DISTRIBUERT PROGRAMUTVIKLING I ET VIRTUELT MILJØ

Design og evaluering av en prototype for Scrum i Second Life

Av

Christer Johannessen
Morten Christian Andreassen

Institutt for informasjons- og medievitenskap
Universitetet i Bergen
Høsten 2009

Fullført som del av kravet til graden
«Master i informasjonsvitenskap»

VEILEDER: FRODE GURIBYE

Abstrakt: Scrum er en metode for å organisere arbeidet til programmeringsteam. Slike team benytter ulike verktøy og møter for å kommunisere og koordinere sitt arbeidet. En fremtredende trend er Scrum team som benytter seg av distribuert arbeid. Multi User Virtual Environments (MUVE), slik som Second Life, er en plattform for å støtte distribuert arbeid.

Denne studien presenterer en analyse av hvordan et MUVE kan bli benyttet for å støtte distribuert programutvikling. Denne analysen er basert på prinsippene fra designforskning. Først beskrives en kvalitativ studie som bestod av intervju og observasjon. Scrum eksperter og profesjonelle ble intervjuet for å analysere hvordan arbeid blir organisert i programvareutvikling og hvordan et MUVE kan støtte denne prosessen. Observasjon hos et programutviklingsfirma blir deretter beskrevet. Empirien ble etter dette analysert for å hente ut krav til en prototype for Scrum, i Second Life. Prototypen ble designet på bakgrunn av kravene. Deretter ble den evaluert ved hjelp av rimelige evalueringsteknikker.

Nøkkelord: Programvareutvikling, Scrum, MUVE, Second Life, Mechanics of Collaboration, Heuristic Evaulutation, Groupware Walkthrough, Awareness, Design Science, CSCW, Common Information Spaces, Articulation Work.

1 INTRODUKSJON 1

2 TEORETISK RAMMEVERK OG METODOLOGIER FOR SYSTEMUTVIKLING 7

2.1 CSCW: SYSTEMER FOR DATASTØTTET SAMARBEID 7

2.1.1 Klassifisering av Second Life 10

2.2 COMMON INFORMATION SPACES 12

2.3 ARTICULATION WORK 13

2.4 SAMARBEIDSMEKANISMER 15

2.5 AWARENESS RAMMEVERK FOR SANNTIDS-GROUPWARE 16

2.5.1 Egenskaper for awareness 16

2.5.2 Situasjons awareness 16

2.5.3 Workspace awareness 17

2.5.4 Vedlikehold av awareness 18

2.5.5 Workspace awareness rammeverk 18

2.6 METODOLOGIER FOR SYSTEMUTVIKLING 19

2.6.1 Agile metoder 19

2.6.2 Ekstremprogrammering (XP) 21

2.6.3 Scrum 24

3 TIDLIGERE FORSKING 32

3.1 PROGRAMVAREUTVIKLING 32

3.1.1 Distribuert programvareutvikling med Scrum 32

3.1.2 Programvareutvikling ved hjelp av et MUVE 33

3.2 MUVE OG SECOND LIFE 35

3.3 POSISJONERING AV STUDIEN 36

4 FORSKNINGSTILNÆRMING OG METODEVALG 38

4.1 FORSKNINGSDESIGN 38

4.2 DESIGNFORSKNING 40

4.2.1 Designforskning på informasjonssystemer 43

4.3 KVALITATIVE METODER 45

4.4 ETIKK 46

4.5 PROTOTYPING 46

4.6 EVALUERING 47

4.6.1 Målekriterier for usability 47

4.6.2 Rimelig usability evaluering 48

5 DATAINNSAMLING OG METODE 55

5.1 KVALITATIV STUDIE 55

5.2 UTVALG 56

5.3 INTERVJU OG OBSERVASJON 58

5.3.1 Universitet i Bergen 59

5.3.2 Programutviklere 59

5.3.3 IKTBergen 59

5.4 DATAINNSAMLING 59

5.5 BEARBEIDING OG ANALYSE 60

6 RESULTATER FRA DEN KVALITATIVE UNDERSØKELSEN 63

6.1 ENDRING AV ARBEIDSPRAKSIS HOS IKTBERGEN 63

6.2 SCRUM HOS IKTBERGEN 64

6.3 GJENNOMGANG AV INTERVJUTEMA 71

6.3.1 Sosiale mekanismer og relasjoner for samarbeid 71

6.3.2 Awareness 74

6.3.3 Samlokaliserte team 76

6.3.4 Distribuerte team 79

6.3.5 Skepsis til MUVE som samarbeidsløsning ... 86
6.4 OPPSUMMERING AV HOVEDFUNN ... 88

7 PROTOTYPEDESIGN I SECOND LIFE ... 89
7.1 VALG AV RAMMEVERK ... 89
7.2 FORBEREDELSER ... 89
7.3 ANALYSE ... 91
7.4 DESIGN ... 92
7.4.1 Fra ide til designvalg ... 93
7.4.2 Tekniske løsninger for møter og samarbeid ... 94
7.4.3 Rettigheter til objekter og land ... 95
7.4.4 Design av Scrum møterom ... 96
7.4.5 Design av rom for opplæring og møter ... 98
7.4.6 Design av samarbeidsrom ... 99
7.4.7 Rom designet for sosiale aktiviteter ... 100
7.4.8 Design av rekreasjonsområde ... 101
7.4.9 Design av nettside ... 102
7.5 KODING OG VIDEREUTVIKLING ... 102
7.6 TILLEGGSMOMENTER OM DESIGNVALG ... 103
7.7 OPPSUMMERING ... 103

8 EVALUERING AV PROTOTYPE ... 104
8.1 HEURISTIC EVALUATION OF GROUPWARE ... 104
8.1.1 Heuristic 1 - Provide the means for intentional and appropriate verbal communication ... 104
8.1.2 Heuristic 2 - Provide the means for intentional and appropriate gestural communication ... 105
8.1.3 Heuristic 3 - Provide consequential communication of an individual's embodiment ... 105
8.1.4 Heuristic 4 - Provide consequential communication of shared artifacts (i.e. artifact feedthrough) ... 106
8.1.5 Heuristic 5 - Provide protection ... 107
8.1.6 Heuristic 6 - Management of tightly and loosely-coupled collaboration ... 108
8.1.7 Heuristic 7 - Allow people to coordinate their actions ... 109
8.1.8 Heuristic 8 - Facilitate finding collaborators and establishing contact ... 109
8.1.9 Oppsummering ... 110
8.2 GROUPWARE WALKTHROUGH ... 112
8.2.1 Walkthrough prosess for scenario 1 - Daglig Scrum møte ... 113
8.2.2 Walkthrough prosess for scenario 2 - Holde kurs ... 117
8.2.3 Walkthrough prosess for scenario 3 - Diskutere kildekode ... 119
8.2.4 Walkthrough prosess for scenario 4 - Parprogrammering ... 121
8.2.5 Walkthrough prosess for scenario 5 – Dele biblioteksressurs ... 124
8.2.6 Walkthrough prosess for scenario 6 - Spille spill ... 126
8.2.7 Oppsummering ... 128
8.3 OPPSUMMERING AV EVALUERING ... 129

9 KONKLUSJON ... 130
9.1 EVALUERING AV STUDIEN ... 133
9.2 VIDERE FORSKNING OG FREMTIDIGE UTFORDRINGER ... 134

10 LITTERATURLISTE ... 135

11 APPENDIKS A – KVALITATIV UNDERSØKELSE ... 142
11.1 LOG FRA DAGLIG SCRUM MØTE HOS IKTBERGEN ... 142
11.2 INTERVJUGUIDE ... 144
11.3 INTERVJUSPØRSMÅL ... 146

12 APPENDIKS B – DESIGN I SECOND LIFE 148
12.1 LAGE BRUKERKONTO OG LOGGE PÅ 148
12.2 BYGGEPROSESSEN 150
12.2.1 Inventar mappen 151
12.2.2 Bygge via objekter 152
12.2.3 Tekstiler (Textures) 155
12.2.4 Scripts 156
13 APPENDIKS C - GROUPWARE WALKTHROUGH – GROUP TASK MODEL 158
13.1 SCENARIO 1 - DAGLIG SCRUM MØTE 158
13.2 SCENARIO 2 - HOLDE KURS 162
13.3 SCENARIO 3 - DISKUTERE KILDEKODE 165
13.4 SCENARIO 4 – PARPROGRAMMERING 167
13.5 SCENARIO 5 – DELE BIBLIOTEKSRESSURS 169
13.6 SCENARIO 6 - SPILLE SPILL 170

