NTNU
Det skapende universitet

Sluttbrukerkomponering av tjenester i smarte hjem

En komparativ studie av grensesnittmetaforer

Bjarne Hammersvik Muri

Master i informatikk
Oppgaven levert: Juni 2009
Hovedveileder: Dag Svanæs, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Oppgavetekst

Brukergrensesnitt for konfigurering av "smarte hjem"

Prosjektet er et samarbeid med forskningsavdelingen til Telenor. Nye boliger blir i dag ofte utstyrt med en rekke sensorer og styrbare elementer. Dette gir store muligheter for styring av varme, lys, luft og diverse elektroniske innretninger i huset. For at disse mulighetene skal kunne bli utnyttet av vanlige brukere er det viktig at de selv kan styre alt dette ved hjelp av enkle og effektive brukergrensesnitt. Oppgaven går ut på å lage prototyper på slike brukergrensesnitt, og teste dem ut med potensielle brukere. Prosjektoppgaven kan med fordel tas videre til en hovedoppgave med samme tema.

Sammendrag

I et smarthjem er det behov for tilpassninger av huset i forhold til brukerens ønsker og liv. Alle hjem er forskjellige og har forskjellige enheter som lamper, alarmer, tv og kaffetrakter med mer. Beboere av et smarthjem kan få mulighet til å komponere egne tjenester og tilpasse disse til sitt eget liv.

Sluttbrukerprogrammering har vist seg mulig innenfor enkelte domener og i denne oppgaven ser vi på hva som kreves for å tilpasse et brukergrensesnitt for sluttbrukerkomponering av tjenester. For å lykkes innen sluttbrukerprogrammering er det viktig å velge et avgrenset domene og ikke prøve å lage programmering for sluttbrukere. Denne oppgaven bruker smarte hjemdomenet, som begrenser mulighetene for hva som skal komponeres. Et tidligere forsøk innenfor samme domene feilet blant annet fordi det var for tungvint brukergrensesnitt.

Det ble utviklet fem ulike prototyper, alle var mest mulig like hverandre og med de samme mulighetene, men med forskjellige metaforer. Metaforene vi brukte var både billedlige, utradisjonelle og mer klassiske. Flere av metaforene er hentet fra tidligere vellykkede sluttbrukerprogrammeringstjenester, men fra andre domener enn smarte hjem. Prototypene ble utviklet for å undersøke hvilken metafor som var best egnet for sluttbrukerkomponeringer av tjenester i smarte hjem.

Potensielle brukere av et smarthjem er hovedsakelig unge i etableringsfasen og det ble derfor brukt et utvalg av denne gruppen til brukbarhetstesting av de ulike prototypene. Utvalget ble videre delt inn i to brukergrupper; en teknisk og en ikke teknisk gruppe. Den tekniske gruppen var programmerere fordi det var ønskelig å se forskjellen mellom programmerere og andre i valg av metafor.

Spørreundersøkelser, et intervju og en brukbarhetstest legger grunnlaget for resultatet hvor den klassiske metaforen WIMP (vindu ikon meny peker) blir best egnet. Tett etter følger puslespillmetaforen som er mer utradisjonell i vanlige dataprogrammer.

Forord

Jeg kontaktet Dag Svanæs og lurte på om han kunne veilede meg. Han lurte på hva jeg ville gjøre og til min store glede hadde han en oppgave som passet meg godt. Oppgaven var gitt av Telenor og i starten hadde vi mange møter for å avklare oppgavens omfang. Før vi gikk over til valg av metaforer og evalueringsmetoder. Under implementering av metaforene jobbet vi tett hvor jeg implementerte, og veiledere fra NTNU og Telenor kom med tilbakemeldinger underveis. Vi støttet oss på anerkjente prinsipper for å gjøre prototypene så brukervennlige som mulig.

Arbeidet med oppgaven har vært varierende, spennende, utfordrende og til tider endeløst. Etter å ha jobbet med denne oppgaven sitter jeg igjen med mange gode erfaringer jeg er takknemlig for. Nå i ettertid finner jeg det rart at det er slutt allerede. Det å skrive en masteroppgave kan sammenlignes med en lang kjøretur. I starten ser du ikke målet, men trives med det du gjør, full av energi. Spiser litt snop, prater litt med reiseføIget og skaper en trivelig ramme for resten av turen. Midt i turen føler du at mye er gjort, du ser enda ikke målet, men vet at du har minst like mye igjen. Det er en tyngre fase før du plutselig nærmer deg målet. Det er først da du innser at du har det fint og kunne fortsatt et godt stykke til. Forventningene til å bli ferdig/komme frem har vært så store at det har overskygget den gleden du har hatt underveis. For det å bli ferdig er slett ikke så bra, det betyr bare at en fin studietid er forbi.

Denne oppgaven er en del av et større prosjekt hos Telenor, Tellu og NTNU. Prosjektet heter ISIS og fokuserer på smarte hjem og mange personer har vært til stor hjelp under prosessen med å produsere den. Jeg ønsker å takke involverte parter, hovedveileder Dag Svanæs for konstruktiv kritikk og en stor, faglig kunnskap. Veileder hos Telenor, Yngve Dahl har vært til stor motivasjon, gitt faglig kunnskap og presset på fremdriften. På NSEP fikk jeg god støtte av tekniker Terje Røsand til bruk av tekniske fasiliteter og jeg skylder han en stor takk. Videre ønsker jeg å takke Ole Alsos for motivasjon og pilottesting. En takk rettes også til Tuva Foldøy Klingsheim og Cecilie Engebretsen for illustrasjonsbilder i forbindelse med brukbarhetstesting. Til slutt ønsker jeg å takke alle studenter ved MMI linjen på arbeidsplassen Gribb for gode faglige diskusjoner og et godt og trygt studiemiljø.

Bildet brukt på kapitteloverskrifter er en manipulasjon av giannisgx89, en grafisk designer ved deviantart.com

X

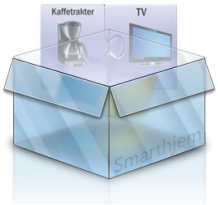
Bjarne Muri

Innhold

Figurliste

Tabelliste

Liste over lister



1. Introduksjon

Datamaskiner er blitt allemannseie. 86 % av alle husholdninger hadde i 2008 en datamaskin stående, de aller fleste av disse er i tilegg koblet til internett (1). Datamaskiner er kraftige verktøy og kan brukes til mange forskjellig oppgaver, så vel underholdning som regnskapsføring. Det er utvilsomt flere bruksområder som ennå ikke er oppdaget og denne oppgaven ser nærmere på hvordan et hjem kan dra enkelte fordeler ved hjelp av en datamaskin. Her menes ikke å kontrollere de enkelte individer men å tilrettelegge hverdagen for den enkelte ved hjelp av egenkomponerte tjenester i hjemmet.

En tjeneste i hjemmet er en eller flere komposisjoner bestående av en hendelse og en aksjon. Dette kan for eksempel være at TV-en viser en melding (aksjon) når noen ringer på døren (hendelse). Verken tjenester eller komposisjoner trenger å være nyttige, men kan være et morsomt bidrag til hverdagen. Et morsomt liv gir økt livskvalitet i enkelte tilfeller(2) og kan derav også ses på som nyttig.

1.1 Smarte hjem

En studie hvor det var mulig å styre et ”vanlig” hjem via en mobil, en datamaskin og en media terminal konkluderer blant annet med at datamaskinen er den mest velegnede enheten å gjøre avansert konfigurasjoner på (3). En avansert konfigurasjon var mer avansert enn tjenestekomponering, men det var ikke store forskjellene. De konkluderer videre med at det er to måter å benytte et smart hjem. 1. dirkete kontroll hvor brytere erstattes med digitale fjernkontroller eller direkte styring og 2. mønsterkontroll, ting som skjer basert på hendelser. Denne siste måten å bruke et smart hjem på bruker denne oppgaven som domene.

1.2 Sluttbrukerprogrammering

Mange ser på datamaskinen som en pakke med programmer til å bruke. Dette byr på problemer da brukere ofte ønsker annen funksjonalitet enn programmerere forutså da de lagde programpakken. Derfor er det ønskelig og la brukere konfigurere og programmere sine egne programmer. Det og la brukere legge til den funksjonaliteten de bruker til vanlig og la de tilpasse funksjoner vil gi dem større glede av programmet. De fleste sluttbrukere har dessverre ikke tid eller ressurser til å lære seg tradisjonell programmering, derfor må det inngås kompromiss med tanke på funksjonaliteten en sluttbruker kan få. Begrensning av sluttbrukerprogrammeringen til et domene vil være med å gjøre sluttbrukerprogrammering enda enklere (4).

Allen Cypher har laget en vidt omspennende definisjon av sluttbruker og sluttbrukerprogrammering på side 587 i (5):

Sluttbruker: En bruker av et dataprogram, ikke en programmerer, men en person som bruker datamaskinen i dagliglivet, enten på jobb eller i fritiden. Personen er ikke interessert i datamaskinen i seg selv, men i bruken av den.

Sluttbrukerprogrammering: Når sluttbrukere som nødvendigvis ikke har lært å skrive tradisjonell kode i vanlige programmeringsspråk skriver dataprogram. Dette inkluderer for eksempel regneark formler og makroer.

Denne oppgaven ser ikke på sluttbrukerprogrammering som et alternativ til tradisjonell programmering. Fokuset her er hvordan brukere selv kan komponere tjenester i hjemmet og ikke hvordan de skriver dataprogrammet dette gjøres i.

1.3 Motivasjon

Det skapes stadig nye muligheter til å gjøre hverdagen enklere og morsommere. Flere ønsker å flytte inn i et hus hvor jeg kan konfigurere og tilpasse det til mitt eget liv. Det å være med å gjøre teknologien brukervennlig for den vanlige bruker opplever jeg som viktig og motiverende. Teknologien rundt smarte hjem er ikke ferdig utviklet enda, men når den blir det håper jeg mitt bidrag kan være med å forenkle prosessen. På denne måten føler jeg at denne studien er med å forkorte ventetiden på et smarthjem. Det har i tilegg vært lite fokus rundt brukervennligheten av et smarthjem(6), noe denne oppgaven er med på å belyse og skape mer kunnskap rundt domenet.

1.3.1 Brukergrupper

Denne oppgaven utforsker hvordan ulike brukergrupper ønsker og klarer å bruke et smarthjem. Da dette er teknologi for fremtiden har vi valgt å fokusere på unge personer, disse personene er potensielle huseiere og vil en dag stå ovenfor et valg om de vil ha et smarthjem eller ikke. Den ene gruppen er teknologer og høyst datakyndige, vi skilt mellom de ulike brukergruppene ved hjelp av programmeringsevne. Programmere tenker på sin måte og forstår lettere hvordan en datamaskin virker enn andre. For å finne personer med ønsket programmeringserfaring valgte vi siste års masterstudenter på dataliner ved NTNU. Den andre brukergruppen er siste års arkitektstudenter med mye mindre programmeringserfaring, men en stor interesse for bolig. Dette er alle studenter som skal ut og kjøpe bolig om kort tid, og denne oppgaven utforsker hva de finner viktig og interessant i forbindelse med brukervennlighet for smarte hjem.

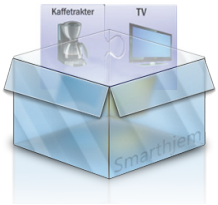
1.4 Problemstilling

Problemstillingen består av fire forskningsspørsmål RQ1-4 hvor RQ4 er sterkt knyttet til de foregående.

Liste 1 Forskningsspørsmål

1. RQ1: Hvilke brukergrensesnittmetafor er best egnet for sluttbrukerkomponering av tjenester i smarte hjem?
2. RQ2: Er det noen forskjell mellom brukergrupper i forhold til preferanser av slike metaforer?
3. RQ3: Hvilke faktorer er viktige i brukernes argumentasjon for hva som er gode og dårlige metaforer?
4. RQ4: Basert på RQ1-3, hva vil være den best egnede metaforen

Videre i oppgaven vil forskningsspørsmålene bli referert til som RQ1 til RQ4



2. Teori

Denne oppgaven berører flere forskningsområder innenfor informatikk, sluttbrukerprogrammering og bruk av metaforer er de viktigste. Andre områder er benyttet men da uten å ha innvirkning på resultatet. Dette da brukerne kun tester en prototyp og ikke oppdager fordelen ved bruk av patterens og andre tekniske spesifikasjoner.

2.1 Sluttbrukerprogrammering

Når brukere skal lage egne tjenester hadde det vært en fordel om alle kunne programmere. Dessverre kan de færreste vanlige brukere et programmeringsspråk. Det kan virke skremmende for enkelte brukere å vite at de skal programmere tjenester selv, da programmering ofte forbindes med noe vanskelig. Programmering handler egentlig om å lage regler, altså fortelle et system hva som skal skje hvis det og det skjer. Det letteste hadde vært om datamaskiner forstod mennesker og skjønte hva de skulle gjøre. Dessverre er det enda flere år til datamaskiner forstår vanlig språk og i mellomtiden er det beste en datamaskin kan gjøre å komme med kvalifisert gjetting. Gjetting er ikke tilstrekkelig for kritiske applikasjoner som å skru av kaffetrakteren (en av de største brannfellene i en bolig). En utvikler kan lage tjenester til hjemmet og selge dette, men da vil brukeren miste mange muligheter(7), eller han/hun må gå til anskaffelse av dem, som er både tungvint og fort kan bli dyrt.

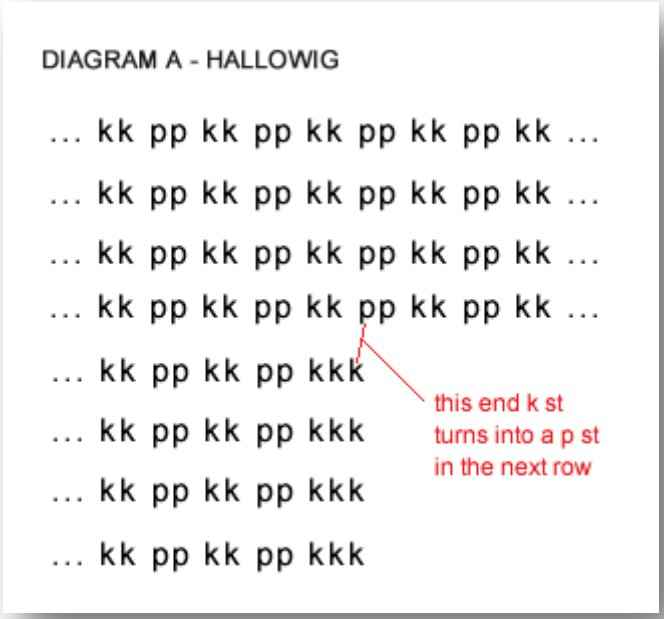
Tradisjonell sluttbrukerprogrammering prøver ofte å la vanlige mennesker (ikke programmerere) lage datasystemer, det har gjennom tiden været flere forsøk på dette, men ikke noe har slått igjennom og programmerer yrket består. Det er derfor naturlig å se på sluttbrukerprogrammering innenfor et mer snevert domene, dette ble gjort i en studie med smarte hjem som domene. I versjon 1 av regelbyggeren i prototypen til (7) klarte bare 3 av 10 datastudenter å bygge egne regler. Denne prototypen var basert på et eget språk som var noe komplekst å lære. De utviklet en versjon 2 med en puslespillmetafor istedenfor språket i versjon 1. Her klarte hele 75 % av datastudentene å lage egne regler. Dette gir en peker på at grensesnittene må utforskes videre i forbindelse med sluttbrukerprogrammering. Det handler ikke nødvendigvis bare om hvor lett det er å lære å bruke programmet, men også om hvor lett det er å verifisere at ønsket funksjon blir nådd ved hjelp av den produserte regelen. Et viktig aspekt er også om nok kompleksitet kan utrykkes i programmet når ulike metaforer benyttes.

KidSim et verktøy laget til barn for å lage egne animasjoner, med fokus på interaksjonsmetoden. De endte opp med programmering ved demonstrasjon som den beste løsningen. Hvor de ulike reglene demonstreres på skjermen av brukeren og programmet spiller de av etterpå. For å unngå at reglene ble for kompliserte å demonstrere ble de delt opp i mange små regler. Dette betyr at brukerne lager små komposisjoner som kan settes sammen til det endelige resultatet (5). Dette verktøyet lot barn programmere sine egne spill og utviklerne lekte med tanken om å lage noe tilsvarende for voksne, der det var mulig å programmere på tilsvarende måte med

agenter og demonstrasjon(8). KidSim slo aldri igjennom, men vokste videre på dette og er i dag det mest populære utvikler miljøet for Mac OS X programmering(9)(10).

Det er knyttet uvisshet til hvordan en datamaskin blir når brukerne programmerer den, i dag er det vanlig at en programmerer gjør sine antagelser om brukeren og lager programmene der etter. I de fleste programmer er der muligheter hvor brukeren kan tilpasse programmet, flere medieprogram kan for eksempel bytte utseende. Dette er ikke programmering da det kun kan velges mellom forhåndsdefinerte muligheter. Hadde det derimot vært mulig å spesifisere hver enkelt knapp, gjerne med egen oppførsel, eller bytte plasser på dem, ville det vært sluttbrukerprogrammering.

Bonnie Nardi har forsket mye på sluttbrukerprogrammering og særlig de vellykkete konseptene. Den største suksessen; regnearket, er meget kraftig, men samtidig veldig enkelt. Dette betyr at brukeren enkelt kan gjøre summeringer og bruke de fire regneartene. Det har i den seinere tid også blitt enkelt å programmere grafer og illustrasjoner. Regnearket var i starten en digitalisering av regnskapsboken og fordelene ved å ha dette digitalt førte til en stor vilje for å ta de nye systemene i bruk. CAD programmer har også vist seg å være en vellykket form for sluttbrukerprogrammering. Begge har en viktig fellesnevner at brukeren skaper innholdet og programmerer hvordan programmet skal virke. I et regneark er det brukeren som forteller hvilke celler som skal summers osv, dette gjør at utviklerne slipper å skreddersy programmet for brukeren og dermed begrense mulighetene. CAD programmer er også viden kjent for å ha mange muligheter og at brukerne bestemmer hvordan ting skal løses. Den viktigste fellesnevneren for disse ulike programmene er et oppgaveorientert språk. Et eksempel på dette kan være en strikkeoppskrift som vist Figur 1. Denne vil være vanskelig om ikke umulig å lese for en som ikke kan strikke, mens den er selvforklarende for en strikker. Det å lage et språk for å løse bestemte oppgaver er i følge Nardi veien å gå mot sluttbrukerprogrammering. Dette kan videre begrunnes med at fjerdegenerasjons programmeringsspråk fortsatt er programmering og ikke noe vanlige sluttbrukere er i stand til å benytte (11).



Figur 1 oppgavespesifikt språk, strikkeoppskrift (47)

2.2 Metaforstudier

En brukergrensesnittmetafor er et forsøk på å lage brukergrensesnittet likt noe brukeren kan kjenne seg igjen i. En av de mest brukte metaforene er et skrivebord, både Apple Macintosh og Microsoft Windows har brukt og bruker denne metaforen. Her blir fysiske objekter fra den virkelige verden representert på skjermen, dette gjelder dokumenter, verktøy (programmer) og

søppelbøtten. Det hele styres ved hjelp av en pekerenhet (mus) og et tastatur for å skrive kommandoer. På en Macintosh kan brukeren dra en fil ved hjelp av pekeren til søppelbøtten for å slette den. Den er ikke helt borte enda, for akkurat som i den fysiske verden ligger filen i søppelbøtten. Der blir den værende helt til du som bruker velger å tømme søppelbøtten, når det er gjort, har filen forsvunnet til evig tid. Denne type og andre metaforer lar brukerne lettere danne en mentalmodell av systemet. Det er viktig at alt er bygget logisk opp og virker fornuftig i forhold til brukerens mentale modell. Apple gjorde et rart valg, i motsetning til x86 maskiner hadde de ikke en fysisk knapp for å løse ut disketter, dette måtte gjøres ved hjelp av skrivebordet. Her ville kanskje en skuffe vært en passende analogi, men de valgte at brukeren måtte dra disketten til søppelkurven, for å løse den ut. Dette førte i starten til store forvirringer da brukerne var redd for å slette innholdet på disketten. Heldigvis kom disketten ut av seg selv når maskinen ble skrudd av og dette ble løsningen for brukere som var redd for å slette innholdet på en diskett. Apple ble etter hvert klar over problemet og løste det ved å skrive fysisk på maskinen hvordan en diskett kunne løses ut (12). Det er slutt på diskettens dager og nyere maskiner fra Apple kommer nå med CD stasjon, der er en egen knapp på tastaturet for å løse den ut og det virker ikke lenger å dra den til søppelbøtten. Det eksisterer mange kjente og mindre kjente metaforer, felles for de alle er at de har noen fordeler og ulemper. Ikke alle metaforer er like godt egnet i alle sammenhenger og det må alltid vurderes når og hvilke metafor om noen i det hele skal benyttes.

Metaforer kan også kombineres, det er ingen hindring å bruke flere metaforer i ett og samme program. Det viktige er hvilke metaforer brukerne kjenner fra før og hvordan de passer sammen med den konseptuelle modellen (se avsnitt 2.8). Det er viktig å bruke en metafor, eller kombinasjon av metaforer, hvor de ulike objekttypene med egenskaper og relasjoner i den konseptuelle modellen blir godt ivaretatt. Brukerne vil danne seg en mental modell over hvordan systemet virker på bakgrunn av metaforen og hvordan de opplever systemet. Denne modellen bør ha samme logikk som den konseptuelle modellen, da det vil forhindre logiske brister. Problemet er at utviklerne lager metaforen ut ifra en konseptuel modell som brukerne aldri ser(13) Den mentale modellen er noe som skapes i hodet til den enkelte bruker og er forskjellig fra bruker til bruker. Modellen hjelper brukeren og forklare hvorfor systemet oppfører seg som det gjør og hva som er mulig. Den mentale modellen blir derfor stadig utviklet hos den enkelte bruker etter hvert som han/hun bruker systemet(14).

I kommunikasjon mellom mennesker er ansiktet og dets utrykk noe av det viktigste for å fremheve et budskap. Bruk av datamaskinen er kommunikasjon via skjermen til brukeren og tilbake, ideelt sett burde da skjermen på en datamaskin fylles med et ansikt (15). Det å lage dataprogrammer som er lette å bruke er ofte krevende. Det kreves brukerinvolvering i hele utviklingsfasen, og god kunnskap om menneske maskin interaksjon. For hjemmeprogrammering er det vanskelig å se for seg at et ansikt kan brukes som metafor eller som hovedskjermbilde. Problemet blir at brukerens mål med å bruke et program ofte er datamanipulasjon, produsere data eller se data/informasjon. I sluttbrukerprogrammering skal det være lett for brukeren å utrykke seg, da teknologien ikke er moden nok for tale og utrykks gjenkjenning kreves det skjermplass for å verifisere brukerens valg. Dersom dette ikke skal verifiseres må det vises på skjermen, ved at brukeren gjør programmeringen visuelt på skjermen ved hjelp av visuelle

programmeringsspråk. Visuelle programmeringsspråk vil hele tiden gi tilbakemelding til brukeren og krever stor skjermplass. Et ansikt til skjermbilde ville derfor ikke her heller være ønskelig. I tilegg er det ikke vanlig å gjøre manipulasjoner på menneskers ansikt ved vanlig ansikt til ansikt kommunikasjon. Det blir derfor rart om en bruker plutselig skal komponere tjenester på et ansikt på skjermen.

2.3 Leksikalsk, syntaktisk og semantisk

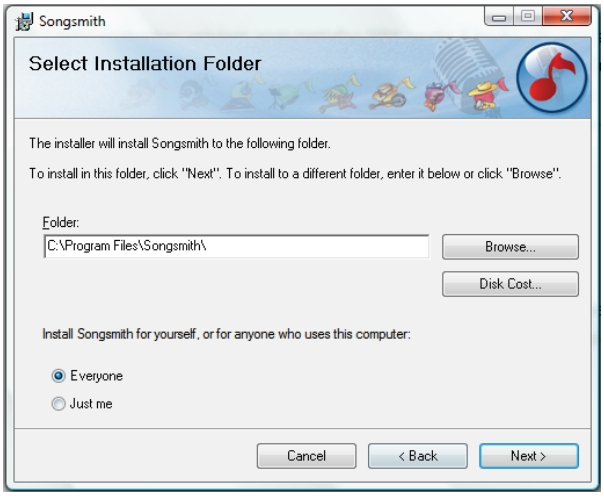
Et Brukergrensesnitt kan deles opp i tre ulike deler, hvor den semantiske er koblingen fra brukergrensesnittet mot det underliggende, dette ble også kalt programgrensesnittet tidligere. Denne delen er ikke relevant i denne oppgaven og vi lar det ligge med at brukergrensesnittet trenger tilbakemelding fra noe underliggende for å respondere riktig ovenfor brukeren. Det Leksikalske er hvordan ting presenteres på skjermen, alt fra ord i en meny, til de ulike objektene på skjermen. Dette kan sammenlignes med ord i et vanlig språk. For at et språk skal komme til sin rett må det bygges setninger, det samme gjelder i et brukergrensesnitt. Det syntaktiske er setningsbygningen, som definerer hva brukeren kan gjøre på skjermen og forklarer hvordan interaksjonen virker. Det syntaktiske setter begrensninger på hva brukeren kan og ikke kan gjøre (16). De to sistnevnte, leksikalsk og syntaktisk, utgjør til sammen en brukergrensesnittmetafor og det er viktig å se disse to i en sammenheng og ikke utvikle de hver for seg.

2.4 Ulike metaforer

I denne oppgaven fokuserer vi på fem ulike metaforer, to av dem, puslespill og objekt kobling er bildelige og utradisjonelle. De andre er basert på tradisjonelle retningslinjer for brukergrensesnitt. Under vil vi presentere teori og hvordan vi har funnet inspirasjon om de ulike metaforene vi ønsker å teste i et smarthjem domene.

2.4.1 Veiviser

Metaforen veiviser stammer fra den verdenskjente boken; The Wonderful Wizzard of Oz (17). En barnenovelle om en trollmann (veiviser), han hjelper ulike ting/mennesker/dyr som kommer til han for å få søke hjelp. Hjelpen de får og måten det skjer på er ikke helt som de tilreisende hadde sett for seg, men resultatet blir bra.



Figur 2 Veiviser installering av Songsmith (18)

I dataverden har veiviser som metafor blitt vanlig å bruke når noe skal installeres eller settes opp for første gang. Det er ikke trolldom som skjer, men mer at brukeren blir spurt på en enkel måte hvordan de ulike tingene skal være. Spørsmålene som blir stilt er ofte enkle og få, ofte er der også egne sider eller steg i veiviseren som bare viser informasjon. Dette gir brukeren god veiledning og forklaring på hva som vil har skjedd og vil skje videre. Navigering i en veiviser består vanligvis av de fire valgene presentert under:

Liste 2 Navigeringsvalg i en vanlig veiviser

- Avbryt
- Tilbake
- Frem (neste)
- Ferdig (fullfør)

Navigeringen skjer mellom de ulike sidene eller stegne. Stegene i en veiviser bruker en arktavle analogi, hvor brukeren skifter ark og følgelig alt innhold endres ved å bytte mellom stegene. Det er ikke uvanlig at brukeren må gjøre et aktivt valg på et steg (ark) for å komme videre til neste steg. Dette benyttes for at brukeren ikke skal glemme å avgjøre viktige valg. Figur 2 viser veiviseren for installasjon av Songsmith (18), og vi ser at brukeren får valg om hvor programmet skal installeres. Dette valget er ferdig utfylt i tilfelle brukeren ikke har noen preferanser på hvor det skal installeres. I tilegg kan brukeren velge hvilke andre brukere av datamaskinen som skal ha tilgang til det ferdig installerte programmet. Veiviseren vist her er veldig klassisk med navigeringsvalg nede til høyre. For veivisere på websider har det også blitt vanlig med en statuslinje for hvor langt brukeren har kommet i prosessen.

En veiviser er lett å lære for brukere, de gjør få feil og løser oppgaver ganske hurtig ved hjelp av en. Til tross for dette er den ikke alltid like egnet, det har vist seg at enkelte brukere finner den kjedelig og lite attraktiv å bruke(19).

2.4.2 Vindu, ikoner, menyer og peking (WIMP)

WIMP er en klassisk og velbrukt metafor fra Xerox Star og det var denne som gjorde datamaskiner tilgjenglig for alle fra midten av 80-tallet. Da metaforen slo igjennom begynte flere og flere å bruke den, og brukerne ble vant til den. Dette førte videre til at utviklingsverktøy begynte å støtte utviklingen av WIMP som standard brukergrensesnitt, noe som gjør det enklere og billigere å bruke denne metaforen i forhold andre metaforer. Selve metaforen kan oppsummeres i følgende punkter:

Liste 3 Vanlige handlinger i WIMP

- Velg ønsket objekt ved å trykke på det (med musen)
- Velg en handling fra en meny eller tastaturet
- Fyll ut noen felter i en boks
- Trykk på <OK> for å bekrefte handlingen og se resultatet

Dette er en kompleks vei å gå for å gjøre endringer, og handlingen gjøres ikke direkte på objektet. Denne måte å jobbe på er indirekte manipulasjon, som krever mer opplæring av brukeren(20).

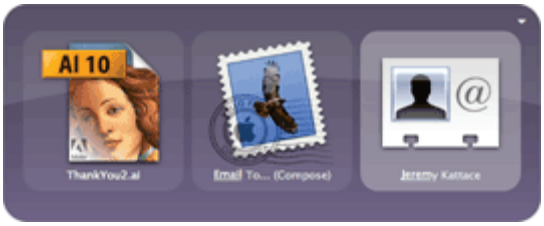
2.4.3 Grafisk kommandolinje

En Kommandolinje kan sammenlignes litt med WIMP:

Liste 4 Steg i en kommandolinje

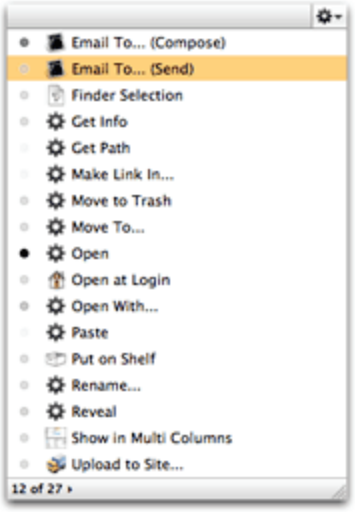
1. Skriv inn navn på ønsket objekt (i stedet for å velge med musen)
2. Skriv inn ønsket handling (i stedet for å velge fra meny)

3. Skriv inn ønskede parameter (i stedet for å fylle ut felter)
4. Trykk Enter (i stedet for ok knappen)



Figur 4 Quicksilver send et bilde med e-post

Innskriving av kommandoer kan være vanskelig, særlig for de som sjeldent bruker systemet. Det er lett å glemme hvilke parameter de ulike kommandoene har og ikke minst hvilke kommandoer som er tilgjengelige på hvilke objekter. Fordelen er at det går veldig fort når brukeren har lært de ulike kommandoene. Ideen til en grafisk kommandolinje er hentet fra Blacktree sin Quicksilver applikasjon for Mac os X 10.6. Figur 4 viser dette programmet når det er konfigurert til å sende et bestemt bilde med e-post til en bestemt kontakt. Det vi ser er en ferdig komposisjon laget med den grafiske kommandolinjen, det som gjenstår for å kjøre komposisjonen er et trykk på <Enter>. For å velge hvilke filer og handlinger brukeren ønsker å gjøre noe med, benyttes tastaturet. Ved å skrive bokstaver vil en liste tilsvarende den vist i Figur 3 bli synlig. Denne listen er sortert etter hvilke bokstaver brukeren skrev på tastaturet. Listen kan navigeres i, ved hjelp av pilene på tastaturet, og et valg bekreftes med <Tab> for å gå videre. Det er ikke et fast antall steg i denne metaforen, dette varierer etter hva som velges. Hadde det blitt valgt åpne i stedet for send e-post i Figur 4 ville ikke det siste valget ha kommet opp (valg av kontakt). Det siste valget i denne komposisjonen er et konfigurasjons valg som ikke er relevant for denne oppgaven da vi kun forholder oss til komponering og ikke konfigurasjon. Brukere av Quicksilver i datamiljøet ved NTNU har utalt at den er genial, men dog litt vanskelig å bruke. Dette er et nytt konsept hvor det har blitt gjort lite forskning, noe som gjør den til en spennende metafor å undersøke nærmere.



Figur 3 Quicksilver valg av handling

2.4.4 Puslespill

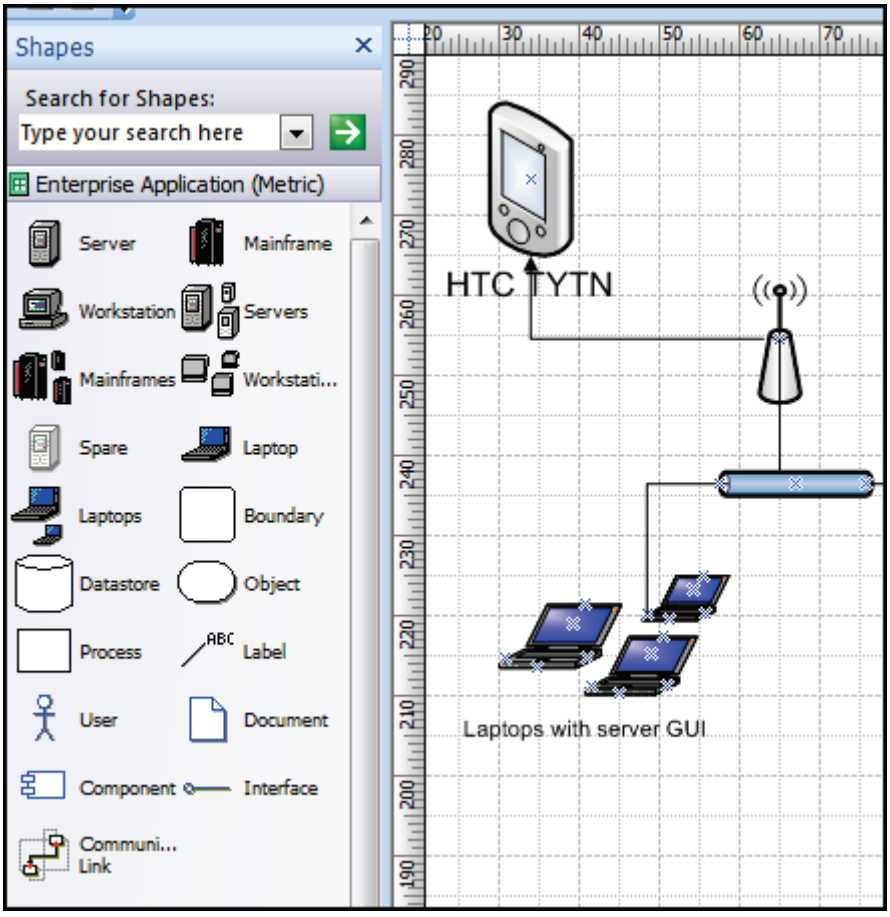
Puslespill er en velkjent metafor hvor brukeren legger et puslespill på skjermen for å løse en gitt oppgave. Dette er lettvint og mange brukere kjenner seg igjen, men det er fort at det blir for liten plass på skjermen. Det endelige resultatet blir heller ikke et bilde som i den virkelige verden, men en samling brikker.

I en Lego™ metafor blir det endelige resultatet et byggverk, på lik linje med det fysiske resultatet av Lego bygging. Problemet er at en dataskjerm er 2D og Lego ligger i planet, dette er derfor en metafor som passer bedre til komplekse oppgaver der det er behov for en ekstra dimensjon i forhold til puslespillet. Lego™ i 2D blir det samme som et puslespill og derav de samme egenskapene(21). Det er viktig å poengtere at ett byggverk av brikker er en komposisjon. En samling komposisjoner blir på lik linje med puslespillet en samling byggverkt eller brikker og ikke en by av Lego.

2.4.5 Objektkobling

Objektkobling er en metafor for å koble sammen ulike objekter fysisk, ideen er hentet fra ulike modelleringsverktøy: Microsoft Visio (vis i Figur 5), Dia og moderne websmørjer (mashup) som Yahoo pipes og popfly (avsnitt 2.5.3). Denne metaforen er basert på direkte manipulasjon hvor brukeren kobler objekter sammen ved hjelp av piler. Objektene kan ofte konfigureres ved hjelp av å skifte nivå. Dette gjøres vanligvis ved hjelp av å klikke på et objekt, eller et symbol og en dialog hvor enkelte elementer kan spesifiseres blir åpnet.

Figur 10 på side 17 viser hvordan skifte i nivåer er løst i popfly. Denne modellen er kjent for å være effektiv i bruk og krever lite opplæring (avsnitt 2.5.2). Der er også et problem knyttet til hvor mye en bruker får plass til på skjermen og hvordan skjermen skal organiseres. En løsning er å sette restriksjoner på hvordan elementene kan plasseres og gjøre det mulig å flytte en gruppe objekter. Funksjoner for å sette objekter på linje eller kjente mønster gir bedre plass på skjermen. Alt dette lar brukeren raskere lage en komposisjon med mindre feil(22).

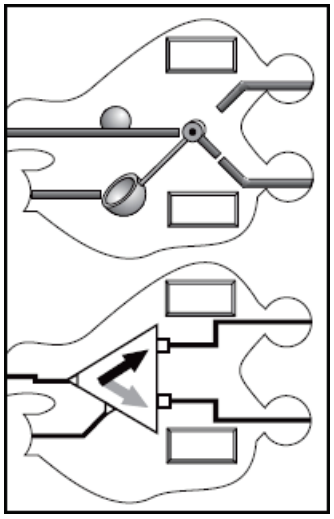


Figur 5 Microsoft visio

2.5 Visuellprogrammering

Visuelle programmeringsspråk kom i 1975, den gangen var det instruksjoner i FORTRAN som ble visualisert(23). Utviklingen gikk sakte fremmover og i 1993 ble det holdt en workshop hvor de reiste spørsmål rundt hvem visuell programmering var for; programmerere eller sluttbrukere.

Workshopen fokuserte også utfordringene med skalering til større applikasjoner. I dette arbeidet konstanters det at tekst er mer kompakt enn visuelle objekter på skjermen. Det ble også pekt på utfordringer rundt hvordan en sekvens kan representeres visuelt (24). I 1999 ble flere av disse spørsmålene prøvd besvart i form av å begrense visuellprogrammering til et bestemt domene og lage en kontekst for sluttbrukere. Forsøket viste at tiden en sluttbruker bruker på å lære et system synker ved bruk av metaforer. Dog var tiden de brukte på å løse en oppgave lik med og uten metaforer. Metaforen de benyttet var av bildelig og velkjent for de fleste, en ball som ruller. De gjorde videre et forsøk hvor metaforen var mer abstrakt i form av et diagram, resultatet ble at programmerere var 100 % bedre enn nybegynnere (23).



