som när de är lösta innebär att hela uppgiften också är löst.

Allwood (1991) fortsätter med att skriva att strategierna förändras ju mer kunskap som individen ackumulerar inom det aktuella användningsområdet. De strategier som individen använder i sitt tänkande kommer att bli mer och mer *specifika* desto mer områdeskunskaper som individen samlar på sig. En specifik strategi kan på ett mer effektivt sätt föra individen närmare det utstakade målet och därmed lösa uppgiften. Eftersom de specifika strategierna är mindre kognitivt ansträngande kommer en individ att med all rimlighet hellre begagna sig av dessa snarare än de *allmänna* strategierna vid sitt problemlösande i den givna situationen.

Att använda *specifika* strategier innebär att det är möjligt att utnyttja kunskaper inom det område som uppgiften ligger i, kunskaper som allmänna strategier inte kan använda sig av. Ett enkelt sätt som Allwood (1991) belyser skillnaden på allmänna och specifika strategier är genom att studera de olika sätt att lösa multiplikationen 6*5. En lösning genom en allmän strategi är att addera fem, sex gånger. Medan en lösning genom en specifik strategi är att hämta svaret direkt från långtidsminnet.

3.3.4.3 Analogiskt tänkande

Allwood (1991) påpekar att en ytterst viktig egenskap hos tänkandet är att det är analogiskt. Detta innebär att hur individen tänker är i hög grad beroende av dennes förkunskaper. Utgångspunkten för individen blir hela tiden vad denne redan vet och kan. Att utnyttja redan befintliga kunskaper inom ett område på ett annat område kallas analogiskt tänkande. Detta åstadkoms genom att individen ser ett objekt i termer av ett annat. Exempelvis när datorn betraktas som en skrivmaskin. Datorn i exemplet benämns som *tillämpningsdomän* medan skrivmaskinen benämns som *ursprungsdomän*. Och när färdigheter från ett område tillämpas på ett annat område talas det om *transfer*.

Allwood (1991) resonerar vidare hur analogier kan skilja sig åt i ett strukturellt avseende.

- Analogier kan vara olika noggrant representerad hos individen. Olika individer har olika kunskaper om, till exempel, skrivmaskiner.

- Antal egenskaper som matchar mellan ursprungsdomän och tillämpningsdomän kan variera mycket. Exempelvis liknar skrivmaskiner datorer mer än vad bilar liknar datorer.
- Tydligheten av vilka egenskaper från ursprungsdomänen som matcher vilka egenskaper från tillämpningsdomänen kan variera. Likheten mellan två datorer från olika fabriker är tydligare än likheten mellan en dator och en skrivmaskin.

Allwood (1991) skriver därefter att för att närma sig *användarens psykologiska verklighe*t är det inte bra att tala om de aktuella analogiernas egenskaper i sig. Det är istället bättre att tala om *hur* ursprungs- och tillämpningsdomänerna antas vara representerade hos användaren.

3.3.4.4 Metakognition

Enligt Allwood (1991) handlar metakognition om individens kunskap om sin egen och andras mentala processer. Det gäller individens förståelse av sina egna kunskaper och färdigheter, styrkor och begränsningar. I vilken utsträckning individen förstår vad som krävs av denne för att kunna lösa ett visst problem. Förmågan att anpassa sina kognitiva förmågor till uppgiftens krav.

3.3.4.5 Kognitiv lättja

För de flesta människor är aktivt tänkande det samma som att försöka undvika kognitiv ansträngning. Med kognitiv ansträngning menas ett tänkande som innebär att många premisser tas i beaktande samtidigt vilket i sin tur kräver långvarig koncentration och uppmärksamhet i kombination med systematiska sökningar i minnet. En tendens för kognitiv lättja framträder oftast när motivationen att lösa uppgiften eller att nå målet är låg. Människor prioriterar dessutom oftast att nå målet högre än att förbättra sin förståelse av verkligheten. (Allwood, 1991)

3.3.4.6 Mentala modeller

Enligt Allwood (1991) anses en viktig egenskap hos tänkandet vara att det är *aktivt*. Som ett resultat av det aktiva tänkandet skapar individen hypoteser eller *konceptuella modeller* (mentala modeller) om verkligheten. Dessa hypoteser stäms sedan av mot verkligheten per automatik med hjälp av de erfarenheter som individen under sin livstid samlar på sig. Det här leder till att individen kontinuerligt bygger upp en mer eller mindre rättvisande bild av sig själv och sin omgivning.

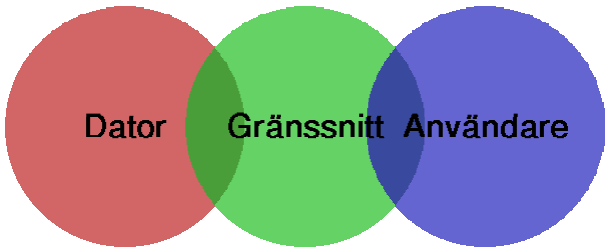
När en person sätts i en ny situation försöker hon ofta tolka den genom att jämföra den, ofta omedvetet, mot en befintlig modell. (IBM, 1992) Genom att ta denna konceptuella modell till hjälp kan användaren förutsäga ett systemets uppträdande utan att behöva memorera några abstrakta och godtyckliga regler (Lynch, 1994). Eftersom vi ofta lär oss genom att experimentera underlättar inte bara användandet utan även vidare inlärning. (Cox & Walker, 1993)

Dessa hypoteser och modeller är dock inte alltid rättvisande eftersom människor tenderar att vara överdrivet spekulativa och konstruera förklaringar med utgångspunkt

från väldigt tunn information, vilket svar som helst är bättre än inget alls. Dessutom är det vanligt att människor gör systematiska fel vid sina bedömningar. Vid uppskattning av sannolikhet för att en viss händelse skall inträffa tenderar människor att tilldöma en viss typ av information alltför stor vikt jämfört med andra informationstyper. Sådan information kan vara att man påminner sig ett konkret exempel av företeelsen i fråga. (Allwood, 1991)

Några vanliga tankefel enligt Allwood (1991):

- *Tillgänglighetsfel* – Om individen kan erinra sig ett konkret exempel på företeelsen som det är frågan om överskattas sannolikheten för att händelsen skall inträffa.
- *Förankringsfel* – Individer tenderar att haka upp sig på hur situationen är nu och hur det brukar vara. Ett exempel på detta är händelsen den 11 september 2001 i New York. Man överrumplades av händelseutvecklingen eftersom man inte trodde att det skulle kunna hända.
- *Funktionell fixering* – Individen tar för givet att ett objekt endast kan användas på det sätt som det är brukligt att använda det på. Exempelvis utfördes det ett försök där deltagarna hade svårt att förstå att en elektrisk krets kunde användas som en vikt. Särskilt om försökspersonen var ingenjör eller nyss hade använt den som elektrisk krets.
- *Anpassningsbarhet* – Om individen precis har löst ett problem på ett visst sätt och att denne sedan löser nästa uppgift med samma metod även om det finns ett bättre och effektivare sätt.



Gränssnittsmodellen överbryggar gapet mellan implementationsmodellen och användarens konceptuella modell

En användarens konceptuella modell av hur ett system fungerar stämmer inte alltid överens med hur det faktiskt fungerar, dess *imple-mentationsmodell*. Det är här *gränssnittsmodellen* i bilden bredvid kommer in och översätter, eller tolkar, mellan användarens modell och implementationsmodellen. (Cooper, 1995)

Analogier, som nämnts ovan, används ofta för att kommunicera den gränssnittsmodell ett program uppvisar. Genom att välja lämpliga visuella och funktionella analogier förväntas användaren använda sin kunskap från källdomänen till den okända måldomänen, systemets funktionalitet. (Lund, 1997) Dessa gränssnittsanalogier underlättar förståelsen genom upplevelser och reaktioner: det är inte fråga om att memorera kommandon utan att reagera på en rik uppsättning information som användargränssnittet presenterar. (Lynch, 1994)

Flera har observerat att den direkta manipuleringen liknar *the willing suspension of disbelief*, att man *villigt glömmer tvivlen*. Detta uttryck myntades av Samuel Coleridges för att beskriva åskådares intensiva inlevelse i teaterpjäser. På ett liknande sätt kan ett välutformat gränssnitt, som erbjuder ett konsistent och förutsägbart uppträdande, göra att användaren glömmer bort skillnaderna och börjar använda objekten som om de var verkliga. (Lynch, 1994)

Nämnas bör att gränssnittsanalogier ibland kallas *gränssnittsmetaforer*. Cox och Walker (1993) anser dock att termen metafor är missledande eftersom det aldrig kan röra sig om en total mappning mellan käll- och måldomänen utan det bör istället kallas partiella metaforer. En partiell metafor är samma sak som en analogi, varför den termen är lämplig att använda.

Enligt Lund (1997) kan man säga att vårt tänkande och vår förståelse i hög grad bygger på metaforer, liknande konceptuella modeller enligt ovan. Nya koncept skapas genom en evolutionär process där gamla koncept metaforiskt projiceras på nya situationer. Lund (1997) refererar till Georg Lakoff och Mark Johnson (1980) som menar att koncept som används dagligen är metaforiska; inte bara att vi medvetet talar om saker i termer av andra, vi tänker även metaforiskt. Våra vardagliga upplevelser struktureras enligt olika slags metaforer:

- **Strukturella metaforer**
  Ett koncept struktureras enligt ett annat koncept. Till exempel kan diskussioner eller gräl struktureras som krig.
- **Orienteriska metaforer**
  Våra upplevelser struktureras i termer av rumslig orientering. Är man till exempel ledsen är man "nere".
- **Ontologiska metaforer**
  Abstrakta koncept, som inflation eller kunskap, struktureras som om de vore fysiska objekt.