Liste med Figurer

Figur 1 – Domene og samarbeidsoppgaver (Gutwin & Greenberg 2002, s. 418) 17
Figur 2 – ” The perception-action cycle” (Gutwin & Greenberg 2002, s. 418 fra; Neisser 1976) 18
Figur 3 - XP livssyklus (Abrahamsson, Salo, Ronkainen & Warsta 2002, s. 19) 22
Figur 4 - Illustrasjon av Scrum prosessen (Abrahamsson, Salo, Ronkainen & Warsta 2002, s. 28) 26
Figur 5 - Praksis og input til en sprint (Abrahamsson, Salo, Ronkainen & Warsta 2002, s. 33) 30
Figur 6 - Studiens forskningsdesign 40
Figur 7 - Groupware Walkthrough oversikt, basert på (Gutwin & Pinelle 2002) 50
Figur 8 – Arbeidsflytmodell 50
Figur 9 - Hierarkisk inndeling av oppgavene, oversatt fra Gutwin & Pinelle (2002, s. 457) 51
Figur 10 – Analysediagram av scenarioet diskutere kildekode 53
Figur 11 - Analyseskisse som utgangspunkt for utarbeidelse av intervjueguide 61
Figur 12 - Åpent kontorlandskap hos IKTBergen 69
Figur 13 - Statustavle i inngangspartiet hos IKTBergen. 70
Figur 15 - Fussballbord hos IKTBergen 71
Figur 14 - Fussballstatistikk hos IKTBergen 71
Figur 16 – Eksempel på manuell Scrum Backlog liste 78
Figur 17 - Analyse-design-kode-test modell, etter Hoffer, George og Valacich (2005, s.16) 89
Figur 18 – Prototypens arkitektur 94
Figur 19 – The Generate/Test Cycle, oversatt til norsk fra Hevner, March, Park & Ram (2004, s. 89) 94
Figur 20 – Scrum møterom. 97
Figur 21 – Product Backlog 97
Figur 22 – Rom for opplæring og møter 98
Figur 23 – Ressursbibliotek. 99
Figur 24 – Samarbeidsrom 100
Figur 25 – Rom for sosiale aktiviteter 101
Figur 26 – Rekreasjonsområde. 101
Figur 27 – Sittegruppe 102
Figur 28 – Scenario walkthrough av daglig Scrum møte 114
Figur 29 – Scenario walkthrough av å holde et kurs 117
Figur 30 - Startsiden til Second Life 148
Figur 31 – Opprette avatar 149
Figur 32 - Velge startområde 150
Figur 33 - Inventarmappen 151
Figur 34 - Lage objekt 152
Figur 35 - Strekke og tilpasse objekt 153
Figur 36 - Rotere objekt 153
Figur 37 - Lage bordben 153
Figur 38 - Plassere bordben 153
Figur 39 - Linke objektene 154
Figur 40 - Velge alle objektene 154
Figur 41 - Det ferdigstilte bordet 155
Figur 42 - Legge på tekstil og farge på objekt 156

Figur 43 – Stegene i groupware walkthrough, oversatt til norsk fra Gutwin & Pinelle (2002, s. 458) 158
Figur 44 – Analysediagram for scenario 1: Holde møte, del 1 161
Figur 45 – Analysediagram for scenario 1: Holde møte, del 2 162
Figur 46 – Analysediagram for scenario 2: Holde kurs 164
Figur 47 – Analysediagram for scenario 3: Diskutere kildekode 166
Figur 48 – Analysediagram for scenario 4: Parprogrammering..... 168
Figur 49 – Analysediagram for scenario 5: Dele biblioteksressurs 170
Figur 50 – Analysediagram for scenario 6: Spille spill 171

Liste med tabeller

Tabell 1 – Second Life i en Tid/Sted matrise..... 11
Tabell 2 – Samarbeidsmekanismene som tar sted i samarbeid. Vår oversettelse fra Gutwin og Greenberg (2000, s. 99-100) 15
Tabell 3 – Fremgangsmåter og teknikker i ekstremprogrammering..... 23
Tabell 4 – Retningslinjer for designforskning (vår oversettelse fra Hevner, March, Park & Ram 2004, s.83)..... 41
Tabell 5 – Scenariospesifikasjon (Gutwin & Pinelle 2002)..... 51
Tabell 6 – Informantoversikt..... 57
Tabell 7 – Tema for intervju 62
Tabell 8 – Krav fra den kvalitativ undersøkelsen 92
Tabell 9 – Elementer for workspace awareness i relasjon til nåtid (Tabell fra Gutwin & Greenberg 2002 oppimot denne studiens empiri)..... 111
Tabell 10 – Elementer for workspace awareness i relasjon til fortid (Tabell fra Gutwin & Greenberg 2002 oppimot denne studiens empiri)..... 112
Tabell 11 – I hvilken grad samarbeidsmekanismene er støttet i Prototypen..... 128
Tabell 12 - Scenariobeskrivelse for daglig Scrum møte 159
Tabell 13 - Scenariobeskrivelse for å holde et kurs 163
Tabell 14 - Scenariobeskrivelse for å diskutere kildekode 165
Tabell 15 - Scenariobeskrivelse for parprogrammering..... 167
Tabell 16 - Scenariobeskrivelse for å dele biblioteksressurs 169
Tabell 17 – Scenariobeskrivelse for å spille et spill..... 171

Forord

Det å skrive en masteroppgave innen datastøttet samarbeid har vært spennende og utfordrene. Dette fordi fagområdet grenser opp til flere andre fagfelt. Dermed blir det naturlig å se til disse når man gjør en studie innen datastøttet samarbeid.

Først og fremst vil vi takke vår veileder, Frode Guribye, for konstruktive tilbakemeldinger gjennom hele prosessen. Han har stilt kritiske spørsmål og gitt tilbakemeldinger for å få oss til å bli bedre.

Det å være to personer som samarbeider om en slik studie er både motiverende og utfordrene. Mye skal koordineres og diskuteres for å komme fram til et endelig produkt som begge to kan stå inne for. Det hadde ikke vært mulig å gjennomføre denne studien uten hjelp fra eksterne ressurser. Vi vil med dette sende en stor takk til IKT firmaet i bergensområdet som lot oss observere deres arbeidspraksis. Videre vil vi også takke alle som har stilt opp til intervju, og evaluererne som har testet prototypen utviklet i denne studien.

Avslutningsvis vil vi takke dem som har motivert oss videre. Sist men ikke minst vår nærmeste familie som har latt oss få lov til å bruke så mye tid på å skrive masteroppgave. Uten deres støtte hadde vi ikke kommet i mål.

1 Introduksjon

Denne studien fokuserer på hvordan programvareutvikling kan sees på som Cooperative Work, og hvordan en slik praksis kan støttes i et distribuert miljø ved hjelp av Computer Support. For å svare på dette innehar studiens forskningsdesign, basert på prinsippene fra designforskning, to forskningsspørsmål. I det første blir det spurt om hvordan distribuerte programutviklingsteam samarbeider, og hvilke verktøy de benytter i sin arbeidspraksis. Dette blir besvart ved å foreta en kvalitativ undersøkelse. Den blir benyttet for å svare på det andre forskningsspørsmålet som søker å få klarhet i hvordan en slik praksis kan støttes av et MUVE. Resultatet er en groupware prototype som blir evaluert i henhold til kravene som blir innhentet gjennom den kvalitative undersøkelsen.

Programvareutvikling har blitt viktig, ved at man er avhengig av velfungerende systemer for å understøtte dagens arbeidspraksis. IKT systemer befinner seg på alle nivå, fra småbedrifter til multinasjonale selskaper, og knytter disse sammen ved hjelp av kommunikasjonsløsninger. Programvareutvikling har tradisjonelt benyttet en prosesstyrt ingeniørpraksis for å utvikle programmer, men i følge Dittrich, Randall og Singer (2009) begynte noe å endre seg på 1990-tallet.Dette kom som en reaksjon på mislykkede programvareutviklingsprosjekt som kostet milliarder av dollar. På bakgrunn av dette innså man at den rigide, lite smidige, ingeniørpraksisen var en mulig årsak.

Dittrich, Randall og Singer (2009) forklarer at kvalitative metoder som arbeidsstedsstudier, etnografiske inspirerte undersøkelser og intervju blir benyttet for å forbedre prosessene. Med andre ord benyttes CSCW studier for å bistå programvareutviklere. For å forstå designsiden av CSCW må man forstå hvordan man samarbeider når man designer og utvikler programvare (Suchman 1994). Man har med andre ord studier av, og studier for programvareutvikling. Dette fagområdet er av interesse for CSCW forskning fordi programvareutvikling er Cooperative Work (Dittrich, Randall & Singer 2009).