Figur 6 Visuell programmering med og uten metafor

Labview er et vellykket eksempel på et visuelt programmeringsspråk, første versjon kom ut for over 20 år siden og er enda mye brukt for å styre prosesser. Labview er en visualisering ved hjelp av grafer som gjør det mulig for kompetente sluttbrukere å lage programmer i språket G. Dette språket har kun muligheten for en delmengde av mulighetene i et fullverdig programmeringsspråk som C. Til gjengjeld er språket mer effektivt på enkelte analytiske funksjoner, og i så måte velegnet for domenet(25).

2.5.1 Uttrykksfullhet og kompleksitet

Der er en avveiing mellom hvor mye et system klarer å utrykke og hvor vanskelig sluttbrukerprogrammering er å bruke. For eksempel blir et puslespill vanskeligere å legge desto flere brikker det er å velge i mellom. Det å lage et sluttbrukerprogrammerings program er derfor ingen enkel oppgave, utviklere må hele tiden tenke på hvor mye sluttbrukeren skal kunne programmere i forhold til hvor vanskelig det blir å bruke. En løsning for å få flere muligheter er å dele programmet inn i flere nivå. Et nivå vil si å dykke inn i et element ved hjelp av et nytt skjermbilde, dette kan brukes for å løse mer kompliserte/konkrete problem. For eksempel kan du trykke på en puslespillbrikke og komme til et nytt puslespill eller skrive et manus for den valgte brikken. Her er det viktig å ivareta en ”en til en” knytting mellom de ulike nivåene. Hvis du i det første puslespillet trykker på en brikke, bør tilsvarende brikker også ha mulighet til å bli trykket på. Det må videre være mulig å knytte de andre brikkene eller elementene i en valgt brikken sammen uavhengig av hva som blir gjort i et konfigurasjonsnivå. Ved bruk av ulike nivå, der brukeren kan dykke inn i systemet er det viktig å passe på at brukeren ikke må skifte nivå for ofte. Dersom brukeren må opp og ned fra de ulike nivåene for å gjøre vanlige ting kan systemet fort blir for tungt å bruke.

De ulike nivåene kan gjerne være skilt ved ulike interaksjonsmuligheter og hva som vises på skjermen. I den moderne dataverden er det både bilder, tekst, video, lyd i mange dataprogrammer. Dette kan styres ved hjelp av tastatur, mus, lys (webkamera) og stemmestyring. Bruken av disse nye mulighetene må gjøres forsiktig så brukeren ikke må lære unødvendige nye konsepter. Det er også ytterst viktig å passe på at brukeren ikke må lære noe nytt mellom de ulike nivåene (26).

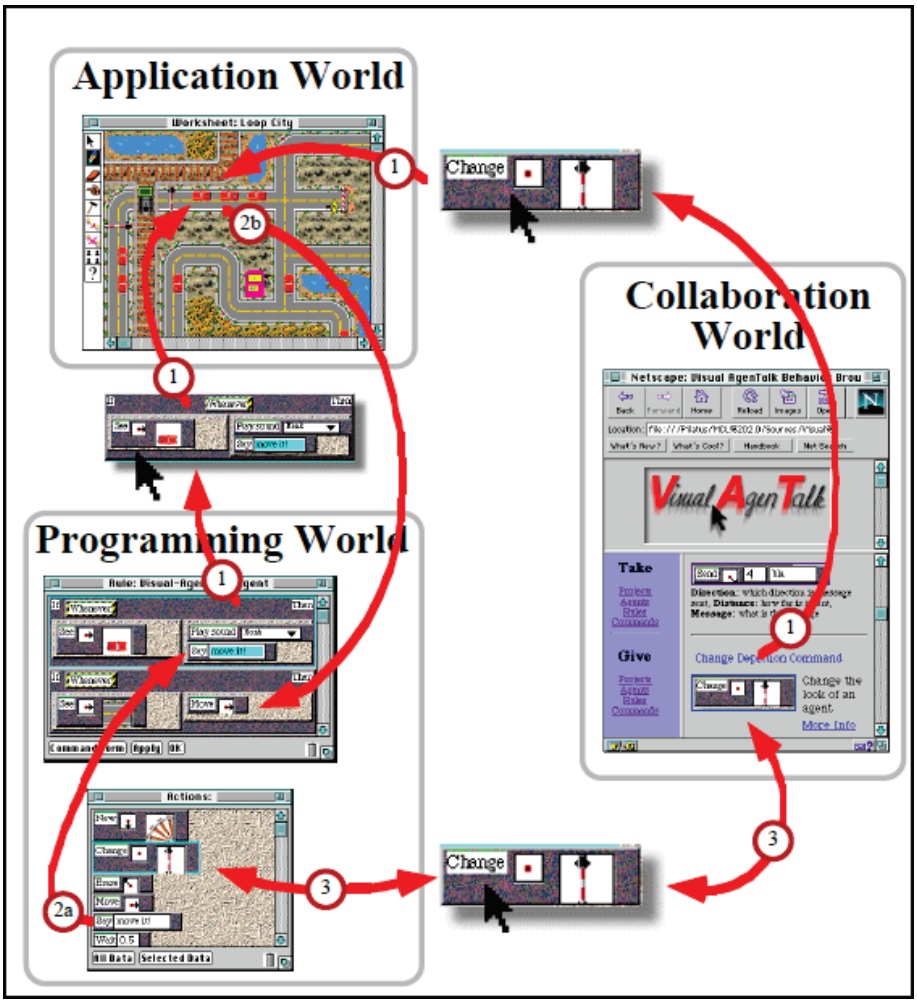
Det er ikke nødvendigvis ett sort hvitt blide å hevde at jo mer sluttprogrammering kan utrykke jo vanskeligere blir det å bruke. Med rett inndeling, bruk av interaksjonsmuligheter og nivåer er det mulig å utrykke mer enn ved tradisjonelle sluttbrukersystemer. Ser vi på et sluttbrukerprogrammeringsspråk isolert sett blir det vanskeligere å bruke jo mer det kan utrykke. Denne avveiingen mellom hvor mye sluttbrukeren trenger å utrykke og kompleksiteten til det ferdige systemet, vi mener her kompleksiteten ovenfor sluttbrukeren og ikke utviklerne.

2.5.2 Taktilprogrammering

Visuellprogrammering er først og fremst egnet for programmerere og i utgangspunktet for vanskelig for vanlige sluttbrukere. Taktil programmering er en videreføring av visuell programmering der de sosiale aspektene av programmering blir tatt høyde for. Dette gjør det enklere å dele "koden" og samarbeide om å produsere et program. Et annet mål for taktil programmering er å overvinne kompleksiteten til tradisjonell programmering og samtidig beholde mulighetene (avsnitt 2.5.1). I Figur 7 er et eksempel på et taktilt programmeringsrammeverk, hvor det mulig å lage visualiseringer til SimCity[1]. Vi ser at det er laget i ulike nivåer etter hva som skal gjøres, disse nivåene er her kalt verdener. Der er en verden for programmering, en for å se resultatet og en for å dele informasjonen. Dette gjør at mer kan utrykkes samtidig som det er enkelt å bruke. Det er lettere å dele informasjon i delingsverdenen enn å programmere vanlige tcp tilkoblinger for deling av informasjon. Vi ser også at det er en tett kobling mellom de ulike verdene. Denne koblingen er viktig og gjør det enkelt for brukerne å bytte mellom verdene uten at de må lære noe nytt. For at verdenene skal være like er det viktig å utvikle dem samtidig og inngå kompromiss mellom dem.

Det holder ikke lenger å lage bedre og bedre visuelle programmeringsspråk til sluttbrukerprogrammering, det kreves mer. Det å krysse grenser mellom ulike aspekter ved programmeringen er en viktig faktor og da er taktil programmering veien videre(27).

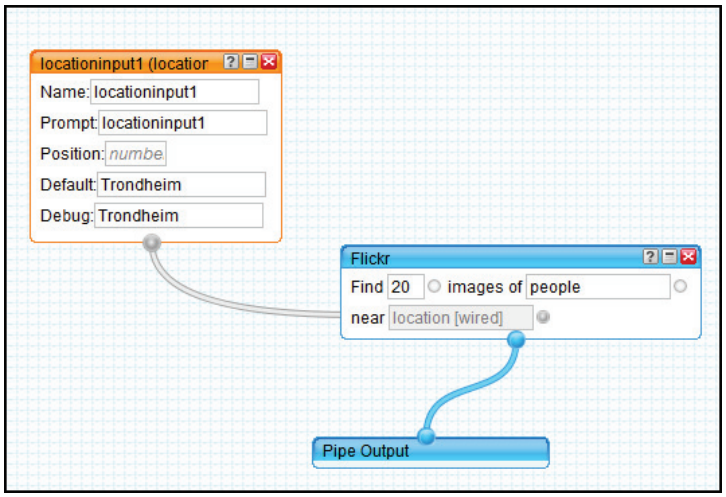
[1] Et spill for å lage og simulere livet til mennesker i en by, ta kontroll over en by eller å bygge sitt eget virtuelle samfunn (49).



Figur 7 Eksempel på taktil programmering (27)

2.5.3 Websmørjer (mashups)

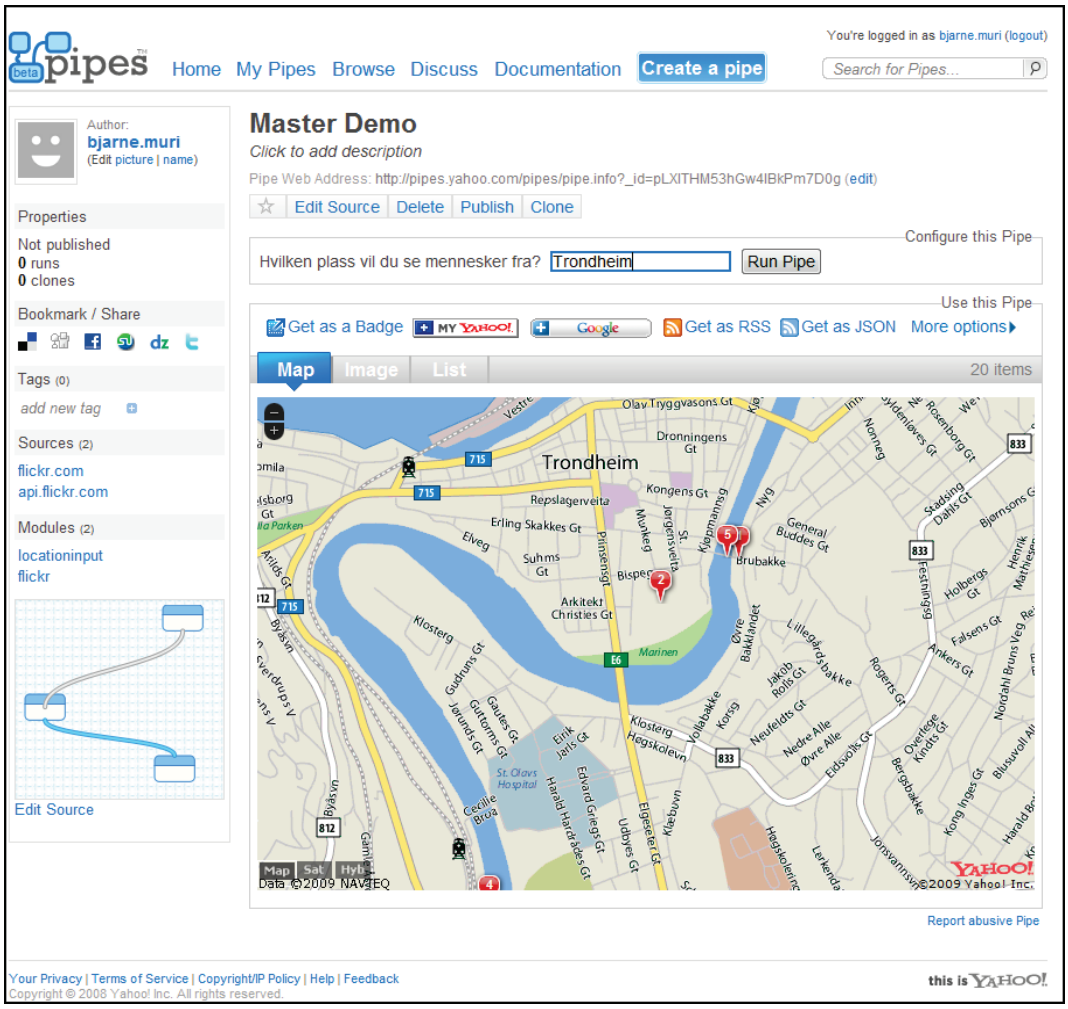
Det har blitt mer og mer vanlig å lage ulike applikasjoner som henter data og informasjon fra ulike nettsider for så å presentere resultatet på en annen nettside. Dette har fått det engelske navnet mashup, det samme som sammenslåing av musikksanger. Websmørjer (mashups) er veldig populært i enkelte miljøer og lar brukere lage avanserte tjenester på kort tid. En websmørje består av en datakilde, enten lesing fra andre nettsider, tjenester eller brukerdata. En annen inrettning til



Figur 8 Komposisjon i Yahoo pipes

behandling av innsamlede data og til slutt en siste inrettning for å spesifisere presentasjonen av resultatet for brukeren (28). Yahoo Pipes er en av de mest brukte og best likte verktøyene for å lage websmørjer. Yahoo pipes bruker programmering via eksempel. Den enkleste måte å finne hjelp til hvordan lage en pipe (Yahoo sitt navn for websmørje) er via videoer med eksempler i, for eksempel (29).

Figur 8 viser hvordan en komposisjon i Yahoo pipes ser ut. Dette er en veldig enkel komposisjon hvor brukeren blir spurt etter en lokasjon (den røde innrettningen). Der er også mulig å skrive en standard lokasjon og en egen lokasjon til bruk ved feilsøking. Lokasjons innrettning er knyttet til den blå Flickr[2] innrettningen. Dette betyr at Flickr innrettningen får informasjon om lokasjonen fra den røde innrettningen. Vi ser videre at det er spesifisert at den blå innrettningen søker etter mennesker (people). Til slutt knyttes flickr til "pipe out" for å signalisere at vi er ferdig med oppsettet. Det å koble sammen ting på denne måten krever en del programmeringserfaring da det er en del ukjente begreper og konsepter for vanlige brukere(28). Dette er en særdeles enkel pipe og det er ikke uvanlig med mer avanserte ting som regulæreutrykk.

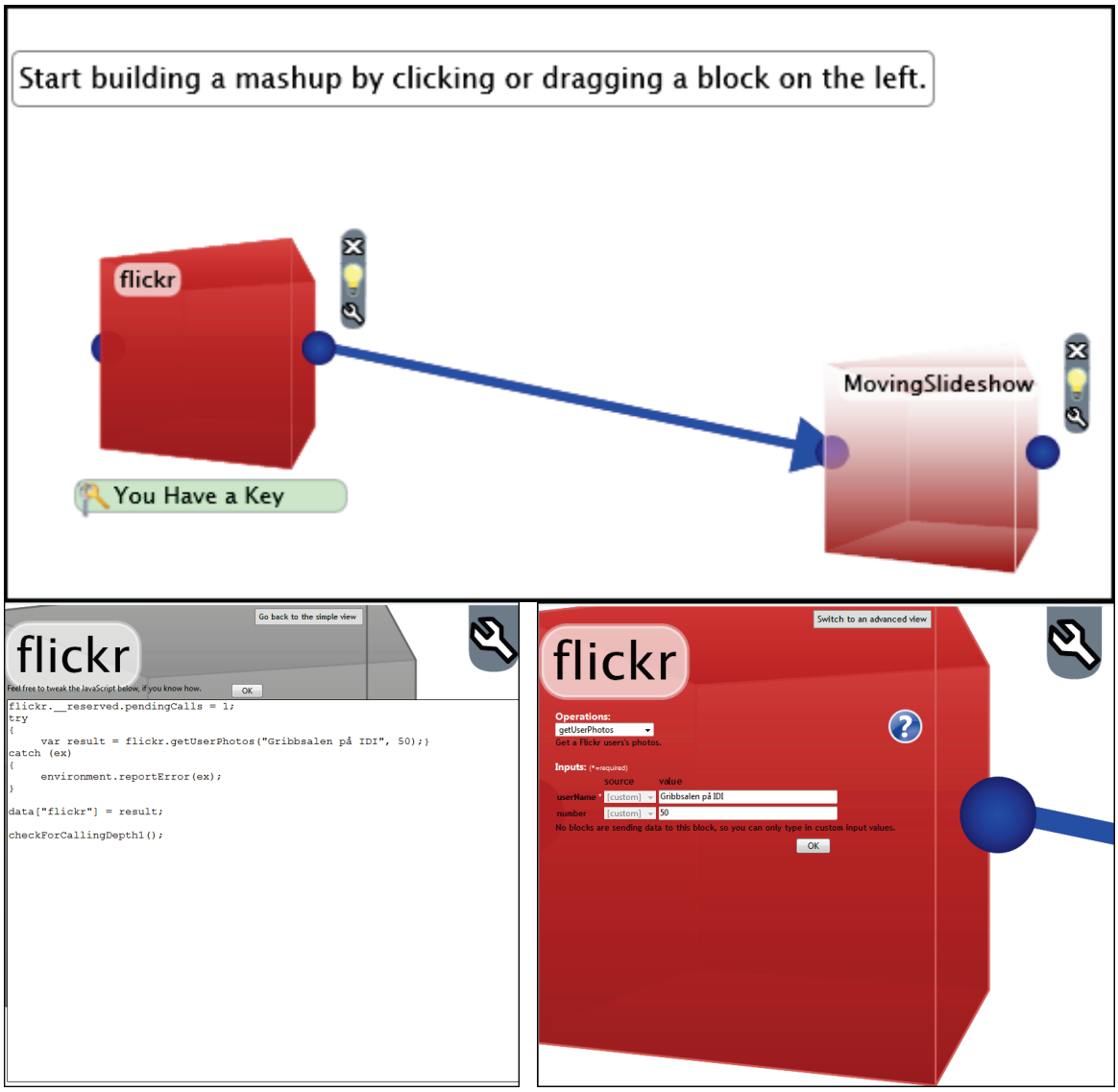


Figur 9 Visning av resultat Yahoo pipes

[2] Bildedelingssamfunn på internett, http://www.flickr.com/

Yahoo pipes er veldig poppulært for kyndige, da de bruker resultatet videre i en ny tjeneste. For eksempel er det vanlig å lage egne nyhetsmatinger (rss-feed) med informasjon fra ulike nettsteder. Det er få muligheter i Yahoo pipes til å spesifisere utseende til resultatet visuelt sett. Det er mulig å spesifisere hvordan tekststrenger og XML data er bygd opp, men ikke noe visuelt som en sluttbruker ofte er interessert i. Figur 9 viser resultatet av websmørjen vi bygde i Figur 8. Vi ser at den blå innrettningen er byttet ut med et tekstfelt og en knapp. I tilegg ser vi at resultatet presenteres i et kart, dette helt uten at det er spesifisert et eneste sted. I selve konfigurasjonen jobbet vi med lokasjon og bilder, ikke kart. Det er kanskje her Yahoo pipes og Microsoft popfly er mest forskjellige. Øverst i

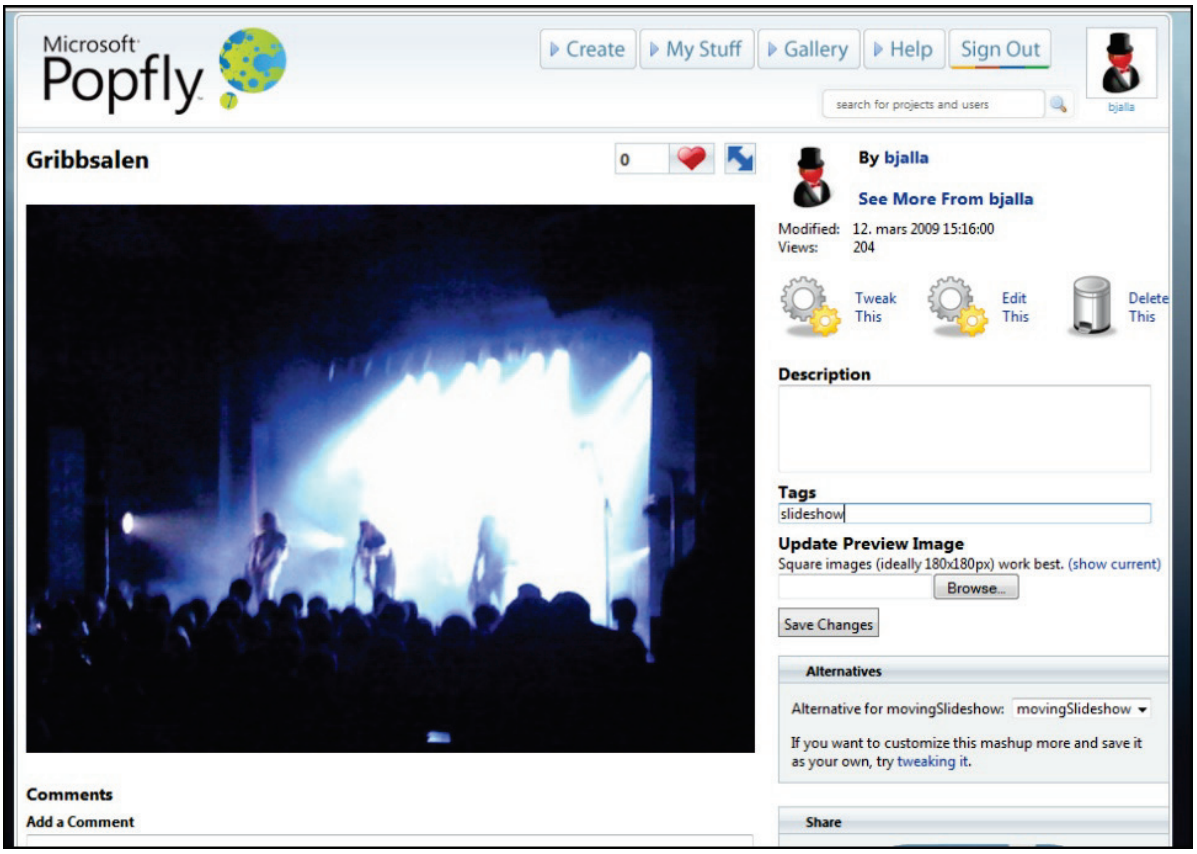
Figur 10 Er det bilde av en tilsvarende komposisjon i popfly, her har vi koblet flickr til en lysbildefremvisning. Det er samme måte å koble sammen elementer på, ved å dra streker. Videre skiller de seg ved at popfly benytter ulike nivå (se avsnitt 2.5.1) og gjør det dermed enklere for brukerne å lage mer avanserte komposisjoner. Nederst i figuren til høyre ser vi standard innstillingene til flickr boksen/innrettningen. Her kan vi velge hva vi vil ha fra flickr og legge på noen parameter for å spesifisere mer avanserte valg. Dette er egentlig bare en forstørret versjon av den opprinnelige komposisjonen. Nede til venstre i figuren ser vi enda et nivå, programmeringsnivået, her kan kompetente personer skrive egen kode for å konfigurere innrettningen/komponenten/boksen enda mer enn de forhåndsdefinerte feltene (mulighetene).



Figur 10 Komposisjon av websmørje i popfly, øverst hele komposisjonen, nede til venstre avansert, nede til høyre er vanlig konfigurasjon

Figur 11 er resultatet av websmørjen i Figur 10, her er lysbildefremvisningen vi valgte og bildene roterer og panorerer akkurat som spesifisert i den røde innrettningen ”MovingSlideshow”. Her er det på lik linje med Yahoo pipes mange muligheter for å dele innholdet. Hovedvekten av delingsinformasjon er på en kodesnutt du kan lime inn på hjemmesiden din for så å få sett lysbildefremvisningen på din egen hjemmeside.

Popfly og Yahoo pipes er litt forskjellige, men samtidig er de ganske like. Yahoo pipes kom først og blir mye brukt, mens popfly har en mindre tilhengerskare. Problemet med websmørjer i sluttbrukerprogrammerings sammenheng er at vanlige brukere ikke skjønner konseptene bak de ulike innrettningene (28), noe som gjør det vanskelig å bruke.



Figur 11 Resultat av komposisjon popfly

2.6 Sluttbrukerkomponering

Det er viktig å velge den rette metoden for en bruker å samhandle med en datamaskin for å komponere tjenester. Datamaskinen snakker et språk mens mennesker et annet, en løsning er at brukerne lærer seg programmering. Dersom et domenespesifikt språk blir brukt og motivasjonen til brukerne er sterk nok går dette bra. Både formelspråket i regneark og makroer i tegneprogrammer er bevis for dette. Brukere som skal komponere egne tjenester i hjemmet og er motiverte for det vil derfor kunne lære seg et eget et språk for dette. Fra den andre siden ville det vært lettere for brukeren å skrive inn sitt eget språk og la datamaskinen tolke dette. Dette ville ført til vansker med dialekter, verbformer og ulike skrivemåter av ting med lik betydning. Datamaskinen ville ofte misforstå og frustrasjon hos brukeren kan oppstå. Et eget oppgavespesifikt språk vil også ha en tett relasjon til det datamaskinen virkelig gjør, i motsetning til naturlig språk. Visuell programmering får høyere abstraksjonsnivå ved bruk av bilder på objekter. Til fordel kan bilder leses av alle uansett hvilket språk de snakker. Ved bruk av bilder er der mye å være oppmerksom på; bilder kan for eksempel ha ulik betydning i ulike kulturer. Skjermplassen er ofte en begrensning og bilder tar mer plass enn tekst. Ved bruk av bilder blir det da større fare for å få skaleringsprogram som følge av plassmangel på skjermen (11).

Det å komponere tjenester i smarte hjem skiller seg litt fra programmering da brukeren typisk ikke vil gjøre for avanserte oppgaver, og det hele skjer innenfor et bestemt domene. Bildelige objekter har et skaleringsproblem, men er lett å forstå og vil derfor være egnet innenfor et

domene med begrenset antall objekter og oppgaver der skalering ikke er noe problem. Abstraksjonsnivået ved bruk av bilder kan være en fare, da brukeren fort kan føle at de mister kontrollen over systemet, eller ikke helt skjønner hva som skjer (11). For å unngå dette er det viktig med "en til en" knytting mellom brukerens mentale modell og det som presenteres på skjermen.

2.7 Smarte hjem

I løpet av de siste 15 årene har teknologien blitt allestedsnærværende, alt fra vaskemaskiner til veggur inneholder små mikroprosessorer. Disse har et stort potensial i samhandling og kommunikasjon mellom seg. Kommunikasjon mellom ulike enheter er ingen ny ide, allerede på 80-tallet var det tanker rundt intelligente hjem som det het den gangen. Det har samtidig skjedd mye, med hjemme våre opp igjennom historien, det er nå vanlig med elektrisitet, telefon, varmt og kaldt vann samt et avløps system. Der er allerede mange maskiner for å gjøre hverdagen letter, vaskemaskin, oppvaskemaskin, brødbaker og strykerjern er bare noen av mange ulike hjelpemidler vi har. Noen av dem er intelligente og tilpasser seg etter andre forhold, en markise kan for eksempel gå opp og ned avhengig av været. En annen retning innen smarte hjem er underholdning og multimediasystem. Multimediasentre er ikke intelligente på samme måte som markisene da de oftest er brukerstyrte og kontrollerte. Der finnes multimediasystem hvor lyden er juster i forhold til bråk i rommet, dette blir en krysning mellom de intelligente og brukerkontrollerte systemene. Brukere av denne teknologien er også brukere av et hus. Beboere er det som gjør et hjem til et hjem og blant dem er det store forskjeller i synet på et smarthjem, ikke alle vil ha intelligente persienner og multimediasystem (6).

Figur 12 Sentral i et smarthjem (6)

Noen finner det skremmende at teknologen gjør inntog i hjemmet, mens andre er overrasket over hvor kort vi har kommet på dette feltet. Det å bli skremt av dette er ikke noe nytt, da elektrisiteten kom var mange redd den. Enkelte passet for eksempel alltid på å ha en stikkontakt i alle støpsler for at strømmen ikke skulle renne ut. Dette eksempelet viser at før smarte hjem kan bli en realitet må brukere se fordeler med det og være overbevist om at det er sikkert. For å oppnå dette trengs det en sikker infrastruktur og husene må være bygget for å støtte smarte løsninger(6).

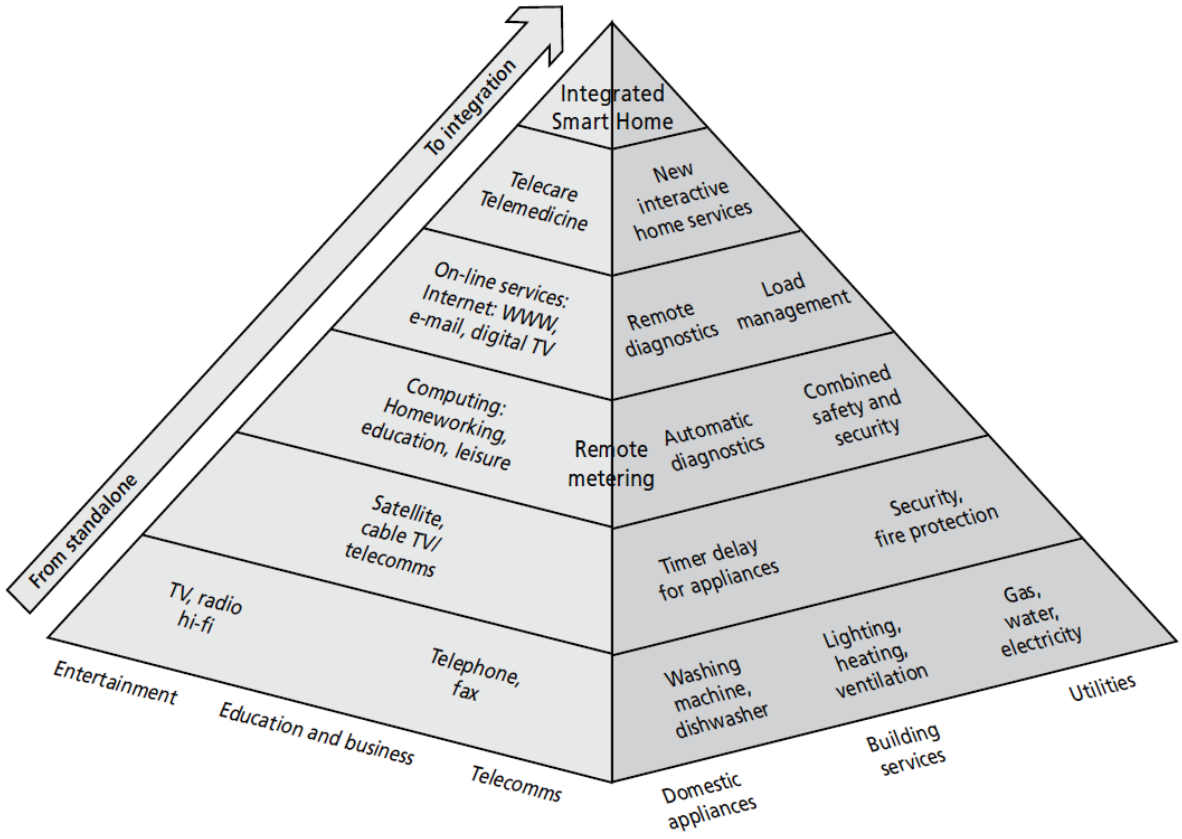
Allerede på 80-tallet og begynnelsen av 90 årene eksisterte det løsninger for smarte hjem. Figur 12 viser sentralen i et av de tidlige smarthjem løsningene, denne var lite brukervennlig og potensielle kunder fant den for vanskelig i bruk. Det har vært en mangel på brukbarhetsforsking og de sosiale aspektene ved smarte hjem. Til nå har teknologens muligheter ledet løpet og brukerne har delvis blitt utelatt og mange spørsmål rundt hvem et smarthjem er tenkt for og hvordan det skal brukes er ikke besvart. I innledningen ble det nevnt at det var ulike aspekter ved et smarthjem, og vi har sett at noen av de vanlige enhetene i et hjem kommuniserer med hverandre den i dag. Det er ikke uvanlig at brannvarslere snakker seg i mellom, eller varsler brannstasjonen ved utløsing. En av grunn pilarene i et smarthjem er kommunikasjon enheter seg i mellom eller mot

ytre faktorer. Smarte hjem er et stort konsept og det kan være vanskelig å sette fingeren på hva det egentlig er. Det som var et smarthjem for 20 år siden er ikke nødvendigvis det i dag. Figur 13 er en pyramide som viser hva som kjennetegner og klassifiserer et smarthjem. Denne kategoriseringen passer til vår tid, men den ville gitt liten mening for 30 år siden. I bunn av figuren finner vi vanlige enheter som i liten grad kommuniserer med omverden og ikke med hverandre, dette er et vanlig hjem slik det fremstår i dag. Går vi et hakk opp ser vi at noen ting er koblet sammen, her finner vi bla eksempelet med brannalarmen over. Neste steg er fjernavlesing for enkelte av systemene i huset, ikke her heller kommuniserer de ulike systemene med hverandre. Videre oppover kommer vi til nye interaktive tjenester i hjemmet, det er det denne oppgaven handler om. Vi ser at dette ligger på nivået under et fult integrert hjem som også tar med dagsformen og helsen til beboerne i hjemmet (6). Dette er ikke den eneste måte å klassifisere hva et smarthjem egentlig er på. Allerede i 1989 ble det lansert et smarthjem, det var omtrent det samme som vestlige lever i nå. Det var vaskemaskin, datamaskiner og lys med sensorer, med mer. Alt dette har blitt vanlig i dag. Terminologien rundt smarte hjem er forvirrende og klassifiseringer er nødvendig. Det er foreslått 5 kategorier av (30) for smarte hjem (her fritt oversatt og litt forkortet):

Liste 5 Kategorier av smarte hjem

1. Hjem med intelligens – lys med lyssensorer og automatiske persienner
2. Hjem med intelligens og kommunikasjon – de smarte objektene kommuniserer seg i mellom
3. Tilkoblete hjem – Enheter har grensesnitt så de kan overvåkes både lokalt og fjernt.
4. Lærende hjem – Huset lærer basert på gjentagende mønstre i brukerens liv.
5. Omsorgsfulle hjem – Huset lærer hvor folk er i huset og tilpasser funksjoner deretter og ikke bare etter bruksmønstre som i punkt 4.

På denne skalaen er denne oppgaven på 3, hvor enheter ikke kan overvåkes, men de kan styres og konfigureres. Det kan diskuteres om den egentlig er på 2.5, men komponering krever mer enn bare kommunikasjon, den krever et grensesnitt.



Figur 13 Pyramiden mot integrasjon (6)

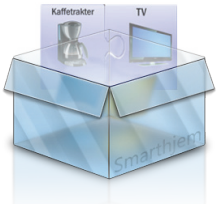
En måte å tilnærme seg smarte hjem er behovet for tilrettelegging for eldre. Det blir flere og flere eldre og mange av dem ønsker å bo hjemme dersom det er mulig. Et smarthjem kan hjelpe eldre med fysiske vansker å bo i sitt eget hjem. Den siste tiden har det også vært et større fokus i media på energisparing og en grønn hverdag, smarte hjem kan være med å redusere energiforbruket. Huset vil ikke bruke energi der det ikke er behov, lys vil være slukket når det ikke er noen der. Varmen kan justeres mer nøyaktig og utstyr som ikke er i bruk kan slukkes. Ikke minst et smart hjem vil være morsomt for enkelte å bo i. Den viktigste grunnen for at dette ikke har kommet for fult enda er uenigheten om hvordan ulike enheter skal kommunisere med hverandre. De siste 20 årene har hver verdensdel etablert sine egne standarder og mye tid går med på å bli enig om en felles standard. En felles standard for kommunikasjon mellom enheter er første steg i retning av et smart hjem (6).

2.8 Konseptuelle modeller

Før et dataprogram kan utvikles er det viktig å tenke over, hvordan brukeren forestiller seg at det virker under panseret. En webportal kan for eksempel være en hierarkisk samling av sider eller en sverm av sammenlenkede sider. Det vil alltid være et valg av det ene eller det andre vedrørende den konseptuelle modellen og ofte kan det være flere vanskelige valg å ta. De ulike løsningene av den konseptuelle modellen er gjerne forbundet med fordeler og ulemper. Det kan

være fristende å utsette valget så lenge som mulig og kanskje unngå det helt. Dette er en stor fallgruve for utviklere da det lett kan føre til forvirrede brukere. En konseptuel modell må utvikles i samråd med brukergrensesnitt metaforen da ikke alle metaforer støtter alle konseptuelle modeller og visa vers. Det er derfor ikke mulig å lage den konseptuelle modellen ved hjelp av fossefalsmetoden, men den må ses i sammenheng med iterativ prototyping. De viktigste konseptene til den konseptuelle modellen bør være det gjeldende for valg av metafor, så på den måten er den konseptuelle modellen med disse delene viktig å gjennomføre før selve utviklingen. Det kan dog være at en ønskelig metafor ikke kan oppnås ved en gitt konseptuelmodell og en avveiing mellom den og metaforen må gjøres. I forbindelse med denne type avveiinger er det viktig å utvikle i en iterativ prosess så det ikke blir brister mellom systemet og den konseptuelle modellen. Den konseptuelle modellen blir *ryggraden* i systemet ditt, og brukergrensesnittet som blir lagt utenpå hjelper brukerne å forstå hva det er og hvordan det kan brukes. Brukeren behøver ikke å vite at der er en konseptuel modell eller hvordan den ser ut. De vil lage sitt eget bilde av systemet, basert på hvordan brukergrensesnittet fremstiller den konseptuelle modellen. Dette bildet gjør det lettere for brukerne å forstå dataprogrammets oppbygging. Brukerens bilde vil gjøre det innlysende at like elementer kan behandles likt og vil oppfører seg likt. Bilde den enkelte bruker danner seg kalles mental modell og er noe bare brukeren har i hodet og ikke nødvendigvis er i stand til å forklare eller dele med andre.

Over har vi argumentert for hvor viktig å lage en god konseptuel modell og følge den i videre utvikling. En god konseptuelmodell vil i tilegg gjøre arbeidet med selve brukergrensesnittet mye lettere da utviklerne vet hva som hører sammen og hvordan det er koblet sammen med andre elementer(31). Et godt eksempel på dette er visuell programmering (se avsnitt 2.5) hvor det er viktig og ikke blande ulike kodeelementene sammen og la hver enkelt instruksjon være et objekt på skjermen. Med uklare koblinger og elementer her kan det lett oppstå spagettikode som det endelige produktet (21).