Alla koncept kan emellertid inte struktureras enligt andra. Det finnas urkoncept som inte är metaforiska. Enligt Lakoff och Johnson (1980) är dessa hur vår fysiska omgivning upplevs. Våra kroppar ser ut på ett givet sätt, gravitationen påverkar oss etc. Dessa livsförhållanden påverkar i mycket stor utsträckning hur vi upplever, tolkar och finner mening i vår omgivande värld i allmänhet. Lakoff och Johnson kallar dessa grundläggande strukturer *bildscheman* (image schemata). Några av de bildscheman Lakoff (1987) tar upp är:

- **Behållare**
  Detta liknar grundläggande mängdlära: något är innanför eller utanför en gräns; det kan inte både vara innanför och utanför samtidigt.
- **Centrum/periferi**
  Man har ett centralt objekt och andra objekt är nära detta centrum eller längre bort, ute i periferin.

- **Vertikalitet**
  Metaforen, *en person som är ledsen är **nere***, bygger på detta schema. Även koncept som kvantitet och kvalitet bygger på detta. Högre är mer och bättre.

Lund (1997) tar upp ett exempel på hur gränssnitt kan utformas enligt dessa teorier. Exemplet kallas SchemaSpace, ett system för att organisera bokmärken, till exempel i en webbläsare. Bokmärken grupperas i staplar som i sin tur placeras i koncentriska cirklar i koner. Viktigare staplar placeras centralare i konen och konernas höjd markerar antal bokmärken i dem. För att kunna läsa bokmärkena måste man befinna sig inne i konen. SchemaSpace har stora likheter med William Gibsons virtuella värld *cyberspace* i boken Neuromancer (Gibson, 1984), där olika informationsdatabaser representeras av objekt som exempelvis rektanglar och pyramider. För att kunna nå informationen måste man gå in i objektet i fråga och objektens storlek avspeglar mängden data i dem.

En annan mental modell är *idiom* som är fasta uttryck vars innebörd inte framgår av de ingående ordens betydelser (Nationalencyklopediens Ordbok 1995, band 2). Det är i stort sett obegripligt den första gången det påträffas. Dock räcker det ofta att få det förklarat för sig någon enda gång för att dess betydelse skall vara klart förståelig. Det kan även i datorsammanhang talas om idiom, men i detta sammanhang rör det sig ofta om visuella eller funktionella idiom. Ett exempel på ett *datoridiom* är en rullningslist eller en mus. Till att börja med behöver dess funktion inte vara uppenbar, men när den väl är inlärd, något som ofta går väldigt fort, är den ett mycket användbart verktyg.

3.3.5 Perception

Enligt May, Scott och Barnard (1995) är perception en aktiv process. De menar att vår visuella omgivning är uppbyggd av objekt snarare än egenskaper eller egenheter. När vi ser oss omkring i ett rum ser vi olika objekt, till exempel några böcker på ett skrivbord. Trots att det som når våra ögon är ett mönster med olika färgnyanser och skuggor av en viss area som vi riktar vår uppmärksamhet mot är det inte på det sättet som vi *ser* arean. Perception är den process som strukturerar den visuella information som vi mottar från vår omgivning på ett sådant sätt att vi kan interagera med den. För att kunna göra detta menar May et al. (1995) vidare att vi behöver kunna se en uppsättning av olikfärgade plan och ytor som tillhörande samma objekt, till exempel en bok, avskilda från andra plan och ytor som i sin tur representerar skrivbordet och de andra böckerna. Om vi skulle plocka upp boken förväntar vi oss att alla visuella delar som vi förknippar med den eller som *tillhör* den att de kommer att röra sig tillsammans på ett förutsägbart sätt medan vi också förväntar oss att alla delar som tillhör skrivbordet kommer att stanna kvar och förbli opåverkade. Om vi däremot skulle plocka upp en hög med böcker vet vi att de individuella böckerna inte nödvändigtvis kommer att förbli en *hög* och att högen med böcker inte kan behandlas på samma sätt som de böcker som den är uppbyggd av. (May et al., 1995)

May et al. (1995) fortsätter med att skriva att den visuella informationen inte explicit innehåller beskrivningar av olika objekts struktur. Strukturerna måste tolkas fram genom att kombinera den visuella informationen med den kunskap om omvärlden som vi innehar, kunskap om *omvärlden* som vi har tillägnat oss genom livslång interaktion med den. Det är därför som May et al. (1995) argumenterar att perception är en aktiv

process, eftersom olika intryck vi mottar genom våra sinnen blandas med kunskap och erfarenhet. Våra interaktioner med omvärlden påverkas av hur vi uppfattar dess strukturer genom våra sinnen, och våra uttolkningar av dessa strukturer påverkar i sin tur hur vi interagerar med omvärlden.

Vidare menar May et al. (1995) att när vi tar in en visuell scen, vare sig det är fråga om en 2- eller en 3-dimensionell scen, formar den visuella informationen, innehållande färger, former och texturer, olika objekt. Hela den *scen* som vi intar den visuella informationen ifrån är en stor struktur av olika objekt. De olika objekten har olika egenskaper, de separerar sig från bakgrunden och är distinkta enheter som ofta har namn. Om man tar en närmare titt på ett objekt är det möjligt att urskilja att det också har en struktur. Det kan vara uppbyggt av flera mindre objekt som i sin tur kan vara uppbyggda av ännu fler och mindre objekt. Världen kan uppfattas genom flera nivåer, från en global nivå ner genom många olika detaljnivåer.

Som exempel tas bilden som visar hur du kan stå i dörren till ett kontorsrum och se ett *kontor* - ett rum med olika objekt i det. Du kan sedan fokusera din uppmärksamhet på den bortre väggen som är en platt yta med möbler framför. På den detaljnivån kan du urskilja ett fönster, en stol och ett arbetsbord. På arbetsbordet kan du urskilja ett pappersblock. Blocket har en penna som ligger på det och det finns dessutom skrift på den sida som är uppslagen. Du kan urskilja skriften genom att vandra genom blockets struktur, och genom att vandra ytterligare en nivå nedåt i dess struktur kan även de enskilda orden urskiljas (om man stod tillräckligt nära). (May et al., 1995)

Det finns ett antal enkla sätt för att gruppera objekt på en tvådimensionell datorskärm:

Ej grupperade

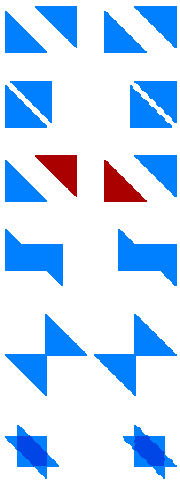
Närhet De ligger nära varandra.

Färg De har samma färg.

Gräns De ligger kant i kant.

Knutpunkt De har en gemensam punkt.

Överlappning De ligger ovanpå varandra.



I vissa fall kan grupper av objekt vara tydligare än enskilda objekt på en skärm. Om man vill att ett objekt inte skall ingå i en gruppering kan man ändra på en eller flera av ovanstående variabler. Detta kan vara användbart om man vill markera ett visst objekt för att på så sätt få användaren att snabbt lägga märke till det. (May et al., 1995)

Redan de gamla grekerna hade tankar som påminner om denna perceptuella process, särkilt Platon och hans idé om den perfekta *idévärlden*. Han menade att vår värld endast är en blek skugga av idévärlden och ett objekt som vi ser är endast en skugga av en idé. Idéerna fungerar som mallar eller mönster när vi ser ett objekt. Den mall som bäst

passar in på ett objekt förknippas med objektet i fråga. Ser vi något som ser ut som en häst, tror vi att det är en häst. Och vi förknippar det med vissa egenskaper som att det är möjligt att rida på den. (Var kanske Platon den förste objektorienteraren?) (Gaarder, 1995)

4 Gränssnittsrekommendationer

Det är inta alla nedanstående rekommendationer som har blivit härledda ur ovanstående teori utan i vissa fall kommer de från egna erfarenheter av allmän datorinteraktion. Rekommendationerna är främst tänkta att rikta sig mot design av gränssnitt för små skärmar och mobila klienter. Dessutom har det hela tiden förekommit en strävan att basera samtliga rekommendationer i teorin.

Vidare kan inledningsvis sägas att trots att de grafiska användargränssnitten med dess direkta manipulation har existerat i mer än 20 år finns det fortfarande en del brister. Den största skillnaden mellan dagens datorsystem och de system som togs fram på Xerox PARC är att dagens datorer har högre upplösning och fler färger. Det finns de som därmed hävdar att de grafiska användargränssnitten nästan är för bra och att de därmed har hindrat den fortsatta utvecklingen av gränssnitt. Trots att datorkraften har ökat hundrafalt har inte interaktionssättet på ett nämnvärt sätt förändrats. (Holmqvist, 1997) Pek-och-klicka gränssnitt anses även vara alltför begränsande. Eftersom det enbart tillåter enkel och mycket primitiv kommunikation som enbart rör objekt i vår omedelbara närhet. I vår interaktion har vi förlorat språkets kraftfullhet och vi kommunicerar med datorn på samma primitiva nivå som vi skulle göra om vi åkte till ett annat land utan att kunna språket. Vi skulle även då vara reducerade till att peka och göra någon enkel gest i hopp om att någon skall förstå. Gränssnitten klarar inte den typen av abstrakt interaktion som ett påstående som detta innebär: *meddela mig när det kommer ett nytt meddelande från Astrid*. (Genter & Nielsen, 1996)

Vidare är det även intressant att notera att den teori som har framlagts i föregående kapitel innehåller många bitar som inte har blir applicerade på gränssnitt i detta kapitel. Det finns därmed gott om utrymme för att gräva fram mer.

Vad som också skulle vara intressant vore att ytterligare konkretisera dessa rekommendationer och formalisera dem till en mera specifik utvärderings- och utvecklingsmetod för användargränssnitt.

4.1 Definition: små och mobila klienter

Med uttrycket *små och mobila klienter* menas datorer som har dimensioner som gör dem portabla. Dock inkluderas inte så kallade *portföljdatorer* utan fokus ligger på handhållna datorer som Palm, PocketPC-maskiner som iPaq, Linux-baserade maskiner som Zaurus och YOPY men även mobiltelefoner ingår. Dock ligger inte fokus på just mobiltelefoner utan mer på de lite större och handhållna datorerna, vilka är lite mer anpassade för att kunna köra *vanliga* applikationer.