Innenfor programvareutvikling har det oppstått en ny trend som svar på utfordringene med en rigid ingeniørpraksis. Denne trenden går under samlekategorien agile

metoder, der Scrum og ekstremprogrammering inngår. Metodene beskriver hvordan man organiserer små team og arbeidsgrupper, basert på best practice[1], med det mål å kunne ta raske beslutninger. I motsetning til ingeniørteknikkene ligger ikke fokuset her på tunge prosesser og mengder med informasjon.

Det er bred enighet om at ulike prosjekter trenger forskjellige fremgangsmåter. Man må derfor benytte de metoder, teknikker og prosesser som er best egnet i henhold til omgivelsene og omstendighetene (Dittrich, Randall & Singer 2009). Tradisjonelle utviklingsløp, som benytter en hierarkisk fossefallsmetode, kan være best egnet når man skal designe programvare hvor kravene er kartlagt og definert før prosjektet starter, og hvor tilpasningsdyktighet ikke er et tema. Derimot vil mer tilpasningsdyktige og smidige metoder være foretrukket når man skal utvikle programvare i samarbeid med kunder, hvor behov og krav ikke lar seg definere så lett før prosjektet starter. Forskningsstudier har forsøkt å forstå de nye fremgangsmåtene for programvareutvikling, ved å fokusere på kommunikasjon og samarbeid, fremfor (kvantitativ) kontroll. Kvalitative metoder har blitt benyttet som de foretrukne forskningsmetodene for å forstå denne overgangen (Dittrich, Randall & Singer 2009).

Norge er et land med stor geografisk spredning. Større prosjekter som skal utvikle programvare kan dermed ha behov for å benytte ressurser som befinner seg på ulike steder. Bjerrum og Bødker (2003) forteller at utvikling av office teknologi ”[…] by structuring and enabling communication, have made distributed organisations possible” (s. 202). Det er dermed slik at prosjektmedlemmene ikke nødvendigvis befinner seg på samme sted. På grunn av høye reisekostnader, og trange marginer, er det ønskelig å samarbeide distribuert. Dette har blitt enda mer aktuelt i dagens globale marked med multinasjonale selskaper og aktiv bruk av utkontraktering av programvareutvikling til lavkostland. Dette skaper nye utfordringer for programvareutviklingsprosessen, hvor man trenger nye metoder og verktøy, som igjen har resultert i ny forskning. Denne forskningen går under betegnelsen Globaly Distributed Software Development (GSD) og har ofte til hensikt å støtte distribuert utvikling på tvers av tid og sted. Omoronyia, Ferguson, Roper og Wood (2009) forklarer at datastøttede verktøy blir bindeleddet mellom personer når man arbeider

1 Håkon Lorentzen forklarer at best practice er ”forstått som inspirasjon fra de som har fått noe til” (2007, s. 71)

distribuert ved at verktøyene innehar støtte for awareness. I denne studien blir virtuell virkelighet en plattform for å støtte GSD.

Virtuell virkelighet er teknologi som ble utviklet på 1960-tallet, og har gjennomgått flere utviklingsfaser. Teknologien har sitt utspring i forskning på flysimulatorer for det amerikanske forsvaret, og forskningsarbeidet til professor Ivan Edward Sutherland. Han lagde "Sword of Damocles", det første systemet for virtuell virkelighet i 1968. På den tid arbeidet han på MITs Lincoln forskningslab, og var leder for (D)ARPAs informasjonsprosesserings teknologikontor.

Denne teknologien var i starten forbeholdt forskningslaboratorier og det militære, men på 1980-tallet ble det utviklet løsninger med hansker og briller, i form av virtuelle spill, som man kunne benytte på kjøpesentre. Ulempen var at man ikke hadde mulighet for at flere kunne delta samtidig, og utstyret som ble benyttet var både tungt og klumpete. Dette har endret seg, og har i de senere år blitt en plattform for distribuert samarbeid, som gjerne betegnes som Multi User Virtual Environment (MUVE), fordi flere nå kan benytte den. Denne teknologiske utviklingen har vært mulig fordi man har kunnet benytte Internett og de kommunikasjonsmulighetene som finnes der.

Salem og Earle (2000) beskriver MUVE som et databasert visualisert miljø, hvor det er ment at brukere skal interagere med hverandre ved hjelp av avatarer. En avatar er en tredimensjonal manifestasjon av en bruker som befinner seg i et virtuelt miljø. Avataren bør være synlig for sin egen bruker, samt andre brukere av systemet, og er essensiell for å representere en brukers tilstedeværelse, orienteringsevne og plassering i et MUVE. Jonathan Samuel Steuer (1995) forklarer at deltagelse i et MUVE kan gi brukeren inntrykk av å befinne seg på et annet sted. Dette kan sammenlignes med "telepresence". Dette er en teknologi som Scott Fisher (1990) mener lar fjernarbeidere få den sanseinformasjonen de trenger for å føle at de virkelig er tilstede på den fjerne lokasjonen og at de kan utføre ulike oppgaver der. Når man samhandler via et MUVE er forskjellen at man samhandler via en datamaskin inn i et simulert miljø, mens man i telepresence kobler seg til et fysisk miljø.

Victor Kaptelinin (2003) argumenterer for at et system som skal støtte datastøttet samarbeid må inneha funksjoner for å kommunisere med de øvrige brukerne, og i tillegg la brukerne fokusere på arbeidsoppgaven de skal utføre, og ikke på bruken av systemet. MUVE er i hovedsak ment for å støtte samarbeid på samme tid på ulik lokasjon. Det medfører at de har samme form for tekstlig interaksjon som man har ved "Instant Messaging"[2] (IM). Nardi, Whittaker og Bradner redegjør for hvordan man kan benytte IM for formell og uformell kommunikasjon (2000). I tillegg innehar teknologien noen aspekter man ikke har i IM, eksempelvis verbale samtaler via "Voice over IP" (VoIP) teknologi, som er beskrevet i Kristin Mitchell (2009) sin artikkel om MUVEet Second Life, og intensjonell gestikulering gjennom brukernes avatarer. VoIP er en metode for å ta analoge audiosignaler, slik man hører når man snakker i telefonen, og forvandle dem til digitale data som kan bli overført via Internett.

Olivier og Pinkwart (2007) trekker frem at enkelte forskere hevder at MUVE som Second Life er et av de viktigste fremskrittene til Internett. De stiller spørsmål ved om MUVE er hype eller håp for CSCW, og kommer frem til at MUVE har et stort potensial som en arena for CSCW løsninger. De poengterer at det like fullt er behov for mer forskning for å forstå styrken og svakhetene til avatarer i 3D verdener sett i lys av CSCW (Olivier & Pinkwart 2007).

Dieterle & Clarke (2009) antar at teknologien vil endre grensesnittet mellom mennesker og informasjonsteknologi ved å tilby nye måter å utveksle informasjon, visualisere prosesser, og muliggjøre nye måter for kunstnerisk utfoldelse. MUVEs "[…] have become a major force, shaping how we communicate, participate, learn, and identify ourselves" (Dieterle & Clarke 2009, s. 7).

Et eksempel på et MUVE er SUN Microsystems sin satsning på et virtuelt arbeidsmiljø, MPK20: Project Darkstar. Dette prosjektet er beskrevet i artiklene til Jim Waldo (2008) og Nicole Yankelovich (2007), hvor de forteller at Sun så behovet for å ha et kontorlandskap i en virtuell verden for å kunne knytte sammen ansatte på ulike lokasjoner i et felles arbeidsmiljø. Prosjektet hadde som mål å skape et delt

2 Nor.: Lynmelding

workspace[3] hvor de ansatte kunne gjøre alt sitt arbeid. Det var ikke meningen at dette skulle være noe en logger på kun for å ha et møte, men skulle være verktøyet de benyttet i sin vanlige arbeidsdag. Dette lar seg gjøre ved at ansatte kan dele sitt workspace i den virtuelle verdenen med kolleger og la dem benytte tastatur og mus på andres maskiner for å jobbe i felles programmer.
I denne studien har det blitt sett på hvordan distribuerte Scrum team samarbeider, og hvilke verktøy de benytter i eksisterende praksis. Dette har blitt gjennomført ved hjelp av en kvalitativ studie, som har bestått av intervju og observasjon. I løpet av intervjuene ble det vist et videoklipp av MPK20 for å demonstrere hvordan et MUVE fungerer. Her ble det samlet inn empiri om dagens arbeidspraksis til et programutviklingsfirma i bergensregionen. I tillegg ble det intervjuet personer fra Universitetet i Bergen og programmerere fra andre firma. Dette er altså i tråd med den foretrukne måten å forstå arbeidspraksisen til programutviklere som benytter agile metoder. Groupware Walkthrough, fra Gutwin og Pinelle (2002), har blitt benyttet for å identifisere og modellere samhandling i eksisterende praksis (se avsnitt 4.6.2.2). I modellene har samhandlingen blitt brutt ned i sine enkelte mekanismer, i henhold til artikkelen ”The Mechanics of collaboration” av Gutwin og Greenberg (2000). På bakgrunn av den kvalitative undersøkelsen har det blitt sett på hvordan denne arbeidspraksisen kan bli støttet av et MUVE.