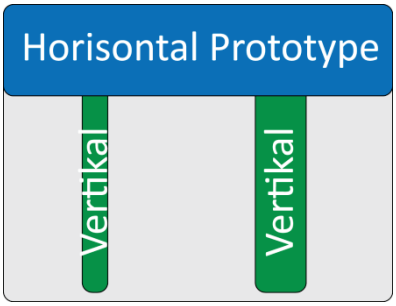
3. Forskningsmetode

Dette kapittelet beskriver hvilke evalueringsmetoder vi benyttet i forbindelse med denne studien, både under utvikling og analysering av resultatet. Alle nevnte metoder har til hensikt å være et ledd i en samlende forskningsmetode for sammenligning av brukergrensesnittmetaforer. Kapittel 4 Forskningsdesign beskriver hvordan de ulike evalueringsmetodene i dette kapittelet settes sammen til en forskningsmetode for å svare på forskningsspørsmålene.

3.1 Prototyping

Tradisjonelt er en prototyp den første enheten som blir produsert, for eksempel den første bilen som kommer ut av en fabrikk. Denne bilen har alle funksjoner og er klar til bruk, den ble produsert med tanke på å avduke problemer i produksjonen. Det gir liten mening å prate om en prototyp på denne måten for programvare. Det er ikke sånn at første person til å laste ned et program innehar prototypen av det. I programmeringsverden er prototyping et steg i utviklingen av det endelige produktet og ikke en versjon av det ferdige produktet. En prototyp er et program som virker under bestemte forutsetninger, dette er ikke et ferdig program, men et program ment for å demonstrere hvordan det ferdige produktet er tenkt. Kode fra prototypen kan benyttes i det ferdige produktet men det er ikke et krav. Det er ikke uvanlig å kode en prototyp i et annet språk enn språket den ferdige versjonen av systemet bruker (32). Prototyping blir således et metodesteg for å avduke problemer og muligheter på et tidlig stadium i utviklingen. En prototyp kan også brukes til demonstrasjon av et uferdig system ovenfor potensielle kunder.

Det er ingen universell måte å lage en prototyp på, heller ikke når i utviklingsfasen den skal lages. En Prototyp må utformes etter hva den skal brukes til. Grovt sett kan vi dele prototyper inn i to grupper:



Figur 14 Horisontal og Vertikal prototype

Liste 6 Kategorier av prototyper

- **Horisontal**prototyp – viser skjermbilder, gjerne i endelig form men funksjonaliteten er enda ikke implementert.
- **Vertikal**prototyp- Lite av skjermbildene, toppen av systemet, er klart og gjerne bare noen enkle versjoner av dette. Funksjonaliteten er på plass og kan testes.

Ofte brukes en kombinasjon av horisontal- og vertikalprototype, dette er illustrert i Figur 14. I Figur 14 er deler av funksjonaliteten implementert (de grønne barene) samtidig er det du ser på skjermen(brukergrensesnittet) til stede (den blå baren). Det du ser på skjermen er ikke

nødvendigvis det endelige produktet, men et forslag(32). En prototyp kan i teorien (33) benyttes til å teste hvilken som helst egenskap ved designet og dette helt uavhengig av hvilket verktøy den er utviklet i. Prototyper kan videre deles inn i low- og hig-fidelity prototyper, de skilles ved hjelp av fem dimensjoner:

Liste 7 Nyansert inndeling av prototyper

- utseende på applikasjonen
- mengden med funksjoner
- dybden på funksjonene
- hvor interaktivt den er
- hvor mye testdata som er tilgjengelig

Selv om en av de overnevnte dimensjonene er høy, betyr det ikke at de andre dimensjonene må være høye. Det er fult mulig å lage en mixed-fidelity prototyp, for eksempel ved å ha mange funksjoner og et lite funksjonelt utseende. Mixed-fidelity er lite utbredt og det er vanlig å holde de forskjellige dimensjonene på noenlunde likt nivå(34).

En papirprototyp faller for eksempel inn under low-fidelity, dette fordi den setter begrensinger i hvor stort datasett det er mulig å holde styr på i form av papirlapper. Det er også vanskelig å lage animasjoner og gi knapper en god oppførsel. I ordet oppførsel ligger måten selve knappen reagerer ved berøring og trykk. Utseende blir også litt annerledes på papir i forhold til en ekte dataskjerm. Dette skyldes at lyset reflekteres på et papir, mens fra skjermen stråler lyset ut.

Hvilken type prototyp et system trenger er ofte avhengig av hva som skal testes og hvor mye penger det er i prosjektet. Prototyper blir ofte laget hurtig og koster lite å utvikle, vertikale prototyper med mye funksjon kan kreve at deler av systemet er implementert, noe som setter sterke begrensninger i hvor tidlig en prototyp kan testes. Er målet å teste kjøretiden og hurtigheten kan det ofte være lurt å velge vertikal prototyp.

3.2 Brukbarhetstesting

En brukbarhetstest er å la personer fra en bestemt gruppe komme til et egnet sted for å løse bestemte oppgaver i et produkt. Dette produktet kan spenne fra ferdige dataprogrammer til tidlige prototyper. Selve testen gjennomføres vanligvis ved at en person sitter foran datamaskinen og løser utdelte oppgaver, personen vil ikke få hjelp med mindre han/hun forlater maskinen og oppsøker en testleder som sitter i testlokalet. Testlederen som sitter i lokalet og observerer befinner seg typisk i et hjørne eller litt tilbaketrukket fra selve testområdet (35).

Oppgavene testpersonene løser utarbeides slik at ønsket del av systemet blir testet. Det er for eksempel fult mulig å teste installasjonsdelen av et system og ikke resten av systemet. Brukbarhetstesting er med å knytte programmeringsverden og brukerverden sammen. Ofte sitter programmerere i sin egne lille boble og lager et system de tror brukerne vil ha. I en annen boble sitter brukerne og venter i spenning på et system og lager forventninger til det. Disse ulike verdene bør møtes i utviklingsfasen når det er mulig å gjøre endringer og ikke når produktet er ferdig og endringer er dyrt. Møte mellom de ulike verdenene kan skje på flere måter og det er viktig å skille en betatest fra en brukbarhetstest, da betatesting ofte først skjer når produktet er

nesten ferdig. I tillegg løser ikke betatestere bestemte oppgaver og dataene utviklerne får fra en betatest er ofte usystematiske. Som utvikler har du også sjelden muligheten til å observere hva betatesterne gjør og går dermed glipp av følelsene og forvirringen brukerne har ved bruk av systemet. Det brukes ofte mange betatestere, mens brukbarhetstesting krever få deltakere for å avduke feil ved et system. Dersom få brukere benyttes vil det ikke være mulig å bruke statistiske metoder for å avdekke fenomener og gyldige resultat. Dette er stort sett ikke noe problem da en brukbarhetstest først og fremt benyttes for å avduke feil med et system(35). Faktisk kan fire pluss minus en bruker være nok til å avdekke de fleste feilene ved et system(36).

Det å gjennomføre en brukbarhetstest krever god planlegging og riktig utstyr. Selv med få testpersoner tar det forholdsvis mye tid å kalle de inn, gjennomføre testen og analysere resultatet. Det er derfor viktig å planlegge dette nøye så testpersonene kan komme fortløpende uten at utviklingen blir ventende på resultatene. Det er en stor fordel med video og lydopptak av selve testene til den videre analysen. Video og lydopptak krever planlegging og oppsett av utstyret før testdeltagerne kommer(35).

For å oppnå relevante resultater fra en brukbarhetstest må de riktige personene og det rette antallet rekrutteres. Med riktige personer menes potensielle brukere av systemet og at de får løse relevante oppgaver i testen. Det er også viktig at testgruppen er representativ for de endelige brukerne(35). Oppgavene testpersonene skal løse er den viktigste faktoren for gode resultater av en brukbarhetstest. Det er viktig å bruke tid på gjennomføringen og utviklingen av en brukbarhetstest, da det tar mye tid og koster store resurser å gjennomføre en ekstra test(36).

3.2.1 "Tenke høyt" metoden

Innsikt i brukerens tanker, meninger og ikke minst hva de tror dataprogrammet vil gjøre er både lærerikt og nyttig. En av de nyttigste og mest brukte metodene til å få et innblikk i brukernes verden, er å la sluttbrukeren tenke høyt under gjennomføringen av en brukbarhetstest. Denne metoden er kjent fra psykologiforskning og har blitt brukt i lang tid innen psykologiske eksperiment. Den har også vist seg å gi god effekt under brukbarhetstester. Når testpersonen tenker høyt forteller han/hun hva han/hun tenker og tror vil skje. De vil forklare hvorfor den knappen ble trykket og ikke den andre osv. Dette gjør det enkelt å oppfatte når i en dialog eller flyten brukeren faller ut og misforstår. Dessverre er det unaturlig for de fleste å tenke høyt på denne måten. Ofte må testlederen minne testpersonen på å tenke høyt. Dette gjøres med spørsmål som, hva tenker du nå? Hva er det du vil gjøre? Hva betyr egentlig det der? Testpersoner ønsker alltid å gjøre det best mulig, de føler at de blir satt på en prøve og evaluert etterpå. Det er derfor viktig at brukeren skjønner at det er systemet som testes og ikke seg selv. Testpersoner kan også være redd for at videoopptak skal komme på avveie eller at mange får se hva de gjorde. Derfor skal alltid opptakene holdes konfidensielle og brukeren gjøres oppmerksom på dette(36).

3.3 Intervju

Intervju av en person kan være krevende, men kommer intervjuer på bølgelengde med intervjuobjektet er det ofte lettere. En viktig forutsetting for å være på bølgelende er å ha samme bakgrunn og forstå hverandre(37). Det er også viktig å lytte mye og ha kontakt med intervjuobjektet, noe som kan være vanskelig ved notering av viktige poenger. Lydopptak av

intervjuet gjør at intervjuer kan ha mer fokus på intervjuobjektet og ikke på notatene sine. I den forbindelse er det viktig å spørre om tillatelse til å ta lydopptak. En annen viktig fordel ved lydopptak er muligheten til å sitere intervjuobjektene direkte i ettertid (38). Et sitat gir sterkere resultater enn vanlig gjengivelse av intervjuet.

Intervjuprosessen er tidkrevende, gode forberedelser er med på å korte ned tiden. På forhånd kan avtaler lages og tilgang til området eller intervjuobjektet sikres. I etterkant er transkribering en lengre prosess, men den gjør det mulig å bruke informasjonsgjenfinningsmetoder på dataene i etterkant. Dette krever mye mer tid enn for eksempel en internettbasert spørreundersøkelse og det krever mye av den som intervjuer, da han/hun hele tiden må passe på å være objektiv. Så ikke intervjuer sine meninger farger resultatet. Det er vanlig at intervjuobjektet prøver å gi de svarene intervjuer helst vil høre, da mennesker av natur vil prøve å tilfredsstille hverandre (37). Dette kan bli ytterligere forsterket ved å ta opp intervjuet, det har vist seg at mennesker er mindre åpne og ærlige når de blir tatt opp. Dette viser at det ikke bare er fordeler ved opptak av intervjuet og ved kun bruk av transkripsjon vil også de ikke verbale elementene forsvinne (39).

Et intervju kan grovt deles i tre forksjellige hovedkategorier:

Liste 8 Kategorier av intervju

- Strukturert
- Semistrukturert
- Ustrukturert

Et strukturert intervju er egentlig en muntlig fremføring av et spørreskjema, fordelen ved å ta det muntlig er at misforståelser og uklarheter i spørsmålene kan avklares. Avsnitt 3.4 gjelder derfor også for strukturerte intervju. Et ustrukturert Intervju er mer å betegne som en samtale hvor det ikke er definert faste spørsmål. Til semistrukturerte intervju er det forbredt et sett med spørsmål, men her må nødvendigvis ikke alle spørsmålene stilles og rekkefølgen på dem kan være mer tilfeldig. Her kan også intervjueren spørre mer i dybden om interessante tema som dukker opp. Går intervjuer i dybden vil informasjon som ikke ville blitt avduket ved spørreskjema komme frem og intervjuet mer å betegne som en samtale mellom intervjueren og intervjuobjektet. Både ustrukturerte og semistrukturerte intervju får få data til å trekke konklusjoner ut av, men er best egnet for å oppdage nye elementer og hva som krever mer forskning. Intervju er også egnet til å samle inn informasjon rundt følelsesmessige tema og emner av sensitiv karakter. Emner som vanskelig kan beskrives eller detaljer innen et emne er også ofte lettere å få fatt i ved hjelp av intervju i forhold til et spørreskjema(37).

3.4 Spørreundersøkelse

Spørreundersøkelser er velegnet å sende ut til en stor gruppe mennesker, gjerne 100 eller flere. Resultatene samles før forskeren begynner å lete etter mønster i svarene. Det er viktig å se over spørsmålene både en og to ganger før undersøkelsen sendes ut, da det er vanskelig om ikke umulig å kontakte alle deltagerne i ettertid dersom et viktig spørsmål er glemt. For eksempel hvis alder eller kjønn er utelatt fra en undersøkelse rundt ulikheter mellom kjønnene. I planleggingsfasen er det også viktig å finne grunner som motiverer personer til å svare. Da det ikke er uvanlig med under ti prosent svar på en spørreundersøkelse. Bedre svar prosent kan

forventes dersom det skrives hvorfor folk bør svare og hvis det er mulig kontakte de som ikke har svart på en høflig måte. For analyse av resultatene er det viktig å ta høyde for eventuelle grupper som ikke har svart. Dette kan gjøres ved å kontakte noen av de som ikke svarte og spørre dem ut over telefon. Dersom en spørreundersøkelse skal ha gyldige resultater for en hel populasjon kreves det ofte et stort antall svar, er populasjonen for eksempel 5000 stykker kreves det 760 svar på spørreundersøkelsen for å ligge innenfor et 95 % konfidensintervall med pluss minus tre prosent nøyaktighet(37).

Spørsmålene i spørreundersøkelsen må være korte og konsise. Under ingen omstendigheter skal det benyttes ord med flere betydninger. Under utformingen av selve spørsmålene er det flere valg å ta hensyn til, et spørsmål kan formuleres på følgende måte:

Liste 9 Spørsmålstyper til intervju (37)

- **Åpne spørsmål** – her svares det med fritekst på et avgrenset område.
- **Lukkede spørsmål** – det må svares på ett av svaralternativene
  - **Ja/Nei** – svaret er enten ja eller nei
  - **Kvantitative spørsmål** – for eksempel alder eller hvor mange barn har du?
  - **Enig/Uenig** – er du enig eller uenig med et utsagn
  - **Enighets skala** – sterkt enig, enig, verken enig eller uenig, uenig, sterkt uenig
  - **Vanlig skala** – veldig dårlig, dårlig, passe, bra, veldig bra
  - **Semantisk differensial skala** – to motsigende utsagn og du setter kryss i retning av den du liker. For eksempel Bra innhold |_|_||_|_||_|_||_|Dårlig innhold
  - **Rangering** – rangerer et gitt antall elementer i forhold til hverandre

De forskjellige typer spørsmål bør varieres i en undersøkelse da det gjør testen mer morsom å gjennomføre. Brukes det for mange forskjellige spørsmål blir det fort forvirrende og mange kan komme til å krysse feil. På enkelte av spørsmålene er det naturlig å ha med valget: ”vet ikke”. Mange kan komme til å krysse ”vet ikke”, da det er den letteste utveien. På den andre siden kan det tvinge personer til svare feil dersom de virkelig ikke vet svaret og denne muligheten ikke er presentert. Under utviklingen av en spørreundersøkelse må det derfor avveies hvor og om det skal være med en ”vet ikke” mulighet. Hvor mange svarmuligheter det skal være på en skala varierer litt, det mest vanlige er fra 5-9. Her er det viktig å tenke på om det skal være mulig å svare midt i, eller ikke. For eksempel midten av 5 er umulig å krysse av i en rute(37).

3.5 Kvantitativ analyse

Kvantitative data er brassert på nummer og er typisk data fra spørreundersøkelser og strukturerte intervju(37). En brukbarhets test kan også i noen tilfeller generere kvantitative data. For eksempel hvis det blir målt tiden en testperson bruker på en gitt oppgave.

Den enkleste form for kvantitativ analyse er å tegne grafer og tabeller med dataene i. Dette lar forskeren og eventuelle lesere enkelt se mønstre i dataene og kan trekke konklusjoner på bakgrunn av dem. Mer avanserte statistiske metoder kan avduke enda flere mønstre. Det er viktig å kontrollere at dataene er gyldige og ikke bare er en tilfeldighet når statistisk analyse blir brukt. Statistikk kan være tidkrevende å regne for hånd og derfor benytter de fleste et verktøy for utregningene. SPSS og minitab er spesialiserte verktøy for statistisk analyse og er meget

effektivt ved større opprasjoner. Det kan være litt tungt å lære for den utrente og da eksiterer der mindre kraftige verktøy, for eksempel Microsoft Excel. I den senere tid har det vist seg at mange forskere bruker denne type verktøy for å imponere med statistiske data uten helt å kunne statistikk. Dette har ført til en skepsis blant lesere og andre forskere, dette er en ikke ønskelig situasjon så det er viktig å forstå de ulike metodene før de brukes til videre analyse. Forklaring av hvilke metoder og begrunnelse for dette er med på å øke aksepten for resultatene i forskningsmiljøet.

Tall er ikke bare tall, det finnes flere ulike måter tall kan bli brukt på. **Nominelle data** er når tallene blir brukt til å beskrive en kategori og ikke tallets numeriske verdi. For eksempel kan mann bli kodet til 2 og kvinne til 1, på denne måten er ikke 2 større enn 1, da menn ikke nødvendigvis er større enn kvinner, alt tallene sier er at de er forskjellige. På denne type data har det ingen hensikt å gjøre aritmetiske opprasjoner, da kvinne pluss kvinne ikke er mann, men to, samtidig er en pluss en, to. Derfor er eneste nyttige analysene her å bruke frekvensanalyser på dataene. Dette står i kontrast til **ordinale data** hvor tallet betyr hvor godt en person liker eller er enig i ett utsagn. For eksempel vil en som setter jordbær til fire, like jordbær bedre enn en med jordbær to. Dette gjør at vi kan benytte et sett aritmetiske opprasjoner på dataene, men ikke alle. Ofte er dataene her fra en skala der brukeren har krysset av på ett av valgene: liker, passe, dårlig osv. Det er umulig å si hvor mye bedre ”liker” er enn ”dårlig” og vi kan derfor ikke si at fire er dobbelt så bra som to eller motsatt. På **intervalldata** er forholdet mellom tallene de samme. Dette betyr at forskjellen mellom en og fem er den samme som mellom fem og ti. Heller ikke her kan vi si noe om to tall i forhold til hverandre, da vi ikke vet hvor stort et intervall egentlig er. Det eneste vi vet er at forholdene er propporsjoner så vi kan derfor ikke benytte multiplikasjon og divisjon på tallene. **Forholdsdata** er når der eksiterer en ekte null verdi, for eksempel en alder kan være null år. Det å ha en ekte nullverdi gjør det mulig å benytte alle de aritmetiske opprasjonene. Her kan vi si at en person på ti er dobbelt så gammel som en på fem år. Videre kan de ulike datatypene kategoriseres i heltall hvor kun hele tallverdier er gyldig. Det er for eksempel ingen som har 0,7 barn. Den andre muligheten er kontinuerlige data, for eksempel tid, det er en kontinuerlig skala, og her må det bestemmes på forhånd hvor nøyaktig svar som trengs(37).

Det eksisterer mange statistiske metoder, og mange av dem kan kun brukes på bestemte data. Følgende statistiske metoder er mye brukt i forskning:

Liste 10 Statistiske metoder

- **Gjennomsnitt** – Den gjennomsnittlige verdien, dette gir lite eller ingen mening på et lite datasett, eller dersom det er stor variasjon i resultatet. Gjennomsnitt kan ikke benyttes på nominelle data.
- **Median** – det midterste tallet i datasettet når det er sortert og kan ikke benyttes på nominelle data.
- **Mode** – Det tallet med hyppigst frekvens
- **Variasjonsbredde** – Differansen mellom høyest og laveste verdi, her kan ekstreme verdier gi stort utslag
- **Fraksjoner** – Dele datasettet opp i ulike blokker, for eksempel median deler datasettet i to.
    - **Kvartil** – Deler datasettet i fire
    - **Prosent** – Deler datasettet i 100
- **Standardavvik** – Dette er trolig den mest brukte metoden for å måle distribusjonen av dataene. Standardavviket forteller gjennomsnittlig distanse for hver dataverdi fra gjennomsnittet. En stor standardavvik betyr derfor stor spredning i dataene.

Det å bruke overnevnte og flere metoder har klare fordeler og ulemper i forhold til hverandre. Noen av dem brukes til å fortelle om variasjonen i dataene, mens andre brukes for å si noe generelt om datasettet. Neste steg i analysen er å finne sammenhenger mellom de ulike variablene i datasettet. Her ser vi på to og to variabler vi antar alltid at der ikke er noen sammenheng, en null hypotese. En null hypotese kan for eksempel være: Der er ingen sammenheng mellom programmeringskunnskaper og mattematikkevner. Statistiske tester for dette returnerer en sannsynlighet for at der ikke er noen sammenheng mellom de ulike dataene. Her er det vanlig å si at, hvis det ikke er en sammenheng sjeldnere enn 1 av 20 ganger så er der en signifikant forskjell på variablene. Dette betyr at $p < 0,05$ avkrefter nullhypotesen(37).

Chi-kvadrat er en metode for å finne sammenheng mellom ulike variabler i et data sett. For eksempel om det er forskjell på kvinner og menn i forhold til programmeringsevner. Denne metoden er mye brukt i forsking og kan brukes på nominelle, ordinale, intervall og forholdsdata. Dersom den skal regnes ut utenfor et statistikkprogram krever den litt kunnskap da den bruker antall rader og kolonner til å beregne graden av frihet. En annen ulempe med chi-kvadrat er at det krever store datasett, minst 30 elementer i hver oppføring.

T-test brukes for å se om det en forskjell mellom to datasett. Selv om gjennomsnittet er forskjellig på to datasett behøver ikke det å bety at der er en signifikant forskjell mellom dem. T-test bruker nullhypotese og $p < 0,05$ betyr at der er signifikant forskjell mellom datasettene. Det finnes flere varianter av en t-test, for eksempel om det er lik eller ulik varians i de to datasettene, eller om elementene i de to datasettene er paret. Parete sett oppstår dersom det er en og samme person som er spurt om det samme på forskjellig tidspunkt. I analysearbeid er

det viktig og presisere hvilen variant av t-test som er benyttet for at andre skal kunne verifisere arbeidet(37).

Friedman test brukes på samme måte som en t-test, bare at den brukes på tre eller flere datasett. Friedman test antar ikke noe om distribusjonen av de enkelte datasettene, og benyttes der de samme personene har tatt de samme testene flere ganger. Eller der ulike personer har gjennomført de samme testene. På lik linje med en t-test, bare at den er for flere en to sett(40). Dersom nullhypotesen blir avist av en friedman test må en paret sammenligningstest til for å avgjøre hva det er som skiller seg ut. Nemenyi prosedyren kontrollerer alle variablene mot hverandre og finner ut hva som er likt og ikke likt(41). Ved undersøkelse av mange variabler er det ofte mulig å finne misvisende svar. Det meste av statistikk regnes innenfor et 95 % konfidensintervall. Dette betyr at hvis du har 20 ulike variabler vil et av dem statistisk sett være signifikant. Ofte leter forskere etter en løsning på et problem, for eksempel hva som fremkaller kreft. For å unngå dette må resultatene korrigeres for flere variabler, den mest benyttede metoden er bonferroni. For analyse av kortrangeringen er variablene knyttet sammen og bruk av denne er ikke anbefalt da det vil gi for konservative resultater(42).

Kvantitative data har mange testede og aksepterte analyse metoder, mange forskere mener dette er den eneste gyldige forskningen. Det er derimot en fare ved å bruke dem, det er fort å miste de ikke kvantitative aspektene av en test. Den subjektive opplevelsen til en testperson kan være like interessant som det faktiske resultatet. Resultatet av en kvantitativ analyse blir uansett ikke bedre enn datasettet og det er derfor ikke sikkert at resultatene sier stort(37).

3.6 Kvalitativ analyse

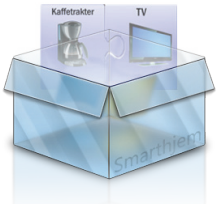
Kvalitative data er alt som ikke er tall, for eksempel meninger, følelser, ord, bilder og annet en testdeltager gjør og presenterer under en test. Mye av dette kan gjøres kvantitativt, for eksempel hvor mange ganger et ord forekommer. Der vil fortsatt være data det ikke er mulig å sette et tall på, og det er her den kvalitative analysen kommer til sin rett. Faren med å gjøre alt kvantitativt er for eksempel at enkelte ord kan ha ulik betydning avhengig av konteksten.

Der er ikke faste metoder og velprøvde formler for kvantitativ analyse og det kreves derfor mer erfaring av forskeren til å trekke ut de viktige elementene. Ofte er det snakk om enorme datamengder, bare en ukes feltstudie kan resultere i flere hundre sider data. Det er ikke uvanlig at store deler av dataene kommer fra transkribering av lyd og eller videoopptak, en tidkrevende prosess. For å slippe å gjøre dette på nytt er det viktig med sikker lagring av dataene og gjerne flere kopier på ulike steder, da det er allment kjent at datamaskiner krasjer innimellom. Før selve analysen kan begynne er det også en fordel å strukturere dataene likt, det gjør videre arbeid mindre tidkrevende(37).

Selv om der ikke er metoder for kvalitativ analyse som gir et signifikant resultat, er det metoder å utføre en kvalitativ analyse på. Første steget er å lese over alle dataene for å få et generelt inntrykk over hva du har tilgjenglig. Her lønner deg seg å notere relevante ting i forhold til forskningsspørsmålene og ting som gjentar seg ofte. Ikke bare det tekstlige men også utrykk, kroppslig oppførsel så vel som konteksten er viktig. Neste steg er kategorisering av relevante data for forskningen, det er to måter å gjøre dette på. Enten kategorisere etter kjente teorier

rundt emnet, eller kategorisering basert på innholdet av dataene. For den siste er det viktig å være oppmerksom så ikke egne meninger er med å forme kategoriene. Kategoriene du velger må holdes ved like, enkelte vil bli for små, mens andre for store. Det er derfor viktig å ha en liste med kategorier og holde den oppdatert etter hvert som kategoriene endres. Når et element blir satt i en kategori, er det viktig å ta vare på referansen fra hvor det kommer, ofte er det på denne måten mønster blir oppdaget. Det lønner seg å sette dataene inn i en tabell for lett å se et mønster, men det er ikke nødvendig. Etter at et mønster er oppdaget er målet å svare på hvorfor det er sånn. Her kan eksisterende teorier være til stor hjelp. Det er ikke dumt å gjøre dette arbeidet parallelt med datainnsamling da du lettere kan spore opp mer informasjon om mønsteret eller avvise det som en tilfeldighet. For å øke gyldigheten av resultatene er det lurt å ha med fremgangsmåten og tankene rundt det i resultatet, da dette vil gjøre at leseren skjønner hvordan du tenker og hvorfor du har kommet frem til konklusjonen(37).

kvalitativ analyse kan ha med faktorer som ikke er mulig kvantitativt, dette kan for eksempel være oppførsel, lukter, kroppsspråk. Dette er aspekter en kvantitativ analyse ikke dekker alene, men en kvalitativ analyse kan føre til en kvantitativ analyse av det. Resultatene fra en kvalitativ analyse bør ikke behandles som lov, men som foreløpige resultater da forskerens meninger kan smitte over på resultatet og datagrunnlaget ofte er litt lite(37).



4. Forskningsdesign

De ulike evalueringsmetodene finner ulike resultater, det er derfor viktig å velge riktig metode for å finne ønskede resultater. Dette kapittelet omhandler tilpassninger og valg av de ulike forskingsmetodene i kapittel 3. Videre beskriver dette kapittelet hvilke metoder hver enkelt av testsubjektene var igjennom og hvordan vi tilpasset de til forskningsspørsmålene våre. For å få nok data valgte vi å bruke 8 personer fra hver av gruppene, datastudenter og arkitektstudenter. De ble rekruttert via e-postlister og personlig oppmøte på arbeidsplassene deres.

4.1 Brukbarhetstesting

Det ble utviklet fem prototyper for konfigurering av smarte hjem, de ulike prototypene benytter forskjellige metaforer. Utfyllende beskrivelse av dem er i kapittel 5. Prototyper og de er lagt til grunn for brukbarhetstestingen. Prototypene er horisontale prototyper med mixed-fidelity, og tilpasset en brukbarhetstest for avduking av problemer knyttet til komponering av tjenester i et smarthjem.

Det er viktig å fortelle testpersonene at det er systemet som skal testes og ikke de, dette må gjøres ettertrykkelig og på en forståelig måte ovenfor hver deltager før testen begynner. Vi tok et bevist valg i at testleder sitter ved siden av testsubjektet og ikke i et hjørne av rommet som er vanlig. Dette gjorde vi for at det skulle være lettere for testpersonene å benytte ”tenke høyt” metoden og hver enkelt bruker skulle føle seg tryggere. Vi prøvde å gjøre omgivelsen hyggelige og ikke skremme testpersonene med en trykkende atmosfære.

Når testleder sitter ved siden av testsubjektet er det også lettere for å se når testsubjektet får problemer eller oppdager spennende ting. Testleder kan så notere dette for å ta det opp igjen etter at testen er avsluttet. Det ble lagt stor vekt på oppgavene, de skulle være varierte og teste ulike aspekter ved hver modell. Gjennomføring av en brukbarhetstest er beskrevet i kapittel 6 Brukbarhetstesting, her er også de ulike oppgavene presentert. Formålet med denne metoden var å avduke hva de ulike brukerne foretrakk ved de ulike modellene, og hvor vanskelig de var å bruke. Denne metoden bruker vi for å besvare forskningsspørsmål RQ3, ved hjelp av kvantitative data og legge grunnlaget for spørreundersøkelser rundt de ulike metaforene.

4.2 Intervju

Etter spørreundersøkelsene hadde vi et semistrukturert intervju med testsubjektene, her hadde vi et par faste spørsmål til kvantitativ analyse og tok opp kvalitative funn fra brukbarhetstestingen. Det ble blant annet spurt om de ønsket å flytte inn i et smarthus og anså et system lignende det de har prøvd som nyttig. Dette dobbeltspørsmålet er essensielt med tanke på hvor mye energi det enkelte testsubjektet er villig til å legge i læringen av systemet. Nardi skriver i boken sin (11) at brukere er villig til å lære seg mer dersom de er motiverte for å ta systemet i bruk.

Et intervju varte typisk i 10 minutter og hadde til hovedhensikt å avduke hva testpersonene la til grunn for valg av metafor. Dette besvarer RQ3, og varier litt i innhold etter hva testpersonen har sagt under ”tenke høyt” sesjonen. Intervjuet ble også brukt som en oppsummering av brukbarhetstesten, her tok vi tak i problemer og spesielle løsninger testpersonen hadde møtt. Problemene er særdeles viktig i forhold til RQ3. Intervju malen vi benyttet er vedlagt i Vedlegg IV

4.3 Spørreskjema

Dette avsnittet består av fire underpunkter og kun punkt 4.3.4 SUS er et tradisjonelt spørreskjema. De andre punktene er deler av brukbarhetstesten, men de ble skilt ut som en egen del. Vi valgte å bruke spørreskjema for å få kvantitative tall på hva de ulike testsubjektene likte, forsto og hadde problemer med.

4.3.1 Personopplysninger

Like etter at den enkelte testpersonen var ønsket velkommen og presentert for hva han/hun skulle igjennom ble det utdelt et enkelt spørreskjema. Her ble deltagerne bedt om å skrive opp navn og adresse til utsendelse av gavekort. Alle testdeltagere fikk et gavekort på 250kr som takk for hjelpen. Videre ble de bedt om å skrive opp egen alderen og velge programmerings evne ut fra en skala. Denne skalaen var delvis forklart, men det ble gitt muntlig veiledning under utfyllingen. Hensikten her var å avduke hvem som er programmere og ikke, vi valgte å rangere dette i fem ulike kategorier, hvor det kun er de to med mest programmering som betyr at vedkommende er en programmerer. Skjemaet hver deltager fikk utlevert er vedlagt iVedlegg V.

4.3.2 Kortrangering

Etter brukbarhetstesten fikk testsubjektene utdelt et skjermbilde av hvert brukergrensesnitt med beskjed om å sortere de i rekkefølge etter hvor godt de likte dem. Dette er en effektiv måte å kvantifisere hvor godt et testsubjekt liker en prototyp i forhold til en annen. For å unngå at testsubjekter skulle si at de likte alle like godt, var det ikke lov å rangere to prototyper likt. Dette kan by på problemer da enkelte ikke vil klare å skille to modeller fra hverandre. Det ble derfor lagt vekt på å ha en ekstra hyggelig tone under denne delen av testingen. Kortrangeringen har som hensikt å sortere ut hvilken grensesnittmetafor testpersonene foretrekker til komponering av tjenester i smarte hjem. Dette svarer delvis på RQ1 da en viktig faktor for den best egnede metaforen er hva brukerne liker, og denne evalueringsmetoden avdekker også om det er forskjell i de ulike brukergruppene (RQ2).

4.3.3 Tolking av skjermbilder

Det er alltid en fare ved brukbarhetstesting at deltakeren ikke helt skjønner hva de gjør, eller klarer å gjøre om på det de har gjort. Der var lagt inn oppgaver i brukbarhetstesten for at brukerne skulle forstå systemet og ikke bare tippe at de løste oppgavene på en tilfredsstillende måte. For å dobbeltsjekke dette hadde vi laget en komposisjon ferdig og en annen halvferdig i hver prototyp. Vi Skrev ut et skjermbilde av hver prototyp med disse komposisjonene og ba brukerne fortelle hva som skjedde i hjemmet i de ulike skjermbildene. For å gjøre dette i en spørreundersøkelse sjargong bestemte vi oss for at det var to mulige resultater for hver prototyp, enten rett eller galt. Denne evalueringsmetoden er med å besvare RQ1, da det er viktig at brukerne vet hva de gjør og hva som skjer i hjemmet på den metaforen som blir best egnet for sluttbrukerkomponering av tjenester i hjemmet.

4.3.4 SUS

Etter avsluttet brukbarhets test fikk testpersonene utdelt et spørreskjema til den prototypen de rangerte best i kortrangeringen. Spørreskjemaet vi brukte var SUS (System brukbarhets skala) dette er i Vedlegg II, et spørreskjema for å måle brukbarheten av et system. Brukervennlighet består av ulike aspekter; er brukeren i stand til å utføre oppgaver på en effektiv måte? Hvor store resurser kreves for å utføre denne oppgaven? og hvordan opplever brukeren systemet? I ulike systemer vektlegges disse egenskapene ulikt av brukerne og SUS kan derfor ikke brukes til å sammenligne brukbarhet mellom ulike systemer. Spørreskjemaet består av ti utsagn hvor testpersonen avgir på en skala fra 1-5 hvor enig/uenig de er, dette er en semantisk differensial skala (avsnitt 3.5). Resultatet blir ordinale data for videre analyse av brukbarheten, og hvert enkelt spørsmål i denne spørreundersøkelsen har liten mening i seg selv. Derfor ser vi på totalresultatet av testen, og regner ut en poengsum basert på følgende regler:

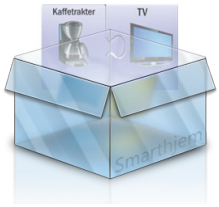
- Spørsmål 1,3,5,7 og 9 (oddetall): poengsum minus 1
- Spørsmål 2,4,6,8 og 10 (partall): poengsum minus 5

Etter å ha regnet ut denne foreløpige summen, summeres tallene og multipliseres med 2.5, dette gjør at skalaen går fra 0-100, hvor 100 er den beste brukervennligheten(43).

SUS avduker hvor brukervennlig de ulike metaforene er i forhold til hverandre, brukervennlighet er et nøkkelpunkt i forbindelse med besvaring av RQ1. Det kan også være at de ulike brukergruppene vekter brukbarheten ulikt, og det kan være besvarelse av RQ2, men dette kommer an på hvor mange som svarer på de samme modellene. Da datagrunnlaget her fort kan bli for lite til kvantitativ analyse.

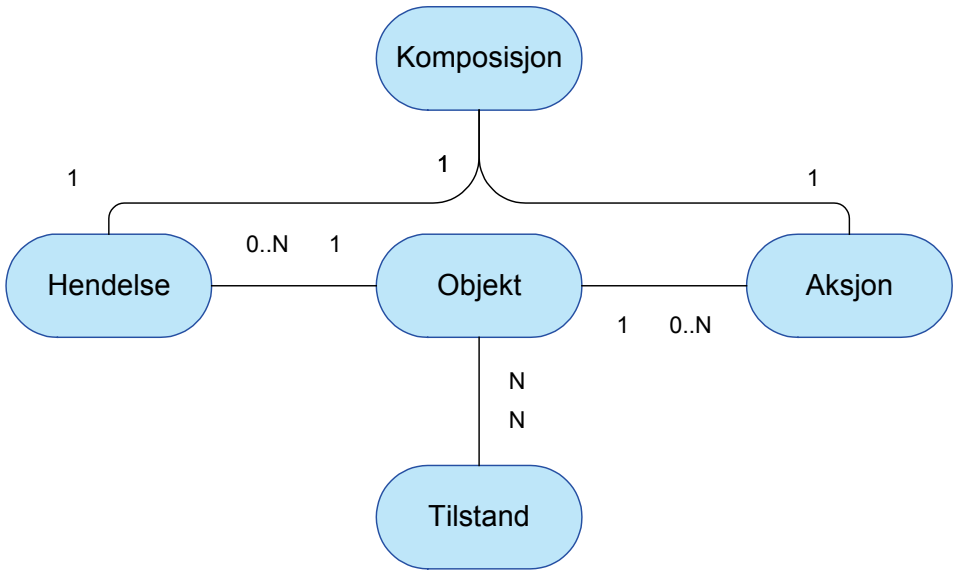
4.4 Kvalitativ analyse

Flere av metodene produserer kvalitative data, brukbarhetstestene alene vil produsere rundt 18 timer med video og lyd. Vi transkriberer ikke disse, da det vil bli en del gjentagelse av de samme problemer og vellykkete ting med en og samme testperson. Mange av oppgavene er noenlunde like i utførelse og det har ikke noen hensikt å transkribere dette. Isteden vil vi redigere hver video så den kun inneholder selve testingen og ikke bytte av prototyper og velkomst sekvensen, da dette er helt likt for hver deltager og ikke gir noen verdifulle data til videre analyse i forbindelse med forskningsspørsmålene. Ved neste iterasjon over videoen skal vi notere sitater og viktige hendelsene, dette kategoriseres i forhold til prototyp, rekkefølge den hadde i testen og hvilken testperson det kom fra. Når dette er gjort kan vi begynne å lete etter mønster og jobbe videre med dataene fra det. I denne kategoriseringen vil vi også legge inn resultater fra intervjurunden, disse utsagnene blir merket med intervju, men kategorisert under de ulike prototypene. Kategoriseringen gjøres ved å la dataene legge grunnlaget for de ulike kategoriene som oppstår. Kategoriene vil i seg selv være en del av resultatet da de er formet av dataene og ikke teorien bak.