Bilden till höger föreställer Sharps kommande handhållna dator vid namn *Zaurus*. (Sharp, 2001)

4.2 Stöd för arbetsminnet

Vital information i ett fönster eller skärmbild som användaren behöver för vidare arbete bör finnas med i nästa fönster för att informationen inte skall glömmas bort, då information i arbetsminnet tenderar att försvagas eller försvinna när nya objekt aktiveras. (Allwood, 1991) För att användaren snabbt skall kunna hitta informationen i det nya fönstret eller skärmbilden bör den finnas på ungefär samma ställe på skärmen för att användaren inte skall behöva aktivt söka efter informationen. (May et al, 1995). Dock är det bäst om det är möjligt att inte byta fönster eller skärmbild alls. (Cooper, 1995)

Programmet bör fylla i lämpliga förvalda värden i dialogfönster och liknande. Dessa förvalda värden kan gärna vara de senast använda värdena eftersom det ofta är troligt att användaren vill ha liknande värden nästa gång. (Cooper, 1995) De värden som fylls i kan även baseras på statistik, de vanligast förekommande värdena eller något liknande. Men det kräver mer av datorn och programmet och är inte lika enkelt som att bara ta de senast använda värdena. Ett annat sätt att göra det på är att spara flera värden. Och det även mellan programkörningar, men det bör vara enkelt att återgå till några inbyggda standardvärden så att användaren inte behöver dras med en massa värden i listor som aldrig eller väldigt sällan kommer att användas igen. Om det finns några mindre antal tänkbara alternativ, särskilt om de olika alternativen är ungefär lika sannolika att inträffa och om det är logiskt att göra valet i samband med att funktionen aktiveras, då kan det vara bra med olika funktioner som öppnar samma fönster men med de olika alternativen olika ifyllda beroende på önskemål. Detta för att användaren inte skall behöva komma ihåg att ange rätt information, något som annars lätt glöms bort och kan upplevas som irriterande att behöva göra om och om igen. Detta görs med någon form av användarstyrda *mallar* för formulären.

Undvik att distrahera användaren, till exempel med dialogfönster som öppnas i tid och otid. (Cooper, 1995)

4.3 Stöd för långtidsminnet

Hjälpfunktioner är viktiga för att hjälpa minnet, för att användaren skall kunna utföra en given uppgift även om denne skulle sakna en viss kunskap eller ha glömt bort hur en viss uppgift utförs. Dessa funktioner bör med fördel anta olika former samt vara lätt tillgängliga. En del vill ha beskrivningar steg för steg medan andra hellre vill ha information åt det mer referensbetonade hållet. Det bör vara möjligt att på en global nivå ställa in så att önskad form av hjälpfunktionerna visas. Det bör dock vara relativt enkelt att byta utformningen av hjälpen om det skulle vara så att den andra formen önskas. Allwood (1991) skriver att ett sätt att göra detta är att använda sig av användarprofiler som innehåller användarens kunskapsnivå, uppgiftsmål och övriga egenskaper. Det här kan utföras på olika sätt:

- **Direkt**
  Användaren gör en beskrivning av sig själv genom att svara på ett antal frågor. Risken finns dock att användaren innehar otillräcklig självkännedom och kan därför inte avge tillräckliga svar.

- **Stereotyp**
  Användarna klassificeras antigen statiskt eller dynamiskt i olika kategorier, stereotyper. Kategorierna och de olika egenskaperna som finns är dock ofta ganska få vilket ofta leder till att hjälpfunktionen ändå inte anpassas särskilt bra till den enskilde användarens behov.

- **Intentioner**
  Programmet försöker dynamiskt anpassa sig efter de avsikter och mål som användaren har. Dock kräver detta relativt mycket intelligens i programmet vilket har medfört att den här tekniken inte används i så stor utsträckning i dagsläget.

Om programmet ibland interagerar med användaren för att bekräfta sina slutsatser angående den profil som programmet har skapat kallas det för en interaktiv lösning och baseras i en av ovanstående metoder.

Vid utformning av hjälpfunktionerna kan det vara bra att komma ihåg att det blir lättare för användaren att verkligen komma ihåg hur uppgiften utförs, även efter att denne har läst instruktionerna, om läromiljön påminner om den verkliga miljön. Det betyder att hjälpfunktioner inte enbart bör erbjuda en lista med instruktioner, steg för steg, utan det bör vara möjligt att användaren åtminstone får se hur det verkligen ser ut. Ännu bättre är om hjälpfunktionerna erbjuder användaren att pröva eller simulera funktionerna, en så kallad *tutorial*. Det här är en teknik som är mycket utbredd bland datorspel där spelaren får lära sig hur spelets kontroller sköts genom att få pröva samtidigt som denne får instruktioner av en röst samt kan läsa dem. Spelaren får inte bara se hur det skall se ut och hur det går till, inte heller får denne enbart en lista med instruktioner, utan spelaren får möjlighet att *uppleva* hur spelet sköts. När spelaren väl börjar spela på allvar behöver denne enbart *känna igen sig* och inte exakt komma ihåg en lista.

Ett lämpligt sätt för att i onödan undvika att belasta minnet är att följa standarder och rent allmänt se till att programmet fungerar på ett enkelt och förutsägbart sätt. Användaren skall med andra ord inte behöva komma ihåg från program till program hur liknande operationer fungerar. Om standarder ibland frångås bör det då vara mycket väl motiverat, till exempel att det blir betydligt enklare eller smidigare att utföra en viss uppgift. Det kan vid sådana tillfällen vara bra om denna skillnad eller avvikelse på något sätt markeras på ett bra sätt.

4.4 Fokusering av uppmärksamhet

Upp-poppande dialogfönster bör användas sparsamt eftersom de tvingar användaren att fokusera om mentalt. Öppna inte ett dialogfönster bara för att informera om att något lyckades. Reservera dessa istället för allvarliga fel eller kritiska händelser som kräver användarens omedelbara eller odelade uppmärksamhet och som programmet inte rimligtvis kan förväntas hantera på egen hand. Det är inte heller alltför lämpligt att använda dialogfönster för att bekräfta val eftersom användaren oftast kommer att bekräfta dem och snart kommer de att bekräftas av gammal vana även när denne inte hade tänkt göra det. Syftet med dialogfönstret har på detta sätt gått förlorad. Det är

mycket bättre att utföra operationen utan frågor men ge användaren möjlighet att vid ett senare tillfälle modifiera eller ångra sitt beslut. (Cooper, 1995)

Öppna inte dialogfönster i onödan i de fall när programmet inte är det program som användaren just nu använder, om denne har växlat till ett annat program och använder det istället. Dialogfönstret kommer att störa detta arbete och om det dessutom reagerar på tangenttryckningar kan det få förödande konsekvenser om användaren just håller på att skriva något. Ett lämpligt sätt att indikera att interaktion med programmet behövs är att visa detta på en statusrad ute i periferin. Om det skulle gälla något viktigt kan indikationen försöka dra till sig uppmärksamhet genom att anta annorlunda färg, vara animerad, använda ljud eller kanske, om så är möjligt, känselsignaler som exempelvis vibration.

Felmeddelanden bör inte placera skulden på användaren. Texten bör vara neutral och saklig. Dialogfönstret bör låta användaren åtgärda felet och få mer information om det och inte bara visa någon kryptisk fel-kod. (IBM, 1992)

Det som kanske mest skiljer de små handhållna datorerna från de stationära, förutom storleken, är den miljö som de används i. Stationära datorer används i bekväma, förhoppningsvis, kontorslokaler eller i hemmets lugna vrå. Medan de handhållna, små och mobila klienterna kan exempelvis användas i morgonrusningstrafiken, i en stojig park, vid övergångsstället eller femton meter från åtta panflöjtspelande peruaner etc. Uppmärksamheten på den uppgift som skall utföras vid användandet av stationära datorer kanske mest tävlar med sällskapssjuka medarbetare eller med en bra låt på radion. Miljön vid användning av en handhållen, liten och mobil klient kan vara mycket mer distraherande. Dock skall nämnas att användningsområdena mellan stationära och handhållna datorer varierar, det är ingen vid sina sinnes fulla bruk som skulle få för sig att skriva sin nya bästsäljare på en liten Palm-dator. I dagsläget är inte de handhållna datorerna tänkta att användas i samma långa tidsintervall som de stationära. Det vill säga, när den används kommer det *oftast* att ske under kortare tidsperioder som därmed kräver mindre uppmärksamhet.

Något att tänka på vid utformning av applikationer för handhållna datorer är vid åkallande av uppmärksamhet. På grund av att den handhållna datorn kan befinna sig i en ficka eller en väska räcker det kanske inte med att den bara börja blinka lite i ena hörnet på skärmen för att åkalla användarens uppmärksamhet. Det beror i högsta grad på hur viktigt det är att användarens uppmärksamhet vänds mot handdatorn och applikationen. Är det inte alls viktigt kan det faktisk räcka med att ett objekt i kanten blinkar när användaren väl får för sig att använda handdatorn, kanske av någon annan anledning. Är det mer brådskande kan till exempel en ljudsignal eller kanske någon form av vibrator användas. Det måste givetvis vara möjligt för användaren självt att kunna styra och ställa in om denne över huvud taget vill kunna bli störd och i så fall på vilket sätt.

Det skulle även vara möjligt att ställa in olika grader av brådska beroende på vilken signal som används eller om handdatorn vibrerar. Normalt har känselsignaler högre prioritet än hörselsignaler men eftersom en vibrator signal kan vara ganska svag, obemärkt och diskret, om användaren inte har maskinen i exempelvis handen, är kanske ljudsignaler fortfarande mer *alarmerande*. (Wickens, 1992)

4.5 Avlasta tänkandet

Utforma programmet med den tänkta målgruppen i åtanke och ta hänsyn till användarnas förväntade förkunskaper. Olika användare har förvisso olika förkunskaper, men det kan man ta hänsyn till i dokumentationen. Det bör då vara lätt att hoppa över delar av dokumentationen man redan kan.