Videre har studien satt dette i praksis ved å utvikle et virtuelt workspace som kan forbedre og forenkle den eksisterende måten å drive distribuert programutvikling. Second Life har blitt valgt som plattform for prototypen fordi denne tilfredstilte kravene til tilgjengelighet, kostnad, og tilpassningsmuligheter som kom frem som behov i den kvalitative undersøkelsen.

Second Life har vært åpen for brukere siden 2003. Rymaszewski, Rosedale, Batstone-Cunningham, Ondrejka, Winters, Wallace og Au (2008) forklarer at innholdet i Second Life har blitt skapt av brukerne ved at det ble lagt til rette for at brukerne selv kan bygge objekter. Dette kan ha gjort det vanskelig for de første brukerne, men ga samtidig store muligheter for pionerene, spesielt med tanke på at Second Life har en valuta med ekte markedsverdi (Linden Labs 2009a).

3 Nor.: Arbeidsflate

Ettersom Second Life legger opp til at brukerne selv kan gjøre hva de vil med applikasjonen er det mulig å tilrettelegge for brukstilfeller innen ulike fagområder, slik som design (virtuell prototyping), utdanning og arkitektsamarbeid. Her benytter man seg av Linden Scripting Language for å programmere funksjonalitet til objekter (Heaton 2007). Second Life har standardisert på et åpent grensesnitt som har åpnet opp for at ulike miljø kan lage sin egen klient. Dette kan gjøre det mulig for programmerere å skreddersy en løsning som kombinerer programutviklingsverktøy og Second Life i samme grensesnitt.

Prototypen ble til slutt evaluert for å vurdere om prototypen tilfredsstilte de krav som kom frem i den kvalitative undersøkelsen. Evalueringen bestod av to rimelige evalueringsteknikker, hvor Workspace Awareness rammeverket til Gutwin og Greenberg (2002) ble lagt til grunn. Først ble heuristikker[4] basert på "Heuristic Evaluation of Groupware based on the Mechanics of collaboration", utarbeidet av Baker, Greenberg og Gutwin (2001), benyttet for å fange opp brukervennlighetsproblemer. Deretter ble gjennomgangsprosessen til Groupware Walkthrough utført. Ifølge Gutwin og Pinelle (2002) har evalueringsmetoden til hensikt å guide testerne i å følge arbeidsoppgavene og evaluere groupware[5] grensesnittet i et testmiljø, for å få frem usability[6] problemene til prototypen. Testmiljøet ble tilrettelagt i henhold til de retningslinjene gitt i designforsking, som beskrevet av Hevner, March, Park og Ram (2004).

I neste kapittel presenteres teori som blir benyttet i studien, etterfulgt av en presentasjon av tidligere forskning. I kapittel 4 blir forskningsspørsmålene og fremgangsmåten som har blitt benyttet for å svare på disse utdypet. I det påfølgende kapittelet blir det forklart hvordan den kvalitative undersøkelsen har blitt gjennomført, og hvordan empirien har blitt analysert. I kapittel 6 blir funnene fra undersøkelsen presentert. På bakgrunn av disse ble det laget en kravtabell til en prototype, som presenteres i kapittel 7. Videre i dette kapittelet blir det beskrevet hvilke designvalg som ble tatt med i utviklingen av denne. I kapittel 8 blir prototypen evaluert. Til slutt blir studiens resultater oppsummert i kapittel 9.

4 Eng.: **Heuristic** is an adjective for experience-based techniques that help in problem solving, learning and discovery (oppsummert på bakgrunn av Baker, Greenberg & Gutwin 2001)
5 Nor.: Sammarbeidssystem
6 Nor.: Brukskvalitet

2 Teoretisk rammeverk og metodologier for systemutvikling

Dette kapittelet redegjør for det konseptuelle rammeverket, samt metodologiene for systemutvikling som blir benyttet i studien. Førstnevnte består av CSCW, common information spaces, articulation work, samarbeidsmekanismer som benyttes for å samarbeide i et delt workspace, og workspace awareness rammeverket til Gutwin & Greenberg (2002). De agile metodologiene Scrum og ekstremprogrammering (XP), som setter viktige føringer for arbeidskonteksten, blir til slutt gjennomgått. Førstnevnte blir ofte benyttet som styringsverktøy for programmeringsteam, mens man gjerne henter konkrete teknikker i fra XP.

2.1 CSCW: Systemer for datastøttet samarbeid

Irene Greif (1988), en av opphavspersonene til begrepet Computer Supported Cooperative Work, forteller at ”over the last half dozen years, Computer Supported Cooperative Work has emerged as an identifiable research field focused on the role of the computer in group work” (s. 5). Bannon og Schmidt (1989) forklarer at dette har skjedd fordi forskningsområde er definert på bakgrunn av en problemsituasjon, slik at forskere har klart å bli enig ”on the nature of the uncertainty common to a set of problem situations” (Bannon & Schmidt 1989, s. 2). Det blir videre forklart at CSCW ikke kan bli definert på bakgrunn av teknikken som blir benyttet, men at det som forener CSCW som forskningsfelt er ”[…] the support requirements of cooperative work” (Bannon & Schmidt 1989, s. 3). Bannon og Schmidt uttaler at CSCW bør bli forestilt som en felles innsats for å forstå naturen og karakteristikkene av samarbeid. Formålet vil være å tilegne seg nok kunnskap, slik at man kan designe datamaskinbaserte teknologier, som kan støtte samarbeid på en hensiktsmessig måte. Fokuset er altså på å *forstå*, slik at en bedre kan *støtte*, samarbeid (oversatt til norsk, fra Bannon & Schmidt 1989, s. 3).

Motivene for å benytte CSCW kan være mange, men ofte ønsker man å forsterke det eksisterende samarbeidet. Ifølge Bannon og Schmidt (1989, s. 2-3) er det likevel essensielt at det ikke fokuseres utelukkende på å støtte aktiviteten ved hjelp av teknologi, men at det fokuseres like mye på å hindre at bruken av datamaskiner ikke ødelegger eller forstyrrer den eksisterende aktiviteten. For at innføring av datastøtte til den eksisterende praksisen skal gi positive resultater, må en god forståelse av

arbeidskonteksten ligge til grunn. I denne studien blir det som nevnt fokusert på programutvikling, hvor arbeidet i all hovedsak foregår ved hjelp av datamaskiner. Observasjon av et slikt miljø blir presentert i avsnitt 6.2.

CSCW som forskningsfelt er i kontinuerlig utvikling, og begrepet groupware viser til de tekniske systemene som resulterer fra denne forskningen. Bannon og Schmidt (1989) stiller seg spørsmålet om man kan definere groupware som programvare som støtter grupper av mennesker som jobber mot felles mål. I artikkelen kritiserer de tanken om å ha en enkelt definisjon på groupware. De mener at å avgrense groupware begrepet til grupper av mennesker som jobber mot et felles mål, gjør at denne definisjonen blir mindre omfattende enn CSCW (i hvert fall om man snakker om groupware versus CSCW som forskningsfelt). En gruppe mennesker kan sies å være et relativt lukket og fastsatt fellesskap, som deler 'mål', og som kommuniserer med hverandre direkte og ofte. Selv om man løser opp definisjonen av en gruppe, og sier at et individ er medlem i en gruppe dersom det tenker på seg selv som del av et "vi", bør ikke samarbeid være begrenset på denne måten. Tanken om gruppearbeid dekker ikke den rike og komplekse virkeligheten, og dermed bør man benytte begrepet cooperativ work, som er "[…] the general and neutral designation of multiple persons working together to produce a product or service" (Bannon & Schmidt 1989, s. 5). Her har man gjerne med mange aktører som ikke nødvendigvis kan karakteriseres som en fast gruppe, og som heller ikke nødvendigvis har et klart definert felles mål. Gruppearbeid er for eksempel ikke den typen samarbeid som man har i moderne fabrikker (Bannon & Schmidt 1989, s. 361-363). I andre arbeidsdomener arbeider man på avstand, uten:

> "[…] direct communication and without necessarily knowing each other or knowing of each other, via a more or less shared *information space*, that is, a 'space' comprising data, personal beliefs, shared concepts, professional heuristics etc" (Bannon & Schmidt 1989, s. 361).