5. Prototyper

Vi utviklet fem prototyper for å styre et fremtidig smart hjem. Prototypene er fult kjørende men de er ikke koblet til et system for husstyring. Prototypene er veldig forskjellige da det er benyttet forskjellig metaforer, men med unntak av dette er de helt like. For at de skulle bli så like som overhodet mulig utarbeidet vi først en konseptuel modell for programmene se Figur 15. Denne modellen er den enkleste form for komposisjon som er mulig å lage.



Figur 15 Konseptuel modell

Prototypene er vedlagt denne besvarelsen, prototyper.zip. Installasjonsveiledning for disse er i Vedlegg VIII

En komposisjon består av to objekter som er koblet sammen, det kan for eksempel være at et lys er koblet til Tv-en. Den konseptuelle modellen viser at det ikke er selve objektene som er koblet sammen, men en hendelse til et objekt og en aksjon til samme eller et annet objekt. Dette kan i et tenkt tilfelle med lys og tv for eksempel være hendelsen; TV slås på er knyttet til aksjonen lys av. Vi ser videre at alle objekter kan ha en eller flere tilstander uavhengig av hendelse og aksjon. Dette betyr at et lys vil ha aksjonen slå på selv om det har tilstanden lys på. Vi har sett det som hensiktsmessig å styre en aksjon selv om det blir smør på flesk i forhold til tilstanden, da målet til brukeren er lys på.

Dette er den enkleste form for komposisjon som er mulig å oppnå, kun to objekter og et sett hendelser og aksjoner. Vi har valgt en så enkel modell da sluttbrukerprogrammering har hatt vansker med å slå igjennom. Prototypene skal ikke være en form for direktestyring av et hjem, men derimot komponering av tjenester. Ikke alle objekter er i stand til å motta en hendelse,

”min posisjon” er en av dem, dette da det ikke er mulig å manipulere sin egen posisjon ved hjelp av å skru av eller på et lys. Teleportering hører fremtiden til og vi har følgelig valgt å bryte en til en kobling mellom den konseptuelle modellen og objektet ”min posisjon”. Dette gjelder også for været. Vi ønsker å gjøre systemet sluttbrukervennlig og har derfor ikke benyttet en sensoranalogi, men fysiske ting på linje med den faktiske kaffetrakter.

For å bygge prototypene brukervennlig hadde vi Jacob Nilsen sine ni punkter for et brukbart system (36) i tankene under utviklingen. Disse er fritt oversatt til norsk under med en kort forklaring på hvordan de er tatt i bruk. Under de ulike prototypene vil det bli vist konkrete eksempler på dem.

Liste 11 Jacob Nilsens ni punkter for et brukbart system (36)

- **Bruk en enkel dialog** – Dialogene ble holdt korte og konsise.
- **Snakk brukerens språk** – All teksten i programmet er skrevet i en muntlig sjargong.
- **Minimer hvor mye brukeren må huske** – Det er forsøkt å gjengi valgt informasjon så brukeren slipper å huske dette fra steg til steg
- **Vær konsistent** – De same navnene er brukt over alt og ting kobles sammen likt
- **Gi tilbakemelding** – Brukeren får alltid informasjon om hva som er gjort
- **Gi klare muligheter ut** – Det er alltid en måte å gjøre om gjorte ting på
- **Muliggjør snarveier** – Flere ting kan gjøres på mange måter
- **Gi gode feilmeldinger** – Brukerne får feilmeldinger i form av tekst
- **Forhindre feil** – Ting som ikke kan kobles sammen i virkeligheten kan heller ikke det i systemet.

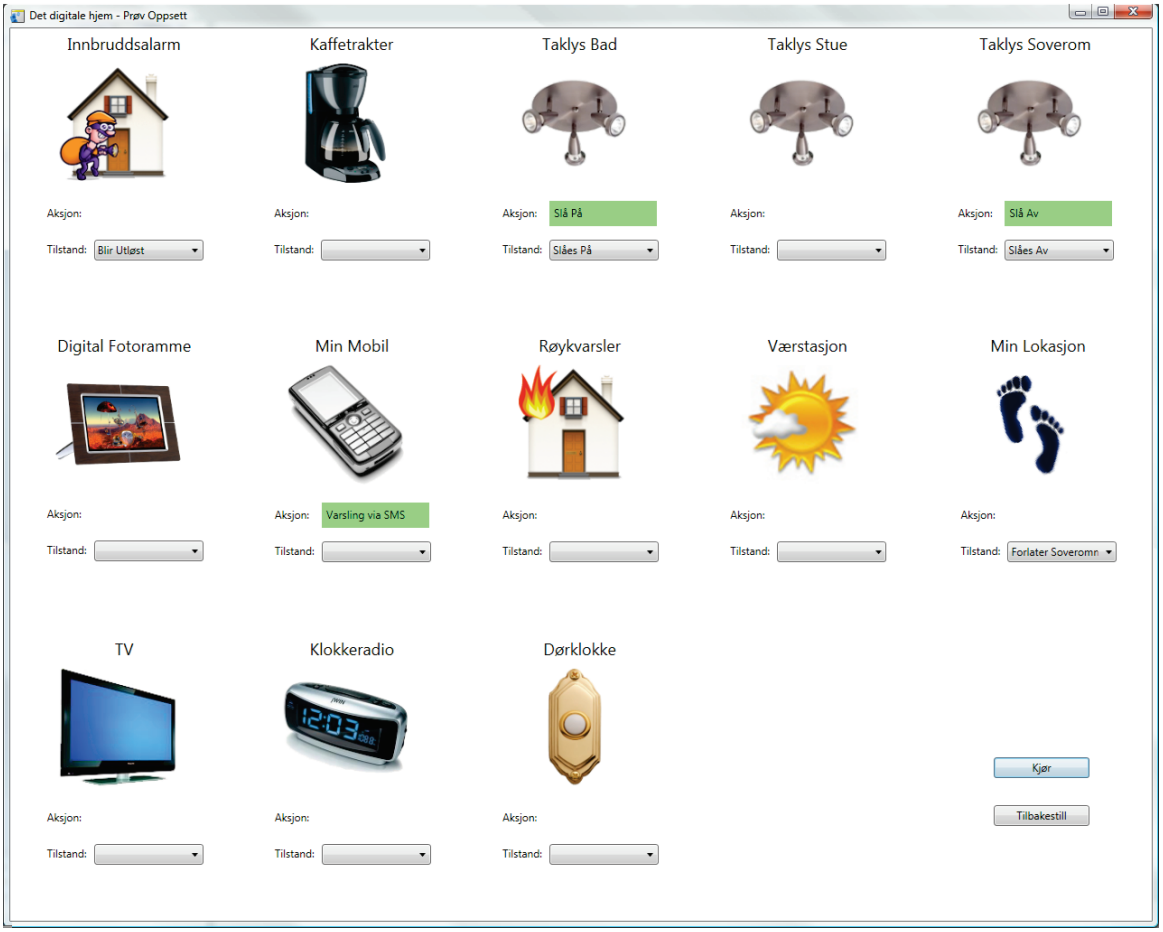
Alle prototypene er veldig like leksikalsk (se avsnitt 2.3 ovenfor). Vi har brukt de samme hendelsene og aksjonene i alle prototypene. Det er også de samme objektene med i alle prototypene og den bildelige illustrasjonen av dem er lik. Leksikalsk sett er der forskjell på hvordan bildet er pakket inn, i noen modeller er det i en liste, mens i andre er det i en brikke. Det syntaktiske har også mange likheter og det er de samme muligheter i alle prototyper. Hvordan brukeren kommer frem til mulighetene og bygger regler er den store syntaktiske forskjellen mellom metaforene. Bakgrunnen for de ulike prototypene er omtalt i av avsnitt 2.4 ovenfor og derfor ikke gjengitt her. Alle prototypene er tiltenkt å kjøre i fullskjerm med en oppløsning på 1280x1024, ved kjøring av prototypene vil de automatisk åpnes i fullskjerm, uavhengig av oppløsning på skjermen.

5.1 Simulator

Vi utviklet en simulator for at brukeren enkelt skulle kunne verifisere komposisjonene de laget. Denne simulatoren er lik for alle modellene og er beregnet som et hjelpemiddel under brukbarhetstesten og derav ikke med i selve testen. Testpersoner som åpner den vil umiddelbart få en forklaring på hva det er av testlederen. Dersom simulatoren ikke gir ønsket utslag vil ikke testpersonen få en forklaring, for da har ikke vedkommende laget den komposisjonen han/hun ønsket. Da blir brukeren heller stilt spørsmålet hvorfor skjedde det ikke noe nå, og hva ville du skulle skje?

Da simulatoren er lik for alle prototypene blir den kun forklart her og ikke nevnt videre under beskrivelse av de ulike prototypene. Alle prototyper har en knapp <prøv oppsett>, brukerne kan trykke for å komme til simulatoren.

Simulatoren viser et bilde alle objektene i huset med et forklarende navn over. Hvert objekt er plassert i en rute med en tilhørende nedtrekkmeny og et felt for aksjon, dette er vist i Figur 16. Nedtrekkmenyen er ønsket tilstand til de ulike objektene (hendelse fører til en tilstand), hendelsen som er starten på en komposisjon. Her kan brukeren velge en tilstand, for eksempel på innbruddsalarm kan en velge; blir utløst. For å se hva som skjer i huset ved skifte til denne tilstanden trykkes <kjør> knappen ned en gang og objektene som påvirkes av denne tilstanden vil få et grønt felt under seg. I det grønne feltet står det hvilken aksjon objektet i samme rute får. I bildet er dette vist ved at for eksempel min mobil får aksjonen varsling via SMS.



Figur 16 Simulator til prototypene

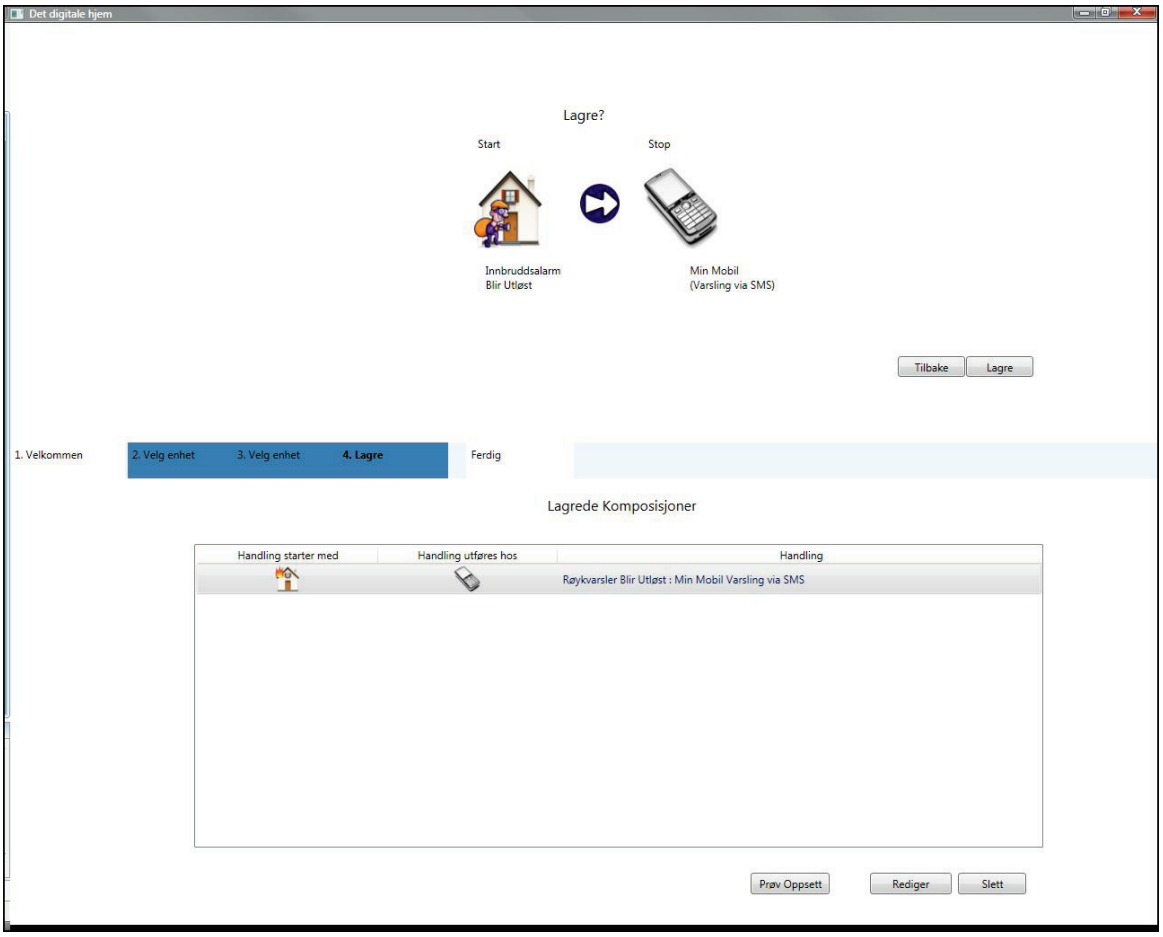
Simulatoren har også støtte for at enkelte objekter kan ha hendelser som påvirkes av aksjonen. Da vi ikke har alle aksjoner og hendelser i systemet vårt er det ikke sikkert dette er tilfellet. For eksempel vil et lys som mottar aksjonen slå på skifte tilstand til hendelsen slås på. Dette fører til at flere komposisjoner kan bli utløst av en hendelse (skifte i tilstand). I Figur 16 er det dette som har skjedd, der er laget en komposisjon for at lyset på soverommet går av når beboeren forlater rommet. En annen komposisjon sier at lyset på badet skal gå på når lyset på soverommet blir

slukket. Når ”min posisjon” blir satt til forlater soverommet vil følgelig lyset på badet skrues på. For å avduke evige løkker som følge av denne type komposisjoner vil simulatoren stå og blinke grønt på de aktuelle aksjonene.

Ved trykk på knappen <tilbakestill> blir alle nedtrekkmenyene nullstilt (blanke) og alle satte aksjoner med grønn bakgrunn forsvinner. Dette har ikke innvirkning på de lagrede komposisjonene, kun på simulatoren. For å retturnere til prototypen simulatoren ble startet fra, trykker brukeren på det røde krysset øverst til høyre. Prototypen simulatoren ble startet fra vil ikke være mulig å benytte så lenge simulatoren er åpen.

5.2 Veiviser

Veiviser er en metafor vi har valgt da den er kjent for de fleste, det er sjeldent den kjører i fullskjerm så den avviker litt fra standarden her. Det er mulig å navigere frem og tilbake ved hjelp av knappene <tilbake> og <neste>. Figur 17 viser prototypen rett før lagring av en komposisjon. Det er det fjerde og nest siste steget i veiviseren. Skjermbildet er todelt, det øverste er selve komponeringsflaten. Det er denne delen som bruker en veivisermetafor og det er dette området vi ønsker å teste med prototypen. Det aktuelle steget viser til venstre hvilket objekt komposisjonen starter med, røykvarsler. Under bildet vises den valgte hendelsen til røykvarsleren, ”blir utløst”. Mottakeren for komposisjonen er vist på høyre side av pilen, ”min mobil” med ”varsling via SMS” som aksjon. Da dette steget er et bekreftelsessteg og siste steget i en komposisjon har teksten på <neste> knappen blitt byttet ut med lagre. Dette for å gi brukeren en indikasjon på at han/hun nå er ferdig med den gjeldende komposisjonen.

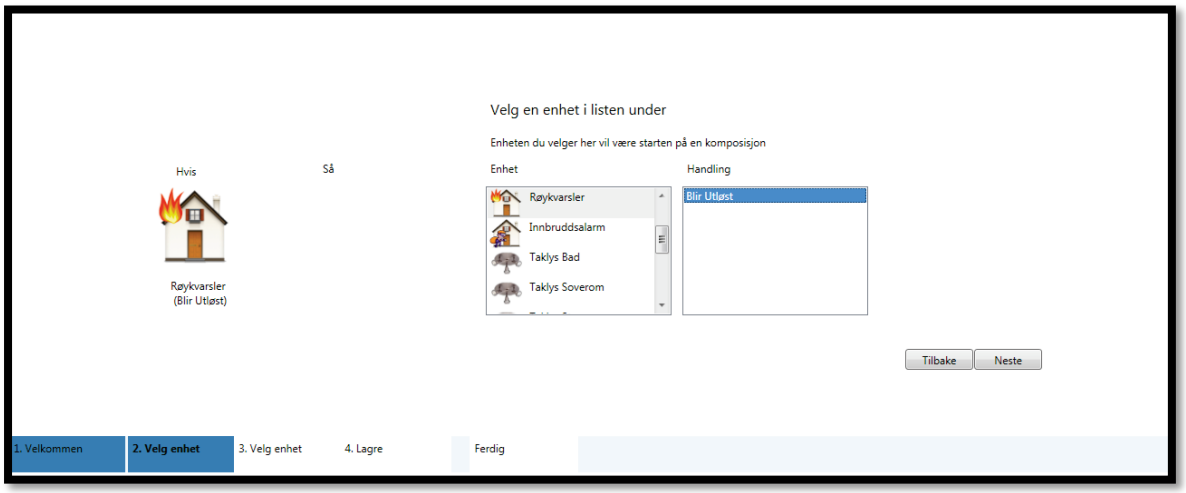


Figur 17 Hele Veiviserprototypen steg 4

De ulike delene i veiviser prototypen er adskilt ved en blågrå horisontal linjen midt på skjermbildet, under denne finner vi en liste over tidligere lagrede komposisjoner. Denne listen går igjen i flere av prototypene og vil kun bli beskrevet her, for siden å bli referert til som liste over lagrede komposisjoner. Listen viser de ulike og lagrede komposisjoner horisontalt. Først begynner den med et bilde av start objektet til venstre. En kolonne lenger til høyre er det bilde av mottaker objektet. I kolonnen lengst til venstre står en tekst med navn på objektene og hvilken handling som utføres. For å gjøre dette hurtig å lese har vi laget en egen syntaks: ”Objekt handling starter med + hendelse for startobjektet +:+ objekt som utfører en handling + aksjonen som skal utføres ”. I listen er det mulig å velge en komposisjon og tykke på knappene <rediger> og <slett>. Rediger bringer den valgte komposisjonen opp i komponeringsdelen av prototypen og her kan den endres. Ved trykk på slett blir den valgte komposisjonen fjernet fra listen og ikke lenger en komposisjon i hjemmet. Det er også mulig å benytte knappene: <del>, <delete>, <backspace> og <crtl> +< backspace> for å slette en komposisjon.

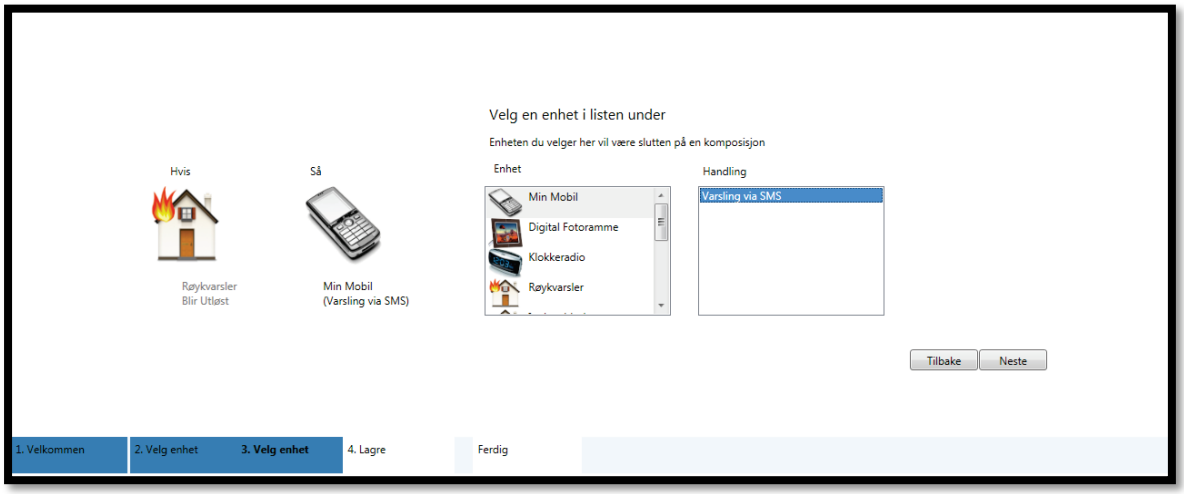
Figur 18 Veiviserprototypen steg 1

Det første som møter en bruker ved oppstart av veiviserprototypen er en velkomstskjerm, denne er vist i Figur 18. Av plassmessige forhold har vi kun tatt med den øverste delen av skjermbildet for de ulike stegene i veiviseren. Ved velkomstskjermen er det ingen tilbake knapp, da der ikke er noe å gå tilbake til. Her er kun en valgmulighet for brukeren, å gå videre til neste steg. Velkomst skjermen forteller kort hva programmet er ment å gjøre.



Figur 19 Veiviserprototypen steg 2

Trykker brukeren <neste> fra velkomstskjermen kommer han/hun til steg 2 hvor det er mulig å velge et startobjekt for komposisjonen. Når startobjektet er valgt vil det bli mulig å velge en hendelse. Hendelseslisten er først aktiv etter at et objekt er valgt, og innholdet i den vil variere etter hvilke objekt som er valgt. Når en handling er valgt vil <neste> knappen bli aktiv. Dette steget er vist i Figur 19. Under teksten: ”hvis”, i skjermbildet, er valgt objekt vist med aksjon under. Dette er på lik linje med oppsummeringsbildet i steg 4.



Figur 20 Veiviserprototypen steg 3

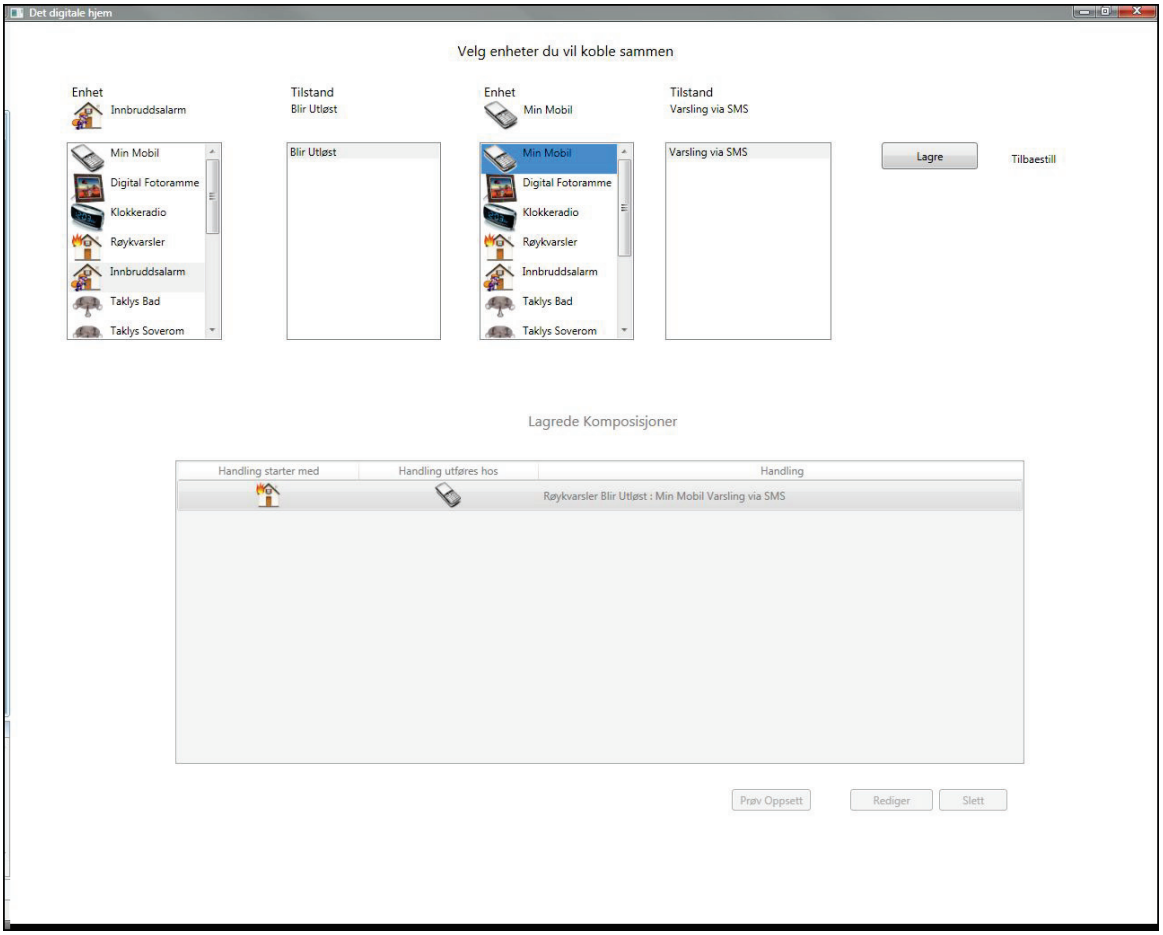
Etter at startobjektet med tilhørende hendelse er valgt kommer brukeren til steg 3, dersom <neste> knappen blir trykket. Dette er vist i Figur 20 og den viser at under ”så” er ”min mobil” valgt, dette er mottaker objektet for komposisjonen. Dette steget virker helt likt som steg 3, bare at det er mottaker som velges og ikke starten. Ved trykk på <neste>, kommer brukeren til steg 4 som er lagring og bekreftelse av laget komposisjon, denne er forklart og vist over.

Dersom brukeren velger å lagre komposisjonen er veiviseren ferdig og brukeren kommer til et stadium hvor listen med lagrede komposisjoner blir aktiv. Denne listen er deaktivert under komponering av en komposisjon for ikke å forvirre brukeren. I dette mellomstadiet blir veiviseren stående i en posisjon der brukeren må trykke på en knapp for å starte en ny komposisjon. Ved oppstart av en ny komposisjon hopper den over velkomst skjermen, da den vil være helt lik for hver komposisjon, og heller begynner i steg 2. Skjermbildet i mellomstadiet er vist i Figur 21.

Figur 21 Veiviserprototypen steg 5

5.3 WIMP

Denne prototypen er bygget på vindu, ikon, meny og peking (WIMP) metaforen. Dette er en av de mest vanlige metaforene i tradisjonelle dataprogrammer og skal derfor være lett gjenkjennelig for brukeren.



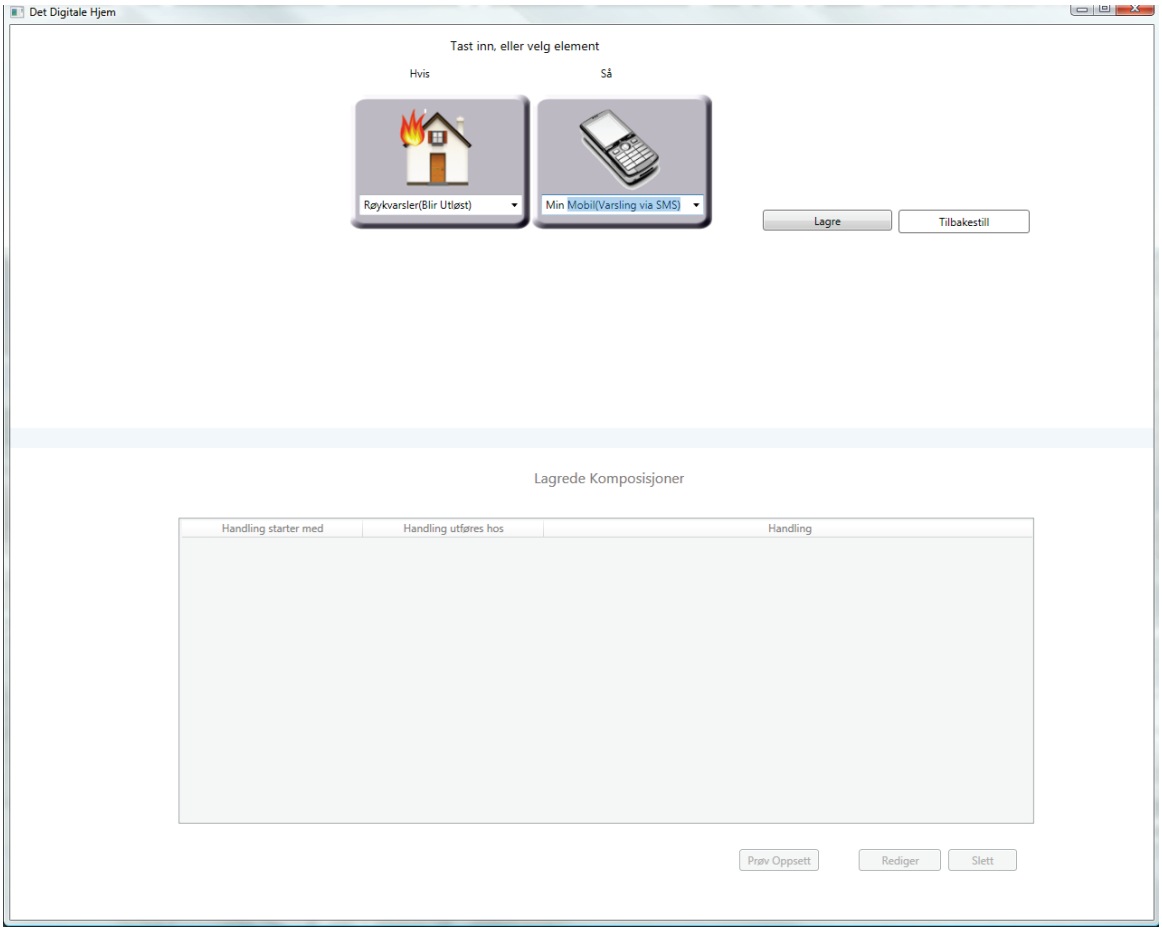
Figur 22 WIMP-prototypen

I Figur 22 er et skjermbilde av prototypen, vi ser at denne også har ett todelt design med lagrede komposisjoner i listen nede. Dette er den samme listen som i veiviser prototypen. Den øverste delen, komposisjonsdelen består av fire lister med informasjon om valgt element i listen rett over hver liste. I forbindelse med listene er der to knapper. En tydelig lagre knapp og en mer lenkeaktig tilbakestillings knapp. Knappene er inspirert fra flere internett sider (44) som alle benytter en stor påloggingsknapp/bekreftelsesknapp og en lenke for avbrytelser eller registrering før pålogging.

De fire listene i prototypen benyttes for å velge hvilke objekter og hendelser som skal inngå i en komposisjon. Den første listen er for å velge et startobjekt, og den andre listen for å velge en tilhørende hendelse. Den andre listen vil ikke være aktiv før det er valgt et objekt i den første listen. Dette gjelder for de påfølgende listene også, en liste vil ikke være aktiv før det er valgt et element i listen til venstre for den. De to siste listene er for valg av mottakerobjekt og dets aksjon til hendelsen fra startobjektet. Vi kaller ikke et objekt i huset for et objekt da det er blir teknisk, vi kaller det derfor enhet som er mer generisk og allment.

5.4 Grafisk kommandolinje

Denne modellen er inspirert av den hurtige måten en bruker kan åpne filer og programmer ved hjelp av Quicksilver (avsnitt 2.4.3). For å være hurtig krever metaforen aktivt bruk av tastaturet, det kan være tungt å lære dersom programmet ikke benyttes ofte og derfor har vi gjort den fult mulig å bruke kun med mus.



Figur 23 Grafisk kommandolinjeprototype

Metaforen er på lik linje med veiviser og WIMP todelt, med den samme listen over lagrede komposisjoner i bunn. Figur 23 viser et skjermbilde av prototypen under komponering av en tjeneste. Prototypen består også av to knapper (<Lagre> og <Tilbakestill>) tilsvarende dem i WIMP ovenfor. Det store skillet er at startobjektet og dets hendelser er knyttet sammen i en og samme nedtrekkmeny. Det samme gjelder for mottakerobjektet med dets aksjoner, dette fører til mange valg på nedtrekkmenyen. Valgene her er ikke sortert, men de ulike objekter og dets handlinger er gruppert sammen. Vi valgte usorterte lister for å hinte om en mer effektiv måte å bruke prototypen på enn å bla i nedtrekkmenyene. For videre å hinte frem på om tastaturet kan benyttes ble teksten: ”Tast inn eller velg et element” plassert øverst i prototypen. For å indikere hva som er utløsende faktor, altså starten på en komposisjon er teksten ”Hvis” og ”Så” plassert over boksene.

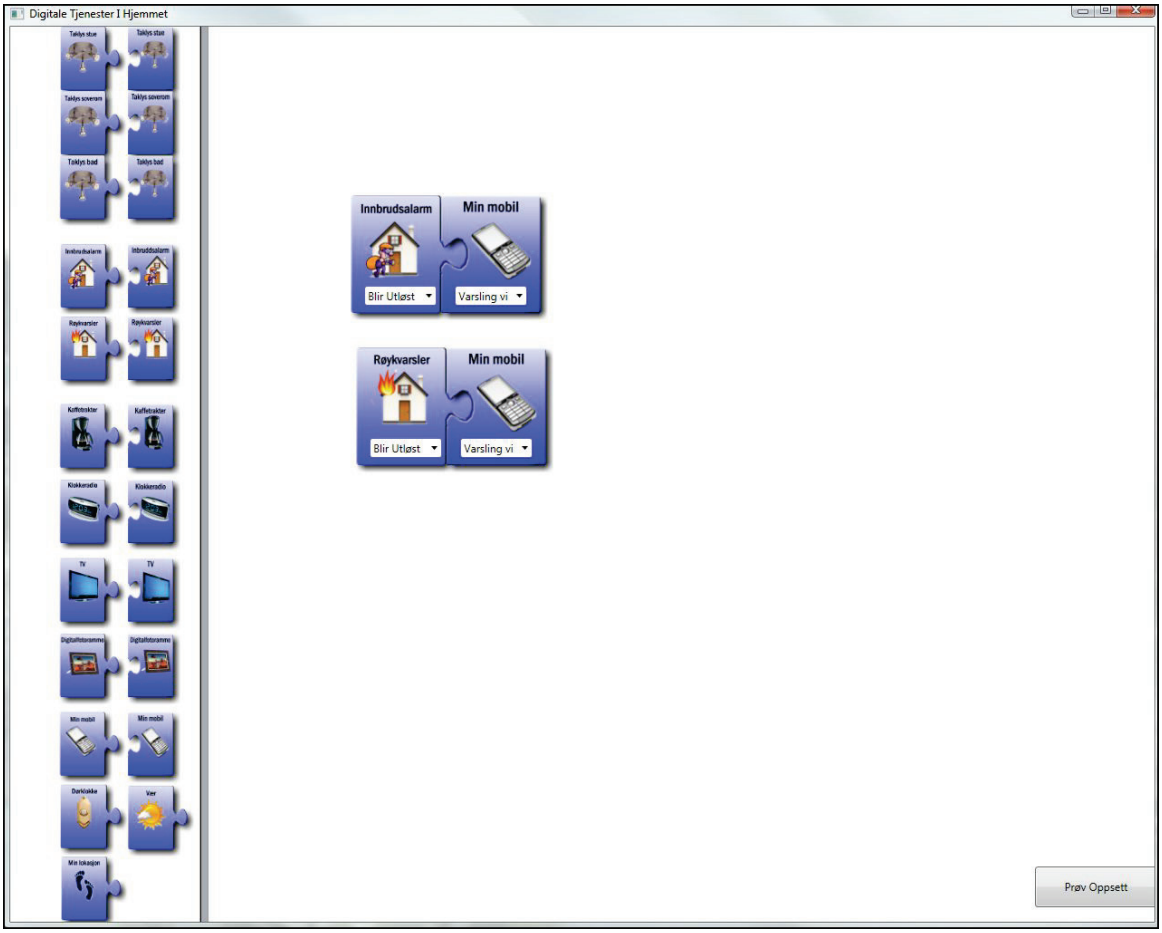
For å lagre en komposisjon må det være valgt objekter i begge de grå boksene. Først når dette er gjort vil lagreknappen bli aktivert. Valg av en komposisjon kan gjøres ved trykk på en

nedtrekkmeny eller en av de grå boksene. Når dette er gjort vil den valgte nedtrekkmenyen åpnes og et element kan velges ved hjelp av musen. Når den ene er ferdig kan tilsvarende fremgangsmåte benyttes på den andre nedtrekkmenyen/boksen.

En hurtigere variant vil være å bruke tastaturet, så fort brukeren begynner å skrive inn tekst vil den første boksen åpnes og de elementer som begynner med teksten brukeren har skrevet inn blir foreslått. Dette er vist på den grå boksen til høyre i Figur 23, dessverre lukket nedtrekkmenyen seg i forsøk på å ta skjermbilde. Når brukeren har funnet riktig valg kan hun/han trykke <Enter> og vil da gå videre til den andre grå boksen og dens nedtrekkmeny. Dersom denne allerede er utfylt vil Enter medføre et fokusskifte til <lagre> knappen. Et trykk til på <Enter> og komposisjonen blir lagret.

5.5 Puslespill

Puslespill er en metafor brukt i enkelte sluttbrukerprogrammerings programmer med mange ulike former på brikkene. Da den konseptuelle modellen vår er den enkleste form for en komposisjon har vi ikke behov for mer enn to typer brikker. En for startobjektet og dets hendelse som kan kobles sammen med et mottakerobjekt med aksjon på.



Figur 24 Puslespillprototypen

Puslespillet vårt er delt inn i to områder, et lite område for brikker som ikke er i en komposisjon. Til høyre finner vi et stort område hvor brikkene kan settes sammen til ulike komposisjoner.