Både generella och specifika strategier skall kunna användas för att använda programmet och lösa olika problem. De generella strategierna bör vara av en mer informativ karaktär, för att användaren efter hand skall kunna lära sig mer om programmet och därmed även så småningom kunna begagna sig av mer specifika strategier. Inriktningen av hjälpfunktionen bör vara att ge användaren specifika strategier för att lösa olika uppgifter. Om det finns alternativa strategier kan är det lämpligt att även dessa omnämns för att användaren skall kunna välja den strategi som passar denne bäst.

Utforma programmet så att användaren kan uppfylla sina mål, både de direkt uppgiftsrelaterade, men även de mer generella, ofta personliga, målen. (Cooper, 1995)

För att användaren inte skall dra felaktiga slutsatser bör dokumentationen och programmet vara tydliga. Undvik även onödig eller redundant information som kan verka distraherande eller förvirrande.

Undvik att användaren skall kunna göra fel. Om en funktion inte skall vara tillgänglig då skall den inte heller gå att aktivera bara för att resultera i ett felmeddelande. Med andra ord bör programmet undvika *fällor* för användaren. Programmet bör kunna känna igen situationer när vissa funktioner inte skall gå att aktivera.

4.5.1 Mentala modeller

Olika användare har olika mentala modeller, detta kommer sig mycket beroende på olika erfarenhet och förkunskap. Dessutom är det svårt att beskriva dessa *modeller*, särskilt med tanke på att användaren i sig inte är medveten om dem. Ansvaret att skapa sig en uppfattning om användarens modeller för att sedan nyttja sig av en lämplig sådan för användargränssnittet vilar på konstruktören av programmets användargränssnitt. Att prata med användarna, analysera och studera det språk som används är sätt genom vilket det är möjligt att bilda sig en uppfattning om användarnas mentala modeller. (IBM, 1992) För att det skall vara lättare för användarna att ta till sig och förstå modellerna bör dessa ligga nära användarna och vara enkla. För vanligt förekommande uppgifter är det därmed lämpligt att använda allmänt accepterade modeller. Om olika program använder olika modeller för liknande eller rentav samma funktioner belastas minnet eftersom användaren måste komma ihåg vilket program som använder vilken modell, detta kan leda till frustration.

Tre klassiska problem med analogier och tillämpning av dem är (Gentner & Nielsen, 1996):

1. Måldomänen har funktioner som inte finns i källdomänen.
2. Källdomänen har funktioner som inte finns i måldomänen.

3. Funktioner som finns i båda domänerna fungerar olika.

Det finns en risk att programmet blir begränsat om det förekommer en strävan att följa analogin alltför exakt. I tillägg till detta kan datormiljön ytterligare lägga till begränsningar och göra datorversionen sämre än originalet. Som med exemplet med att jämföra en dator med en skrivmaskin, om datorn bara ses som en skrivmaskin medför detta tydliga begränsningar för vad som kan göras med datorn. Det handlar om förmågan att tillämpa eller kunna se bortom analogin. Det gäller därmed att inte alltid exakt följa en analogi eller att ens använda en analogi utan i första hand handlar det om att eftersträva funktionalitet. Analogier kan vara svåra att förstå eftersom de ibland kan vara kulturellt betingade, inte bara mellan länder, utan även mellan grupper vilket gör att de kan vara svåra att förstå. När analogier används bör det finnas en medvetenhet om de begränsningar som existerar och vilka problem som kan uppstå på grund av analogins användning. Analogier kan trots allt vara mycket användbara för att strukturera en tillämpning dock skall inte gränssnittet begränsas, och därmed tillämpningens funktionalitet, bara för att passa en viss analogi. (Cooper, 1995)

Det bästa är dock inte alltid att finna lämpliga analogier, något som är mycket bättre är att hitta lämpliga idiom. De idiom som används bör vara enkla eller utgöras av enkla idiom som kombineras ihop. Ett bra exempel på detta är direkt manipulation med en mus. Med hjälp av ett fåtal grundläggande operationer med en mus som kombineras ihop kan nästan allt i gränssnittet utföras. (Cooper, 1995)

4.6 Underlätta för perceptionen

Den erfarenhet som vi har ackumulerat om den värld som omger oss angående vad som anses vara den normala *hastigheten* för olika förlopp anger att datorprogram behöver reagera minst lika snabbt. Detta för att användaren inte skall missledas att tro att datorn har missförstått dennes avsikter. Det kan även orsaka förvirring om *orsak och verkan*. Exempelvis kan det vara en kort fördröjning på en sekund innan någonting händer. Och under den tiden hinner användaren göra en ytterligare operation och kan därmed förledas att tro att det var den senaste operationen som är orsak till resultatet, även om så inte är fallet. Därmed kan sägas att även små fördröjningar i respons från datorn, en fjärdedels till en halv sekund, kan missleda och förvirra användaren. (Lynch, 1994)

Om det är möjligt att manipulera ett objekt eller inte bör helst vara direkt synligt. Mobila klienter använder inte muspekare i samma utsträckning som vanliga datorer, att bara modifiera utseendet på muspekaren blir därmed inte ett bra alternativ och det kan leda till att användaren inte inser att ett objekt är manipulerbart. Om objekten inte är direkt synbara kommer det att belasta minnet. Eftersom en mer specifik strategi för att hitta manipulerbara objekt är att lagra deras position i minnet. Om dom är klart synliga behöver användaren enbart känna igen dem utan att först behöva utföra en sökning.

När användaren löser olika uppgifter med programmet kommer användarens fokus att röra sig över skärmen. Ha i åtanke hur kontroller placeras på skärmen så att deras inbördes belägenhet antar ett logiskt mönster för att användaren snabbt skall kunna lösa uppgiften utan onödigt stora förflyttningar med pekdon eller fokus. Arbetsminnet störs dessutom av att det sker en ständig omfokusering av uppmärksamheten mellan enstaka

kontroller till grupper av kontroller, en ständig omfokusering är även tidskrävande. (May et al., 1995) Vad som kanske blir vanligare vid lite mer komplicerade applikationer för handdatorer är ett ständigt skärmskiftande på grund av att det inte finns fysisk plats på en enstaka skärmbild. Se till att fördelningen mellan de olika skärmarna är logisk och att behövlig information flyttas med till nästa skärm så att användaren inte skall behöva komma ihåg den. Gör det även snabbt och enkelt att backa tillbaka till den förra skärmen.

Tydlighet och igenkännbarhet bör karaktärisera de ikoner som finns på skärmen. (May et al., 1995). Använd gärna stora ikoner med kort bildtext om programmet används mindre ofta och länge. Medan det är av mycket vitalare vikt för program som används ofta och länge att är ikonerna mindre för att inte slösa på viktigt utrymme. (Cooper, 1995) För små och mobila klienter kan det dock vara en fråga om hur mycket information som skall finnas på skärmen både om det är fråga om program som används ofta eller sällan. Använd därför inte stora och fina ikoner om det har den bieffekten att programmet breder ut sig på flera skärmbilder.

Vid utformning av ikoner är det tillrådligt att följa standarder så att olika funktioner ser likadana ut och blir lätt igenkännbara; exempelvis att en sax alltid betyder *klipp ut*. Det här betyder dock inte automatiskt att identiska ikoner skall användas hela tiden. Genom att använda tydligt igenkännbara ikoner men med lite annorlunda utseende ger det som resultat att programmen får lite olika prägel vilket i sin tur gör det lättare för en användare att se och uppleva skillnad mellan programmen. Det blir även intressantare att använda programmen om det förekommer lite variation mellan dem. (Gentner & Nielsen, 1996)

På en skärmbild med ett antal olika symboler eller ikoner kommer likadana ikoner att grupperas ihop och särskilja sig från de andra. Exempelvis i Windows har alla Word-dokument samma ikon och det blir därmed lätt att se vad det är för sorts fil och hur den kan användas. Det finns även fall där det är önskvärt att skilja ikoner och de filer eller funktioner som de representerar åt. Exempelvis om ikonerna representerar olika program som har vitt skiljda funktioner. I det fallet är det inte bra att användaren grupperar ihop dem och blir tvungen att skilja dem åt genom att gå ner i dess struktur och läsa textetiketten för att se vilket program som representeras av vilken ikon. (May et al., 1995)

Det är även tillrådligt att inte använda små detaljer i ikoner, Särskilt inte för att skilja olika ikoner åt. Av den anledningen att små detaljer kan vara svåra att se och användaren blir tvungen att vandra nedåt i de olika strukturerna för att över huvud taget kunna uppfatta dem. (May et al., 1995)

Vad som får en ikon att på ett bra sätt återspegla en idé finns det många olika åsikter om från olika personer. Därför är det en bra idé att försöka använda en ikon som så många som möjligt upplever på ett bra och naturligt sätt återspeglar den givna idén och därigenom blir intuitiv och förståelig. Det kan vara en bra idé att användaren självt får möjlighet att i programmet ändra ikoner till en uppsättning som passar just denne. För att undanröja de kvarvarande tvivlen kan en enkel hjälptext användas. Det finns även olika åsikter om funktionalitet eller estetik skall prioriteras högst, vissa människor

föredrar det ena framför det andra. Att därmed kunna ändra ikoners utseende skulle därmed kunna tillfredsställa även detta. Detta kan vara en anledning till att program med utbytbara *skal* som Winamp (http://www.winamp.com, 2001-12-24) har blivit så populära jämfört med andra liknande program.

Vid skärmbyte är det viktigt att *det psykologiska subjektet* bevaras. (May et al., 1995) Om det är fråga om ett objekt som har blivit valt och som skall behandlas bör information om vilket objekt det är finnas med på den nya skärmen. Användaren inte skall behöva backa tillbaka för att kunna ta reda på vilket det är. Om så är möjligt och för applikationen lämpligt, försök att ge intrycket av att det sker en in-zoomning i en struktur när ett objekt markeras och det sker ett skärmbyte. Den större *strukturen* där de olika objekten som kan markeras finns, markerandet av ett objekt leder till att dess *understruktur* och dess objekt visas. I mån av plats kan kontroller för att manipulera objekt i de olika strukturnivåerna finnas med. Det får dock inte bli plottrigt. Zooma hellre in ännu ett steg på de manipulerbara och visa möjligheterna för manipulation.