I avsnitt 2.2 vil vi komme nærmere innpå hva et information space innebærer, og hvordan det påvirker denne studien.

Bannon og Schmidt (1989) forteller videre at:

> ”[…] we reject the equation of Groupware with CSCW because of its technological focus and its narrowness in the face of the multiplicity of social forms of cooperative work manifest in the world” (s. 6).

Forholdet mellom begrepene er imidlertid ikke særlig turbulent i dag. CSCW har blitt det foretrukne begrepet i forskningsmiljø, selv om mange kommersielle rapporter fremdeles refererer til groupware som et forskningsfelt (Schmidt & Bannon 1992, s. 9-10). Irene Greif (1988) forsto tidlig at fremtidens applikasjoner ville ha innebygget samarbeidsfunksjonalitet og dermed være groupware til en viss grad. Likevel hevder hun at CSCW som forskningsfelt vil bestå fordi det tar opp bredere spørsmål om design og forbedring av groupware (Greif 1988, s. 12). Selv om flere systemer vil bli betegnet som groupware, betyr ikke det at man ikke fremdeles har behov for ”core CSCW and groupware research and for the definition of basic features or characteristics of collaboration support tools” (Koch & Gross 2006, s. 166). Sammenhengen kan forklares enklere ved å tenke på CSCW som forskningsfeltet, og groupware som de tekniske systemene som er utviklet ut i fra teori fra forskningen (Koch & Gross 2006, s. 2).

Prototypen i denne studien kan karakteriseres som groupware. For å få frem hvilke behov det var til en slik løsning har det blitt gjennomført en kvalitativ undersøkelse for å forstå hvordan programutviklere samarbeider. Ved å tilby et grensesnitt til et shared workspace har målet med løsningen vært å støtte et team som jobber med et felles prosjekt (eller mål). Nå er det imidlertid ikke slik at samtlige brukere innenfor dette problemområdet nødvendigvis har de samme oppgavene eller mål. Dette fordi deltagerne fra arbeidskonteksten er programutviklere, testere, dokumentasjonsspesialister, brukerstøttepersonell, ledelse, selgere, kunder (interne og eksterne) og andre interessenter. Programmerere koder, testerne tester kvaliteten på koden, funksjonaliteten dokumenteres, kunden vurderer om funksjonalitet er i henhold til krav, og så videre. Ulikhetene innad i de forskjellige rollene kan bedre forstås dersom man skiller mellom rollenes underordnede og overordnede mål. Et underordnet mål kan, for eksempel, være programmererens utvikling på bakgrunn av et krav fra kunden, eller at resultatet fra utviklingen blir godt dokumentert. Selv om

de underordnede målene skiller seg fra hverandre, kan man hevde at begge jobber mot et felles overordnet mål - å ferdigstille en applikasjon.

Groupware begrepet og forskningsfeltet CSCW har vært av betydning for denne studien. Førstnevnte bidrar til å skildre applikasjonen som har blitt utviklet, og CSCW teori presiserer viktigheten av å forstå arbeidskonteksten man ønsker å støtte. Senere i studien blir groupware begrepet benyttet i forbindelse med evaluering av prototypen, hvor heuristisk evaluering (Baker, Greenberg & Gutwin 2001) og Groupware Walkthrough (Gutwin & Pinelle 2002) har blitt benyttet. Dette blir belyst i avsnitt 4.6.2.1 og 4.6.2.2.

2.1.1 Klassifisering av Second Life

Når man skal analysere systemer for CSCW (se Grudin 1994) er det to dimensjoner som fremstår som viktig; tid og sted. Man må ta hensyn til om systemene er designet for å støtte ansikt til ansikt samhandling eller ment for brukere som er spredd over ulike steder. I tillegg kan groupware være designet for å støtte kommunikasjon og samarbeid ved sanntidsinteraksjon, synkron, eller ikke sanntids, asynkron, interaksjon. Tabell 1 illustrerer hvordan disse dimensjonene og kombinasjonene av dem passer inn i Second Life. Dette har blitt gjort på bakgrunn av Tid/Sted matrisen til Ellis, Gibbs og Rein (1991), og Oliver og Pinkwarts (2007) tabell over ”Computer Mediated Communication” i Second Life, samt våre observasjoner. I Second Life er det primært skillet mellom offline/online som er viktig. Videre er det slik at systemet tilbyr de samme funksjonene uavhengig av om brukerne befinner seg på samme eller ulikt sted. De samarbeidsmekanismene og verktøyene brukerne har behov for når de arbeider sammen på samme sted, må likevel få virtuelle substitutter hvis Second Life skal støtte cooperative work på en god måte (se tabell 1).

Tabell 1 – Second Life i en Tid/Sted matrise

	Samme tid synkron	**Ulik tid** asynkron
Samme sted	**Ansikt til ansikt interaksjon** Brukere møtes i Second Life ved hjelp av avatarer og befinner seg samtidig på samme fysiske sted. Det er ikke innebygget støtte for beslutningsrom, delte bord, tavler, gule lapper, med mer, som man benytter når man samarbeider på samme sted.	**Kontinuerlige oppgaver** Det er ingen store fordeler med å arbeide i Second Life på samme sted til ulik tid. Dette fordi det mangler innebygget støtte for grupperom, prosjektstyringsverktøy, med mer. Brukere kan utføre en oppgave i Second Life over en lenger periode, ved at objekter som lages forblir der, og man kan jobbe videre på samme objekt over en lengre periode.
Ulikt sted	**Stedsuavhengig interaksjon** Brukere kan møte hverandre som avatarer i Second Life uavhengig av sted. Her er det god og fleksibel støtte, og Second Life forsøker å skape co-locatedness når man møtes med avatarer. Det er her styrken til Second Life befinner seg, ved at den dekker behov for videokonferanse, IM, delte skjermer, telefoni, med mer.	**Kommunikasjon og koordinasjon** Second Life har flere asynkrone kommunikasjonsmidler som kan benyttes for å koordinere arbeidet når man befinner seg på ulikt sted til ulik tid. Dette er notat, offline IM som blir sendt som e-post, objekt informasjon, med mer. Brukere kan utføre en oppgave i Second Life over en lenger periode, ved at objekter som lages forblir der, og man kan jobbe videre på samme objekt over en lengre periode, uavhengig av hvor man befinner seg.

Second Life er i hovedsak konstruert for at mennesker skal interagere i sanntid. Dette fører til at mesteparten av mulighetene i Second Life faller inn under de kvadrantene som omhandler synkront samarbeid, slik som vist i tabell 1. Som modellen viser er det mulig å finne eksempler som også støtter asynkront samarbeid, ved at man kan legge igjen informasjon og objekter, men det er ikke her styrken til Second Life ligger. Det har ikke blitt identifisert mange bruksscenarioer som omhandler asynkront samarbeid på samme sted hvor det virker hensiktsmessig å benytte Second Life, men dette kommer igjen an på hvor bredt man definerer samme sted. Ettersom Second Life er et distribuert system, er ikke stedvariabelen like utslagsgivende for bruksområdet i forhold til andre teknologier, eksempelvis en tavle som krever at alle brukerne befinner seg på samme sted. Når to brukere, på ulik fysisk sted, er representert i en virtuell verden til samme tid, kan det gi en illusjon av at de er tilstede på samme sted og tid, slik illustrert i artikkelen til Olivier & Pinkwart (2007). Second Life er en applikasjon som etterstreber å støtte eksisterende arbeidspraksis, og det vil det dermed være nødvendig for applikasjonen å inneha funksjonalitet som lar brukerne samhandle

gjennom avatarene, som om brukerne befant seg på samme fysiske sted. Metodologiene for systemutvikling Scrum og ekstremprogrammering, som det blir sett på i denne studien benytter seg av flere slike teknikker, for eksempel det å feste gule lapper på en vegg for å skissere en prosess. Selv om det ikke er innebygget støtte for dette, kan man scripte og bygge verktøy i Second Life som emulerer dette. Studien benytter Second Life sin innebygde støtte for samarbeid, og har hatt til hensikt å utvide disse ved å lage støtte for virtuelle versjoner av beslutningsrom, tavler, gule lapper, med mer, som man benytter når man befinner seg på samme fysiske sted.