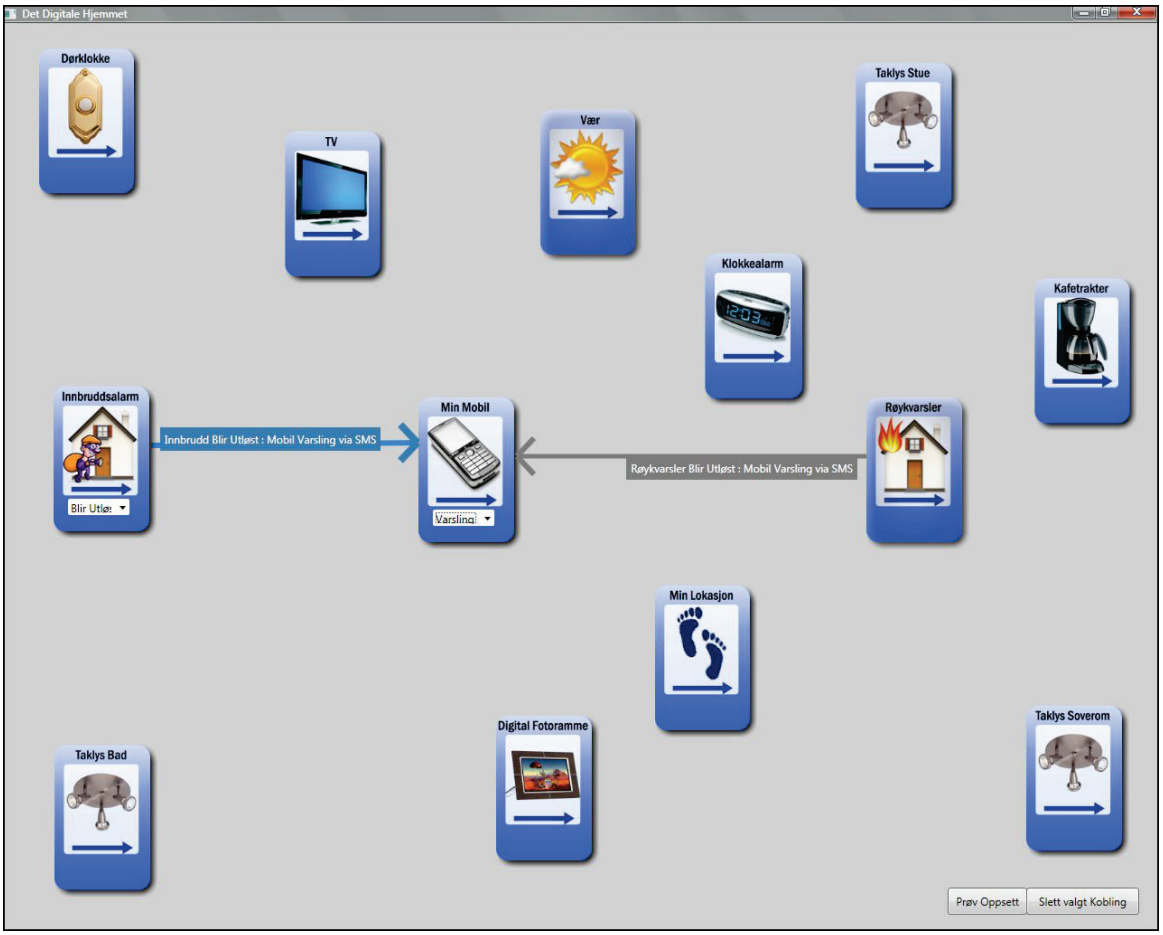
Figur 24 viser et skjermbilde av puslespill prototypen, hvor det er laget to komposisjoner. Brikkene kan flyttes ved hjelp av å føre musepekeren over en brikke og trykke ned museknappen. Under forflyttning vil brikken hele tiden være synlig for brukeren. En komposisjon lages ved å trekke en brikke fra venstre over til komponeringsflaten på høyre side. Når brikken passerer den vertikale stripen som skiller de ulike delene blir brikken større og får en nedtrekkmeny på seg. Det vil samtidig komme en tilsvarende brikke tilbake i området til venstre. Den nye brikken kommer fordi et og samme objekt kan være med i mange ulike komposisjoner.

I komponeringsområdet er det ingen begrensninger på hvor en brikke kan plasseres. Når en startbrikke og mottakerbrikke kommer i nærheten av hverandre med åpningen eller piggen vil de sprette sammen automatisk. For å koble de fra hverandre er det bare å dra den ene brikken bort fra den andre. Dersom en brikke skal fjernes dras den tilbake til høyresiden og den vil forsvinne. Vi har valgt at startobjektet alltid plasseres til høyre for mottakerobjektet og henholdsvis hendelse og aksjon settes på de respektive objektene sine nedtrekkmenyer. Brikkene har også fått skygge under seg, dette valget har vi gjort for å oppmuntre brukerne til å flytte dem rundt, vi ønsket å skape en illusjon av at brikkene lå oppå et bord.

5.6 Objektkobling

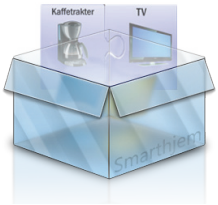
Objektkoblingsmetaforen er inspirert av ulike modelleringsverktøy og de populære webtjenestene, pipe og popfly. Vi har valgt å gjøre prototypen forholdsvis enkel med kun et nivå. Her kunne vi brukt to nivå, hvor du dykket ned i et nytt nivå for å sette hendelse og aksjoner. Dette avsto vi fra, da det ville skjule informasjon for brukerne og gjøre det lett å glemme å sette aksjon og hendelse på objektene.

I prototypen er hvert objekt kun presentert en gang, og der er kun en del programmet i motsetning til de øvrige prototypene. Hvert objekt kan her både være et startobjekt og et mottakerobjekt, retningen på pilen mellom dem avgjør hvilken rolle objektet har. Et objekt kan flyttes fritt rundt på skjermen ved å peke i rammen på det og trykke ned museknappen for å dra det til ønsket sted. Under flytting av et objekt vil det hele tiden være synlig. Prototypen er vist i Figur 25 og vi ser at pilene mellom brikkene har tekst i tilknytning til seg. Denne teksten er bygget opp etter samme syntaks som teksten i listen over lagrede elementer fra veiviser, WIMP og grafisk kommandoline. Da et objekt kan ha mange piler inn og ut fra seg, er det ikke mulig å vise hendelse og aksjoner på objektene. Disse egenskapene må derfor forflyttes til pilene for å være synlig hele tiden. En pil i fokus, den pilen brukeren sist tegnet eller har trykket på, vil skifte farge til blått. Det er denne pilens hendelse som blir synlig i nedtrekkmenyen på de involverte objektene. Dette betyr at ved tegning av en pil, vil det bli synlig en nedtrekkmeny på start- og mottakerobjektet. I denne nedtrekkmenyen kan brukeren angi hendelse og aksjon. For å minne brukeren på dette vil det stå ”?:?” på pilen helt til hendelse og aksjon er satt.



Figur 25 Objekt koblingsprototypen

En komposisjon består av en pil og de to objektene den er koblet sammen med. For å lage en komposisjon flyttes musepekeren innenfor den hvite rammen på et objekt. I det pekeren treffer rammen på objektet blir pekeren til et flyttesymbol, føres den inni det hvite blir den en penn for å indikere at det er mulig å tegne. Dette er en skjult egenskap, for å gjøre det lettere å oppdage denne egenskapen. Dersom brukeren trykker på pilen under objektet vil brukeren få en melding om hvordan elementer kobles sammen. Når brukeren har en pen som musepeker kan han/hun tegne en strek til objektet han/hun vil koble det sammen med. Da vær og ”min lokasjon” ikke kan være mottakere vil disse få en rød ramme om de blir forsøkt tegnet til, de øvrige objektene får en grønn ramme. Den grønne rammen indikerer også at brukeren kan slippe musepekeren og objektene blir koblet sammen.



6. Brukbarhetstesting - gjennomføring

Dette kapittelet beskriver hvordan brukbarhetstestingen ble gjennomført og hvilke utfordringer vi møtte og løste på ulike måter.

6.1 Oppgaver

Oppgavene ble formulert som problemer i hverdagen for å kontrollere at testdeltagerne forstod hva de gjorde. Vi valgte derfor å bruke andre navn på objekter i oppgaveteksten enn på dataskjermen. Oppgavene ble gitt i samme rekkefølge til alle deltagere, og de løste oppgavene for hver av de ulike prototypene. Under er oppgaven gjengitt sammen med forklaring og løsningsforslag til hver enkelt oppgave. Alle oppgavene kan løses på flere måter og ingenting er mer rett enn noe annet.

Liste 12 Oppgaver til brukbarhetstest

1. Jeg ønsker å bli varslet raskt ved et eventuelt innbrudd eller branntilløp i boligen.
   Hensikt: Oppvarmingsoppgave for å gi brukeren mulighet til å utforske prototypen og metaforen.
   Løsning: Oppgaven har flere mulige løsninger, brannvarsler og innbruddsalarm kan kobles til ulike gjenstander for varsling.
2. Lyset på soverommet skal være på når jeg er der, og av når jeg ikke er der.
   Hensikt: Bruk av lokasjon i en komposisjon
   Løsning: Flere mulige løsninger, mest logisk er å koble ”min posisjon” til lyset på soverommet.
3. Jeg tar ofte bilder med mobilen når jeg er ute og reiser. Det hadde vært fint om bildene jeg sendte til huset havnet på veggen i stuen.
   Hensikt: En komposisjon som er aktiv når du ikke er hjemme
   Løsning: koble mobiltelefon sammen med en digital fotoramme
4. Jeg har det ofte travelt om morgenen før jeg drar på jobb, men trenger som regel en kopp kaffe å våkne på. Det hadde vært praktisk å kunne stått opp til nytraktet kaffe!
   Hensikt: Lage noe praktisk og nyttig i hverdagen
   Løsning: Koble vekkerklokken sammen med kaffetrakteren
5. Jeg lurer ofte på hvordan været er ute når jeg våkner om morgenen før jeg skal på jobb.
   Hensikt: Artig oppgave, leke litt rundt
   Løsning: koble vær og vekkerklokke sammen, for å skifte alarmlyd på vekkerklokken etter værforholdene
6. Når jeg sitter og ser på TV hører jeg ikke alltid om noen ringer på døren.
   Hensikt: Praktisk oppgave hvor det er lett å kjenne seg igjen i situasjonen
   Løsning: Koble dørklokken til tv

7. Jeg angrer på at jeg har bedt huset om å lage kaffe for meg når jeg våkner.
   Hensikt: Endre/fjerne en tidligere komposisjon
   Løsning: Slett løsningen på oppgave 4

6.2 Pilottest

En pilottest er en test av testopplegget til en brukbarhetstest. Pilottesten har ikke til hensikt å teste prototypene, men selve testopplegget. Resultatet fra testen kan derfor ikke brukes til videre analysearbeid vedrørende prototypene. I kapittel 13 i boken (35) står det at en pilottest bør gjennomføres to dager før første testperson ankommer. Vi var usikre på deler av testopplegget og hvor lang tid en test ville ta. Vi valgte derfor å gjennomføre pilottesten tre uker før første testperson skulle komme på brukbarhetstest. Vi valgte så god tid for å rekke små endringer i prototypene og endring av oppgavesettet og rammen rundt det. På grunn av denne usikkerheten var det viktig å sette av nok tid til å gjennomføre en ny pilottest, for i tilfellet det ble store endringer ville det være lurt å gjennomføre en ny pilottest før testsubjektene kommer.

Under pilottesten oppdaget vi at testen tok for lang tid, vi valgte derfor å kutte ned på antall oppgaver. Det ble også oppdaget et par problemer med prototypene, to av dem krasjet under testen og måtte videreutvikles. Da vi ikke gjorde endringer som ble synlig for brukeren og det ikke ble store endringer i oppgavene gjennomførte vi ikke en ny pilottest.

6.3 Oppsett

Både pilottesten og brukbarhetstester foregikk i lokalene til Norsk senter for elektronisk pasientjournal (NSEP). De har et brukbarhetlaboratorium i samarbeid med NTNU. Laboratoriet er delt i to, en testavdeling og en teknisk avdeling til opptak. Det tekniske rommet, vist i Figur 26, samler alle lyd og bildestrømmer fra testavdelingen. Ved hjelp av en miksepult kan lyd og ulike bildekilder kombineres til en video for videre analyse.

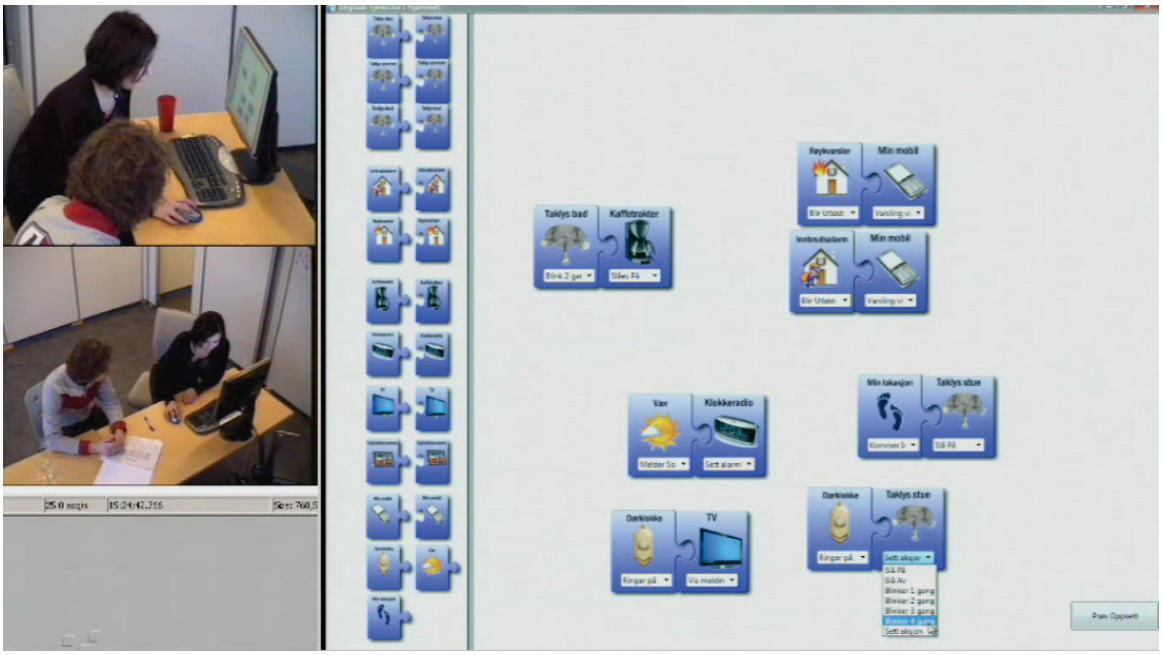
Figur 26 Teknisk rom

Testlokalene består av et stort rom med flyttbare vegger, dette er tilpasset en mobil brukbarhetstesting i en klinisk setting. Vi hadde kun behov for å teste brukerne i et vanlig rom og benyttet derfor bare et vanlig rom laget av flyttbare vegger. Figur 27 viser testlokalene, med en tiltenkt testperson foran datamaskinen til høyre og testlederen til venstre. Rommet er veldig sterilt med få eller ingen forstyrrende elementer. På bordet er det en skjerm, trådløst tastatur og mus. Oppgavene testsubjektet skal løse ligger også på bordet, disse er bundet inn i en flippoverblokk med laminerte sider som gjør det lett å bla. På veggen helt til høyre henger en plakat i A3 størrelse over de ulike elementene i smarthjemmet. Dette er de samme elementene testpersonen vil finne i de ulike prototypene og denne plansjen er lagt ved i Vedlegg VI

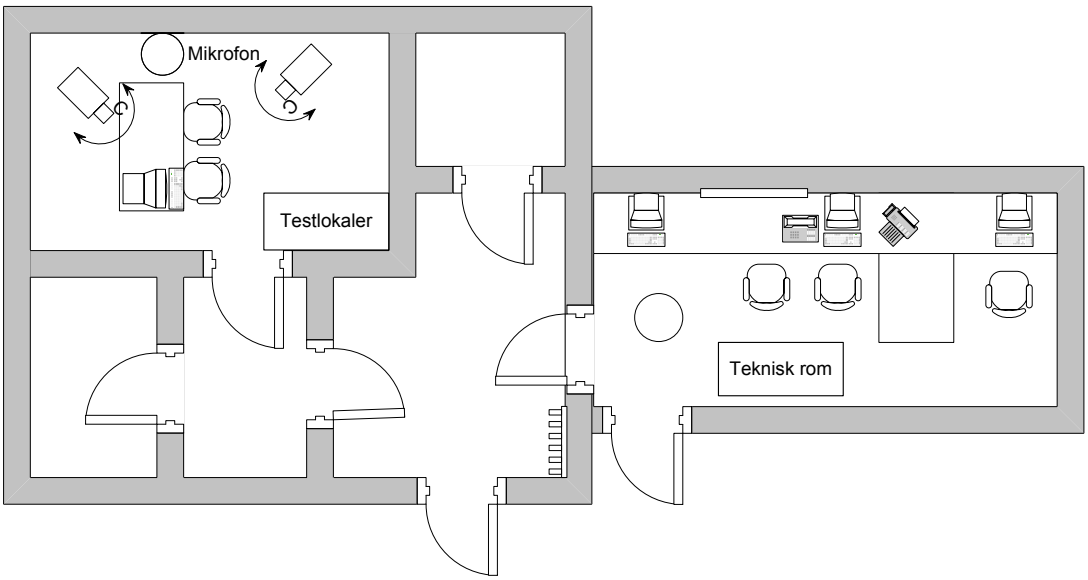
Figur 27 Testlokaler

I testlokalene er det også montert to kamera og flere mikrofoner i taket. Disse tekniske detaljene er veldig diskret og utenfor synsfeltet til testpersonen. I forbindelse med dataskjermen er det montert en liten boks som ligger bak datamaskinen for å ta opp skjermbildet av det testpersonen gjør på datamaskinen. Det er data fra disse elementene som samles til en videostrøm i det tekniske rommet.

I Figur 28 er et skjermbilde av den endelige videoen vist, fra hver test blir det en video tilsvarende denne. Testleder slipper å notere alt under selve testen da analysene kan foregå i ettertid ved hjelp av videoene.



Figur 28 Video strømmer fra testlokalet



Figur 29 Oversikt over brukbarhetslaboratoriumet

En planskisse over laboratoriet er illustrert i Figur 29, denne er ikke målbar, men viser godt hvordan vi bare bruker det ene rommet i den mobiltilpassede omgivelsen. Det tekniske rommet og testlokalene ligger avskilt, så det er ikke problem med støy mellom dem. Dette gjør det til et velegnet område å gjennomføre brukbarhetstestene på.

6.4 En vanlig test

Vi tilstrebet å holde de 16 distinkte testene så like som mulig, dette for å kunne sammenligne resultatene i ettertid. Denne beskrivelsen av en vanlig test vil derfor være gjeldene for alle testene og fremgangsmåten er det endelige resultatet av pilottesten.

6.4.1 Introduksjon

Hver deltager ble møtt i ytterdøren til NSEP og fikk tilbudet om en kopp kaffe eller te før testen begynte. Dette ble gjort for å få en hyggelig velkomst. Liker etterpå tok vi med drikken inn i rommet merket testlokaler på Figur 29 og testdeltageren fikk en introduksjon om hva som skulle skje. Introduksjonen begynte med hva en brukbarhetstest egentlig er, og hva som skulle testes. I løpet av denne introduksjonen ble det presisert flere ganger at det er systemet som skal testes og ikke testpersonen. Like etter fulgte en introduksjon av hva et smarthjem er, og det ble gitt et eksempel på en komposisjon av en tjeneste. Denne delen ble forklart ved hjelp av plansjen over hjemmet som hang på veggen, på denne måten fikk testpersonen en liten følelse av hvilke elementer de hadde tilgjengelig i hjemmet sitt. Ikonene og navne på objektene på plansjen er de samme som i prototypene. Etter dette satt vi oss ned ved bordet og testpersonen fikk sitte foran datamaskinen. Her var tastatur og mus lagt til side og testpersonen måtte fylle ut et skjema med personopplysninger (avsnitt 4.3.1). Etter dette skjemaet ble det spurt om tillatelse til å ta opp lyd og bilde av selve testen under forutsetning at resultatene ble holdt konfidensielt og at kun et fåtall personer fikk se videoene. Hvis testpersonen svarte ja på dette ble videoopptaket startet.

6.4.2 Testing

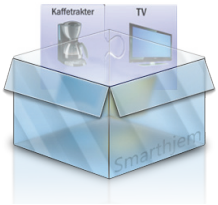
Selve testen foregikk ved at testleder presenterte en og en prototype i tilfeldig rekkefølge. Rekkefølgen var bestemt ved hjelp av trekning i forkant av hver enkelt test. Dette medførte at testleder måtte starte og stoppe hver prototype under testen. For hver prototype fikk brukeren presentert de syv oppgavene i stigende rekkefølge. Hele tiden under testen ble det gjort oppmerksom på at testdeltageren måtte tenke høyt for å hjelpe oss å forstå systemet. Vi mistenkte også at testdeltagerne ikke kom til å benytte tastaturet på grafisk kommandolinjeprototypen. Det ble derfor hintet om denne muligheten på følgende måte etter oppgave 4: ”her kan du også bruke tastaturet”. Etter testen foretokk vi en kort oppsummering før vi fortsatte med spørreundersøkelser (avsnitt 4.3) og et lite intervju (avsnitt 4.2).

6.5 Problemer

Vi hadde et lite teknisk problem ved noen av testene, dette førte til at vi ikke fikk opptak av all videoen, men lyden er med på alle. Lyden og ved at vi kjenner rekkefølgen prototypene ble testet i, gjør det mulig å bruke opptakene til kvalitativ analyse. Således er ikke dette et reelt problem og kan sees bort ifra.

Flere av brukerne hadde vansker for å tenke høyt, dette gjorde det utfordrende for testleder som hele tiden passe på at testpersonen tenkte høyt. Vi ønsket ikke at testdeltageren skulle føle seg maset på og stresset, vi prøvde derfor å si ifra pent innimellom når testdeltageren tydelig lurte på noe, eller var usikker.

Det viste seg også vanskelig å få testdeltagerne til å komme fortløpende, dette til tross for at de ble kompensert med 250 kr for å stille opp. Dette medførte at brukbarhetstestingen trakk ut i tid. Vi hadde som mål å gjennomføre 3-4 tester per dag. En time per deltager, pluss eventuelle tilegg og at testdeltagerne kom 10min for tidlig eller seint. Derfor valgte vi å sette av 2 timer per deltager og begynte kl 10.00 hver dag. Dessverre ble det en del hul i timeplanen og mye tid gikk til spille. Dette løste seg ved at laboratoriet var mye ledig i den aktuelle perioden, og vi kunne reservere det for et lengre tidsrom.



7. Resultater

Dette kapittelet beskriver resultater fra metodene i kapittel 4 Forskningsdesign.

7.1 Deltagere

Datastudentene var forholdsvis enkelt å få fatt i da vi arbeider i samme miljø. Det ble valgt datastudenter fra ulike retninger, med og uten barn og fra ulike etniske samfunn. Dette ble gjort for å reflektere den reelle brukergruppen så godt som overhodet mulig. Det var utfordren å få et likke variert utvalg av arkitekter og det ble et flertall jenter (7 jenter og en gutt). Dessverre hadde en av datastudentene ikke noe erfaring med programmering og hvordan dataprogrammer virket. Den personen er derfor behandlet på linje med en arkitektstudent. For arbeid videre har hver testperson blitt anonymisert, arkitektpersoner fra A1-A8 og datapersoner D1-D8. De ulike testpersonene og deres programmeringserfaring er oppsummert i Tabell 1. Programmeringsferdighet er her omkodet (se avsnitt 4.3.1 for kategoriene) til tall hvor 0 er ingen programmeringsferdighet og 4 er størst programmeringsferdighet. Dette er nominelle data, og det er derfor umulig å si noe om hvor mye bedre 4 er enn 2.

Tabell 1 Testpersoner

Navn	Kjønn	Programmerings ferdighet	Alder
A1	Jente	0	24
A2	Jente	1	27
A3	Jente	0	25
A4	Jente	0	23
A5	Jente	0	24
A6	Jente	2	29
A7	Jente	1	26
A8	Gutt	1	27
D1	Gutt	4	23
D2	Gutt	4	25
D3	Gutt	3	29
D4	Gutt	3	29
D5	Gutt	4	25
D6	Jente	3	23
D7	Jente	3	25
D8	Jente	2	26

7.2 Brukbarhetstesting og intervju

De ulike prototypene blir vurdert både kvantitativt og kvalitativt, Her vil vi presentere de kvalitative resultatene fra brukbarhetstesten med fokus på hva som er bra og dårlig med de ulike prototypene. Vi har her valgt å inkludere både utsagn under brukbarhetstesten og intervjuet i samme del da testpersonene hadde de samme meningene under intervju og brukbarhetstest. De faste spørsmålene fra intervjuet og deres kvantitative analyse vil bli holdt utenfor.

Prototypene ble presentert i tilfeldig rekkefølge og noen av utsagnene kommer fordi det er første eller siste modell de prøver. En gjenganger på første modellene er at de utforsker brukergrensesnittet en del i begynnelsen og prøver seg frem. Etter fire tester husket deltagerne mange av oppgavene og gikk følgelig fort frem. Dataene er presentert etter en mal hvor hver prototype presenteres hver for seg. En prototyp er videre delt inn i to tabeller, en for positive kommentarer, hendelser og forbedringsmuligheter. Denne tabellen inneholder også et felt for andre hendelser, her er kommentarer vedrørende interaksjonsmetoden som ble benyttet. Dette feltet er utelatt dersom mus er eneste brukte interaksjonsmetode. Den andre tabellen inneholder negative hendelser og kommentarer, samt forbedringsmuligheter. Denne presentasjonen av dataene er presentert iVedlegg VII, og blir referert til ved hjelp av tabell nummer videre. For hver av prototypene i den videre analysen vil forskjellene mellom de ulike brukergruppene bli fremhevet, samt hva som er bra eller dårlig med en prototyp.

7.3 Kategorisering av kvalitative data

Alle utsagnene fra Vedlegg VII ble skrevet på post-it lapper, en farge for hver prototyp. I tillegg inkluderte vi dataene fra spørreundersøkelsen som ikke går direkte på en bestemt prototyp, dette fikk en egen farge. På hver post-it ble det merket hvilken brukergruppe den representerte og maks et utsagn på hver lapp. Alle lappene ble så hengt opp på en tavle, de som var like eller hadde noe med hverandre ble plassert i nærheten av hverandre, dette ble gjort uavhengig av prototyp og brukergruppe.

Figur 30 Kategorisering av kvalitative data

Etter at alle lappene var plassert på tavlen, og stokket litt om på sto vi igjen med 13 grupper, hvor flere av gruppene er undergrupper til en annen gruppe. I Figur 30 er et bilde av tavlen etter at lappene var hengt opp. Under vil de ulike gruppene og undergrupper bli presentert:

Liste 13 Liste over kvalitative kategorier

- **Visuelt:** Alle utsagn om utseende til en prototyp, fargevalg, fin, stygg, rar, søt osv.
- **Effektivitet:** Hvor effektiv føles prototypen i bruk? Raskt, sakte, mye trykking med mer. Her har vi kun notert ytterpunktene og det er kvalitativt målt.
- **Implementering:** Hvordan representerer prototypen metaforen? Er det i overensstemmelse med brukerens mentale modell? Hvilke problemer oppsto som en følge av dette?
    - **Muligheter:** Er der noen objekter som har for få/feil muligheter? Eller er det rare objekter vi har valgt å ha med i testen? Denne har ikke noe med den enkelte prototyp å gjøre, da det er like muligheter i alle prototypene.
- **Interaksjon:** hvilke interaksjonsmuligheter er benyttet, merk her har vi ikke skrevet noe dersom alle har brukt mus alene.
- **Krasj:** Når testen har stoppet opp og brukeren måtte ha hjelp videre.
- **Metafor:** Generelle punkter rundt metaforen, uvant, tungvint, forstyrrende og lignende.
    - **Opplevelse:** Hvordan føler brukeren prototypen er? trivelig, sikker, rasjonelt, kjedelig osv
    - **Rekkefølge/forståelse:** Er det problem med rekkefølgen av komposisjoner?
    - **Utforskningsvilje:** Har testpersonene prøvd mulighetene? Leker eller søkes det alternative løsninger?
- **Konseptuelmodell:** Er der hendelser eller kommentarer som strider mot den konseptuelle modellen?
- **Sammenligning:** Utspill rundt sammenligning av prototypene og andre programmer eller konsepter brukerne er kjent med.
- **Smarthjem:** Utsagn knyttet til domenet

De overnevnte kategoriene vil bli brukt under videre presentasjon av resultatene og er med på å besvare RQ3.

7.3.1 Kvalitative resultater puslespillprototypen

Denne prototypen er veldig grafisk og viser alt i ett og samme vindu. Tabell 26 viser at programmere finner det positivt at det er oversiktlig samtidig som alt presenteres i ett og samme vindu. De finner det også bra at det går fort å løse prototypen, samtidig som det er lett å se hva som er hva i en komposisjon. Den andre gruppen finner prototypen morsom å jobbe med, samtidig som det er lett og forstå hva som kan gjøres. For å gjøre prototypen enda bedre foreslår begge grupper en annen ny måte å presentere hendelser og aksjoner på. En ”ikke programmerer” foreslår at nedtrekkmenyene bør åpnes i det en brikke flyttes ut i komposisjonsområdet. En av Programmererne forventet å finne egne brikker for hendelser og aksjoner. Egne brikker for dette er ikke med i prototypen men er mulig å gjennomføre for metaforen. Mange programmerere ønsket å bygge mer avanserte puslespill, altså å koble flere brikker sammen. Det er ikke mulig da det ville bryte med den konseptuelle modellen. En ikke

programmerer ønsket også en større form for bekreftelse for når en komposisjon er ferdig, gjerne at de skifter farge.

Der er også flere ulemper og problem ved puslespillet, ”ikke programmerere” finner det vanskelig å se sammenhenger. De er også enig med programmerere at det er stor fare for å få en uryddig skjerm. Det er et reelt problem da alt vises samtidig og det er begrenset hvor mye plass det er på en skjerm. Begge brukergruppene syntes brikkene til venstre er for små, dette er lett å gjøre noe med, så vi fokuserer heller på større problem som at en ”ikke programmerer” ikke forstod forkskjell på start- og mottakerobjektet i en komposisjon. Dette er et reelt problem da eneste indikasjon på start og mottaker er rekkefølgen fra venstre til høyre. I tilegg var det to ”ikke programmerere” som prøvde å lage komposisjoner i det høyre feltet, altså ikke dra brikkene over på komponeringsflaten. Dette er alvorlige problemer med prototypen. Tabell 27 inneholder de kvalitative dataene, og vi ser her at ”ikke programmerere” møtte flere problemer og fant flere negative elementer enn programmerere.

Det er en forskjell mellom programmerere og ”ikke programmerere” for puslespillet, programmerere forstår bedre hvordan en komposisjon skal lages, altså hvor de skal legge puslespill brikker, og i hvilken rekkefølge. De forstår bruken av dette, mens ”ikke programmerere” har problemer med å se rekkefølgen av en komposisjon og hvor en komposisjon kan lages.

Tabell 2 Kategorisering av puslespillet

Kategorier	**Brukeropplevelse**	
	Programmerere	**Andre**
Visuelt	Stilig	Søt
Effektivitet	Hurtig	Hurtig
Implementering	Liten skrift på brikker til venstre, Egen brikke for handling/aksjon Lese hele teksten i en lukket nedtrekkmeny	Tydeligere at noe er lagret Åpne nedtrekk automatisk liten skrift på brikker til venstre
Muligheter		
Interaksjon		
Metafor	Rotete Alt i samme vindu Rett på sak Tydelig Ikke for lett i bruk Vanskelig å se om det skjer for mye	Så enkel at den blir dårlig
Opplevelse	Morsomt	Trivelig, innlysende, gøy
Rekkefølge	Blandet	Utfordrende
Utforskningsvilje	Ser nye muligheter	Leker seg til nye løsninger
Sammenligning	Mer tydelig enn grafisk kommandolinje	

Puslespillet oppsummeres ved hjelp av kategorier fra Liste 13 i Tabell 2, viser blant annet at begge brukergrupper har en positiv opplevelse ved bruk av prototypen.

7.3.2 Kvalitative resultater WIMP-prototypen

WIMP-prototypen fikk gode tilbakemeldinger fra begge brukergruppene, men det varierer hva de syntes er bra. Tabell 28 viser at programmerere finner metaforen oversiktlig, mens den andre gruppen har flere forskjellige positive poenger. De sier også at den er oversiktlig, men viktige aspekter som å føle seg trygg, rask, logisk og finne muligheter, er blant argumentene. Dette er den mest klassiske metaforen, noe som påpekes av en ”ikke programmerere”.

En av programmererne foreslo en gruppering av start og mottaker delene for å gjøre det lettere å skille hva som er starten og slutten på en komposisjon. Han forklarte under testen hvordan dette burde gjøres og denne forklaringen er gjengitt i form av en skisse tegnet basert på hans forklaring. Denne skissen er vist i Figur 34 hvor hans forslag er ekstra uthevet med den sterke blåfargen og stor skrift på ”Hvis” og ”Så”. Vedkommende kommenterte også at det ikke burde hete enhet og tilstand på begge, men kanskje enhet – hendelse og enhet – aksjon.

Figur 31 Forbedringsmulighet for WIMP-prototypen

Forslaget til forbedring er en direkte følge av negative tilbakemeldinger vedrørende representasjonen av start- og mottakerobjekt. Det er verdt å merke seg at det kun er programmerere som finner dette rart på denne modellen (fra Tabell 29), i motsetning til puslespillet hvor det var ”ikke programmere” som hadde problemer med rekkefølgen.

En kategorisert oppsummering av prototypen er i Tabell 3 og vi legger spesielt merke til at begge grupper mener det er en hurtig prototyp.

Tabell 3 kategorisering av WIMP

Kategorier	Brukeropplevelse	
	Programmerere	**Andre**
Effektivitet	Hurtig	Hurtig
Implementering	Listene burde vært sortert Steg 5 fra veiviser burde vært her	Tydeligere rekkefølge
Interaksjon	Tastaturet er tungvint	
Metafor	Fin til å sette opp mye med Oversiktlig	Oversiktlig Lett å forstå Logisk
Opplevelse	Smart	Trygg, likte den ikke
Rekkefølge	Blandet	Forståelig
Utforskningsvilje		Leker seg til nye løsninger
Sammenligning	Ligner de andre, mer oversiktlig	Klassisk og ligner puslespill

7.3.3 Kvalitative resultater veiviserprototypen

Veiviser er en metafor mye brukt til installering av programmer, noe en av ”ikke programmererne” gjenkjenner og påpeker (Tabell 30). Vi ser at ikke programmerere generelt sett har mye positivt å si om prototypen. De finner den lett, gir forståelse av hva de ulike objektene er, samtidig som de finner den oversiktlig. Programmerere hadde lite bra å si om denne modellen, annet enn at den sikkert er fin å bruke første gangen noe skal gjøres.

Et stort savn for programmerere ved veiviser prototypen er at det ikke er mulig å navigere ved hjelp av å trykke på statuslinjen som skiller komponeringsområdet og listen over lagrede elementer.

Tabell 31 inneholder mange negative elementer så vel som problemer. En av programmererne likte overhodet ikke veiviser som metafor uavhengig av hva den skulle brukes til. Mange av de andre programmererne følte den tok lang tid å bruke, og var tungvint. I begge gruppene var der problemer med å skille start- og mottakerobjekt i en komposisjon, enkelte hadde også vansker ved å se sammenhengen mellom det første og andre objektet de valgte. Det var knyttet flere problemer til det å lage en komposisjon. Her skilte ”ikke programmererne” seg ut, ved at de hadde større problemer enn programmererne. I oppsummeringen i Tabell 4 viser at begge gruppene likte metaforen rent visuelt, men at den gikk for tregt å bruke.

Tabell 4 Kategorisering av veiviserprototypen

Kategorier	Brukeropplevelse	
	Programmerere	**Andre**
Visuelt	Penest	Proft
Effektivitet	Treg	Rydding er tregt
Implementering	Steg 2 og 3 samme navn Statuslinjen burde være trykkbar	Bekreftelse før lukking
Interaksjon		Mye musetrykking
Metafor	Tung å bruke Hater veivisere på generelt grunnlag Må tenke mye	Liker den ikke Bra for gamle Bra med oppsummering Vanskelig
Opplevelse	Fæl	Lett, omstendelig, komplisert
Rekkefølge	Vanskelig	Utfordrende
Utforskningsvilje	Grei første gang, Vanskelig å se muligheter	Lett og umulig å se muligheter Gir god forståelse
Sammenligning		Minner om installering

7.3.4 Kvalitative resultater grafisk kommandolinjeprototypen

Grafisk kommandolinje er en uvanlig metafor, og optimalisert for bruk av tastaturet. Under annet i Tabell 32 har vi derfor valgt å ta med interaksjonsmetoden de benyttet. Vi ser at begge gruppene foretrakk å bruke mus fremfor tastaturet. For programmerere skyldes dette delvis at søkefunksjonaliteten ved tastaturet var for dårlig, sett i forhold til forventningene. Denne gruppen setter også pris på at den er oversiktlig og gjør det enkelt å se sammenhenger. Den andre gruppen, ”ikke programmerere”, oppdager nye muligheter og leker generelt mer med mulighetene, og finner arbeidet lett.

Prototypen har skjulte muligheter ved bruk av tastaturet og ingen av brukerne oppdaget dette uten at de ble fortalt det. En av programmererne påpekte at <Tab> knappen burde ha samme funksjon som <Enter> når det gjelder å skifte mellom start- og mottakerobjektet.

Flertallet av ”ikke programmerere” finner denne prototypen tidkrevende, da de leter mye etter riktig objekt i nedtrekkmenyene. Programmererne har en lang liste over negative elementer (se Tabell 33) hovedsakelig handler det her også om at det er vanskelig å finne riktig element i listen. Det blir for mange valg, som igjen gjør det uoversiktlig og kjedelig i bruk. En hevder også at det er en ulempe at alt er skjult, dette gjør det vanskelig å få overblikk. Dette er oppsummert ved hjelp av kategorier i Tabell 5.

Tabell 5 Kategorisering av grafisk kommandolinje

Kategorier	Brukeropplevelse	
	Programmerere	**Andre**
Effektivitet	Treg	Veldig treg
Implementering	Sorterte list dårlig søkefunksjon Tab og Enter like	Forvirrende at nedtrekk åpnes av seg selv Kun et bilde til hver type
Interaksjon	Greit med tastatur, men mus er best	Foretrekker mus
Metafor	Alt i samme vindu For mange valg Oversiktlig For mye skjult Glad jeg ikke startet med denne Enkelt å forstå Dårlig	Hadde ikke klart noe om jeg startet med denne
Opplevelse	Kjedelig	Minner om programmering
Rekkefølge	Måtte tenke	
Utforskningsvilje	Lett med sammenhengen	Leker seg til nye løsninger
Sammenligning	Ligner WIMP	

7.3.5 Kvalitative resultater objektkoblingsprototypen

Objektkoblingsmetaforen er delvis hentet fra nett tjenester som har tatt av i det siste, og med det i tankene er det ikke overraskede at testpersonene har mange lovord om den. Tabell 34 viser at begge gruppene finner den morsom å bruke. Programmerer mener i tilegg at den har et bra konsept og føles veldig riktig.

Programmerer har også mange forbedringsforslag til denne modellen, de mener det er vanskelig å se hvilken pil som er aktiv og teksten på pilene er litt for kryptisk. Et veldig interessant innspill er muligheten til å bruke denne metaforen til å lage komposisjoner i, også heller vise lagrede komposisjoner i en liste tilsvarende den brukt i veiviser, grafisk kommandolinje og WIMP. Dette er litt motstridene med en annen som anser prototypen fin til å se sammenhenger i. Da han/hun finner det oversiklig mener vedkommende det burde være mulig å eksportere komposisjoner fra en annen prototyp til denne. Dette for lettere å kunne se hvor mange enheter som varsler mobilen din for eksempel.