5 Applikationen

Nedan följer först en beskrivning av den applikation som jag har gjort. Sedan följer ett stycke med analys av applikationens användarvänlighet, främst med utgångspunkt från de ovanstående egenframtagna rekommendationerna men även till viss del från stycket *Användarvänighet*.

5.1 Beskrivning av applikationen

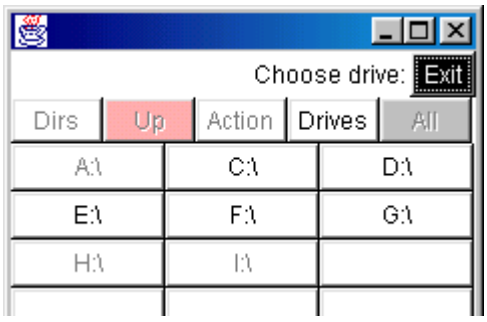
Den applikation som blivit gjord i anslutning till det här arbetet är en *filhanterare* med små och handhållna klienter i åtanke. Den filhanterare som har konstruerats har fått namnet *File* Quest.

En filhanterare är en applikation som hanterar de filer som är lagrade på en dator. Applikationen skall kunna erbjuda användaren vissa grundläggande möjligheter vid filhantering. Exempelvis: kopiering och flyttning av filer, namnbyte, radering, kunna starta program och öppna de filer som kan öppnas. Med det sista menas förmågan att kunna visa bild-filer, spela musik-filer eller öppna dokument antigen i någon form av ordbehandlare eller åtminstone visa texten. En filhanterare skall också kunna erbjuda ett sätt att navigera i datorn filsystem.

Det, numera, kanske mest kända exemplet på en filhanterare är Utforskaren i Windows. Utforskaren är tätt integrerad i hela systemet och vissa av dess funktioner är åtkomliga från samtliga fildialoger som används av program i Windows, åtminstone så länge applikationerna använder standard-dialogerna och inte bygger egna.

Men det är inte Utforskaren som har varit huvudinspirationskälla vid utformandet av File Quest, utan den excellenta filhanteraren Directory Opus. (GPSoftware, 2001)

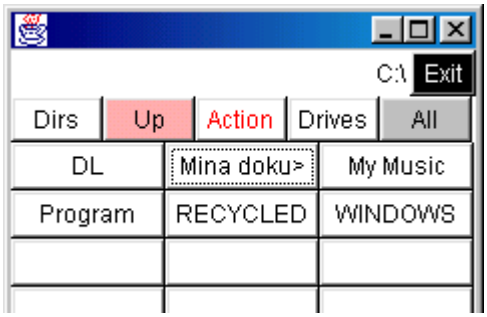
5.2 Funktionalitet



Det första som möter användaren vid start av File Quest är en skärmbild där denne ombeds välja enhet eller filsystem. Samtliga tillgängliga filsystem syns med sina respektive systembeteckningar. De enheter som finns men är tomma, exempelvis en tom diskettstation eller CD-läsare, är *skuggade* och därmed kan inte användaren klicka på dem och misslyckas. Hur viktig den här funktionen är det är osäkert eftersom en handdator kanske enbart har en enhet eller filsystem att välja på och då blir den här operationen ytterst överflödig. Men i en Windows-miljö som prototypen befinner sig i är den här funktionen vital.

I bilden (ovan till höger) visas de tillgängiga enheterna. Som synes finns det ingen diskett i enhet A eller en CD-ROM i enhet H. Skulle användaren vilja utföra

filoperationer på en diskett stoppar denne enbart i en diskett och går upp i *Show* menyn och väljer alternativet *Drives* varvid diskett enheten kommer att bli tillgänglig. För att välja en enhet klicka på den.

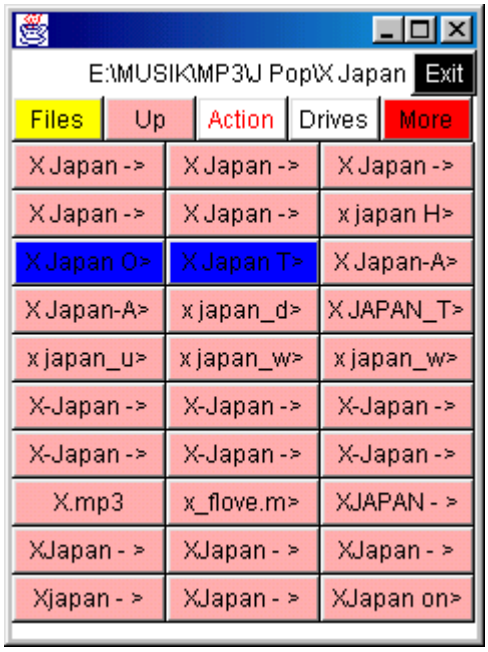


Efter att ha valt en enhet kommer programmet att gå in i den valda enheten och den nästa skärmen som visas (till vänster) kommer att visa de befintliga katalogerna som ligger i *roten* av filsystemet. Även dolda filer visas. För att gå in i en katalog är det bara att klicka på den knapp som har dess namn skrivet på sig.

5.2.1 Knappraden

Över knapparna med kataloger finns en rad med små knappar. Till höger syns hur den ser ut när programmet visar kataloger I roten av C:\ och inte filer. Tilläggas kan att färgsättning av dessa knappar inte är slutgiltig än men deras funktioner är det. Nedan följer en beskrivning av varje knapps funktion:

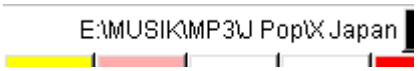
- Dirs Files Genom att klicka på denna knapp byts vy mellan att visa de kataloger som finns i en katalog och de filer som finns där. Om knappen är *vit* visas kataloger medan när knappen är *gul* visas filer. Vad som för tillfället visas indikeras även av den text som knappen har, visas filer står det *Files* medan om kataloger visas står det *Dirs*.
- Up Vid klickning på denna knapp backar File Quest upp ett steg närmare roten och hamnar tillbaka i den katalog som ligger ovanför den nuvarande katalogen. Denna knapp är även tillgänglig när programmet visar filer vilket innebär att användaren inte behöver byta vy till katalogvyn bara för att backa mot roten.



- Action Den här knappens funktion är att växla skärm till den skärm som innehar filoperationer (mer om den skärmen nedan).
- All More Syftet med den här knappen är att indikera om det finns fler filer eller kataloger, beroende på vilken vy som för tillfället visas, i den katalog som programmet befinner sig i. Om vyn visar filer och det finns fler filer än de 30 som får plats på en skärm kommer denna

knapp att vara röd. Om användaren sedan trycker på knappen kommer 30 filer till att visas. Finns det fortfarande fler kommer den att fortsätta att lysa rött men om det inte finns fler kommer den att slockna och vara ljusgrå. Dessutom indikeras förekomsten av fler filer eller kataloger genom text. Finns det fler kommer det att stå *More* på knappen medan om det inte finns fler att visa, det vill säga alla har visats då kommer texten *All* att stå på knappen. Programmet kommer även ihåg och skiljer på katalog- och filvyerna. Om filvyn har en röd knapp men inte katalogvyn kommer knappen att slockna om användaren byter från filvy till katalogvy och vice versa. Om användaren skulle råka bläddra förbi den katalog eller fil som denne letar efter och programmet har visat samtliga filer eller kataloger, knappen har slocknat, då är det bara att börja om genom att återigen trycka på den här knappen. Processen skulle kunna beskrivas som en *loop*, det går dock inte att backa vilket innebär att efter ett visst antal tryckningar på knappen kommer användaren tillbaka.

- 

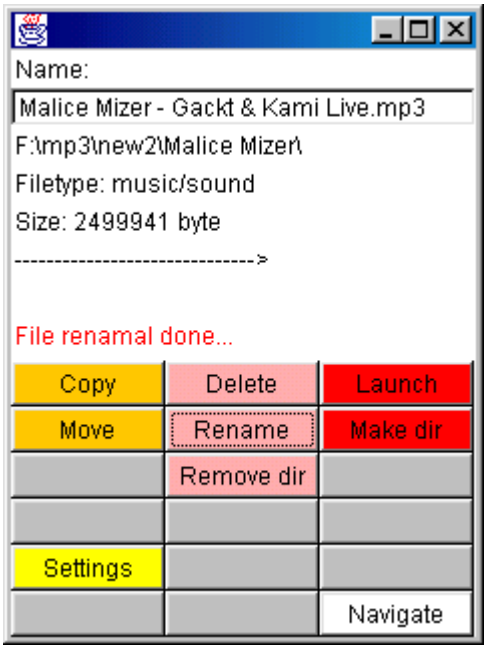
Den här knappens funktion är ganska uppenbar. Trycker användaren på den kommer programmet att avslutas.
- 

Den här textetikettens funktion är att visa sökvägen till den position som programmet har just nu. Får inte texten plats i rutan kommer texten att försvinna in i den vänstra kanten så att åtminstone de närmaste underkatalogerna är synliga.

5.2.2 Färgsättning av filer

Som synes i bilden (ovan i föregående stycke) har olika filer olika färger. Samma filtyp har samma färg. Exekverbara filer är orange, musik-filer är rosa, bilder är gröna, texter och dokument är gula och filmer är blå. Kataloger och enheter har alla alltid samma färg; nämligen vit.

5.2.3 Filoperationer



Om användaren trycker på *Action*-knappen eller om hon klickar på namnet på en fil kommer programmet att visa den skärm som innehar filoperationer samt mer detaljerad information om den fil som hon har valt att dyka upp. (Bild till höger)

I textinmatningsfältet står filens filnamn. Textetiketten nedanför filnamnet innehar namnet på den katalog som filen ligger i. Detta är dock en sanning med modifikation; den sökväg som står är den position i filsystemet som programmet för närvarande har. Om användaren klickar på en fil och informationsskärmen dyker upp då är det sökvägen till den fil användaren valt som kommer

att stå. Medan om användaren skulle återgå till filnavigationsskärmen för att sedan byta katalog och efter det återvända till filinformationsskärmen via Action-knappen (kanske för att hon vill kopiera eller flytta filen) då kommer inte filens egen sökväg stå. Den sökväg som då står är sökvägen till den nya katalogen som användaren gått in i.