2.2 Common Information Spaces

Innenfor fagfeltet CSCW finnes det ulike begreper for å beskrive hvordan flere samarbeidende aktører anvender et felles distribuert system for å utføre både individuelle og samarbeidende aktiviteter. Denne studien tar i bruk begrepet Common Information Spaces (CIS) fordi dette har, ifølge Liam Bannon (2000), blitt spesifisert på en slik måte at man kan benytte det som et konseptuelt rammeverk innenfor CSCW. Bannon forklarer at det finnes andre begreper som forsøker å beskrive det samme fenomenet - shared workspaces, shared information spaces og common communication spaces (2000). Schmidt og Bannon (1992) forteller at fokuset i CIS er på hvordan mennesker i en distribuert setting kan samarbeide i et common information space. Med andre ord vil dette si at man vedlikeholder et sentralt arkiv med organisasjonsinformasjon, med et visst nivå av delt forståelse om meningsinnholdet til informasjonen (som oppstår lokalt), selv om det er klare forskjeller når det gjelder opphav og kontekst for informasjonselementene (Schmidt & Bannon 1992, s. 16).

Det er altså viktig at det tilrettelegges for at samarbeidsdeltagerne har mulighet til å oppnå en felles enighet om meningen til informasjonselementene. Dette kan blant annet gjøres ved å skildre produsenten eller opphavet til informasjonen, samt konteksten der de ulike elementene kommer fra.

I tillegg er det lett å tenke at det å tilby teknologi som muliggjør deling av informasjon er tilstrekkelig, men Schmidt og Bannon forklarer at:

> ”Cooperative work is not facilitated simply by the provision of a shared database, but requires the active construction by the participants of a common

> information space where the meanings of the shared objects are debated and resolved, at least locally and temporarily" (1992, s. 20).

Det å skape en felles forståelse for informasjonen som er tilgjengelig i aktørenes felles arbeidsområde, er en av de største utfordringene med CIS ved bruk av datamaskiner. Dette gjelder spesielt dersom informasjonen er felles mellom flere mennesker med ulik sjargong[7], bakgrunn, og kompetanse (Bannon & Bødker 1997, s. 4). Hvis man i tillegg kommer fra ulike kulturer, noe som er tilfelle når en benytter arbeidskraft fra utlandet, er det spesielt viktig å være på vakt mot ulik tolkning av de samme informasjonselementene. Programmerere har gjerne en annen sjargong enn kunden, og forståelsen av de felles informasjonsobjektene vil ikke nødvendigvis være lik mellom dem. Det samme gjelder salg og ledelse hos et programutviklingsfirma, fordi de har en annen bakgrunn og kompetanse enn programmererne. Denne problematikken kom frem i den kvalitative undersøkelsen som ble gjennomført i denne studien. Dette er like fullt hverdagen i programutviklingsbransjen, og det antas at dette ikke vil by på betydelige utfordringer. I avsnitt 7.4 blir denne tråden tatt opp igjen. Et MUVE kan karakteriseres som et CIS i denne studien.

2.3 Articulation work

Et samarbeid[8] tar til når to eller flere aktører skal løse en oppgave eller oppnå et mål i fellesskap (Nurminen, Fjuk & Smørdal 1997, s. 1). Anselm Strauss (1985) påpeker at prosjekter innehar en form for handling som innebærer en form for arbeidsdeling - både mellom aktører og handlinger. Når man skal samarbeide med andre er det altså viktig at man definerer hva som skal gjøres, hvem som skal gjøre hva, hvordan arbeidet skal utføres, når man skal gjøre det, og hvor de skal gjøre det. Articulation work er et konsept som omhandler denne koordinasjonen av oppgaver mellom aktørene som inngår i et samarbeid. Dette konseptet ble introdusert av Anselm Strauss (1985; Strauss, Fagerhaugh, Suczek & Wiener 1985; 1993; Strauss & Corbin 1993), som et analytisk rammeverk for å forstå og utforske "the interwoven nature of mutually dependent actions of collaborating actors" (Nurminen, Fjuk & Smørdal 1997, s. 1).

7 Hverdagsspråklig uttrykksform som er særegen for en viss gruppe mennesker
8 Eng.: Cooperative Work

Gerson og Star (1986) studerte hvordan informasjonssystemer kan benytte articulation work, og forklarer at artikulasjon består av "all the tasks involved in assembling, scheduling, monitoring, and coordinating all the steps necessary to complete a production task" (s. 266). Dette betyr med andre ord at man må fullføre det en har satt seg fôre, uavhengig av om det skulle inntreffe noen uforutsette muligheter og eventualiteter. Dette kan være at det oppstår feil man ikke forutså, det oppstår konflikt om hva som er den beste måten å utføre oppgaven, eller at en har ufullstendig kunnskap om lokale forhold. Det er altså ikke mulig å konstruere en formell beskrivelse av et system som kan garantere for at det ikke vil forekomme uforutsette hendelser som er inkonsistent med denne. Dette vil si at man må tilrettelegge for at deltagerne i et samarbeid kan artikulere aktivitetene, altså planlegge, koordinere og overvåke, for å være i stand til å fullføre den oppgaven de skal løse på en god måte (s. 266-267).

Konseptet blir videre diskutert av Schmidt and Bannon (1992), som benyttet det for å gi CSCW et fokus mot å forstå sosialt organiserte arbeidssituasjoner. De hevder at articulation work oppstår som en vesentlig del av ethvert samarbeid, i form av det settet med aktiviteter som kreves for å håndtere den distribuerte naturen til samarbeid (s. 18). Dette vil med andre ord si at articulation work kan bli benyttet som et fruktbart begrep for å forstå hvordan distribuerte programmeringsteam samarbeider, slik som det er gjort i denne studien, og av Boden, Nett og Wulf (2008).

Schmidt og Bannon (1992) uttaler at håndtering av arbeidsflyt, som ble nevnt tidligere, bare er det ene viktige aspektet av articulation work for CSCW. Det andre fokuserer på "the construction and management of what we term a *common information space"* (1992, s. 16). Dette er også gjenspeilet i denne studien, fordi et MUVE er karakterisert som et common information space. I kapittel 7 skal det vises hvordan articulation work er støttet gjennom tilretteleggelse for kommunikasjon, planlegging, overvåking og koordinasjon i et MUVE. I avsnitt 3.3 forklares det hvordan articulation work er vesentlig for å posisjonere denne studien, og det skal ytterligere illustreres hvordan articulation work er et viktig aspekt ved programutviklingssamarbeid.

2.4 Samarbeidsmekanismer

Et sentralt bidrag fra CSCW feltet er Gutwin og Greenbergs (2000) konseptuelle rammeverk for å vise samarbeidsmekanismene[9] som benyttes når man samarbeider. Dette anses som et nyttig analysenivå for en gjennomgang[10]. Grunnen er at de gir et svært detaljert bilde av samarbeid, hvor man kan bryte ned samarbeid i dets spesifikke handlinger, som den som evaluerer[11] kan se på enkeltvis. De grunnleggende aktivitetene i delt arbeid vises i tabell 2.

Samarbeidsmekanismene fungerer i denne studien som veiledende krav til basisfunksjonaliteten for prototypen som blir beskrevet i kapittel 7, og er dermed en fundamental del av studien. Ved å benytte disse kan applikasjonens usability forbedres betydelig fra slik den opprinnelig var designet. Dette har blitt gjort kontinuerlig i utviklingen av prototypen til studien, ved at den har blitt redesignet flere ganger, når mangler i forhold til samarbeidsmekanismene har blitt identifisert. Samarbeidsmekanismene utgjør viktige trekk for å opparbeide en felles forståelse av informasjonsobjektene i et datastøttet samarbeidssystem. I kapittel 7 vil det bli beskrevet hvordan disse konseptene har blitt anvendt under design av prototypen. Samarbeidsmekanismene er også vesentlig når man skal evaluere prototypen. Groupware Walkthrough og Heuristic Evaluation, som denne studien gjør nytte av i kapittel 8, baserer seg på å benytte disse mekanismene (se tabell 2) for å analysere samarbeidsaktivitetene som blir utført i systemet.

Tabell 2 – Samarbeidsmekanismene som tar sted i samarbeid. Vår oversettelse fra Gutwin og Greenberg (2000, s. 99-100)

Samarbeidsmekanisme	Beskrivelse
Eksplisitt kommunikasjon	Skriftlig eller verbal kommunikasjon er en viktig del av alle typer samarbeid.
Indirekte kommunikasjon	Informasjon som en aktør gir fra seg uten å være klar over det, som kan være til nytte for andre. For eksempel informasjon som blir gitt fra artefakter når de blir manipulert av andre.
Koordinasjon av handling	Man forsøker å organisere sine handlinger i et delt workspace slik at de ikke kommer i konflikt med andre. Delte ressurser og verktøy krever at handling skjer i en viss rekkefølge. Man lærer å forutse andres handlinger og benytter denne kunnskapen for å få gruppen til å samarbeide mer effektivt. Tegn på dårlig koordinering - Duplisering av handlinger, ressurskonflikt og et kaotisk arbeidsmiljø.