Flere ”ikke programmerere” glemte å sette aksjon og hendelse på komposisjoner, dette er en alvorlig feil som tyder på at de ikke forstår hva de driver med. Denne gruppen hadde også vansker med å se forskjell på start- og mottakerobjektet. ”ikke programmerere” følte også det var for lite tilbakemelding når en komposisjon var lagret og var følte seg generelt usikre. Programmere syntes det var en uryddig prototyp og at pilene tok for mye plass. Dette gjør den støyete å jobbe i, men de klarte å løse oppgavene uten nevneverdige problemer. Alle negative elementer og problemer kan leses i Tabell 35 og prototypen er kategorisert i Tabell 6.

Tabell 6 Kategorisering av objektkoblingsprototypen

Kategorier	Objektkobling	
	Programmerere	**Andre**
Visuelt	Fin, veldig riktig	
Implementering	Rotete implementert tydeligere hva som er merket Skjult informasjon i aksjon/hendelse	Bedre merking av valgt kobling hele teksten vises ikke i nedtrekkmeny
Metafor	Liker ikke å rydde på skjermen Sånn skal det være oversiktlig Bra konsept Lett å se sammenheng mellom komposisjoner Rotete For liten plass på skjermen	Alt er synlig Rotete Lett å glemme aksjon/hendelse Usikker på når en komposisjon blir lagret
Opplevelse	Kaotisk, artig å flytte obj	Kaotisk, usikker, moro
Rekkefølge		Utfordrende
Utforskningsvilje	Lett å se sammenhenger	Leker seg til nye løsninger

7.3.6 Generelle resultater fra brukbarhetstestene

En del problemer knyttet til forståelsen av objektene og deres hendelser er utelatt fra hver enkelt modell. Det var mange som hadde problem med å bruke ”vær objektet” i de ulike modellene. Dette er ikke relevant i forhold til problemstillingen men er et viktig ledd i diskusjonen rundt kompleksiteten av systemet. Likens var det mange som ikke forstod at de skulle slette en tidligere laget komposisjon i siste oppgaven. Her laget de heller en ny komposisjon for å veie opp for komposisjonen de skulle slette. En testperson valgte å slette en komposisjon da vedkommende gjorde noe han/hun selv syntes det var feil, men på siste oppgaven valgte samme person å forhindre kaffelaging om morningen ved en motkomposisjon. Motkomposisjonene besto typisk av å blinke med lyset på soverommet, eller at kaffetrakteren skulle skru seg av når vekkerklokken ringte. Dette var et problem med oppgavetolkning og ikke utførelsen av oppgaven.

Der var også ulike begrunnelser for hvorfor et testsubjekt ønsket et hus tilsvarende dette eller ikke. Noen så på det som morsomt, andre strømsparende og noen ønsket bare en enklere hverdag. Dette er kategorisert i Tabell 7, hvor det er viktig å presisere at tidsavgrensning under konseptuelmodell har vært en gjenganger blant flere.

Tabell 7 Kategorisering av elementer som går på tvers av prototypene

Kategorier	Generelt
Muligheter	Generell Værmelding Uvante muligheter med fotorammen Innbruddsalarm og røykvarsler tolkes som sirener Mobiltelefonen bør kunne ringe og ikke bare SMS
Konseptuelmodell	Mangler tidsavgrensning av komposisjon Ønsker å lage komposisjon av flere enheter
Smarthjem	Alle vil bo i et

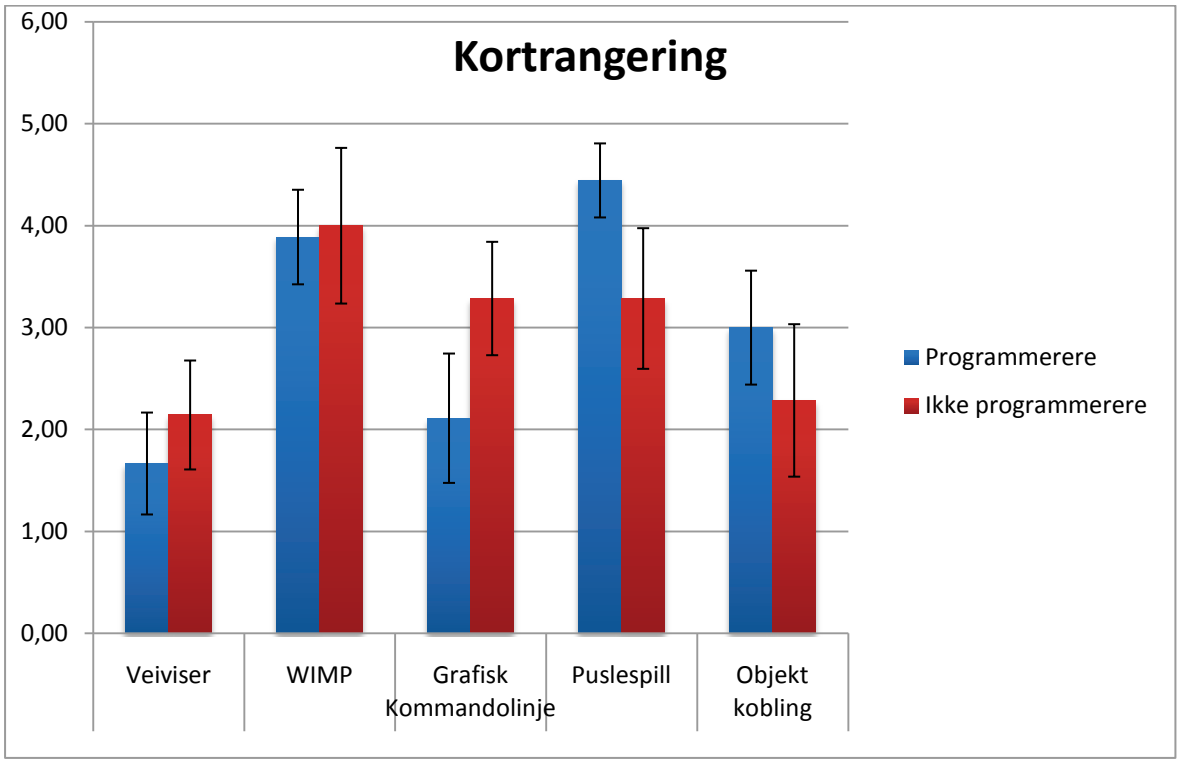
7.4 Kortrangering

Kortrangeringen resulterte i en rangert liste over hvilke metaforer de ulike brukerne foretrakk, se Tabell 24 i Vedlegg IEn høyere poengsum tilsier et bedre resultat i testen, 1 er det dårligste og 5 er det beste. Programmeringserfaring har de selv krysset av med hjelp på følgende skala:

Liste 14 Programmeringserfaring skala

1. Ingen
2. Liten (HTML)
3. Middels (PHP, script språk9)
4. Stor (deltatt i mindre utviklingsprosjekt på skole/jobb)
5. Veldig stor (selvstendig programmerer)

De ulike prototypene ble presentert i tilfeldig rekkefølge når de løste oppgavene, en friedman test på utleveringen av oppgaver gir P = 0,7727. Kortene (bilde av de ulike prototypene) ble presentert for brukeren i samme rekkefølge som de testet oppgavene. Brukerne ble gjort oppmerksom på dette for at de ikke skulle tro rekkefølgen på kortene var det samme som den forrige testperson hadde valgt, eller verre at det var den rekkefølgen vi ønsket oss. Mennesket klarer å huske syv pluss minus to ting, i dette tilfellet har vi bare fem ulike prototyper som brukerne da skulle klare å skille fra hverandre(45). Det viste seg at testen strakk over så lang tid at enkelte av testpersonene ikke husket de første modellene de prøvde. De fikk da muligheten til å prøve de litt igjen. Etter noen sekunder med prøving husket de godt hvordan en prototyp var.



Figur 32 Kortsortering programmerere og ikke programmerere

Figur 32 viser resultatet av kortrangeringen med en standardavvik for hver modell og testgruppe. En friedman test av resultatene for programmerere (de med programmerings erfaring 3 eller høyere fra Tabell 24 iVedlegg I) gir p = 0,001 som betyr at det er forskjell mellom de ulike prototypene i testpersonenes rangering. Programmerere vekter i gjennomsnitt puslespill til 4.44 og WIMP til 3.89 i gjennomsnitt. Standardavviket mellom dem overlapper delvis og det er derfor ikke entydig hvilken av modellene som er best.

7.4.1 Programmerere

For å finne forskjeller mellom de ulike brukergruppene og dermed svare på RQ2, benyttet vi statistiske metoder og presenterer resultatene av dem hver for seg.

Tabell 8 Parvis forskjell kortrangering for programmerere

	Veiviser	WIMP	Grafisk Kommando-linje	Puslespill	Objekt Kobling
Veiviser	0	-2,167	-0,444	-2,778	-1,278
WIMP	2,167	0	1,722	-0,611	0,889
Grafisk Kommandolinje	0,444	-1,722	0	-2,333	-0,833
Puslespill	2,778	0,611	2,333	0	1,500
Objekt Kobling	1,278	-0,889	0,833	-1,500	0

Kritisk Differanse: 1,4609

Nullhypotesen til friedman testen av programmerere ble avist og en nemenyi test viser hva som er forskjellig. Resultatet av nemenyi test for programmerere er gjengitt i Tabell 8, de grønne rutene er de med over 1,4609 i differanse. De grønne feltene betyr således at det er signifikant forskjell i resultatet av kortrangeringen mellom dem. Vi ser her at puslespillet er signifikant forskjellig med alle de andre med unntak av WIMP. Nemenyi testen plasserer de ulike modellene i ulike grupper på bakgrunn av rangeringen de fikk. For programmerere ble det tre overlappende grupper. Tabell 9 viser dette resultatet. Da vi brukte høyeste nummer som det best er gruppe C best, B er middels og A er dårligst.

Tabell 9 Gruppering av metaforer for programmerere

Metafor	Frekvens	Sum av rangering	Gjennomsnittlig rangering	Grupper		
Veiviser	9	15,000	1,667	A		
Grafisk Kommandolinje	9	19,000	2,111	A		
Objektkobling	9	26,500	2,944	A	B	
WIMP	9	34,500	3,833		B	C
Puslespill	9	40,000	4,444			C

7.4.2 Ikke programmerere

Resultatene for "ikke programmerere" (de med programmerings erfaring 2 eller lavere fra Tabell 24 iVedlegg I) er ikke signifikante p = 0,150 det er dog ikke langt unna og visse retningslinjer til videre forskning kan det gi. Selv om nullhypotesen ikke ble avist har vi valgt å gjøre en nemenyi test, for å gi små indikasjoner som kan sammenlignes med resultatet for programmerere. Selv om resultatene videre viser signifikante forskjeller videre, så er det ikke gyldig da friedman testen bekreftet nullhypotesen.

Tabell 10 Parvis forskjell kortrangering for ikke programmerere

	Veiviser	WIMP	Grafisk Kommandolinje	Puslespill	Objektkobling
Veiviser	0	-1,857	-1,143	-1,143	-0,143
WIMP	1,857	0	0,714	0,714	1,714
Grafisk Kommandolinje	1,143	-0,714	0	0,000	1,000
Puslespill	1,143	-0,714	0,000	0	1,000
Objekt kobling	0,143	-1,714	-1,000	-1,000	0

Kritisk differanse: 1,6565

Tabell 10 viser ikke overraskende at der er liten forskjell mellom de ulike modellene. Dette resultatet fra nemenyi testen antyder at der er noen forskjeller i preferanser mellom WIMP, veiviser og objektkobling. Der er ikke innbyrdes forskjell mellom veiviser og objektkobling og vi kan derfor antyde at WIMP skiller seg ut.

Tabell 11 Gruppering av metaforer for ikke programmerere

Metafor	Frekvens	Sum av rangeringer	Gjennomsnittlig rangering	Gruppe	
Veiviser	7	15,000	2,143	A	
Objektkobling	7	16,000	2,286	A	
Grafisk Kommandolinje	7	23,000	3,286	A	B
Puslespill	7	23,000	3,286	A	B
WIMP	7	28,000	4,000		B

For ”ikke programmerere” har vi gitt beste rangering det høyeste nummeret, og den laveste verdien til den metaforen som ble dårligst rangert. Dette er nominelle data, og må behandles der etter. Nemenyi testen gjør dette og har funnet to ulike grupper av metaforer i testen. Den ene gruppen er dårligere enn den andre. Tabell 11 viser dette resultatet, her er B den beste gruppen, og at det kun er WIMP som befinner seg kun i denne. Både puslespill og grafisk kommandolinje er i gruppe B, men de er også i den dårligere gruppen A. Da dette kommer fra en ikke signifikant friedman test, kan vi ikke trekke noen konklusjoner, men foreløpige resultater antyder at WIMP er mest foretrukket blant ”ikke programmerere”. Datasettet her består kun av 7 testpersoner, og variansen blir følgelig større enn ved et stort datasett dersom antagelsene våre stemmer. Grunnet dette er det rimelig å anta at WIMP vil være den mest egnede metaforen.

7.4.3 Alle testdeltagerne

For å besvare RQ1 må vi se alle metaforene under ett, en friedman test over kortrangeringen gir p=0,0004. Nullhypotesen er avist og der er forskjell mellom hvilke metaforer de ulike testpersonene foretrekker.

Tabell 12 Parvis forskjell kortrangering for alle metaforer

	Veiviser	WIMP	Grafisk Kommandolinje	Puslespill	Objektkobling
Veiviser	0	-2,031	-0,750	-2,063	-0,781
WIMP	2,031	0	1,281	-0,031	1,250
Grafisk Kommandolinje	0,750	-1,281	0	-1,313	-0,031
Puslespill	2,063	0,031	1,313	0	1,281
Objektkobling	0,781	-1,250	0,031	-1,281	0

Kritisk differanse: 1,0957

Tabell 12 viser signifikante forskjeller mellom mange av prototypene, og at forskjellene er forholdsvis store. Forskjellen mellom puslespill og objektkobling er meget stor, dette betyr at brukerne er forholdsvis samstemte i kortrangeringen.

Tabell 13 Gruppering av metaforer for alle testdeltagere

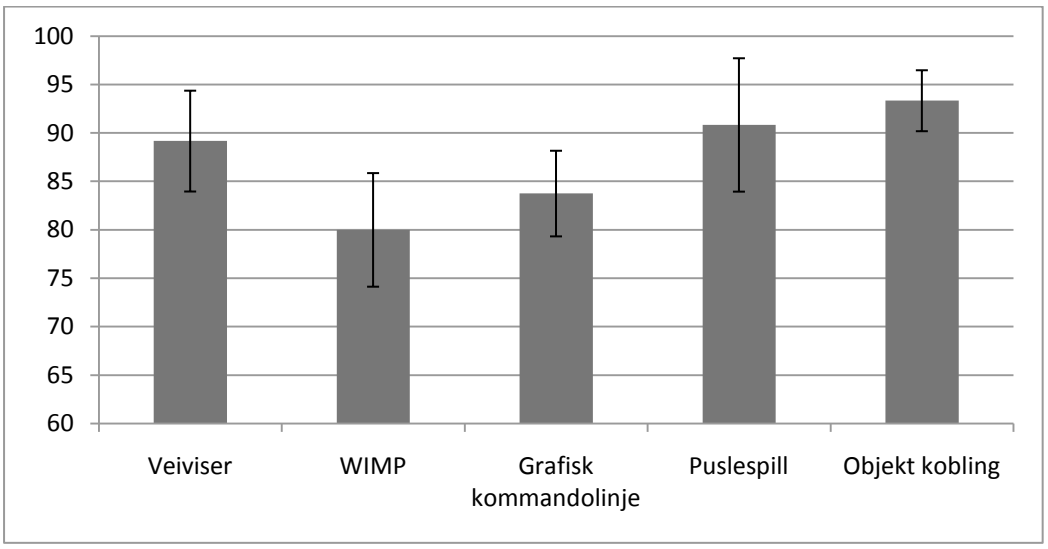
Metafor	Frekvens	Sum av rangeringer	Gjennomsnittlig rangering	Gruppe	
Veiviser	16	30,000	1,875	A	
Grafisk Kommandolinje	16	42,000	2,625	A	
Objektkobling	16	42,500	2,656	A	
WIMP	16	62,500	3,906		B
Puslespill	16	63,000	3,938		B

Grupperer vi resultatene fra Tabell 12 gir nemenyi testen to ulike grupper av metaforer. Dette er vist i Tabell 13. De beste prototypene fra kortrangeringen er de i gruppe B, puslespill og WIMP. Disse er signifikant bedre enn de øvrige modellene og er med på å besvare RQ1.

7.5 SUS

Det ble gjennomført en SUS test for den prototypen de ulike testsubjektene likte best, derfor er det varierende hvor mange som har svart for de ulike modellene. Antall og data for grafen i Figur 33 ligger i Vedlegg III

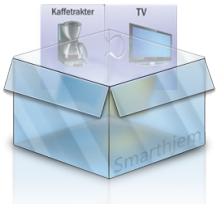
. Da antallet var lite og det er stort standardavvik sier ikke resultatet i Figur 33 noe signifikant. Gjennomsnittlig poengsum ble 87,42 med en standardavvik på 10,79. Alle modellene ligger innenfor en standardavvik av gjennomsnittet med unntak av objekt kobling som får litt mer poeng. Utav dette kan vi trekke konklusjonen at brukergrensesnittene er mer eller mindre like brukervennlige. Deltagerne foretrakk ulike modeller som sin favoritt og det er ikke mulig å trekke noen konklusjoner basert på de ulike brukergruppene. Det var ingen ”ikke programmerere” som besvarte SUS med tanke på grafisk kommandolinjemetaforen, og datagrunnlaget fra modell til modell ble for lite. Ser vi alle modellene under ett er der heller ingen forskjell i de ulike brukergruppene, t-test gir P = 0,70.



Figur 33 Graf over resultatene med SUS

7.6 Tolkning

Etter å ha brukt systemet i brukbarhetstesten gjennomførte alle testdeltagere et sett med tolkeoppgaver. I hver oppgave var det lagret en komposisjon og en komposisjon som holdt på å bli laget. Alle deltagerne klarte alle oppgavene uten problem. Testsubjektene leste fra arkene og forklarte hva som ville skje i huset med full forståelse. Vi kan konkludere med at det ikke er knyttet problemer til forståelsen av hva som skjer i huset etter å ha brukt et dataprogram først.



8. Analyse

For å støtte resultatene fra de enkelte forskningsmetodene ser vi her etter de samme og nye resultatene på tvers av de enkelte forskningsmetodene.

8.1 Kvalitativ analyse

De kvalitative dataene fra underkapittel 7.2 Brukbarhetstesting og intervju gir mye forskjellig informasjon. I denne oppgaven er det hovedfokus på hva som gjør en prototyp egnet eller uegnet. Vi har derfor plukket ut noen påstander/utsagn og sammenlignet dem mellom de ulike gruppene. Her er det hele tiden fokus på prototypene vi testet metaforene i, da enkelte av dem kan ha implementeringer som ikke fremmet metaforen godt nok. I avsnitt 10.2 ser vi på forholdet mellom prototypene og metaforene de representerer. I dette kapittelet knytter vi sammen kvantitative og kvalitative data for å se sammenhengen mellom dem. Dette vil gjøre oss i stand til å svare på de RQ1-4 som antydet underveis i resultat kapittelet.

8.1.1 Sammenligning av de ulike metaforene

I avsnitt 7.2 Brukbarhetstesting og intervju blir hver prototyp presentert hver for seg, her vil vi prøve å poengtere likheter og uliker av prototypene basert på de kvalitative dataene. For å gjøre dette oversiktlig har vi generalisert hovedpoenger innenfor de ulike kategoriene. På denne måten vil enkelte aspekter falle bort, men vi har tilstrebet å beholde mest mulig informasjon. Vi har plukket ut de kategoriene som berører metaforen, og opplevelsen til brukeren ved bruk av systemet. Interaksjon vil ikke være med i sammenligningen da alle foretrakk å bruke mus. Forklaring på de ulike kategoriene og de utelatte kategoriene kan ses i Liste 13.

Visuelt sett er, veiviser, puslespill og objektkobling penere prototyper. De andre prototypene har ikke testpersonen utalt noe om det visuelle og det er vanskelig å ta stilling til. Dersom det hadde vært store forskjeller ville det kommet frem av resultatene, noe det ikke gjør og vi kan konkludere med at de er ganske like.

I effektivitet varierer prototypene mer, i Tabell 14 kommer dette tydelig frem. Brukergruppene er like i uttalelsene sine hvor WIMP og puslespill fremstår som raske prototyper. Veiviser og grafisk kommandolinje er på andre siden av skalaen og oppleves som trege.

Tabell 14 Forskjell i effektivitet

Metafor		Effektivitet
	Programmerere	**Andre**
Veiviser	Treg	Rydding er tregt
WIMP	Hurtig	Hurtig
Puslespill	Hurtig	Hurtig
Objektkobling		
Grafisk Kommandolinje	Treg	Veldig treg

Tabell 15 Forskjell i opplevelse

Metafor		Opplevelse
	Programmerere	**Andre**
Veiviser	Fæl	Blandet
WIMP	Smart	Trygg, likte den ikke
Puslespill	Morsomt	Innlysende, gøy
Objektkobling	Kaotisk, artig	Kaotisk, usikker, moro
Grafisk Kommandolinje	Kjedelig	

Ulike brukergrupper opplever ting forskjellig, en brukergruppe kan være vant til et system, mens en annen gruppe til noe helt annet. Et dataprogram kan være effektivt, oversiktlig og lett å bruke samtidig som det oppleves vondt å bruke det. Det visuelle kan ødelegge for enkeltes opplevelse, mens det er hvor lett det er å gjøre feil som ødelegger for andre. Her skal vi ikke spekkulere i hvorfor enkelte opplever en modell som god og dårlig, men påpeke forskjellene mellom gruppene og modellene. Tabell 15 oppsummerer kategorien opplevelse for alle modellene, og puslespillet liker begge brukergruppene. De er også enige om at objektkobling er kaotisk men morsomt. De andre metaforene er det mindre variasjoner, men veiviseren var vond å bruke for programmere, mens det for andre opplevelses forskjellig.

Tabell 16 Forskjell i rekkefølge av komposisjoner

Metafor		Rekkefølge
	Programmerere	**Andre**
Veiviser	Vanskelig	Utfordrende
WIMP	Blandet	Forståelig
Puslespill	Blandet	Utfordrende
Objektkobling		Utfordrende
Grafisk Kommandolinje	Tankefullt	

De ulike metaforene blir benyttet for å gjøre det lettere å lage komposisjoner i smarte hjem. Det er ikke bare metaforen, men også implementeringen som påvirker hvor lett det er å gjøre en

enkelt komposisjon. Vi ser her på hvor brukerne hadde lett og vanskelig for å finne riktig rekkefølge for objekter i komposisjonen. Tabell 16 viser dette, her har vi brukt vanskelig for programmerere på veiviseren, dette fordi testpersonene til stadighet måtte tenke en ekstra gang på rekkefølgen. For utfordrende og blandet har enkelte testpersoner vært usikre på rekkefølgen i starten. Utfordrende er litt verre enn blandet, da det for blandet også er brukere som finner rekkefølgen av objekter i en komposisjon naturlig. Brukergruppene er ganske samstemte, mens det er stor forskjell mellom prototypene. WIMP skiller seg ut som den prototypen med mest innlysende rekkefølge for objekter i en komposisjon.

I et smarthjem er det mange muligheter, og et system for det vill gi liten glede og nytte for brukeren om han/hun ikke utforsker mulighetene og prøver ulike komposisjoner. Dette kan både være morsomt og nyttig, men for å finne de komposisjonene et individ finner interessant må det prøves ut ulike ting i det aktuelle hjemmet. For å avgjøre RQ1 er det derfor viktig å se hvilken prototype som gir de beste mulighetene for utforskning. Fra Tabell 17 går det frem at det er forskjell mellom brukergruppene, ”ikke programmerere” søker stadig nye løsninger på en måte de trives med. Det er kun i veiviserprototypen at der er ”ikke programmerere” som finner det vanskelig å se mulighetene. Puslespillet gjør at programmerere kan se nye muligheter med systemet, noe som er veldig viktig for å kunne utnytte systemet fult ut i hjemmet.

Tabell 17 Forskjell i utforskningsvilje

Metafor		**Utforskingsvilje**
	Programmerere	**Andre**
Veiviser	Grei første gang, Vanskelig å se muligheter	Lett og umulig å se muligheter Gir god forståelse
WIMP		Leker seg til nye løsninger
Puslespill	Ser nye muligheter	Leker seg til nye løsninger
Objektkobling	Lett å se sammenhenger	Leker seg til nye løsninger
Grafisk Kommandolinje	Lett med sammenhengen	Leker seg til nye løsninger

Den siste kategorien vi vurderer i sammenligningen er kategorien metafor. Dette er den overordnete kategorien for flere av de andre kategoriene og resultatene her er derfor påvirket av mange andre. De andre resultatene er med på å forklare og underbygge disse resultatene, og vi kan derfor ikke se kun på denne kategorien isolert sett. Programmerere finner WIMP-prototypen og dens metafor oversiktlig og grei. De andre modellene har en tendens til å bli rotete, tungvinte eller dårlige. Her er det forskjell mellom gruppene, de andre finner også WIMP oversiktlig, men anser puslespillet som en enkelt.

Tabell 18 Forskjeller i kategorien metafor

Metafor		Metafor
	Programmerere	**Andre**
Veiviser	Tungvint	Delte meninger
WIMP	Oversiktlig	Oversiktlig
Puslespill	Rotete Rett på sak	Enkel
Objektkobling	Rotete, lett å se sammenheng	Rotete, lett å glemme aksjon/hendelse
Grafisk Kommandolinje	Enkel og Dårlig	Vanskelig

8.1.2 Kvalitativ oppsummering

Til nå har vi sett på hver kategori isolert sett, det er viktig å se dem i et større bilde for å finne en total vurdering av en bestemt prototype. Kategoriene her er de samme som brukt ovenfor og kommer fra Liste 13 Liste over kvalitative kategorier.

Tabell 19 Oppsummering av kvalitative forskjeller

Metafor	Gruppe	Visuelt	Effekti-vitet	Metafor	Opp-levelse	Rekke-følge	Utforskning-svilje	Sum
Veiviser	Prog.	X						1
	andre	X						1
WIMP	Prog.		X	X	X			3
	andre		X	X	X	X	X	**5**
Puslespill	Prog.	X	X		X	X	X	**5**
	andre	X	X	X	X	X		**5**
Objekt-kobling	Prog.							0
	andre	X					X	2
Grafisk kom. Linje	Prog.							0
	andre						X	1

For å gjøre det mulig å sammenligne de enkelte prototypene har vi samlet analysen mellom de ulike prototypene fra avsnitt 8.1.1 i Tabell 19. Her er det merket med et kryss for hver brukergruppe og prototype hvilke kategori de er best i. Flere prototyper kan være på beste plassering i en bestemt kategori, ser vi på kategorien visuelt, er både veiviser, puslespill og objektkabling best egnet. Den sistnevnte er dog kun best egnet for ”ikke programmerere”. I den siste kolonnen har vi oppsummert hvor mange kategorier hver gruppe syntes en bestemt metafor er best i. Her er de ulike kategoriene vektet likt, og det er ikke sikkert det er riktig, men ved lik vekting er WIMP og puslespill best egnet for ikke programmerere, mens det kun er puslespillet som er egnet for programmerere.

Der er også en sammenheng mellom effektivitet og opplevelse, dette kan antyde at hvis en metafor er effektiv i bruk vil den også oppleves bra å bruke. For ”ikke programmerere” ser vi

også en sammenheng mellom metafor og opplevelse, rekkefølge og effektivitet. Dette er ikke unaturlig da metaforen på flere punkter er en samling av disse underkategoriene. Det er litt rart at der ikke er større sammenheng mellom utforskingsvilje og opplevelsen til en metafor.

8.2 Hvilken metafor er resultatmessig best?

WIMP og Puslespill var de best egnede metaforene fra kortrangeringen, disse var signifikant bedre enn de øvrige. I kvalitativ analyse var det disse metaforene som også fikk flest topp plasseringer av de utvalgte kategoriene. Puslespillet var blant de beste i hele 10 kategorier (5 for programmerer og 5 for andre) mens WIMP i 8 kategorier (henholdsvis 5 og 3). Puslespillet kommer litt bedre ut, men ikke signifikant bedre ut enn WIMP i kortrangeringen, dette kan være en tilfeldighet, men sett i lys av at puslespillet også kommer best ut av den kvalitative analysen er det ikke nødvendigvis en tilfeldighet. WIMP ble beskrevet som klassisk, noe puslespillet ikke er. Videre har programmerere utalt at puslespillet kan bli rotete. Klassiske ting kan være kjedelig, men samtidig trygt, mens forandring fryder. Rotete er et problem som trekker ned for puslespill. Det er derfor ikke datagrunnlag nok til å skille disse fra hverandre. Resultatmessig er svaret på RQ1 at der er to metaforer som er best egnet til sluttbrukerkomponeringer av tjenester i smarte hjem. Det endelige svaret på RQ1 vil ikke bli klart før etter diskusjon om validiteten til resultatene.

8.3 Hvilke argumenter mener brukerne er viktig for en god metafor?

Fra kvantitative og kvalitative resultater fant vi enkelte forkskjeller mellom brukergruppene, her vil vi fokusere på argumentene i de ulike kategoriene i forhold til resultatet fra avsnitt 8.2 Hvilken metafor er resultatmessig best? Denne analysen har som hensikt å besvare RQ3 ved å fokusere på hva som er bedre med puslespill og WIMP i forhold til de øvrige prototypene.

Effektivitet, det at bruken av prototypen gikk hurtig er felles for puslespill og WIMP, det er kanskje her de skiller seg mest fra de øvrige prototypene. Begge prototypene oppleves smarte og morsomme å bruke, dette i kontrast til de øvrige prototypene som oppleves tunge, vanskelige og kjedelige å bruke.

Metafor, en programmerer ser nye løsninger i puslespillet, dette skiller seg ut og er en viktig faktor for hvorfor denne modellen er likt av programmerere. For ”ikke programmerere” skiller verken WIMP eller puslespill seg nevneverdig fra de øvrige modellene i denne kategorien.

Rekkefølge/Forståelse, et system har liten nytte dersom ingen klarer å bruke det. ”Ikke programmerere” finner det lettere med rekkefølgen av objekter i en komposisjon i WIMP og puslespill enn i øvrige prototyper.

Visuelt, en prototyp kan være visuelt pen, men dette ser ikke ut til å ha direkte sammenheng med de øvrige resultatene. Dette kan skyldes at få personer har nevnt noe om dette. Det er naturlig å anta at testpersonene vill sagt mer om de visuelle om de fant det viktig for valg av prototyper under kortrangeringen.

Opplevelse, der er også en sammenheng mellom hvordan brukerne opplever et system og hvordan det blir rangert i forhold til andre faktorer. På bakgrunn av dette kan vi konkludere med

at et godt brukergrensesnitt oppleves morsomt eller artig. Et dårlig brukergrensesnitt kjennetegnes ved kjedelig eller tungvint opplevelse.

Svaret på RQ3 kan på bakgrunn av dette oppsummeres i en tabell med det gode og dårlige faktorer for brukergrensesnittmetaforer. Dette svaret forutsett at metaforene er representert på en tilfredsstillende måte i prototypene. Tabell 20 viser hva brukere mener er forskjellen på en god og dårlig prototyp innenfor vår begrensede konseptuelle modell og til sluttbrukerkomponering av tjenester til smarte hjem.

Tabell 20 Forskjeller mellom gode og dårlige metaforer

	God	**Dårlig**
Effektivitet	Hurtig	Treg
Opplevelse	Morsom	Kjedelig/Tungvint
Rekkefølge/Forståelse	Forståelig	Vanskelig
Metafor	Oversiktlig, rett frem	Rotete, uforståelig

8.4 Forskjeller mellom grupper

Kortrangeringen viser at der er forskjell mellom de ulike brukergruppene. Programmerere foretrekker puslespillet, de rangerer dette best av alle, med WIMP som en god nummer to. For den andre brukergruppen er det mer usikkert, enten er der store variasjoner mellom individene, eller så hadde vi for få testpersoner. Nullhypotesen ble opprettholdt, så modellene ble likt rangert. Der var dog sterke tendenser til at WIMP var den mest foretrukne modellen. Derfor sammenligner vi prototypene ved hjelp av de kvalitative dataene for endelig å kunne bestemme hvilken prototyp ”ikke programmere” foretrekker. I avsnitt 8.1.2 Kvalitativ oppsummering ser vi at både WIMP og puslespill er blant de beste i 5 kategorier. For puslespill er en av disse kategoriene det visuelle, denne kategorien er ikke viktig for brukerens argumentasjon for en god og dårlig metafor (se avsnitt 8.3). Ser vi bort i fra denne er de fortsatt veldig like, men både kvalitative og kvantitative data peker i retning av at WIMP er den best egnede prototypen for ”ikke programmerere”. Svaret på RQ2 blir da; ja der er en forskjell mellom de ulike brukergruppene i preferanser rundt de ulike metaforene. Hadde de hatt de samme preferansene ville de endt opp med å foretrekke den samme modellen. Her er det verdt å merke at brukergruppene kunne hatt forskjellige preferanser og endt opp med samme modell ved en tilfeldighet.

8.5 Hva er den best egnede metaforen i forhold til RQ1-RQ3?

Forskingsspørsmål RQ4 er et resultat av RQ1-3 og her vil vi analysere svarene av dem for å besvare RQ4. Denne diskusjonen er med forbehold om gyldigheten av resultatene og forskningen vi har gjort.

Det var to prototyper som skilte seg ut som de beste etter resultatene og ble svaret RQ1: WIMP og puslespill. At disse ble vinnere er ingen overraskelse da begge brukergruppene finner de bedre enn de andre og foretrekker hver sin modell som den beste. Det var liten forskjell mellom den beste og nest beste modellen i hver brukergruppe. WIMP var den best egnede metaforen for ”ikke programmerere” og da dette er den største potensielle brukergruppen burde vi tillegge

den mer vekt. På den andre siden er programmerere teknisk kompetente og de vil være i stand til å ta i bruk et smarthjem før det er ferdig utviklet. Begge gruppene ønsket å flytte inn i et smarthjem, men verdien av betatestere (programmerere) er stor og programmerere sine preferanser bør ivaretas. Her er det en interesse konflikt og da forskjellen mellom den beste og nest beste prototypen hos begge brukergrupper var små har vi valgt å vekte dem likt i det videre arbeidet med å finne den beste metaforen.

Etter RQ1 og RQ2 står vi igjen med to prototyper som de best egnede, vi vil her bruke RQ3 for å skille dem fra hverandre. Dette gjør vi ved å kombinere tabellen over viktige argumenter for en god og dårlig metafor med de kvalitative resultatene fra puslespill og WIMP. Dette er en kombinasjon av tidligere viste tabeller og dataene er således vist før, den nye tabellen er Tabell 21.

Tabell 21 skille mellom WIMP og Puslespill

	WIMP	**Puslespill**
Effektivitet	Hurtig	Hurtig
Opplevelse	Smart/Trygg	Morsom/innlysende
Rekkefølge/ Forståelse	Blandet, forståelig	Blandet, utfordrende ser nye muligheter
Metafor	Oversiktlig	Rotete og enkel

Tabell 21 viser at begge prototypene er raske i bruk, det er dog litt vanskeligere å skille opplevelsen. I WIMP føler brukerne seg trygge, mens puslespillet oppleves innlysende, da dette er teknologi brukerne skal ha rundt seg i sitt eget hjem er trygghet viktigere enn innlysende. Dette strider med testresultatene som skiller en god og dårlig prototyp med kjedelig til morsom på opplevelse. Vi har likevel valgt å vekte trygghet her av overnevnte grunn.

Valg av riktig rekkefølge på en komposisjon er elementært for å oppnå ønsket effekt. Her ser vi at det er blandet på begge kategoriene, dette skyldes at flere av brukerne prøvde seg frem i starten for så å finne ut av det. Dette er i praksis likt mellom de ulike prototypene. Vi ser videre på den siste kategorien metafor. WIMP er oversiktlig mens puslespill er rotete og enkel. Det å ha oversikt over hva som skjer i huset er viktig, det er også viktig at puslespillet er enkelt. Vi kan derfor likestille disse utsagnene, men puslespillet har også egenskapen rotete. Dette vil bli verre med flere komposisjoner og er en kjent ulempe ved metaforer hvor det programmeres og lagres resultat i samme område. Dette gjør WIMP til en bedre egnet prototyp på metaforpunktet.

WIMP var litt bedre enn puslespill på to av punktene fra RQ3 og dette skiller den totalt sett fra puslespillet. Svaret på RQ4 blir derfor WIMP med puslespill tett etterfulgt.

8.6 Hvorfor er en metafor god eller dårlig?

Denne oppgaven fokuserer på hvilke metaforer som er best egnet til sluttbrukerkomponering av tjenester i hjemmet. Det er også viktig å se på hvilke faktorer som kjennetegner de mindre egnede metaforene. Fra avsnitt 8.5 avdekket vi kategoriene brukerne legger til grunn for å

avgjøre om en metafor er god eller dårlig, her vil vi analysere de ulike metaforene for å se hva som gjør de gode eller dårlige. Vi vil her fokusere på de elementene i de ulike brukergrensesnittene som antagelig ligger til grunn for tilbakemeldingene innenfor kategoriene. Vi presenterer metaforene i den rekkefølgen de er rangert av alle brukerne fra kortrangering i avsnitt 7.4.3. For de øverste plassene er den i samsvar med vårt svar på RQ4 fra avsnitt 8.5. Vi presiserer at det ikke er funnet signifikante forskjeller mellom alle prototypene og at denne listen må ses på som foreløpige resultater for plass 3-5.