Under sökvägen anger programmet vilken typ av fil som användaren har valt, till exempel en musikfil eller en bild. Sedan anges filens storlek i byte. Underst finns det en textetikett som normalt är tom men som tillhandahåller information om filoperationer lyckas eller misslyckas.

Under de vita textetiketterna finns knapparna som tillhandahåller filoperationerna. Deras olika funktioner är ganska uppenbara.

- 

Kopierar en fil från en plats I filsystemet till en annan. Skapar en exakt kopia. Användaren väljer först vilken fil som skall kopieras. Återgår sedan till navigeringsskärmen och flyttar programmets position i filsystemet till den önskade positionen, återgår till filinformationsskärmen via *Action*-knappen och trycker på denna knapp.
- 

Flyttar en fil från en plats i filsystemet till en annan. Skapar en exakt kopia av källfilen varefter källfilen tas bort efter att kopieringen är avklarad. Proceduren är samma som för filkopiering.
- 

Tar bort den valda filen från filsystemet. Användaren väljer först en fil och trycker sedan på den här knappen.
- 

Byter namn på den valda filen. Efter att användaren har valt en fil skriver hon in det nya namnet på filen i den textinmatningsruta som filnamnet står i. Efter att det nya önskade namnet är skrivet trycker användaren på den här knappen och programmet kommer att försöka byta namnet på filen till det nya önskade namnet.
- 

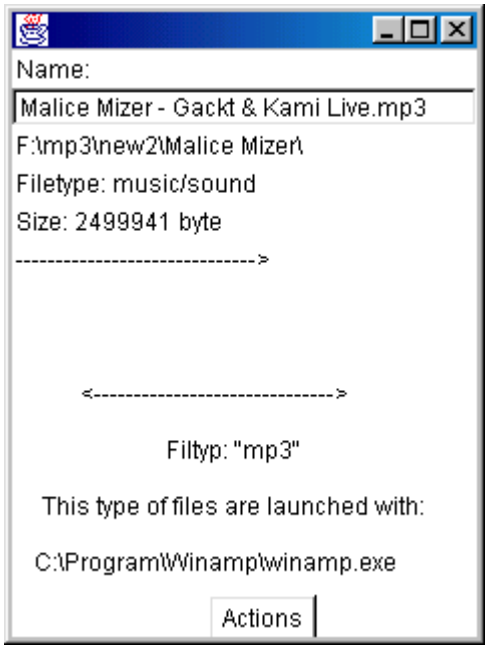
Tar bort en katalog ur filsystemet. Katalogen måste vara tom. Användaren navigerar sig fram till den katalog som hon vill ta bort. När katalogen är nådd klickar användaren på *Action*-knappen eller visar filinformationsskärmen genom att klicka på en fil i katalogen. Om hon vill ta bort katalogen måste samtliga filer tas bort först, därefter klickar användaren bara på *Remove dir*-knappen, katalognamnet måste stå på den textetikett som indikerar vilken som är den aktuella katalogen under filnamnet.
- 

Kör eller visar den aktuella filen. Om filen är exekverbar (för Windows gäller detta för alla filer som har ett efternamn som är EXE, COM eller BAT) kommer *File Quest* att försöka starta det. Om den valda filen inte är exekverbar men den är av en känd typ, det finns en inställd visare, då kommer filen att visas med den förinställda visaren för filen. Annars händer inget.

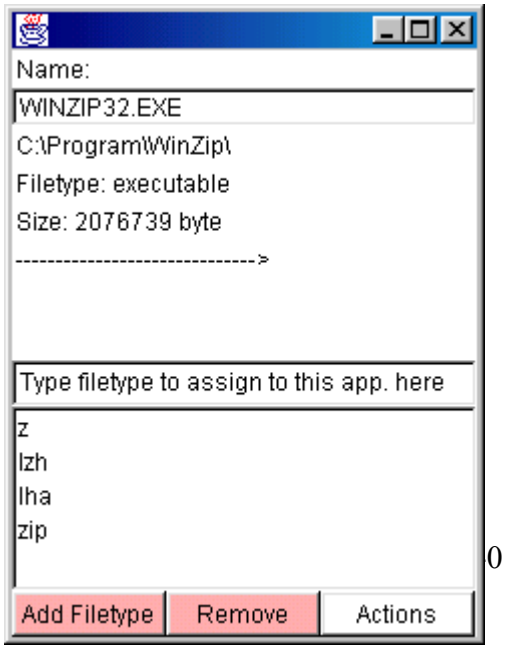
- Make dir Skapar en ny katalog i den katalog som programmet står i. Användaren skriver in det nya namnet i textrutan för filnamn och klickar på den här knappen. Katalogen är därmed skapad.
- Settings Visar den delskärm som har information om vilka applikationer som är inställda på att visa just den typ av fil som den valda filen är. Men om filen är exekverbar kommer det stället att komma upp information om vilka typer av filer som programmet är inställt på att visa. Användaren kan även lägga till och ta bort filtyper. Nästa stycke innehar mer information om hur detta går till.
- Navigate Vid tryck på denna knapp återgår programmet till att visa navigationsskärmen.

5.2.4 *Settings*-skärmen

När amvändaren trycker på *Settings-knappen* kommer knapparna med filoperationer att försvinna och ersättas av information relaterad till vilken applikations som är inställd att visa just den typ av fil som har blivit vald. Om det däremot är fråga om en exekverbar fil kommer det att visas en lista på vilka filtyper som har blivit inställda att visas med den valda applikationen. Det finns även möjlighet att både ta bort associerade filtyper samt lägga till nya.



Om den fil som användaren har valt inte är exekverbar kommer skärmen att se ut som den till höger. Vilken filtyp det är fråga om visas samt vilket program och dess sökväg visas också. Längst ner finns en knapp med texten *Actions*, vid tryck på den knappen kommer filoperationerna att återigen vara synliga.



Som redan har blivit klarlagt ser skärmen inte likadan ut när den fil som användaren har valt är exekverbar, hur skärmen då ser ut visas av bilden till höger. Högst upp finns ett textinmatningsfält som används för att associera nya filtyper till programmet. Om en ny filtyp önskas läggas till skrivs filslutet in i textfältet och användaren trycker på knappen med texten *Add Filetype* och det hela är klart. Den nya filtypen dyker även upp i listan. Om användaren däremot vill ta bort en associering väljer hon vilken som skall bort i listan och trycker på

Remove-knappen och associeringen försvinner. För tillfället är det inte möjligt att skriva över en associering och på det sättet byta program som är associerat med filtypen. Vill användaren byta program som assicieras till en viss filtyp måste hon först ta bort assiciering till det gamla programet varefter hon lägger till associering för det nya programmet.

Tilläggas bör att den här delen av programmet är det nyaste tillskottet och att den därför än inte är helt buggfri. Att lägga till och ta bort filtyper fungerar lite knaggligt än så länge.

6 Analys av applikation

Nedan följer en genomgång om hur väl det skapade programmet, *File Quest*, följer de framtagna gränssnittsrekommendationerna. I det ingår hur programmet följer rekommendationerna och i vilka fall det inte gör det och varför. Men även hur ännu ej implementerade funktioner skall kunna ta hänsyn till dessa rekommendationer.

6.1 Arbetsminnet

Stöd för arbetsminnet handlar främst om att hjälpa användaren att komma ihåg den data som hon just nu behandlar. Flytta relevant data mellan skärmarna så att användaren vet vad som händer, eller just har hänt, utan att hon för den delens skull skall behöva backa ett steg för att se eller komma ihåg och därmed försäkra sig om vad som hände. Fyll även i och visa alla alternativ som är möjliga att i förväg fylla i.

Detta utförs i *File Quest* i främst tre lägen.

- **Navigering.** När användaren förflyttar sig i filträdet går det endast att göra så i ett steg i taget till skillnad från till exempel Utforskaren där det är möjligt att göra stora hopp. För varje steg som görs visas det på en textetikett, överst, vilken den nuvarande sökvägen är vilket ger användaren kontinuerlig information om var någonstans som hon befinner sig just nu.
- **Markering av fil.** När användaren markerar en fil, det vill säga klickar på den, flyttas hon över till den skärm som innehar filoperationer. För att användaren inte skall behöva komma ihåg vilken fil som hon valde skrivs detta ut i en textinmatningsruta. Visserligen hamnar inte informationen på samma ställe på skärmen som filen som användaren klickade på låg, men eftersom det hela tiden är fråga om samma ställe på skärmen som namnet står lär sig användaren snart var som namnet står.
- ***Settings*-skärmen.** När användaren klickar på *Settings*-knappen är det fortfarande tydligt vilken fil som det gäller eftersom all den informationen står kvar på exakt samma ställe.

6.2 Långtidsminnet

Stöd för långtidsminnet innefattar främst implementering av olika hjälpfunktioner för att hjälpa användaren att använda och lära sig själva programmet.

Än så länge finns det ytterst begränsade hjälpfunktioner, eller rättare sagt nästan inga alls. Programmet riktar sig för närvarande till användare som har lite mer erfarenhet med datorsystem eftersom en viss grundläggande förkunskakunskap om hur datorers trädliknande filsystem fungerar antas finnas.

Dock hade det varit lämpligt om det var möjligt för användaren att på ett enkelt sätt få information om vad de olika knapparna gör och vilka möjligheter som varje funktion

erbjuder. Exempelvis genom att det fanns en *Hjälp*-knapp som användaren kunde klicka på varefter hon får information om den nästa knapp hon klickar på. Dock är det viktigt att beakta vid skapandet av en sådan funktion att användaren lätt ser att nu är programmet inställt på att ge hjälp och information.

För verkliga nybörjare av programmet och datorer i allmänhet skulle någon form av *tutorial* vara önskvärd där användaren kunde lära hur filträdet fungerar och programmets funktioner utan att skada eller förändra något, en slags simulator.