9 Eng: Mechanics of Collaboration
10 Eng: Walkthrough
11 Eng.: Usability expert

Planlegging	I et delt workspace er det viktig at man planlegger oppgaver og fordeler dem mellom seg, har mulighet for å reservere deler av workspace, og mulighet for å simulere ulike handlinger.
Overvåking	Flere av de andre samarbeidsmekanismene baserer seg på at man kan overvåke og innhente informasjon om de andre deltagerne i et workspace. Mye av dette er workspace awareness informasjon.
Hjelp	Gruppemedlemmene gir hverandre assistanse ved behov. Man kan uttrykke sitt behov gjennom eksplisitt kommunikasjon eller andre kan benytte awareness informasjon for å forstå at man trenger hjelp. For at dette skal fungere optimalt er det viktig at de andre deltagerne forstår hva man gjør og hvor en er i sitt arbeid.
Beskyttelse	Deltagerne kan endre eller ødelegge andre sitt arbeid. Det er derfor viktig at man følger med på sitt eget arbeid, oppfatter effekten av andres handlinger, og forhindrer de som er skadelige.

2.5 Awareness rammeverk for sanntids-groupware

Gutwin og Greenberg (2002) beskriver et rammeverk for workspace awareness for analysering av sanntids groupware som blir benyttet i denne studien. Dette er et nyttig verktøy for å fremheve enkelte kortvarige aspekter ved samarbeid. De egenskaper av awareness som er relevant for gruppearbeid vil bli gått igjennom under og det vil bli sett nærmere på workspace awareness, og hvordan dette blir opprettholdt.

2.5.1 Egenskaper for awareness

Tidligere forsking har pekt på fire kjennetegn ved awareness. Oversatt fra engelsk fra (Gutwin & Greenberg 2002, s. 416):

- Awareness er kunnskap om tilstanden til omgivelsene festet i tid og rom.
- Omgivelsene er i stadig endring, så bevisstheten er kunnskap som må vedlikeholdes og oppdateres over tid.
- Mennesker interagerer med og utforsker omgivelsene, og vedlikeholder dermed awareness.
- Målet er å fullføre en handling i omgivelsene, ikke å opprettholde awareness.

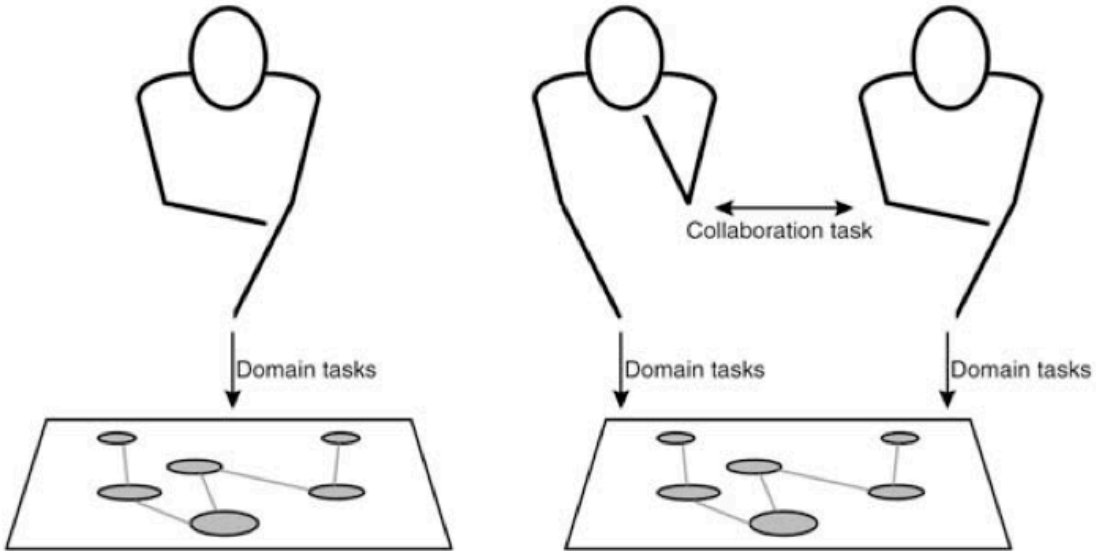
Alle har erfart denne formen for awareness, da det er dette som lar oss bevege oss rundt uten å støte borti noe. Men når situasjoner og omgivelsene blir så komplekse at man har problemer med å henge med, må man tenke mer over ting. Forskere har utforsket dette, og de har gitt fenomenet betegnelsen situasjons awareness (Gutwin & Greenberg 2002).

2.5.2 Situasjons awareness

Det finnes ikke en enkelt definisjon av situasjons awareness, men en dekkende definisjon er ifølge Gutwin og Greenberg (2002) ”the up-to-the minute cognizance required to operate or maintain a system” (s. 417) . Altså har man en spesifikk

situasjons awareness, der det er avgjørende å få med seg de siste hendelser i omgivelsene. I henhold til artikkelen har det blitt forsket på miljøer hvor situasjons awareness spiller en stor rolle, slik som i kommersiell flytrafikk, flygeledelse, og anestesi. Disse miljøene deler følgende karakteristikker ”dynamism, complexity, high information load, variable workload, og risiko (Gaba, Howard & Small 1995 referert i; Gutwin & Greenberg 2002, s. 417). Mica Endsley (1995) fokuserer mer på prosessen, og foreslår en definisjon hvor man har en trelagsstruktur. For det første må en aktør kunne samle inn informasjonen fra omgivelsene, og prioritere de elementer som er mest relevant for den oppgaven det jobbes med. For det andre må en aktør kunne integrere innkommende persepsjonsinntrykk med eksisterende kunnskap, og forstå denne informasjonen i lys av gjeldende situasjon. For det tredje må man kunne forutse endringer i omgivelsene, og hvordan innkommende informasjon vil forandre seg, for å kunne handle effektivt. Disse tre nivåene gjelder også for workspace awareness (Gutwin & Greenberg 1996), fordi workspace awareness er en spesialisering av situasjons awareness (Gutwin & Greenberg 2002).

2.5.3 Workspace awareness

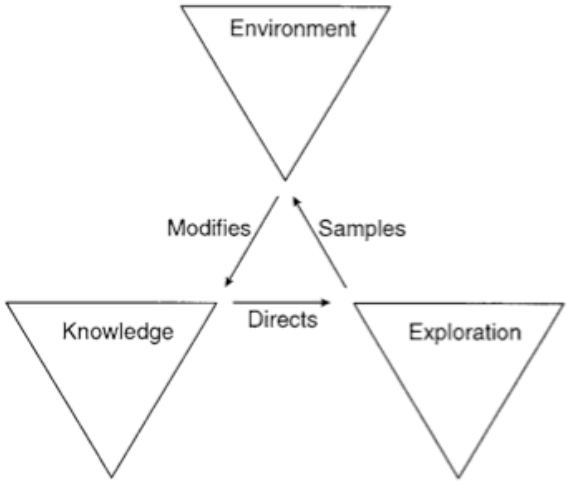


Figur 1 – Domene og samarbeidsoppgaver (Gutwin & Greenberg 2002, s. 418)

Gutwin og Greenberg (2002) definerer Workspace Awareness som den kontinuerlig oppdaterte forståelse av hvordan de andre interagerer i workspace. Denne definisjonen begrenser konseptet på to måter. For det første er awareness begrenset innad i workspace hvor hendelsene tar sted. Når noen jobber alene i workspace vil

deres aktiviteter og deres situasjons awareness kun involvere workspace og domene oppgaven (se figur 1). I en samarbeidssituasjon, må man selv ta ansvar, og dermed må situasjons awareness for den enkelte involvere både domenet og samarbeidet (Gutwin & Greenberg 2002).

2.5.4 Vedlikehold av awareness



Figur 2 – " The perception-action cycle" (Gutwin & Greenberg 2002, s. 418 fra; Neisser 1976)

Neissers modell i figur 2 gir et bilde over hvordan man vedlikeholder workspace awareness. Et viktig prinsipp her er at persepsjon og handling er tett knyttet sammen. Dette er essensielt hvis man ønsker å designe systemer som skal støtte workspace awareness, og er noe man må ta høyde for under evaluering av et system. Modellen fanger opp interaksjonen mellom en aktør og omgivelsene, og inkorporerer forholdet mellom en persons kunnskap og informasjonsinnhentingsaktiviteter. Dette skiller seg fra lineære modeller av informasjonsprosessering ved at den tar høyde for at oppfattelsesevnen er påvirket og styrt av eksisterende kunnskap. Således er det slik at når en person trer inn i et miljø for å gjøre en bestemt oppgave, har man med seg en generell forståelse av situasjonen og en grunnleggende ide av hva man skal se etter (Gutwin & Greenberg 2002).