Liste 15 Hva gir utslag i en kategori fra en metafor

1. **WIMP**
   Denne metaforen kjennetegnes ved å være hurtig, dette fordi brukerne velger objekter og hendelse i et og samme vindu. De kan gjøre dette uten å åpne noen menyer, listene viser mange objekter eller hendelser samtidig, hvilket gjør det hurtig å bla gjennom. Denne åpenheten, at alt er strukturert og vises samtidig gjør at metaforen oppleves oversiktlig i bruk. Trygghetsfølelsen er trolig en følge av at dette er en kjent metafor for brukerne.
2. **Puslespill**
   Dette er også en hurtig prototyp, det er ikke like enkelt å forklare hvorfor, da brukerne må gjøre tyngre opprasjoner for å velge et objekt. Et objekt, eller en brikke, må her flyttes ved hjelp av musen. Dette tar lenger tid enn å trykke på tilsvarende objekt i WIMP metaforen. Valg av hendelse/aksjon er delvis skjult i en nedtrekk meny, dette er med på å gjøre mulighetene mindre oversiktlige. Denne metaforen blir også uryddig ved mange komposisjoner på skjermen.
3. **Objektkobling**
   På linje med puslespillet blir også denne modellen uryddig med mange komposisjoner på skjermen. Den er til gjengjeld veldig oversiktlig ved færre komposisjoner, da strekene gjør det lett å se hva som skjer i huset. Brukerne syntes den er artig å bruke fordi det er en ny måte for mange å jobbe på. Dette gjelder også for puslespillet, men det er usikkert hvor lenge de vil ha denne opplevelsen. Pilene på skjermen går over hverandre og dette blir et visuelt rot, hvilket kan forklare et uryddig skjermbilde selv med forholdsvis få komposisjoner.
4. **Grafisk kommandolinje**
   Denne metaforen ble nest dårligst fordi det er tungvint og tar for mye tid å lete etter ønskede objekter med tilhørende hendelse og objekt. Metaforen skjuler for mye funksjonalitet og det blir usikkert hvilke muligheter der er.
5. **Veiviser**
   Veivisermetaforen var lett gjenkjennbar, men her opplevdes ikke en trygghetsfølelse som var tilfellet for WIMP. Dette kommer trolig av at det var vanskelig å forstå forskjellen på start og mottaker i en komposisjon. Da metaforen skiller disse på ulike steg/bilder er det vanskelig å se en sammenheng og få oversikt over komposisjonen brukeren prøver å lage. Det er også mer trykking for å navigere i en veivisermetafor, noe brukerne finner unødvendig og tungvint. Summen av disse ulempene med metaforen fører til at den er treg i bruk for komponering av tjenester i hjemmet.

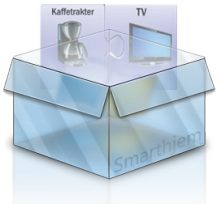
8.6.1 Oppsummering

De ulike konseptene ved en prototyp er oppsummert i Tabell 22. Denne tabellen viser at elementer som skjuler informasjon for brukeren er med å gjøre en prototyp dårlig. Der er en sammenheng mellom hvor mye informasjon en prototyp viser og hvilken plass den havner på i brukerens rangering. Dette stemmer for alle metaforene med unntak av objektkobling, som ikke skjuler informasjon mer enn puslespillet, men kommer dårligere ut i rangeringen. Dette skyldes at pilene og særlig teksten på pilene blir liggende oppå hverandre og informasjonen blir uleselig. Uleselig informasjon kan ses på som skjult informasjon og derfor blir overnevnte oppsummering riktig.

Få valg og lite trykking henger delvis sammen, her impliserer få valg at brukeren har få valg i hva som kan gjøres i metaforen og ikke få muligheter i hjemmet. Veiviser metaforen har for eksempel mange valg, brukeren må hele tiden velge hvilket steg han/hun ønsker å utføre en endring på. Samtidig som det blir mye trykking mellom de ulike stegene.

Tabell 22 Gode og dårlige konsepter ved en metafor

Bra	Dårlig
Lite trykking	Skjult informasjon
Alt synlig	Ikke se en hel komposisjon samtidig
Få valg	Ting oppå hverandre
	Store nedtrekkmenyer



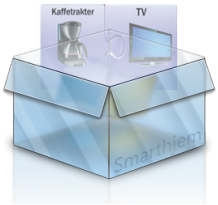
9. Anbefalinger og forslag til forbedring

Dette kapittelet beskriver forfatterens anbefalinger og forslag til forbedringer. Dette er ideer og tanker som har kommet igjennom tiden det har tatt å utvikle, teste og analysere de ulike prototypene. Forslagene som blir presentert her vil derav bli mindre vektlagt i den videre diskusjon og konklusjon, men mindre endringer og forslag til videre forskning vil bli vurdert.

I løpet av testene var det mange brukere med problemer knyttet til veiviseren og navigering i den. Veiviseren tvinger en bruker for å lage komposisjonen sekvensielt, da det ikke er mulig å gå videre til neste steg før et steg er ferdig. Dette ble gjort for at brukeren ikke skulle glemme å fylle ut et steg fullstendig. Skulle vi laget den på nytt nå, ville vi latt statuslinjen være navigerbar, da mange brukere forsøkte dette. For å kontrollere at alle felter var utfylt burde da oppsummeringsvinduet gitt tydelig tilbakemelding om hva som var galt. Denne tilbakemeldingen burde vært navigerbar så brukeren enkelt kunne gå tilbake til det aktuelle steget og fullføre det.

WIMP burde hatt et mer visuelt skille, gruppering av start- og mottakerobjektene i en komposisjon. Dette er en ide fra en av testpersonene, og etter å ha studert mulighetene er dette et viktig grep for å gjøre prototypen enda bedre.

I oppgaven har vi lagt strenge begrensninger på den konseptuelle modellen og kun den enkleste form av en komposisjon har blitt vurdert. Dette for å bekrefte eller avkrefte om sluttbrukerkomponering av tjenester er mulig i smarte hjem. For å få gyldige resultater er det viktig at oppgavene gir mening til testpersonene. Mange av oppgavene er generelle, mens andre kanskje bare bør være gyldige innenfor et tidsaspekt. Mange brukere savnet å knytte tid til hendelser, dette er vi enig i. Uten at vi beveger oss over i konfigurasjon av tjenester er det mulig å knytte en klokke til hver komposisjon. Det er mange måter å gjøre dette på, og ulike alternativ må vurderes.



10. Diskusjon

Dette kapittelet vil diskutere viktige aspekter rundt arbeidet som er utført og resultatet av det. Et resultat er ikke bedre enn arbeidet bak det, og det er derfor viktig å være kritisk til utførelsen så vel som resultatet. Enkelte konsepter vil også bli knyttet mot den ulike teorien som er presentert tidligere.

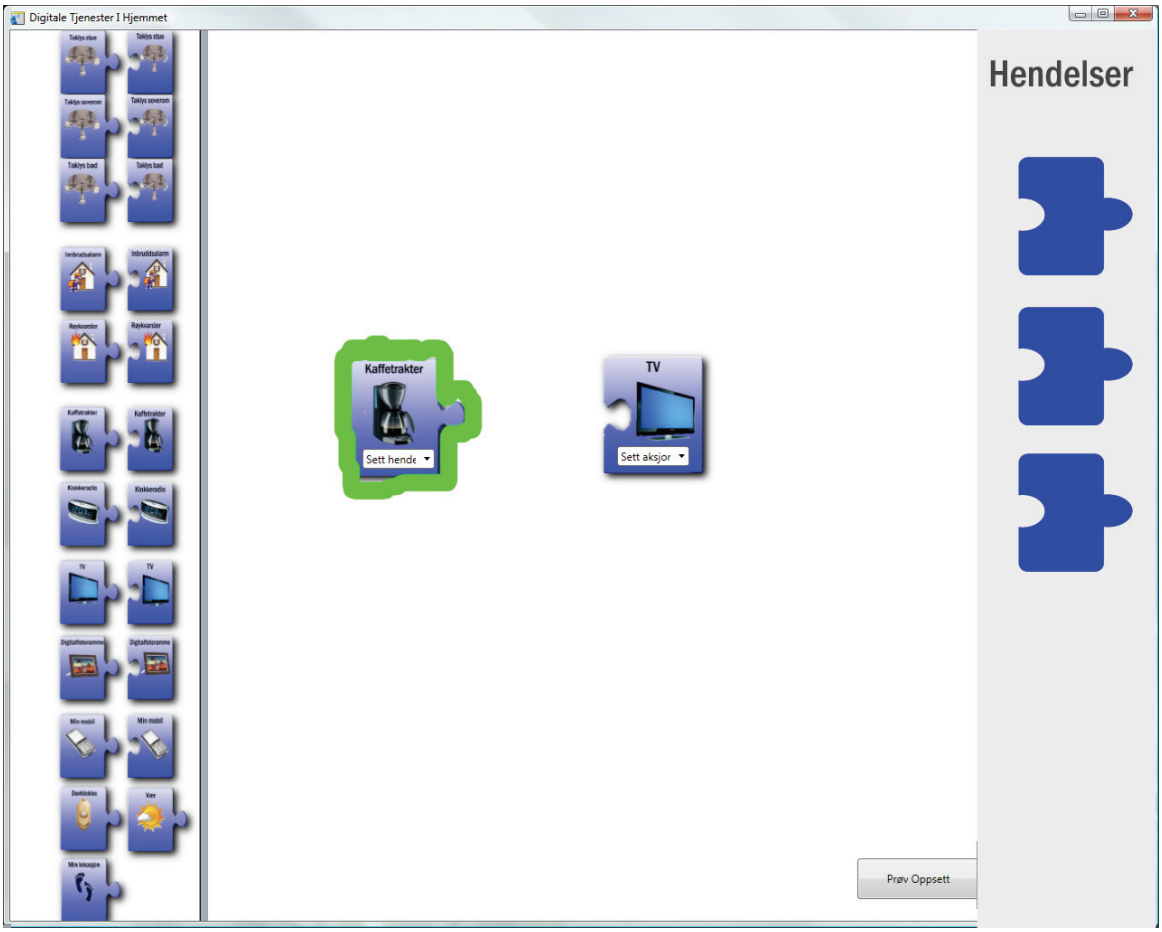
10.1 Forbedringer av prototypene og konseptuel modell

Under brukbarhetstestingen ble det foreslått flere mulige forbedringer, dette er gjengitt i resultatkapittelet. Vi har også sett på enkelte forbedringer til prototypene, disse er presentert i kapittel 9 og vil ikke bli vektlagt like mye i den videre diskusjonen.

10.1.1 Puslespill

En av programmererne trodde der skulle være egne brikker for hendelser og aksjoner i puslespillet. Vedkommende lette en stund etter dem, før han forsto hvordan hendelser og aksjoner kunne settes. En utfordring med dette er at et objekt har ulike hendelser og aksjoner, i tilegg til at de ulike objektene har forskjellige hendelser og aksjoner i forhold til hverandre. Det er mulig å løse ved at en brikke kan merkes, og ved merking kommer det opp en liste over mulige hendelser eller aksjoner som kan kobles sammen. Dette er illustrert i Figur 34 og vil trolig gjøre en puslespillkomposisjon lettere å lese. Bruk av egne brikker til hendelser gjør kanskje aksjoner og hendelser mye enklere å få oversikt over. Samtidig som det tar forholdsvis mye plass på skjermen. Skjermplass er et sentralt problem med sluttbrukerkomponeringsverktøy hvor lagrede komposisjoner vises på samme plass som de lages. Selv om dette ville hatt klare fordeler vil det ikke være å anbefale da det tar opp for mye skjermplass. For å gjøre dette mulig måtte det være mulig å skifte mellom ulike faner med komponeringsflater. Kanskje gruppert de ulike rommene på ulike fane, men det er et sorteringsproblem og er utenfor rammen av denne oppgaven.

Det var også forslag om å kunne flytte hele komposisjoner for og lettere kunne rydde på skjermen. Dette kan løses ved at en komposisjon har ulik oppførsel alt etter hvor musen trykkes ned. I et endelig system er dette noe som bør være med, da det vil gjøre rydding på skjermen betraktelig lettere. Ved å trykke innenfor et avgrenset område i midten av en komposisjon, kan den for eksempel flyttes. Dette kan gjøres synlig ved å bytte musepeker når musen er innenfor det gitte området. En annen løsning vil være å plassere et flyttekryss oppe i venstre hjørnet når musen er over en komposisjon. Føres musen til dette krysset kan brukeren trykke ned museknappen og flytte komposisjonen. Denne løsningen vil ligne på Microsoft Word sin måte å flytte tabeller på.



Figur 34 Forbedringsmulighet av puslespillprototypen

10.1.2 Objektkobling

Objektkoblingsprototypen har de samme plassproblemene som puslespillet da både komponering og lagring av komposisjoner utføres i samme område. En av brukerne foreslo derfor å dele prototypen i to. En del for komponering av tjenester ved hjelp av den samme metaforen som i dag, mens de lagrede komposisjonene skulle lagres i en liste tilsvarende den for WIMP, puslespill og grafisk kommandolinje. De tre sistnevnte prototypene er alle todelte, de har en del for komponering og en for visning av resultatene. Problemet med å gjøre dette i objektkobling er at det ikke blir plass på skjermen til å vise alt samtidig. Testpersoner utalte at det var lett å se sammenhenger mellom komposisjoner og få oversikt over hvor mye som skjedde med en enhet i objektkoblingsprototypen. Dette er aspekter som forsvinner ved å bruke en liste til lagring av komposisjoner. Det vil videre stride med tankene våre rundt valget av den metaforen fra avsnitt 2.4.3. videre vil det vil stride mot oppsummeringen av gode og dårlige prototyper fra avsnitt 8.6.1 og skjule deler av informasjonen. En oppdeling kan allikevel være en god mulighet, ved å dele prototypen inn i ulike verdener tilsvarende det som er gjort i taktil programmering skal det fortsatt være enkelt å bruke (avsnitt 2.5.2). Da dette vil bryte med metaforen vi testet er det ikke mulig å si noe videre om hvordan det vil være og ny brukbarhetstest vil være nødvendig for å avgjøre dette.

10.1.3 Veiviser

I avsnitt 8.1.1 Sammenligning av de ulike metaforene blir navigering ved hjelp av statuslinjen diskutert. Konklusjonen derfra er at vi ønsker å la brukerne bruke statuslinjen til navigering, men at oppsummeringen før lagring da må gi tilbakemelding om noe er glemt, og ikke la brukeren lagre før det glemte er rettet opp.

Ved en endring som dette er det viktig å passe på at veivisermetaforen ikke blir brutt, en veiviser går sekvensielt og hopping mellom de ulike stegene skal være unødvendig. Prototypen viser alltid hva brukeren har gjort på tidligere steg. Problemet med å hindre hopping mellom stegene er at flere brukere ønsker og utforske programmet før de gjør noe. Det virket som om de var redd for å gjøre feil og dermed ville se muligheter før de valgte å gjøre noe. Med dagens løsning er det ikke mulig og utforske prototypen før en komposisjon lages og en versjon med navigering på statuslinje burde prøves. Dette kan potensielt føre til brudd i metaforen, da den ikke nødvendigvis blir brukt sekvensielt, derfor er det vanskelig å si om det er en god eller dårlig løsning. Enkelte hevder at en veiviser skal være navigerbar og ikke en sekvensiell opprasjon (46), og våre resultater tyder på at de har rett.

10.1.4 WIMP

Under brukbarhetstesten kom en bruker opp med et forslag om å gruppere startobjekt og hendelse for seg, samt mottakerobjekt og aksjon. Testsubjektet skisserte hva han mente på skjermen og en tegning er vist under resultatene (Figur 31 Forbedringsmulighet for WIMP-prototypen). Dette er en veldig bra forbedring som ikke har noen andre ulemper ved seg enn at det kan bli støyete. Vi har hele tiden utviklet prototypene etter KISS-prinsippet[3] og ønsket i utgangspunktet å holde de så enkle som mulig. Her ser det ut til at vi har valgt å gjøre denne prototypen litt for enkel da flere brukere hadde problemer med hva som var start- og mottakerobjekt. En endring som dette kan gjøres ved hjelp av gestaltprinsippet (47). Da enhet egentlig er samme type, og hendelse/aksjon er forholdsvis like, ønsker vi å gruppere start og mottaker hver for seg. Dette er i strid med teorien som sier at like objekter skal grupperes sammen. Da start- og mottakerobjekt i utgangspunktet er de samme elementene. Den beste løsningen blir da å bruke forslaget fra det testsubjektet ved å putte start og mottaker i hver sin boks.

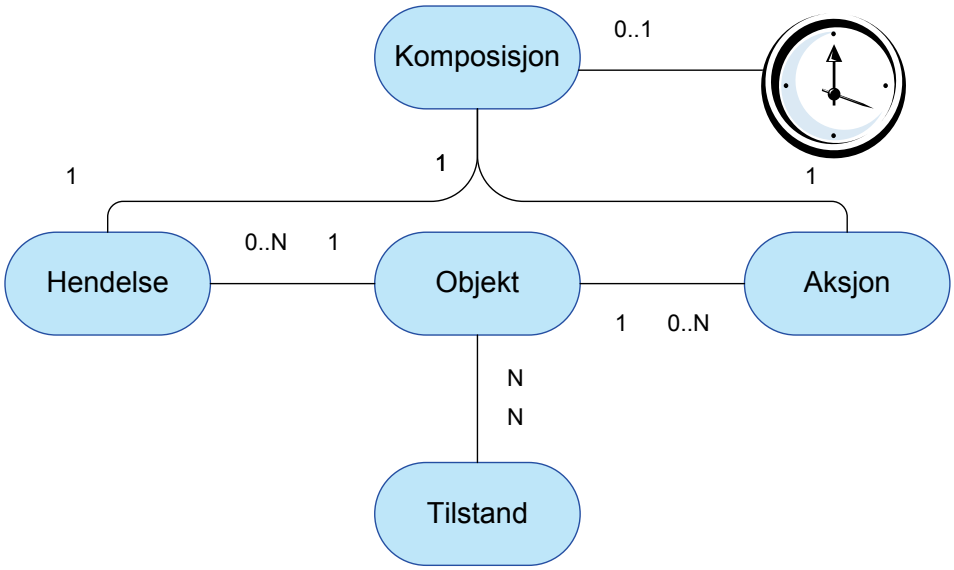
WIMP-prototypen vår inneholder ingen meny og bryter således mot den opprinnelige WIMPmetaforen. Dette førte ikke til at brukerne følte noe manglet i prototypen. De fleste brukere er kjent med denne metaforen og derfor ville det være naturlig av dem å forvente mer av den.

10.1.5 Konseptuel modell

Vi valgte å gjøre den konseptuelle modellen til den ekleste form av en komposisjon: En hendelse og en aksjon knyttet til ett eller to objekter som har en tilstand. Dette er gjort da oppgaven omhandler komponering av tjenester og ikke konfigurasjon. Det viste seg under testingen at brukere savnet muligheten for enkle konfigurasjoner, da det ikke ble helt troverdig system for smarthjem uten. De fleste oppgavene gikk bra, men flere testdeltagere reagerte på at lyset på soverommet skulle være på når de var der. Dette ville være til sjenanse om natten når de skulle

[3] Keep it simple stupid

sove. Mange av testsubjektene ønsket derfor en mulighet til å sette tidsbegrensning på en komposisjon. Dette kan gjøres på flere måter, det er mulig å sette tidsbegrensning for når en hendelse eller aksjon skal være mulig. Dette ville ikke være en hensiktsmessig løsning for objektkobling, da hvert objekt kun er representert ved en brikke. Da måtte tiden stilles to ganger, på samme objekt, en for hendelsen og en for aksjonen. Dette kunne fort virket forvirrende og en bedre løsning ville være å sette tidsbegrensning for hele komposisjonen. Dette kan være en meget enkel klokke hvor det er mulig å angi et tidsrom og hvilke dager i uken den skal gjelde for. Dette er konfigurasjon av en komposisjon og på kanten av oppgaven, men det ville vært med og bedret resultatene hvis det ikke hadde blitt et forstyrrende element i testen. En typisk konfigurasjonsoppgave ville hatt muligheter for å gjøre justeringer på hendelsene og aksjonene i tilegg til objektet. Vårt forslag til løsning for tidsbegrepet i den konseptuelle modellen er vist i Figur 35. Det er vanskelig å si om dette ville virket forvirrende og ødelagt resultatene for metafortesten vi har gjennomført, eller om det ville forsterket den. I avsnitt 10.2 drøfter vi om resultatene av testen ble ødelagt av prototypene og i avsnitt 10.4 kommer vi tilbake til gyldigheten av resultatet. I den sistnevnte blir tidsproblematikken tatt opp på ny.



Figur 35 Konseptuel modell med tidsbegrensning

10.2 Prototypenes rolle ovenfor de respektive metaforene

Det er alltid en fare ved denne type brukertesting at dårlige prototyper ødelegger for resultatet. Her vil de ulike kommentarene på prototypene sett i forhold til hverandre med tanke på faktorer som kan ødelegge for en metafor.

10.2.1 Generelle likheter

Alle prototypene er leksikalsk like i den utstrekning det lar seg gjøre. Prototypene bruker også de samme ikonene og navne på de ulike objektene og har de samme hendelser og aksjoner. Det varierer likevel litt hvordan dette presenteres. Der er ingen begrensning i det syntaktiske utover

de leksikalske byggeklossene og den konseptuelle modellen. Dette betyr at prototypene alle har likhetstrekk og dette gav utspill under testingen da testsubjektene generelt sett ble tryggere på systemets muligheter og begrensninger etter hvert som de hadde prøvd noen prototyper.

For å gjøre sammenligning av metaforene best mulig er det viktig at de ulike prototypene har de samme kvaliteter teknisk sett. Resultatet av SUS-spørreskjemaet bekrefter at alle prototypene er like brukervennlige. Ulike metaforer behøver ikke nødvendigvis være like brukervennlig. Det at vi fikk dette resultatet her skyldes nok delvis at de er leksikalsk like og et for lite datasett.

Alle brukerne klarte tolkeoppgavene, noe som tilser at det er forståelig programmer og ingen forskjeller mellom dem på dette punktet heller. Konklusjonen blir at de ulike prototypene er veldig like, selv om de ser forskjellige ut, og er forskjellige i bruk.

10.2.2 Sammenligning av ulike metaforer i en brukbarhetstest

Noen av metaforene er syntaktisk kjente for brukerne, de baserer seg på kjente metaforer og det er lett å forvente mye av dem, eller finne de kjedelige i forhold til de nye og spennende metaforene. En programmerer utalte at puslespillet var nytt og spennende, det er en kjensgjerning at ting ikke forblir nytt og spennende, men vanlig over tid. Spørsmålet da er om en mer klassisk metafor blir bedre sett over tid.

Klassiske metaforer gjør det enklere å utføre oppgaver og brukerne virket i våre tester tryggere på det de gjorde. Derfor burde en test vart mye lenger og brukeren burde prøvd de ulike metaforene over et større tidsaspekt. En brukbarhetstest gir likevel svar på hvor lett det er å lære å bruke en prototype og hvordan det vil være den første tiden systemet er i bruk.

10.2.3 De ulike metaforene

I **Veiviseren** ble det påpekt at der burde være mulig å navigere ved hjelp av å trykke på statuslinjen, og at stegene for start- og mottakerobjekt burde ha forskjellige navn. Dette kunne gjort at ikke så mange hadde problemer med å se start- og mottakerobjektet. Det sto uansett forklart med tekst på begge stegene hva som skulle velges. Tilbakemeldingene gikk mye på at det var tungt å bruke den, tok mye tid og var mye trykking. Dette er alle argumenter vedrørende metaforen og ikke implementeringen av den. Vi kan derfor konkludere med at prototypen representerte metaforen godt og anse resultatene som representative for en veiviser metafor i sluttbrukerkomponeringer av tjenester i smarte hjem.

I **WIMP** var det vanskelig å skille start- og mottakerobjektet, det ble det presentert en løsning på dette. Disse problemene oppsto kun i starten, og kan ha vært med på å trekke ned den totale opplevelsen. Ettersom testsubjektene ikke hadde vansker med dette senere i testen kan vi konkludere med at dette ikke har stor innvirkning på resultatene da begge gruppene fant den forståelsefull og enkel i bruk.

I **puslespillet** måtte enkelte anstrenge seg for å lese navnene på brikkene til venstre, dette tok litt tid, men var egentlig ikke et problem gjennom testen. Videre var det noen som begynte å legge brikker helt til høyre, de dro ikke brikkene ut i komponeringsfeltet. Dette kan være et problem da enkelte kan komme i fare for å gi opp før de har prøvd mer. Dette vil uansett ikke

innvirke på resultatene av en brukbarhetstest da ingen prøvde dette i mer særs kort tid. Vi kan derfor konkludere med at prototypen representerer metaforen på en god måte.

Objektkoblingsmetaforen fikk kommentarer på at den var dårlig implementert, men de som mente dette likte konseptet. Denne prototypen kan ha vært med å ødelegge litt for metaforen, da pilene er noe klumpete, vanskelig å merke og kan ved enkelte vinkler ligge litt oppå hverandre. For å unngå at dette skulle få stor effekt fikk hver av testpersonene opplyst at det var tidlig i utviklingen og pilene dessverre var stygge. Dette ble ikke sagt før etter de hadde tegnet den første pilen, da det på denne måten ikke fikk innvirkning på forståelsen av konseptet i metaforen. Dette resulterte i at testpersonene forsøkte å tenke objektivt på det de kaller konseptet, som er grensesnittmetaforen. Resultatene bør vurderes her, men dette har ikke innvirkning på de store linjene. Vi ser også av de kvalitative dataene at mange av tilbake meldingene går på andre faktorer enn implementeringen.

Den **grafiske kommandolinjen** var vanskelig å søke i, eller uvant å søke i. Søket opplevdes skuffende i forhold til hva de som prøvde det forventet. Dette var nok med på å trekke ned opplevelsen av å bruke den, men langt fra alle brukerne forsøkte å skrive inn noe med tastaturet. Tilbakemeldingene fra de som prøvde å skrive inn og ikke gjorde det er forholdsvis like. Videre ønsket noen at listene burde være sorterte, det var et bevist valg å beholde de usorterte, for å fremme bruken av tastatur. Denne prototypen ødela således ikke for metaforen.

Tabell 23 Oppsummering av implementering

Metafor		**Implementering**
	Programmerere	**Andre**
Veiviser	Dårlig statuslinje	Bekreftelse før lukking
WIMP	Sorterte Lister	Tydeligere rekkefølge
Puslespill	Liten skrift til venstre	Tydelig sammenkobling
Objektkobling	Dårlige koblinger	Dårlige koblinger
Grafisk Kommandolinje	Dårlig Søk	Tungt å lese liste

Tabell 23 oppsummerer de ulike prototypene ved hjelp av kategorien implementering fra Liste 13 Liste over kvalitative kategorier. Her er det viktigste, eller mest nevnte problemet nevnt. Det betyr at enkelte argumenter fra diskusjonen over ikke omfattes av denne tabellen. De ulike prototypene hadde god nok kvalitet til å representere metaforen på en tilfredsstillende men ikke optimal måte. Dette gjør at vi kan sammenligne resultatene mellom metaforene og ikke bare mellom de ulike implementeringene av dem. Vi skal likevel være litt forsiktig med å sammenligne objektkoblingsmetaforen direkte, da denne var dårligere implementert enn de øvrige modellene. I diskusjoner i analysekapittelet er det forbehold om prototypene representerer metaforen. Vi kan nå anse den diskusjonen til å omfatte metaforer og ikke bare prototyper.

10.3 Sluttbrukerkomponering av tjenester

Sluttbrukerprogrammering har vist seg å være vanskelig, men det er bevist at det er mulig å gjennomføre innenfor et bestemt domene. Denne oppgaven benytter sluttbrukerkomponering av tjenester, noe som er lettere for brukere enn sluttbrukerprogrammering. Vi har også valgt et bestemt domene og begrenset mulighetene. Den konseptuelle modellen begrenser sterkt hvor avanserte komposisjoner det er mulig å lage. Det er flere grunner for at vi har valgt den enkleste form for en komposisjon til å sammenligne metaforene med. Sluttbrukerprogrammering har feilet mange ganger før, da det blir for vanskelig for den vanlige bruker. For å gjøre det enkelt er det derfor viktig å begrense mulighetene samtidig som systemet utrykker nok til å være verdifullt. Det viste seg at brukerne følte dette systemet gav mening, men det var litt for snevert til å være et endelig system da tidsproblematikken ble for stor.

To av prototypene, puslespill og objektkobling er billedlige, i følge (11) er det et problem med skalering da det ikke blir nok plass på skjermen. Dette ble tilfellet i objektkobling da det ble mye på skjermen, i et endelig system må både objektkobling og puslespill videreutvikles for å overkomme denne begrensningen. Det oppsto ikke problemer knyttet til abstraksjonsnivået ved bruk av bilder. Både ”min lokasjon” og vær-objektet, bryter med en direkte kobling mot den konseptuelle modellen da de ikke kan være mottakere av en hendelse. Dette skyldes at disse objektene ikke har noen aksjoner å utføre. Vi opplevde ikke dette som en begrensning, men ser at det kan forårsake problemer dersom koblingen mellom brukerens mentale modell og komponeringsmuligheten blir ytterligere forverret.

Komponering av tjenester, en enkel form for sluttbrukerprogrammering, er fult mulig å gjennomføre på et bestemt domene. Vi har vist det igjennom denne studien og resultatene våre er i tråd med tidligere teori. Det er begrenset hvor mye forskning det er gjort på dette området i den senere tid, og det videre arbeidet med den best egnede metaforen vil være med å vise de nye mulighetene vi har fått som en følge av kraftigere og bedre teknologi. Det gjenstår enda å se om dette kan utnyttes til å bedre brukervennligheten på sluttbrukerkomponering.

10.4 Validitet

Vi har allerede diskutert hvor godt prototypene representerer de ulike metaforene, og hvilke begrensninger de legger på metaforene. Dette er en av mange faktorer resultatets gyldighet er avhengig av og her vil vi se nærmere på flere andre.

10.4.1 Testutforming

Testene ble gjennomført i et brukbarhetslaboratorium og ikke i et ekte smarthjem. Det var derfor umulig å prøve komposisjonene i et ekte smarthjem og det er vanskelig å si hvordan dette vil være. Testene varte rundt en time for hvert testsubjekt, dette gir rundt åtte minutter til hver prototyp. 8 minutter er nok tid til å avduke brukbarhetsproblemer, men det gir ingen resultater på hvordan brukerne opplever teknologien i dagliglivet. Til prototypene var der en simulator så testpersonene kunne verifisere at de hadde laget riktige komposisjoner. Denne simulatoren gav dem også en forståelse over hva som ville skje i huset. Vårt tenkte smarthus besto av en toroms leilighet med eget kjøkken, ikke alle bor i denne type bolig og det kan være vanskelig å tenke seg hvordan det ville være. Resultatene våre sier derfor ikke noe om hvordan en metafor er i bruk i et smarthus, men hvor lett en bruker har for å komponere tjenester i det. Det gir også gode

resultater på hvordan brukeren opplever å bruke programmet isolert sett. Da ingen av testdeltagerne har bodd i et smarthus vil de ha vanskelig for å forestille seg dette konseptet, og vi har følgelig heller ikke fokusert på effekten av teknologien.

10.4.2 Realistiske oppgaver

Den enkle konseptuelle modellen vår legger begrensninger på hvor avanserte komposisjoner det er mulig å lage. Vi tilstrebet å lage komposisjonene forskjellige og prøvde å dekke flere sider ved et smarthjem. Noen av oppgavene var nyttige, andre morsomme og noen informative. Alle oppgavene ble godt tatt i mot, med unntak av en der de skulle vite været når de våknet om morgenen. De fleste ønsket her å trekke opp persiennen og se ut, da været kan ha så mange ulike nyanser. Et testsubjekt pleide å se hva folk hadde på seg for å avgjøre været, mens en annen så etter vind i fjellene. Dette er aspekter et smarthjem på nivå 3 etter skalaen til (30) har mulighet til å kjenne til, men det virket ikke som brukerne var interessert i det. Utover dette opplevde testsubjektene oppgavene som mer eller mindre realistiske. Alle kunne tenke seg et system som det de prøvde under testen, men enkelte ønsket mindre modifikasjoner. Den viktigste modifikasjonen var tidsbegrensningen på komposisjoner, dette er diskutert i avsnitt 10.1.5 ovenfor. Oppgavene fremsto realistiske, noe som gjorde det lettere for testpersonene å forstå hva de prøvde og på den måten gi bedre tilbakemeldinger.

10.4.3 Domenet smarthjem

Brukerne fikk en innføring i hva et smarthjem var, og brukte ikke lang tid på å forestille seg et. Teknologien har kommet langt og unge mennesker omgir seg med teknologien til vanlig. Flere lurte på hvorfor dette ikke har kommet allerede, og hvor lende de må vente før de kan få et system som dette i hjemmet sitt. Alle var oppriktig interessert og ville være med på å utvikle fremtiden. Domenet appellerte til testpersonene og var således meget treffende. Det at domenet appellerer til testpersonene er viktig for de enkelte testdeltagers motivasjon til å gjøre testen så realistisk og god som mulig. Flere av testpersonene var ivrige og kom sågar med forbedringsmuligheter, noe et spennende domene er med på å fremskaffe. Da vi brukte arkitektstudenter som ”ikke programmerere” er det naturlig at de var opptatt av et hjem og interessert i å se hvordan fremtiden kan bli.

10.4.4 Diskusjon av metode

I denne oppgaven er det benyttet mange forskjellige evalueringsmetoder som til sammen danner forskningsmetoden. Vi har valgt å benytte flere ulike evalueringsmetoder for å bedre kunne ende opp med et riktig resultat. En enkelt metode kan i enkelte tilfeller gi et feilaktig resultat, og vi trekker derfor ingen konklusjoner basert på en enkelt hendelse eller metode.

Ved bruk av kvalitative data er det alltid fare for at forskerens meninger og antagelser er med å forme resultatet. For å unngå dette valgte vi å lage post-it lapper med alle utsagnene og gruppere de som hørte sammen, i nærheten av hverandre. På denne måten er det dataene som danner kategoriene og ikke oss. Det er allikevel en fare for at kategoriene er påvirket av våre tanker. Vi har derfor valgt å legge kvantitative analyser til grunn for siden å støtte eller avkrefte den ved hjelp av de kvalitative resultater. De kvantitative dataene består av en kortrangering og noen spørreskjema. Ved bruk av SUS hadde vi ventet å finne noen prototyper mer brukervennlig enn andre. Det at vi ikke gjorde dette betyr at de er like brukervennlige, noe som tilsier at

resultatene fra kortrangeringen ikke omhandler hvor brukervennlig et system er, men andre faktorer. Ved hjelp av kvalitativ analyse fant vi disse faktorene og ser at det er en sammenheng mellom hva brukerne har foretrukket og resultatet fra kortrangeringen. De kvalitative kategoriene vi fant omhandler brukervennlighet, men er mer nyanserte og tilpassede domenet enn SUS. I tilegg har vi kvantitative data på hvorvidt testsubjektene ønsker seg et smarthjem, dette er data for å validere forskningen ved å vise at testpersonene var motiverte for et system som de vi har presentert.

SUS blir brukt for å måle forskjellig brukervennlighet mellom ulike programmer, eller for å finne forskjeller mellom individer eller grupper i oppfatning av brukervennlighet. For å finne den generelle brukervennligheten til systemene baserte vi oss på kvalitative data og en evalueringsmetode for tolking av skjermbilder. Denne metoden gav oss et litt uventet resultat, vi hadde forventet at brukerne fant det enklere å lese komposisjoner i noen prototyper enn andre. Dette var ikke tilfellet, men vi kunne i stedet legge dette resultatet sammen med de kvalitative dataene som et pluss i retning av brukervennlighet. I ettertid er det da lett å se at dette ikke var den mest effektive måten å finne ut dette på, ett spørsmål i intervjuet om hvor lett systemet var å bruke i forhold til et par andre kjente programmer ville trolig gjort samme nytte og gått mye fortere.

De kvalitative dataene ble samlet inn under brukbarhetstesting og intervju. Under intervjuet hadde det vært verdifullt og spurt deltagerne om forskjeller på prototypene i forhold til de ulike kategoriene vi har presentert. Dette lot seg ikke gjøre av to grunner, kategoriene er delvis en følge av intervjuet, og vi hadde sterkt tidsbegrensning. Det vi i stedet burde ha gjort for å sikre gyldigheten av kvalitative data var å sende testdeltagerne et spørreskjema med fokus på de ulike kategoriene kort tid etter testen.

De kvantitative dataene bygger på et tilstrekkelig antall testsubjekter og gav tidvis særdeles signifikante resultater. Det var kun for ”ikke programmerere” at det kan såes tvil om resultatene, noe vi har tatt høyde for ved hjelp av andre evalueringsmetoder. Dette er helt klart et usikkerhetsmoment og vil være med på å svekke de endelige resultatene.

Evalueringsmetodene vi har benyttet er anerkjente og gir gode resultater, når vi i tilegg har sammenlignet resultatet fra dem kan vi konkludere med at de gir gode og gyldige resultater for det vi testet.

10.4.5 Oppsummering

I dette underkapittelet har vi diskutert testen, gjennomføringen og metodene vi har benyttet og i 10.2 diskuterte vi prototypenes representasjon av metaforene. Dette er alle ulike faktorer med innvirkning på validiteten til de endelige resultatene. Dette er mange faktorer og noe lite kan ha gått feil med hver og en av dem, noe vi ikke har sett eller noe vi ikke kunne kontrollere. Resultatet er derfor ikke 100 % gyldig, vi har i tilegg litt implementeringsproblem med objektkobling og to av evalueringsmetodene våre gav ikke det resultatet vi hadde forventet. Resultatene fra denne oppgaven har stor validitet, innenfor de best egnede metaforene. WIMP ble den beste metaforen totalt sett, tett etterfulgt av puslespill. Da det er små

usikkerhetsmomenter ved forskningen og evalueringsmetodene vi har benyttet bør ikke WIMP ses på som en klar vinner og puslespillet bør være med videre i utviklingen av et endelig system.

10.5 Generaliserbarhet

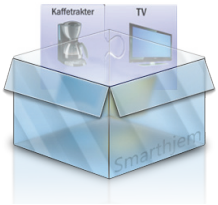
Vi har forsket på ulike metaforer innenfor et bestemt domene. Det å holde seg til et bestemt og gjerne snevert domene er viktig for at sluttbrukerprogrammering skal lykkes. Det kan stilles spørsmål med hvorvidt denne forskningen kan flyttes til andre domener. En forutsetning for dette må være at alle objekter er like og den konseptuelle modellen blir bevart. Det er vanskelig å trekke noen konklusjoner på bakgrunn av dette, men resultatene fra dette kan være grunnlaget for forskning innenfor andre tilsvarende domener.

I prototypene er det ingen muligheter for å konfigurere de ulike objektene (se avsnitt 2.5.1), dette var et bevisst valg fra starten. I et ferdig sluttbrukerkomponeringsverktøy vil konfigurasjon av de ulike enhetene kunne kreves. En bruker vil forvente å stille lyden og kanalene på et fjernsyn ved hjelp av systemet. Det kan for eksempel være en bruker ønsker å skru Tv-en på når vekkerklokken utløses og da er det stor nytteverdi i å spesifisere kanal og lydnivå. En løsning på dette er å la brukeren velge mellom ulike nivå i dataprogrammet, at brukeren kan klikke på et objekt eller en knapp for å komme til et dypere nivå av det valgte objektet. Dersom dette gjøres på en måte hvor brukeren slipper å lære mye nytt i det nye nivået vil det fortsatt være mulig for vanlige brukere å bruke systemet (26). Dette vil være tilsvarende nivåbytter Microsoft popfly benytter (se avsnitt 2.5.3). For å finne ut hvordan skifte av nivå skal gjøres og om det blir for komplisert, må det forskes videre. Det er viktig å huske at flere muligheter i et system også gjør det mer komplekst (avsnitt 2.5.1).