Programmet skiljer sig åt från redan kända filhanterare som Utforskaren. Den punkt där *File Quest* mest särskiljer sig är i själva navigeringen. Filer och kataloger har blivit separerade på varsin skärm. Detta gjordes främst av den anledningen att när kataloger och filer fanns på samma skärm upplevdes programmet som rörigt. Men även själva filoperatorernas utförande skiljer sig eftersom programmet inte begagnar sig av någon *drag-and-drop*-teknik. Men annars har enbart standardutseende på knappar och dylikt används för att användaren i så lång utsträckning som möjligt skall kunna känna igen sig.

6.3 Uppmärksamhet

I sitt nuvarande utförande har inte *File Quest* några behov av att åkalla användarens uppmärksamhet när programmet inte används. Dock har ansträngningar gjorts för att inte över huvud taget använda några dialogfönster som stör användaren. Felmeddelanden skrivs ut med röd text. Dock kan tilläggas att den text som felmeddelanden skriver ut just nu inte alltid är helt korrekt. Men det är bara en fråga om att modifiera så att programmet ordentligt förser användaren med matnyttig information om vad som har gått fel och förslag på vad som kan göras för att rätta till felet. Möjligtvis skulle programmet även kunna generera en ljudsignal när något gått fel.

Något som även nämndes i stycket om att fokusera användarens uppmärksamhet var även att det är bättre att programmet har en ångra funktion än ideligen uppoppande dialogrutor. Eftersom användaren enbart kommer att klicka på *Ja* och bekräfta valet av gammal vana. En sak som inte för närvarande är bra i programmet är att programmet raderar filer utan att ge användaren en chans att ångra sig. Detta kan lösas med någon form av *sophantering* eller *papperskorg* dit filerna först kommer innan de slutligen tas bort av användaren.

En intressant tanke är att faktisk realisera en *ångra*-funktion, inte enbart för radering av filer, utan även för de andra filoperationerna så att det är möjligt att till en viss del i alla fall återställa filsystemet till sitt usprungliga skick. Och det utan att användaren skall behöva komma ihåg exakt vad hon har gjort. Programmet loggar helt enkelt användarens förehavanden och ger denne en möjlighet att backa ett eller flera steg.

6.4 Tänkandet

Applikationen ger inte användaren möjlighet att nyttja flera olika strategier i sin nuvarande implementation. Vill användaren exempelvis kopiera en fil finns det bara ett sätt att göra det på. Det finns inga specifika strategier.

Något som skulle kunna göras för att skapa mer specifika strategier är att låta användaren definiera ett visst antal *favoriter*. Dessa favoriter skulle kunna visas och ställas in på en egen skärmbild. Det skulle vara fråga om *genvägar* eller *länkar* till olika program eller andra filer som användaren vill ha lätt till hands. En annan idé är att skapa en lista på de vanligaste eller de senast använda filerna. Det kanske även skulle vara en idé att sortera filerna efter filtyp genom att exempelvis presentera de senast spelade MP3:orna.

En annan sak skulle vara att nyttja kortkommandon vid exempelvis skärmbyte. För att användaren inte alltid måste peka på knapparna utan kan använda en och samma knapp på själva handdatorn för att byta skärm i applikationen, då främst mellan navigations- och filmanipuleringsläge.

Ännu en idé skulle vara att ge användaren möjlighet att programmera en knapp på den övre raden i navigationsskärmen med en filoperation, exempelvis kopiering. Detta skulle leda till att användaren skulle kunna markera en fil, leta upp den katalog som filen skall kopieras till och sedan bara trycka på den egenprogrammerade knappen utan att behöva byta skärm.

Ytterligare en idé är att använda dubbelklick. Exempelvis antigen visa eller spela filen om den kan visas eller spelas eller köra filen om den är exekverbar. Och detta utan att byta skärm till filoperatorskärmen.

Något som också har eftersträvats vid skapandet av *File Quest* är att det skall vara omöjligt för användaren att komma åt funktioner som inte skall vara åtkomliga i vissa lägen. Detta för att programmet inte skall krascha eller generera ett felmeddelande bara därför att användaren råkat komma åt en funktion som inte borde vara tillgänglig. Exempelvis blir flera funktionsknappar i den övre raden otillgängliga när programmet visar de tillgängliga enheterna eftersom de inte skall gå att komma åt funktionerna i det läget. I tillägg till detta går det inte att välja de enheter som är *tomma*, exempelvis en tom diskettstation, men dom är fortfarande synliga för att visa användaren att trots att de är otillgängliga så finns de fortfarande.

6.5 Mentala Modeller

Någon av den mest framträdande egenskapen för mentala modeller är utnyttjandet av *analogier* för att förenkla förståelsen för hur programmet används. För en filhanterare som *File Quest* är det enklast att använda de redan befintliga hos det underliggande systemets analogier. Det är främst utnyttjandet av två separata analogier som förenklar förståelsen för hur filsystem eller ett filträd är uppbyggda.

- **Filsystem.** Analogin är att filerna är organiserade som ett arkivskåp där varje katalog representeras av en låda som i sin tur innehåller mappar, engelskans *file*, som representerar filerna.
  Den modifiering som har gjorts till analogin är att varje låda eller katalog kan i sin tur även den innehålla en nya låda som innehåller nya lådor och filer.
  Windows har en lite modifierad version av den här analogin. Varje enhet som är kopplad till datorn ses som en låda som innehåller mappar, kataloger kallas för mappar, vilka i sin tur innehåller olika papper eller filer.

- **Filträd.** En dators filsystem brukar även bli liknat vid ett träd, eller snarare vid ett upp och nedvänt träd. Högst upp ligger *roten* och ut från roten skjuter olika grenar, kataloger, som i sin tur delar upp sig i ännu fler grenar. Varje gren kan ha olika blad som kan ses som filerna.

För att en användare skall kunna med fördel använda en filhanterare är det av yttersta vikt att hon har förstått de här analogierna.

Det finns en del *idiom* som utnyttjas, men det är främst fråga om enkla idiom som peka och klicka. Handlingar som även mycket oerfarna datoranvändare känner igen och har förstått.

6.6 Perception

Understöd för perceptionen handlar om att ikoners och knappars utformning. Men behandlar även den allmänna uppbyggnaden av skärmbilderna och hur dessa olika skärmar interagerar med varandra.

De knappar som används har en utformning som tydligt särskiljer dem som just knappar som användaren kan trycka på. De olika funktionsknapparna har textetiketter som indikerar knappens funktion och inte en snygg bild eller ikon. Estetiskt är bilder mer tilltalande och kan därmed göra programmet mer attraktivt men eftersom det än så länge enbart är en prototyp får den estetiska biten vänta.

Filknapparna bör nämnas för sig. De olika filerna representeras även de av knappar som även en datoranvändare med lite erfarenhet förstår att trycka på. Knapparna antar även olika färger beroende på vilken filtyp som filen den representerar är, exempelvis är dokument och texter gula. På detta sätt grupperas de olika filerna ihop och användaren får snabbt en bild av vad denne kan göra med varje separat fil. Tyvärr grupperas även exekverbara filer ihop med en och samma färg och användaren blir tvungen att läsa textetiketten för att förstå vilket program som det är fråga om.

På grund av den begränsade skärmytan och det faktum att det finns tre kolumner med filknappar får inte alltid hela filnamnet plats på knappen. Detta är särkillt störande när det är fråga om flera filer vars första del av filnamnet är identiskt och vars filnamn är för långt så att det inte får plats. Användaren kan bli tvungen att klicka på en filknapp för att kunna läsa hela filnamnet. Om den fil som användaren få får upp inte är den fil som hon vill åt. Och hon därmed återvänder till navigationsskärmen finns det inget som indikerar vilken fil hon har valt och hon kan råka välja samma fil igen när hon fortsätter att leta. Problemet löses genom att den senast valda filen markeras med till exempel inverterade färger. Användaren skall inte behöva gissa eller komma ihåg vilken filknapp hon klickade på.

Det hjälper inte om det psykologiska subjektet behålls från navigationsskärmen till skärmen med filoperatorer när subjektet inte alls behålls när användaren återvänder till navigationsskärmen. Det sätt som det psykologiska subjektet behålls mellan skärmar är när användaren klickr på en filknapp följer filnamnet med till nästa skärm. Men även när användaren navigerar i filsystemet kan hon se vilken katalog hon just nu står i och vilka kataloger som hon passerade innan dess, detta görs genom att läsa den översta

textetiketten. Ett sätt att lösa problemet är att minska antalet kolumner till två eller kanske rentav en. Om en kolumn används skulle det kanske till och med vara möjligt att visa mer information än bara filnamn, information som till exempel storlek, fast inte i byte utan kanske i megabyte. Medan det för kataloger skulle vara möjligt att visa hur många filer och kataloger som den innehåller eller kanske sammanlagd storlek.

Det är även möjligt att säga att programmet följer den logiska perceptionsprocess som May et al. (1995) beskriver. I det att programmet startar med att användaren får välja enhet (eller *rot* beroende på vilken analogi som vill användas) varefter en inzoomning sker för varje katalog som användaren går in i. Det kan beskrivas som en inzoomning på varje objekts, katalog eller fil, detaljer. De detaljer en katalog har som användaren zoomar in på är fler kataloger och filer. Medan en fils detaljer som användaren zoomar in på är ytterligare detaljerad information om dess egenskaper som dess storlek utförligt filnam och filtyp, om användaren har glömt vad färgen representerar.

Något annat som skulle vara önskvärt vore att användaren självt skulle kunna ställa in vilka färger som skall användas till vilka filtyper för att göra det mer estetiskt tilltalande och personligt. Eller att hon även skulle kunna ställa in antal kolumner och vilken extrainformation som skall visas, om det bara är en kolumn.

7 Slutsats

För att det skall vara möjligt att på ett bättre sätt interagera med de små och mobila klienterna och deras användargränssnitt behövs det prov på större kreativitet. Kreativitet som leder till nya och bättre interaktionsmetoder. Under tiden får vi nöja oss med vanliga grafiska gränssnitt med dess direkta manipulation. Det är just den typen av interaktion som den här avhandlingen behandlar. Både det program som har blivit framtagit och den analys som sedan görs är anpassad efter just det vanliga grafiska gränssnittet.