10.5.1 Skalerbarhet

Et smarthjem vil inneholde flere objekter og muligheter enn det prototypene i denne oppgaven inneholder. Tidligere i oppgaven er det nevnt at de metaforene hvor komponering og lagring av objekter (puslespill og objektkobling) vil møte et problem med mange komposisjoner. Dette kan løses ved hjelp av ulike verdener, dette blir litt på samme måte som i taktil programmering. Det er mulig å ha en verden for hvert rom eller like enheter i huset. En annen løsning er bruk av en kortstokkmetafor. Her kan brukerne lage komposisjoner på hvert kort, og hele kortstokken utgjør alle komposisjoner i huset. En kortstokk vil gjøre det mulig å sette tidsbegrensning på de ulike kortene, aktivere og deaktivere et kort. Dette vil gi systemet en ny dimensjon men det er uklart hvor brukervennlig det vil være. Det kreves videre forskning for å avgjøre om dette er mulig å forstå for sluttbrukere.

De modellene hvor det er adskilt komponeringsområde og lagringsflate vil ikke møte de samme problemene. Her vil det være nødvendig å sortere listene på en måte brukeren forstår, skulle listene over objekter blir for stor kreves andre taktikker. Hva som er for stor liste er avhengig av hvor ofte systemet brukes og hvilke valg brukeren skal gjøre. Teknisk sett er det ikke problematisk, men en ny brukbarhetstest med mange komponenter kan ha stor nytteverdi. Her vil det da ikke være nødvendig å teste med like mange brukere, da vi kun er ute etter en bekreftelse eller avkreftelse på om det blir for mange valg i en liste.



11. Konklusjon

Dette kapittelet besvarer forskningsspørsmålene RQ1-4 og konkluderer basert på det arbeidet som er gjort i studien. Her vil også videre arbeid innenfor fagfeltet og domenet blir presentert.

11.1 Konklusjon

Vi har i denne studien sett på fordeler og ulemper ved fem forskjellige brukergrensesnittmetaforer. Metaforene er valgt fra ulike domener og hvor populære de er i bruk i moderne programmer. En av er hentet fra sluttbrukerprogrammeringsverden; en puslespillmetafor.

Der ble totalt brukt forskjellige brukergrupper til å teste de ulike metaforene, til sammen 16 testpersoner deltok i studien. Opprinnelig var det åtte fra hver gruppe, da den ene testpersonen falt under en annen gruppe enn vi hadde forutsett ble det henholdsvis ni ”ikke programmerere” og syv programmerere.

For å oppnå gode resultater er studien basert på flere ulike evalueringsmetoder; prototyping, pilottest, brukbarhetstest, kortrangering, spørreskjema, intervju og flere analysemetoder for både for kvalitative og kvantitative data. Resultatene av dette er kombinert for å minimere antall feilkilder. Der vil alltid være feilkilder i forskning også i denne studien. Den ene prototypen objektkobling var dårligere implementert enn de øvrige og kan ha fått en dårligere plassering enn de andre på bakgrunn av dette. Vi hadde også litt for få testpersoner til å oppnå signifikante resultater på alle testene, eller så er det stor variasjon i brukernes preferanser.

RQ1: Hvilke brukergrensesnittmetafor er best egnet for sluttbrukerkomponering av tjenester i smarte hjem? *To modeller skilte seg ut som de best egnede for sluttbrukerkomponering av tjenester i smarte hjem. WIMP og Puslespill er signifikant bedre enn de øvrige modellene.*

RQ2: Er det noen forskjell mellom brukergrupper i forhold til preferanser av slike metaforer? *Det er forskjell mellom de to brukergruppene denne studien har evaluert. Programmerere foretrekker en puslespillmetafor, men de andre foretrekker WIMPmetaforen.*

RQ3: Hvilke faktorer er viktige i brukernes argumentasjon for hva som er gode og dårlige metaforer? *Der er ulike faktorer brukerne finner viktig for brukergrensesnittmetaforer i smarthjemdomenet er (i tilfeldig rekkefølge):*

- *Effektivitet – Hurtig er best og treg dårligst*
- *Metafor – Oversiktlig, rett frem er best og Rotete, uforståelig dårligst*
- *Opplevelse – Morsom er best og kjedelig/tungvint dårligst*
- *Rekkefølge/forståelse – Forståelig er best og vanskelig er dårligst*

De ulike faktorene og deres skala er presentert i kapittel 8.3 og argumentene er hentet fra den kvantitative analysen. Disse er beskrevet under resultater i Liste 13*.*

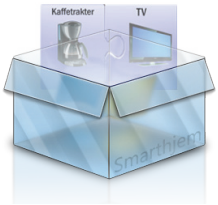
RQ4: Basert på RQ1-3, hva vil være den best egnede metaforen? *Metaforen WIMP egner seg best til komponering av tjenester i smarte hjem. Denne metaforen gjør det litt bedre enn puslespillet på de kvalitative kategoriene fra RQ3. Det er for øvrig ingen andre skiller mellom disse to modellene og begge bør vurderes som gode metaforer til sluttbrukerkomponering av tjenetester i smarte hjem. Det er foreslått et par forbedringsmuligheter for WIMP i studien, dette går på bedre gruppering av start og mottaker i en komposisjon. Dersom disse blir implementert vil WIMP vil skille seg mer fra puslespill.*

Før det kan hevdes at WIMP og Puslespill er det beste metaforene for komponering av tjenester i andre domener må dette bevises ved en studie i et annet domene. Resultatet er kun gyldig for den enkle konseptuelle modellen vi har benyttet, men det antydes at denne kan utvides litt da brukerne savnet noe funksjonalitet.

Denne studien bringer ny kunnskap til sluttbrukergrensesnitt innenfor et smarthjem. Tidligere system har feilet grunnet dårlige brukergrensesnitt(6) og mye av forskningen er fra tiden datamaskiner ikke var vanlig. Brukergrensesnittmetaforer har utviklet seg og denne oppgaven viser moderne metaforer i et område hvor mye av forskningen er fra 80 og begynnelsen av 90-tallet. Resultatene vil derfor være med å legge grunnlaget for videre forskning på veien mot smarthjem.

11.2 Videre arbeid

Det er gjort lite arbeid med smarthjem i den senere tiden og før dette systemet kan bli en realitet trenger vi å utforske bruksmønsteret av et smarthjem. I den videre utviklingen er det viktig å la brukerne være med å styre behovene innenfor smarte hjem og ikke la teknologiens muligheter styre. Det trengs også videre forskning for hvilke metaforer som egner seg til konfigurering av smarte hjem, og hvordan denne metaforen kan kombineres best mulig med WIMP eller puslespillmetaforen. Først når dette er gjort kan smarthjem bli en realitet hvis produsentene av komponenter kommer til enighet om en felles standard for kommunikasjon.



12. Referanser

1. **Statistisk sentralbyrå.** Statistisk sentralbyrå. *Informasjonssamfunnet .* [Internett] ssb, 2009. http://www.ssb.no/ikt/.

2. **Kristoffersen, Svebak og Aasarød.** Livskvalitet, kjønn, plager og humor hos pasienter i dialyse. *Tidskrift for den norske legeforening.* 2008, 8.

3. **Koskela, Tiiu og Väänänen-Vainio-Mattila, Kaisa.** Evolution towards smart home environments: empirical evaluation. *Pers Ubiquit Comput.* 2004, 8.

4. **Goodell, Howie, et al.** What Is End-User Programming? [Internett] ca 1998. http://www.cs.uml.edu/~hgoodell/EndUser/whatsEUP.htm.

5. *KidSim: end user programming of simulations.* **Cypher, A og Smith, D.** s.l. : ACM Press/Addison-Wesley Publishing Co., 1995. ss. 27--34.

6. **Gann, David, Barlow, James og Venables, Tim.** *Digital Futures: Making Homes Smarter.* Coventry : The Chartered Institute of Housing, 1999.

7. **Zhang, Tao og Brügge, Bernd.** *Empowering the User to Build Smart Home Applications.* München, Germany : IOS press, 2004.

8. *KidSim: programming agents without a programming language.* **Smith, D, Cypher, A og Spohrer, J.** s.l. : ACM, 1994, Commun. ACM, Vol. 37, ss. 54--67. 0001-0782.

9. **Cocoa Dev Central .** Coca Dec Central. [Internett] 2009. http://cocoadevcentral.com/.

10. **Andress, Sarah.** KidSim. [Internett] wellesley. http://www.wellesley.edu/CS/courses/CS110/History/KidSim.html.

11. **Nardi, Bonnie A.** *A small matter of programming: perspectives on end user computing.* s.l. : MIT Press, c1993. ss. XVI, 162 s.

12. *Metaphors we compute by: bringing magic into interface design.* **Rohrer, Tim.** Oregon : Department of Philosophy University of Oregon, 1995.

13. **Norman, Donald A.** *Psychology of everyday things.* s.l. : Perseus Books Group, 1990.

14. **Norman, D og Draper, S.** *User Centered System Design; New Perspectives on Human-Computer Interaction.* s.l. : L. Erlbaum Associates Inc., 1986.

15. **Lisetti, Christine L og Schiano, Diane J.** Automatic Facial Expression Interpretation: Where Human-Computer Interaction, Artificial Intelligence and Cognitive Science Intersect. *Pragmatics*

and Cognition (Special Issue on Facial Information Processing: A Multidisciplinary Perspective). 2000, Vol. 8, 1, ss. 185-235.

16. **Proctor, Robert W. og Vu, Kim-Phuong L.** *Handbook of Human Factors in Web Design.* s.l. : Routledge, 2005. ISBN 0805846123, 9780805846126.

17. **Baum, L. Frank.** *The Wonderful Wizard of Oz.* Chicago : George M. Hill Company, 1900.

18. **Microsoft Research.** Songsmith. *Everyone has a song inside...* [Internett] Microsoft corp, 2009. http://research.microsoft.com/en-us/um/redmond/projects/songsmith/.

19. **Sandberg, Karl W., Palmius, Joel og Pan, Yan.** An evaluation of a Wizard approach to Web design. *Electronic Journal.* 2009, Vol. 12, 1.

20. **Beaudouin-Lafon, Michel.** Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. 2000.

21. *Component visualization based on programmers conceptual models (poster session).* **Martin, L og Siemon, E.** s.l. : ACM, 2000. ss. 73--74.

22. *The Usefulness of Constraints for Diagram Editing.* **Wybrow, Michael, et al.** Melbourne : s.n., 2008.

23. *Does metaphor increase visual language usability?* **Blackwell, A F og Green, T R.** 1999. Visual Languages, 1999. Proceedings. 1999 IEEE Symposium on. ss. 246-253.

24. *Visual object-oriented programming.* **Burnett, M.** s.l. : ACM, 1994, SIGPLAN OOPS Mess., Vol. 5, ss. 127--129. 1055-6400.

25. **Johnson, Gary W og Jennings, Richard.** *LabVIEW graphical programming; 4th ed.* s.l. : McGraw-Hill Professional, 2006. ISBN 0071451463, 9780071451468.

26. *Visual AgenTalk: Anatomy of a Low Threshold, High Ceiling.* **Reppening, Alexander og Ambach, James.** Colorado : University of Colorado, 1996. CU-CS-802-96.

27. **Repenning, A og Ambach, J.** *Tactile Programming: A Unified Manipulation Paradigm Supporting Program Comprehension, Composition and Sharing.* Colorado : IEEE Computer Society, 1996. s. 102.

28. *Building Mashups by example.* **Tuchinda, R, Szekely, P og Knoblock, C.** Gran Canaria, Spain : ACM, 2008. ss. 139--148. http://doi.acm.org/10.1145/1378773.1378792.

29. **pipes_team.** JumpCut. *Learn How to Build a Pipe in Just a Few Minutes.* [Internett] http://www.jumpcut.com/fullscreen?id=F4396574585311DC87A2000423CF0184&type=clip.

30. *Smart Homes: Past, Present and Future.* **Aldrich, Frances K.** 0 0 2003, Inside the Smart Home, ss. 17--39.

31. *Conceptual models: begin by designing what to design.* **Johnson, J og Henderson, A.** s.l. : ACM, 2002, interactions, Vol. 9, ss. 25--32. 1072-5520.

32. **Floyd, Christiane.** *A systematic look at prototyping.* Berlin : s.n., 1984. ss. 1--18. SWT FRS-6.

33. *What Do Prototypes Prototype?* **Houde, Stephanie og Hill, Charles.** Amsterdam : Handbook of Human-Computer Interaction, 1997, Vol. 2.

34. *Breaking the fidelity barrier: an examination of our current characterization of prototypes and an example of a mixed-fidelity success.* **McCurdy, M, et al.** s.l. : ACM, 2006. ss. 1233--1242.

35. **Dumas, JS.** *A practical guide to usability testing.* s.l. : Intellect Books, 1999. ISBN 1841500208, 9781841500201.

36. *The usability engineering life cycle.* **Nielsen, J.** Mar 1992, Computer, Vol. 25, ss. 12-22. 0018-9162.

37. **Oates, B J.** *Researching Information Systems and Computing.* s.l. : Sage Publications Ltd.,, 2006. ISBN 141290224X, 9781412902243.

38. **Robson, Colin.** *Real world research: a resource for social scientists and practitioner-researchers.* s.l. : Wiley-Blackwell, 2002. ISBN 0631213058, 9780631213055.

39. *Doing interpretive research.* **Geoff, Walsham.** Cambridge : European Journal of Information Systems, 2006, Vol. 15. doi:10.1057/palgrave.ejis.3000589.

40. **MacFarland, Dr. Thomas W.** Statistics Tutorial. *Friedman Two Way Analysis of Variance by Ranks.* [Internett] 1998. http://www.nyx.net/~tmacfarl/STAT_TUT/friedman.ssi.

41. *Statistical Comparisons of Classifiers over Multiple Data Sets.* **Demšar, J.** s.l. : MIT Press, 2006, J. Mach. Learn. Res., Vol. 7, ss. 1--30. 1533-7928.

42. **Bland, Martin.** Multiple significance tests and the Bonferroni correction. *An Introduction to Medical Statistics.* s.l. : Oxford University Press, USA, 200.

43. **Brooke, J.** SUS: A quick and dirty usability scale. [bokforf.] Patrick W. Jordan. *Usability evaluation in industry.* s.l. : CRC Press, 1996.

44. **Flem, Lars Kristian.** tenketing.net. *Hvor legger jeg log-in?* [Internett] 12 November 2008. http://tenketing.net/2008/11/12/hvor-legger-jeg-log-in/.

45. **Miller, George.** The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. [bokforf.] William P. Banks, James B. Newman Bernard J. Baars. *Essential sources in the scientific study of consciousness.* s.l. : MIT Press, 2003, Vol. 63, ss. 81--97.

46. *Interaction Patterns in User Interfaces.* **Welie, Martijn og Trætteberg, Hallvard.** s.l. : PLoP 2000 conference, 2000. ss. 13--16.

47. **Spokane Falls Community College.** *The Gestalt Principles.* [Internett] http://graphicdesign.spokanefalls.edu/tutorials/process/gestaltprinciples/gestaltprinc.htm#similarity.

48. **Reardon, Megan.** Knitty. [Internett] knitty, 2004. http://www.knitty.com/ISSUEfall04/PATThallowig.html.

49. **Cypher, Allen og Halbert, Daniel Conrad.** *Watch what I do: programming by demonstration.* s.l. : MIT Press, 1993. ISBN 0262032139, 9780262032131.

50. **Johnson, G.** *LabVIEW Graphical Programming: Practical Applications in Instrumentation and Control.* s.l. : McGraw-Hill School Education Group, 1997.

51. **Electronic Arts Inc.** SimCity. *What is SimCity.* [Internett] EA games, 2001. http://simcity3000unlimited.ea.com/us/guide/.

Vedlegg I. Kortrangering

Tabell 24 Kortsortering rangering

Testperson	Veiviser	WIMP	Grafisk Kommandolinje	Puslespill	Objekt kobling	Prog. erfaring	Kjønn
A1	1	4	2	5	3	1	Jente
A2	2	5	3	4	1	2	Jente
A3	3	5	4	1	2	1	Jente
A4	2	5	3	4	1	1	Jente
A5	2	5	4	3	1	1	Jente
A6	1	3	2	5	4	3	Jente
A7	4	1	5	2	3	2	Jente
A8	1	3	2	4	5	2	Gutt
D1	2	5	1	4	3	5	Gutt
D2	2	3	1	5	4	5	Gutt
D3	1	5	2	4	3	4	Gutt
D4	1	4	5	3	2	4	Gutt
D5	1	3	2	5	4	5	Gutt
D6	4	3	2	5	1	4	Jente
D7	1	5	3	4	2	4	Jente
D8	2	4	1	5	4	3	Jente

Vedlegg II. SUS

Noen spørsmål om systemet du har brukt.

Vennligst sett kryss i kun en rute pr. spørsmål.

	Sterkt uenig				Sterkt enig
1. Jeg kunne tenke meg å bruke dette systemet ofte.	1	2	3	4	5
2. Jeg synes systemet var unødvendig komplisert.	1	2	3	4	5
3. Jeg synes systemet var lett å bruke.	1	2	3	4	5
4. Jeg tror jeg vil måtte trenge hjelp fra en person med teknisk kunnskap for å kunne bruke dette systemet.	1	2	3	4	5
5. Jeg syntes at de forskjellige delene av systemet hang godt sammen.	1	2	3	4	5
6. Jeg syntes det var for mye inkonsistens i systemet. (Det virket "ulogisk")	1	2	3	4	5
7. Jeg vil anta at folk flest kan lære seg dette systemet veldig raskt.	1	2	3	4	5
8. Jeg synes systemet var veldig vanskelig å bruke	1	2	3	4	5
9. Jeg følte meg sikker da jeg brukte systemet.	1	2	3	4	5
10. Jeg trenger å lære meg mye før jeg kan komme i gang med å bruke dette systemet på egen hånd.	1	2	3	4	5

SUS
Norsk versjon ved Dag Svanæs
NTNU 2006

Vedlegg III. Resultater SUS

Tabell 25 Resultater fra SUS

Modell	antall	snitt	standardavvik
Veiviser	3	89,17	10,41
WIMP	5	80,00	11,73
Grafisk kommandolinje	2	83,75	8,84
Puslespill	3	90,83	13,77
Objekt kobling	3	93,33	6,29

Vedlegg IV. Spørsmål til intervju

Følgende faste spørsmål til intervjuet ble alltid stilt:

- Tror du noen gang at du kommer til å flytte inn i et smarthjem eller et hjem som dette?
- Tror du at et system som det du nå har prøvd kan være nyttig?
    - Oppfølges med om det ville være morsomt med et sånt system?
- Fokuser videre på interessante konsepter og problemer i løpet brukbarhetstesten
- Grip fatt i eventuelle kreative løsninger, eller tankespill som kom frem i brukbarhetstesten.

Vedlegg V. Person Spørsmål

Digitale Hjem
Konfidensielt

NTNU
Det skapende universitet

Personas

Navn:

_____

Gate:

_____

Postnr og sted :

_____

Alder: _____

Yrke/Studie: _____

Telefon: _____

Dine programmeringsferdigheter

- ☐ Ingen
- ☐ Liten (HTML)
- ☐ Middels (PHP,script språk)
- ☐ Stor
- ☐ Veldig Sor

3 egenskaper som beskriver deg:

1_____

2_____

3_____

Figur 36 Person spørsmål

Vedlegg VI. Plansje over det digitale hjem

Det Digitale Hjem

<table>
<tr><td colspan="2">Stuen
   
Digital Fotoramme TV Taklys Stue</td></tr>
<tr><td>Soverom
  
Taklys Soverom Klokkeradio</td><td>Kjøkken
 
Kaffetrakter</td></tr>
<tr><td>Ute
  
Dørklokke Værstasjon</td><td>I Huset
  
Innbruddsalarm Røykvarsler</td></tr>
<tr><td>Forskjellig
  
Min Lokasjon Min Mobil</td><td>Bad
 
Taklys Bad</td></tr>
</table>

Vedlegg VII. Kvalitative data

Tabell 26 Positive elementer, forbedringer og annet ved puslespill prototypen

Puslespill		
	Programmerere	**Ikke Programmerere**
Positivt	Alt i samme vindu	Dette var gøy
	Morsomt	Leker mye rundt, utvider oppgavene
	Klarte å løse oppgaven med vær her, etter forsøk i 3 prototyper før denne	"Den var søt"
	lett å se start- og mottakerobjekt	Husker oppgavene og går fort frem
	Veldig tydelig	Trivelig og innlysende
	2x Går veldig fort	Finner mange løsninger for å sende bilde til den digitale fotorammen
	Tenker annerledes på vær brikken enn i andre prototyper	Dette var morsomt
	"Lettere enn de andre, trolig fordi jeg kjenner systemet" - etter å ha prøvd andre prototyper først	Mer tydelig enn objektkobling
	Mer tydelig	Går veldig fort og er lite tenking
	Går rett på sak	
	Liker den fordi den ser ”fancy” ut, men er ikke dermed letter å bruke	
Forbedringer	3x vil bygge mer avanserte puslespill	Ønsker at nedtrekkmenyen skal åpne seg når den trekkes ut på komponeringsområdet
	Vekkerklokke skal vekke meg, ikke spille musikk	De burde skiftet farge når de kobles sammen
Annet	Forventet egen brikke for aksjon og hendelser	

Tabell 27 Negative elementer og problemer ved puslespillprototypen

Puslespill		
	Programmerere	**Ikke Programmerere**
Negativt	fare for rotete skjermbilde	Vanskelig å se at mange ting vil forstyrre meg på mobilen
	Burde sett hele teksten i en lukket nedtrekkmeny	Tungt å rydde på skjermen, testpersonen må ha det han/hun jobber med midt på skjermen
	For liten skrift på brikkene til venstre	Forstår ikke forskjell på start- og mottakerobjekt
		Bruker en del tid på å forstå hva som skal gjøres
		Litt for små brikker før de trekkes til høyre
		Dårligere enn de andre da den er for enkel på en måte
Problem	2x Vær kan ikke kun eksistere om morningen	Bruker lys på som min lokasjon, da testpersonen alltid har lys på i rom han/hun er og kobler det sammen med lys av. Verifiserer dette med simulatoren og konkluderer med at det var en dum løsning og oppdager min lokasjon
		Usikker på om en komposisjon er lagret
	roter litt med start- og mottakerobjekt	Klarer ikke å komponere oppgaven med vær ute om morningen
	Har problemer med været og går videre	Kobler innbruddsalarm til røykvarsler, da røykvarsler er en sirene
	Usikker på hendelser til den digitale fotorammen	Løsning på siste oppgave blir å gå til kjøkkenet og stoppe kaffetrakteren
		2 x Legger brikkene til høyre i brikke samlingen i begynnelsen

Tabell 28 Positive elementer, forbedringer og annet ved WIMP-prototypen

WIMP		
	Programmerere	**Ikke Programmerere**
Positivt	Denne er best fordi den er mest oversiktlig	Finner nye muligheter
	Lettere å se hva som er start og mottaker i en komposisjon	Viser større forståelse for sammenkobling av elementer til en komposisjon
	Mer oversiktlig enn puslespillet	Den var logisk
	2x Går igjennom uten å nøle/tenke/hurtig	Gjennomføres raskt
	Hele tiden oversiktlig	2x Leker med ulike komposisjoner
	Fin til å sette opp mye i	3x Går rett på uten å nøle
	Veldig smart	Føler seg veldig trygg
		Lett å forstå
		2x Veldig oversiktlig
		Liker å se alt samtidig
		Lett å se hva som er rett
Forbedringer	Listene burde vært sortert	
	Burde være gruppering av start og mottaker elementer (skisse vedlagt)	
Annet	Veldig lik puslespill	Veldig klassisk
	Usikker på om innbruddsalarm har sirene eller ikke	Veldig lik puslespill
	Veldig lik veiviser, bare alt er presentert samtidig	
	Bruker tastaturet og finnet det som monotont arbeid, og går over til bruk av mus som er raskere å bruke	

Tabell 29 Negative elementer og problemer ved WIMP-prototypen

WIMP		
	Programmerere	**Ikke Programmerere**
Negativt	Ønsker å velge flere enheter samtidig	Likte ikke å bruke denne av uforklarlig grunn
	Usikker på hva som er start og mottaker i en komposisjon	
	Rar presentasjon av start- og mottakerobjekt	
Problemer	Usikker på start og mottaker i en komposisjon i starten	Bruker innbruddsalarm som sirene, bruker denne ulikt i forskjellige metaforer
	Bruker innbruddsalarm for å avgjøre om vedkommende er på soverommet	

Tabell 30 Positive elementer, forbedringer og annet ved veiviserprototypen

Veiviser		
	Programmerere	**Ikke Programmere**
Positivt	Tenker lite og utfører oppgaven uten problem	Oppsummeringen før lagring er oversiktlig og bra
	Denne er sikkert fin første gang	Skjønner plutselig at en røykvarsler ikke er en sirene for innbruddsalarmen
		Får mer oversikt her enn i de andre programmene
		Mer oversiktlig, lett å se hva du kan gjøre
		Dette var lett
		Et grundig program som gir god forståelse
		Dette ser proft ut
		Minner om å installere et program
Forbedringer	Burde kunne navigere ved hjelp av statuslinjen	Burde vært en bekreftelse før programmet lukkes
	Både steg 2 og 3 heter velg enhet, de burde ha forskjellige navn	
	Bruke WIMP modellen for konfigurering, ved trykk på neste kunne steg 5 (oppsummeringen) kommet opp	
	Burde kunne velge flere elementer samtidig	Omstendelig, mye trykking men sikkert bra for ho mor

Tabell 31 Negative elementer og problemer ved veiviserprototypen

Veiviser		
	Programmerere	**Ikke Programmerere**
Negativt	Hater veivisere på generelt grunnlag	Ikke innlysende at aksjon/hendelse teksten i listen kan velges
	For mye hvitt, lite tekst	"Jeg er uvenner med veiviseren"
	Bruker mye tid til navigering	Usikker i starten
	Dette tar mye tid	Forstår ikke helt forskjellen på start og mottaker i en komposisjon
	Tung å bruke	Er ikke helt sikker på hva jeg gjør
	Denne var forferdelig	Vanskelig å se mulighetene
	Er penest og samtidig mest ubrukelig	Leser velkomst teksten og gjesper oppgitt
		Komplisert og uoversiktlig
Problemer	Usikker på om jeg skal velge avsender av et signal eller noe	Står helt fast og får hjelp til å komme i gang
	Prøver å trykke på statuslinjen flere ganger	2x Valgte ikke hendelse og kom ikke videre, fikk hjelp etter mange forsøk
	2x Vanskelig og hvite hva som er starten og slutten på en komposisjon	Lagrer ikke og prøver simulatoren flere ganger, før det lagres
	Er i tvil på om oppgaven er løst eller ikke	Testpersonen gikk seg vill i stegne og ba om å få starte på nytt
		2x Forstår ikke at objektene kobles sammen og ser hvert steg som uavhengig, men etter gjentatte forsøk og mye navigering blir det forståelig
		Forstår ikke hvorfor en komposisjon skal lagres

Tabell 32 Positive elementer, forbedringer og annet ved grafisk kommandolinjeprototypen

Grafisk Kommandolinje		
	Programmerere	**Ikke Programmerere**
Positivt	Lett å se sammenhenger	Oppdager at vær har mange muligheter
	Alt i samme vindu er bra	Nøler mindre enn ved tidligere modeller
	Enkelt	2x Leker med objektene og mulighetene, finner alternative løsning
	Veldig oversiktlig	Dette var lettere enn de andre, for jeg er vant til nedtrekkmenyer
	Liker å leke med tastaturet	
Forbedringer	Bør være tydeligere tilbakemelding når noe slettes	Ønsker varsling via oppringning og ikke bare SMS på mobilen
	2x Listene bør sorteres	For å bedre listene holder det å bruke et bilde av hver type objekt
	Tab bør få samme funksjon som Enter	
Annet	Liker å bruke tastatur	Alle foretrekker mus
	Får tips om tastaturet, og etter å ha prøvd litt kommer uttalelsen: "super fast"	Tastatur er så som så, og jeg er fan av hurtigtaster
	Alle foretrekker mus	Benytter pilknappene og ikke skriving, fortsetter derav med mus
	Veldig lik som WIMP	minner om programmering, hvis så
		Bruker både mus og tastatur, en kombinasjon

Tabell 33 Negative elementer og problemer ved grafisk kommandolinjeprototypen

Grafisk Kommandolinje		
	Programmerere	**Ikke Programmerere**
Negativt	Forvirrende nedtrekkmenyer	4x Tar mye tid å lete etter ønsket objekt
	Liker ikke at nedtrekkmenyene åpner seg av seg selv	Glad jeg ikke begynte med denne, da hadde jeg ikke klart noe
	3x For dårlig søkefunksjon	For mye jobb å leite opp objektene
	Føyer seg i listen over dårlige program	Tungvint å bla i listene
	2x Tenker en stund på hva som er start og slutten av en komposisjon	
	Glad jeg ikke startet med denne	
	Ulempe at alt er skjult	
	Lite oversiktlig	
	Ble for mange valg	
	2x Tar mye tid å lete etter ønsket objekt	
	For mange valg	
	Kjedelig	
	Lite logisk da den skjuler så mye informasjon	
	Tungt å bruke musen	

Tabell 34 Positive elementer, forbedringer og annet ved objektkoblingsprototypen

Objektkobling		
	Programmerere	**Ikke Programmerere**
Positivt	Stort pluss for at objektene kan flyttes	Leker mye rundt
	Interessant prototyp	Visuelt sett veldig bra
	Fin til kompliserte komposisjoner, mange til en osv	Tenker mye mer direkte når alt er synlig
	Liker konseptet, men ikke denne	Lett å tenke komposisjoner bestående av to objekter
	Er bra fordi den ser fin ut, men ikke nødvendigvis lettere å bruke enn de mer klassiske	Moro å dra piler
	Visuelt sett veldig riktig måte, sånn det skal fungere	
	Sekvensielt og enkelt	
	Artig å flytte på objektene	
	Lett å se sammenhenger	
Forbedringer	Burde være større forskjell mellom aktiv og ikke aktiv pil	
	Ønsker denne på trykkfølsom skjerm	
	Vil få opp mulighetene, aksjon hendelse er skjult	
	Hadde vært kult å kunne flytte pilene	
	Tydeligere tilbakemelding når noe merkes	
	Navnet på avsenderen burde vært i fet tekst i skriften på pilene	
	Kan brukes til å lage komposisjoner i, og lagrede komposisjoner kan heller vises i en liste tilsvarende veiviseren	
Annet	Denne hadde vært fin som en oversikt over huset, ser mye lettere sammenhenger	

Tabell 35 Negative elementer og problemer ved grafisk objekt kobling prototypen

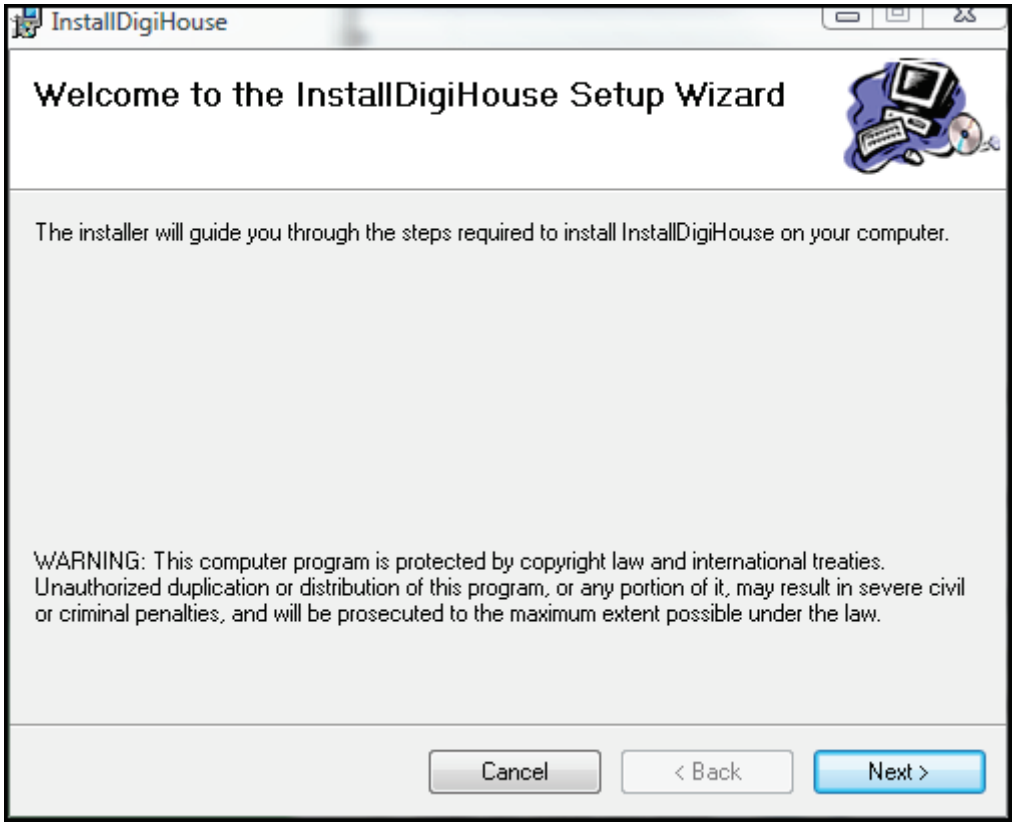
Objektkobling		
	Programmerere	**Ikke Programmerere**
Negativt	Litt kaotisk	Knotete piler
	2x Litte oversiktlig	Vanskelig med bare et bilde av hvert objekt
	Lettere å navigere i lister	Denne likte jeg ikke, vet ikke helt hvorfor egentlig
	Vanskelig å finne ut at objekter kan flyttes	Utbryter: "Nei, huff" i det den åpnes
	Frykter det fort blir for mye på skjermen	Vanskelig å slette en kobling
	Rotete lagt opp, tror det er prototypen, og ikke konseptet	Føler seg veldig usikker
	Teksten på pilene er lite informativ	Knotete når to ting kobles sammen mange ganger
	Pilene tar for mye plass	Usikker på om ting en komposisjon er riktig og lagret
	Vet ikke hvilken strek som er valgt	
	Oppleves støyete å jobbe i	
	Blir for mye på skjermen	
	Vanskelig å finne ulike objekter og komposisjoner	
	Jeg liker ikke å rydde på skjermen, men denne er fin når den er ryddig	
Problemer	Ønsker flere objekter av samme type i begynnelsen	4x Glemmer å sette aksjon og hendelse på enkelte komposisjoner
		Finner ikke ut hvordan en komposisjon kan slettes
		Hjelpetekst for hvordan trekke piler kommer ikke opp, dette blir fortalt
		Usikker på hvilken vei pilene skal settes
		Prøver å legge puslespill av objektene

Vedlegg VIII. Installasjonsveiledning for prototypene

Prototypene er utviklet i WPF og **krever Windows** for å kjøre optimalt, dette skyldes at de krever .NET 3.5 noe som ikke offisielt er støttet av andre OS. Der finnes mulige løsninger for å få dette til, men det er ikke utprøvd i praksis og anbefales ikke.

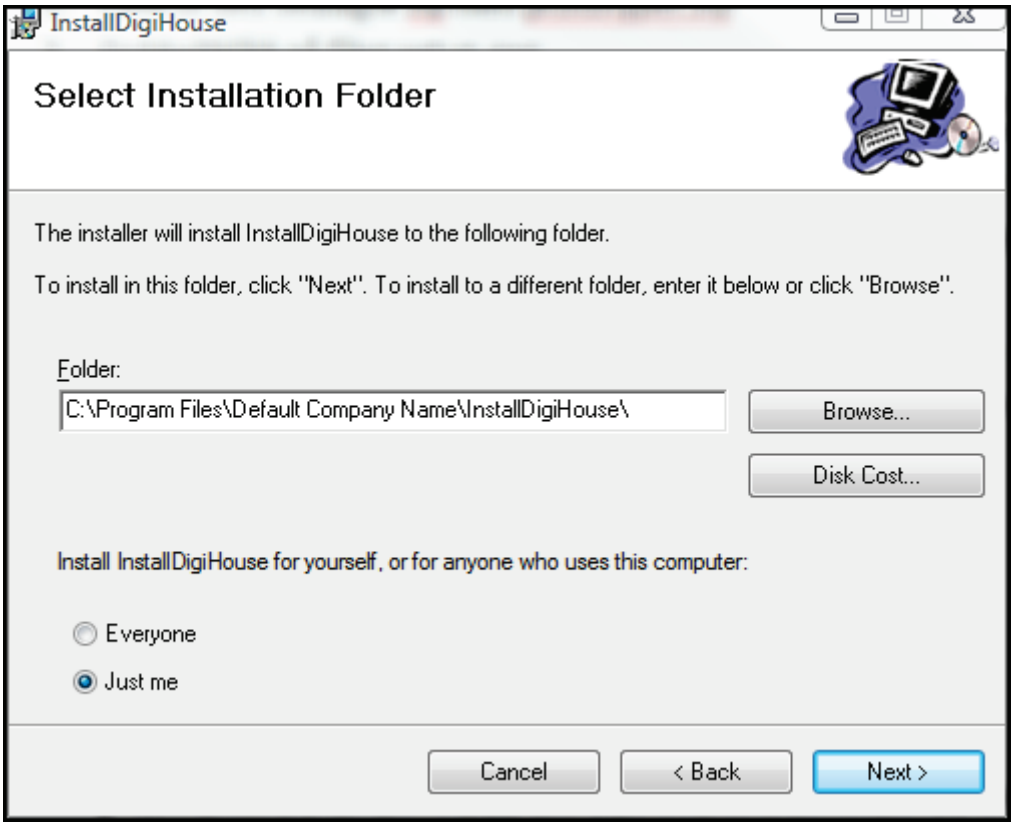
Prototypene må kjøres i 1280x1024 oppløsning for å virke optimalt

1. Pakk ut den vedlagte zip filen prototyper.zip
2. Dobbeltklikk på filen setup.exe
3. Bildet vist i Figur 37 viser bildet som blir vist på skjermen, her trykker du Next>



Figur 37 Velkomstskjermen installasjon av prototyper

4. Bildet vist i Figur 38 viser det neste steget, og her kan du velge hvor programmet skal installeres. Det er sterkt anbefalt å bruke stien som kommer opp som forslag, da det kun er dette som er testet. Velg så neste for å gå videre.



Figur 38 Valg av bane for installasjon av prototyper

5. Trykk Next> på det påfølgende skjermbildet.
6. Lukk veiviseren når den er ferdig
7. Pakk ut filen bilder.zip på c:, den endelige banen for bildene må være ”**C:\Visual Studio 2008\Projects\MasterWPF\quickSilver\img** ” for at programmet skal starte.
8. Gå til mappen du installerte programmet, standard: ” C:\Program Files\Default Company Name\InstallDigiHouse”
9. Dobbeltklikk på de følgende filene for å starte de ulike prototypene:
    a. jigsawTest.exe – Puslespill
    b. MasterWPF.exe – Objektkobling
    c. quickSilver.exe – Grafisk kommandolinje
    d. wizard.exe – Veiviser
    e. wimp.exe – WIMP