Intressant kan vara att lägga märke till att det är enbart på ett fåtal punkter som gränssnittet för små och mobila klienter skiljer sig åt från stationära datorers gränssnitt. Den faktor som främst skiljer är storleken på skärmen och dess inmatningsverktyg, det sistnämnda en del av användargränssnittet som inte alls har blivit uppmärksammat i den här uppsatsen. Det är därmed av yttersta vikt att konstruktören har ett logiskt samband och att relevant information flyttas mellan de olika skärmbilderna, eftersom den begränsade ytan ofta innebär att programmet blir tvunget att breda ut sig på flera skärmbilder istället.

En annan markant skillnad är en direkt påföljd av att klienterna är mobila; det handlar om den miljö som de kommer att användas i. Användaren kan vara på stan en solig dag, vårsolen skiner och ungarna i barnvagnen bredvid på övergångsstället skriker samtidigt som en arg bilist tutar på en fotgängare som bestämt sig för att chansa på att hinna över före. Det är viktigt att ta den användarmiljön i beaktande. Gör det lätt för användaren att hitta tillbaka om hon skulle bli distraherad.

För att uppnå målet att skapa en användarvänlig applikation måste hänsyn tas till användarnas varierande mentala förmågor och behov. För att klara detta krävs stor insikt i hur diverse mentala processer fungerar och vad som driver användarna.

Slutligen, vid utformning av ett gränssnitt som skall vara användarvänligt tänk på följande:

- Ta i beaktande både de generella och de uppgiftsrelaterade mål som användaren har.
- Låt programmet i så lång utsträckning som möjligt hjälpa användaren att undvika fel. Och skulle användaren göra fel skall det vara enkelt att backa tillbaka.
- Hjälp användaren att lära sig programmet.
- Följ standarder för den plattform som programmet körs på så att användaren känner igen sig.
- Undvik att distrahera och därmed avleda uppmärksamheten med ovidkommande företeelser och saker såsom onödiga dialogrutor.
- Belasta inte användarens minne i onödan utan använd datorns.

- För att användaren skall trivas med resultatet är det bra om hon får vara med och bestämma både vid användning, programmet är konfigurerbart, och vid utformning, programmeraren är konfigurerbar, av programmet.
- Var kreativ och nyskapande. Små och mobila klienters begränsade möjligheter att presentera information kan kräva nya angreppssätt vid utformning av gränssnitt för att det skall vara möjligt att nå optimal användarvänlighet och funktionsduglighet.

Intressant att lägga märke till är att samtliga av dessa ovanstående punkter även är tillämpbara på alla typer av gränssnitt, inte bara för små och mobila klienter utan även för stationära klienters gränssnitt. Några punkter som kan tilläggas för just små och mobila klienters gränssnitt är däremot:

- Om applikationen är så pass komplex att flera skärmbilder måste användas skall det finnas en logisk uppdelning av informationspresentationen och dess eventuella inmatning.
- Försök även att ta hänsyn till den miljö som applikationen skall användas i. Om det till exempel är en applikation som skall användas i en bullrig miljö är det inte lämpligt att använda ljudsignaler för att meddela användaren om olika saker. Eller om användaren lätt kommer att bli distraherad av yttre faktorer under sin användning av applikationen är det lämpligt att göra det så enkelt som möjligt för användaren att snabbt återuppta interaktionen och återigen sätta sig in i var någonstans i applikationen som denne befinner sig i.

Slutligen kan användandet av utvärderingsmetoder och kunskap om hur användaren fungerar, både de direkt tänkta användarna och människan i allmänhet, göra att många brister hos användargränssnittet kan undvikas. Det är även bättre att se till att de implementerade funktionerna fungerar på ett tillfredsställande sätt än att ständigt implementera nya funktioner som kanske inte är helt stabila eller enkelt förståeliga.

> *När ett finger pekar mot månen, undersöker den dåraktige fingret.*
> Buddha (Poserina, 2001)

Det gäller med andra ord att se de bakomliggande processerna, inte bara att något är som det är (ett pekande finger) utan även varför det är som det är (fingret pekar mot något nämligen månen). Att vara konstruktör av gränssnitt innebär inte enbart att ha god kännedom om hur en dator fungerar och hur den programmeras utan även att vara en god kännare av hur människor fungerar mentalt och vad som driver och motiverar oss till ett visst handlingssätt. Det är detta som är den här avhandlingens mål att erbjuda en viss insikt i och hur detta sedan kan appliceras på användargränssnitt.

8 Referenser

Allwood, Carl Martin (1991). *Människa-datorinteraktion - Ett psykologiskt perspektiv.* Lund, Studentlitteratur

Andersen, Erling S (1994). Systemutveckling – principer, metoder och tekniker (2:a rev. uppl.). Studentlitteratur, Lund.

Backman, Jarl (1998). *Rapporter och uppsatser.* Lund: Studentlitteratur.

Björk, Staffan (2000). *Flip Zooming – the development of an information visualization technique.* Göteborg: Göteborgs Universitet, Institutionen för Informatik, Studier inom Informatik, Rapport 19, Oktober 2000, ISSN 1400-741X

Cooper, Alan (1995). *About Face: The Essentials of Usre Interface Design.* Foster City, USA, IDG Books Worldwide, Inc.

Cox, Kevin & Walker, David (1993), *User Interface Design*, Singapore, Prentice Hall

Eco, 1995: Eco, Umberto (1995). *The Search for the Perfect Language.* Blackwell Publications.

Gaarder, Jostein (1995), *Sofies Värld: roman om filosofins historia.* Sverige, Rabénförlagen

Gibson, William (1984). *Neuromancer*. New York, The Berkley Publishing Group

Haberlandt, Karl (1994). *Cognitive psycology.* Boston, Mass. London: Allyn and Bacon.

Holme, Idar Magne & Solvang, Berndt Krohn (1991). *Forskningsmetodik: Om kvalitativa och kvantitativa metoder.* Lund: Studentlitteratur.

IBM, 1992: Object-Oriented Interface Design (1992). Cary. USA, Que Corporation

Illustrerad Vetenskap nr 6, 2000 (27/4 – 24/5). *Internet skall tala alla språk.* Bonniers

Kukulska-Hulme, Agnes (1999). *Language and communication: essential concepts for user interface and documentation design.* New York: Oxford University Press.

Lakoff, Georg (1987). *Women, Fire and Dangerous things*. Chicago, The University of Chicago Press

Meldau, Fred J (1968). *Why We Believe in Creation Not Evolution*. Christian Victory Publishing, Denver, CO

Nationalencyklopedin (1990-). Höganäs, Bra Böcker

Nielsen, Jakob & Mack, Robert L (1994). *Usability Inspection methods*. New York

Patel, Runa & Davidson, Bo (1994). *Forskningsmetodikens grunder – att planera, genomföra och rapportera en undersökning (2:a rev. uppl.).* Studentlitteratur, Lund.

Scientific American, september 1979: Scientific American, (september 1979)

Wickens, Christopher D (1992). *Enginering Psychology and Human Performence.* Harper Collins cop., New York.

8.1 Internetlänkar

Computeractive, 26 februari 1998: *Computeractive*. [www-dokument]. http://www.computeractive.co.uk/, 1998-02-26

Eco, 1996: Eco, Umberto (1996). *The Dream of a Perfect Language.* [www-dokument]. http://www.italynet.com/columbia/dream.htm, 25 november 1997

Flem-Ath, Rand & Rose (1996), *Atlantis and the Earth's Shifting Crust.* [www-dokument]. http://www.netfeed.com/pstevens/delaware19.htm, 13 december 1997

Gentner, Don & Nielsen, Jakob (1996). *The Anti-Mac Interface*. [www-dokument]. http://www.acm.org/cacm/AUG96/antimac.htm, 2001-12-23

GPSoftware (2001). *Directory Opus 6 for Windows.* [www-dokument]. http://www.directoryopus.com, 2001-12-26

Holmquist, Lars Erik (1997). *Det senaste inom gränssnittsutvecklingen.* [www-dokument]. http://www.informatik.gu.se/~leh/mdi/text/pdframe.htm, 1998-03-18

Koelle, Dave (feb, 1996). *Intellegent User Interfaces.* [www dokument]. http://www.cs.wpi.edu/Research/airg/IntInt/intint-outline.html, 2001-12-5

Lund, Andreas (1997). *Embodied Interfaces: Towards an experientialist approach to user interface design*. [www-dokument]. http://www.ifi.uio.no/iris20/proceedings/34.htm, 1998-02-04

Lynch, Patrick J. (1994). *Visual design for the user interface, part 1*. [www-dokument]. http://info.med.yale.edu/caim/publications/papers/gui.p1.html, 1998-01-14

Nielsen, Jakob (1994 [1]). *How to Conduct a Heuristic Evaluation.* [www-dokument]. http://www.useit.com/papers/heuristic/heuristic_evaluation.html, 2001-12-17

Nielsen, Jakob (1994 [2]). *Ten Usability Heuristics.* [www-dokument]. http://www.useit.com/papers/heuristic/heuristic_list.html, 2001-12-20

Pedraza Arpasi, Jorge (2001). *Aymara uta: jaya mara aru.* [www-dokument]. http://www.aymara.org, 5 december 2001

Poserina, Jim (2001). *Jim's favourite quote , quip, axiom, and maxim repository*. [www-dokument]. http://www.jimposerina.com/quotes/default.asp, 2002-01-08

Sharp (2001). *Zaurus.* [www-dokument]. http://www.ezaurus.com, 2001-12-26

8.1.1 Postscriptdokument från Internet

Holyer, Andy (1993). *Methods for Evaluating User Interfaces.* [postscript-dokument]. ftp://ftp.cogs.susx.ac.uk/pub/reports/csrp/csrp301.ps.Z, 2001-12-14

May, Jon & Scott, Sophie & Barnard, Phill (1995*). Structuring Displays: a psychological guide.* [postscript-dokument]. ftp://ftp.mrc-apu.cam.ac.uk/pub/amodeus/usemod/um_wp31.ps.Z, 5 december 2001