2.5.5 Workspace awareness rammeverk

Det konseptuelle rammeverket legger frem grunnleggende elementer en designer må ta høyde for når han skal lage workspace awareness støtte i groupware systemer. Rammeverket beskriver tre aspekter ved workspace awareness - de ulike delene, mekanismene som behøves for å opprettholde den, og hvordan det blir brukt ved

samarbeid. De tre delene tilsvarer de tre oppgavene som en utvikler må utføre for å kunne støtte workspace awareness - forstå hvilken informasjon man må tilby, finne ut hvordan informasjonen blir hentet inn, og finne ut når og hvor informasjonen vil bli benyttet. De ulike elementene i workspace awareness gir svar på hvem, hvor, hvordan og hva spørsmål, og fokuserer på fortid og nåtid (Gutwin & Greenberg 2002).

I denne studien har workspace awareness egenskapene til Second Life blitt benyttet for å vise hvem som er til stede i prototypen til enhver tid, gi brukerne mulighet for å diskutere intensjonen bak handlingene sine (gjennom synkron og asynkron interaksjon), og vise hvilke handlinger som har blitt utført på felles objekter. Utover dette har funksjonalitet for å vise hvem som har gjort hva, og på hvilket tidspunkt, blitt benyttet. Dette blir videre diskutert i kapittel 7.

2.6 Metodologier for systemutvikling

Når nye systemer skal utvikles vil de fleste utviklere benytte et sett av standard metoder for å understøtte arbeidet som skal utføres. Arbeidet med å utvikle systemer har som mye annet et livsløp, fra vugge til grav. Dette livsløpet består gjerne av det å bli laget, introdusert i markedet, solgt i en periode, og til slutt utfaset eller erstattet av et annet produkt (Hoffer, George & Valacich 2005, s. 10). Man kan se på en metodologi for systemutvikling som en samling prosedyrer, teknikker, verktøy, samt dokumentasjon til hjelp for utviklere i sitt arbeid med å utvikle nye informasjonssystemer (vår oversettelse fra Avison & Fitzgerald 2003, s. 20).

I dag deles metodologier for systemutvikling normalt inn i tradisjonelle og agile metoder, hvor sistnevnte er fokuset i denne studien. Det er ikke slik at tradisjonelle metoder er utdatert i forhold til agile, det er mer at de har en annen grunnfilosofi og benytter andre metoder og teknikker for å utvikle informasjonssystemer.

2.6.1 Agile metoder

En utviklingsmetode er agil[12] når programutviklingen er inkrementell (små programpakker, med raske sykluser), har samarbeid i fokus (kunder og utviklere jobber konstant sammen med nær kommunikasjon), rett frem (metoden er lett å lære, modifisere og er godt dokumentert), og tilpasningsdyktig (man kan gjøre endringer i

12 Smidig, lett, livlig – hentet fra ordnett.no 20/8-2009

siste liten) (Abrahamsson, Salo, Ronkainen & Warsta 2002). I denne kategorien vil det bli sett nærmere på ekstremprogrammering og Scrum[13].

2.6.1.1 Historikk

Agile metoder kan ses på som en reaksjon til den tradisjonelle måten å utvikle programvare, hvor man innså behovet for et alternativ til en dokumentdrevet og svært tung programvareutviklingsprosess. Et av problemene var at utviklingen innen teknologien og industrien gikk fort, slik at kravene ble utdatert før systemet var utviklet. En annen utfordring var at brukerne hadde problemer med å definere alle krav til systemet i forkant av utviklingen. Det var med andre ord behov for metoder og teknikker som ikke trengte et fullt sett med systemkrav før selve utviklingsprosessen startet. De agile metodene vokste frem som svar på dette behovet. Agile metoder er altså mer enn bare en ting, det er i stedet et paraplybegrep for en samling av ulike teknikker og fremgangsmåter som deler like verdier og prinsipper (Cohen, Lindvall & Costa 2004). I dagens globale finansmarked, hvor marginene er små, er det smidighet og tilpasningsdyktighet som skiller vinnerne fra taperne.

2.6.1.2 Det agile manifest

Det agile manifestet (Beck, Beedle, Bennekum, Cockburn, Cunningham, Fowler, Grenning, Highsmith, Hunt, Jeffries, Kern, Marick, Martin, Mellor, Schwaber, Sutherland & Thomas 2001) ble grunnlagt helgen 11. til 13. februar 2001 når 17 organisasjonsarkitekter, alle forkjempere for agile arbeidsmetoder, samlet seg for å diskutere de nye utviklingsmetodene, med et mål om å finne et felles ståsted. Det var deltagere fra ulike miljø innen Scrum, XP, Crystal og mange andre som var samlet for å finne et alternativ til de tradisjonelle metodene. De ønsket å komme frem til en metodologi som ikke krevde hundrevis av sider med dokumentasjon før selve utviklingsprosjektet kunne starte opp (Cohen, Lindvall & Costa 2004).

Resultatet fra denne samlingen var det agile manifest som har fire grunnsøyler.
Den første søylen er fremhever viktigheten av å ha en tett integrasjon mellom utviklere og brukere. Videre forteller denne at det er viktig å arbeide sammen i team, hvor man sitter tett sammen for å få en god team følelse. Videre er det vitalt at man hele tiden leverer nye programvarekomponenter med korte intervaller. Selv om

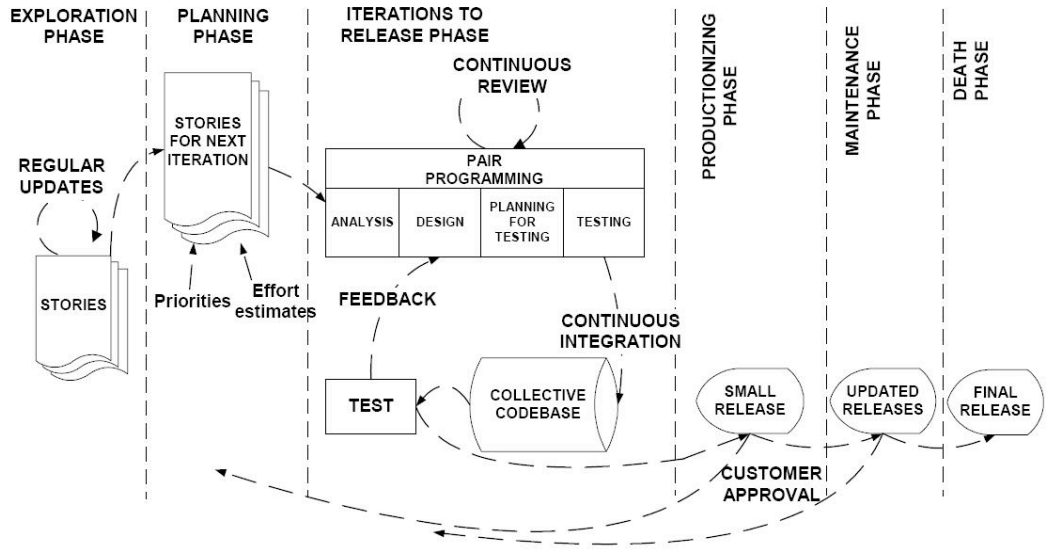
13 I kapittel 6 blir det beskrevet, i form av kvalitativ observasjon og intervju, hvordan IKTBergen benytter disse metodene for å utvikle programvare.

behovet for kontrakt mellom leverandør og kunde fremdeles er like aktuelt, løper kunden en mindre risiko i prosjektet fordi kunden hele veien får levert ny funksjonalitet med faste intervaller. Til slutt er det viktig at kunden kan påvirke resultatene av utviklingen gjennom hele utviklingsprosessen for å kunne ende opp med et best mulig produkt (Abrahamsson, Salo, Ronkainen & Warsta 2002). På agilemanifesto.org kan man lese om hvordan arkitektene ”[…] are uncovering better ways of developing software by doing it and helping others do it”. Gjennom dette arbeidet har de lært å verdsette; [1] individ og interaksjon foran prosesser og verktøy; [2] programvare som fungerer fremfor detaljert dokumentasjon; [3] kundesamarbeid foran kontraktsforhandlinger; [4] at det er viktigere å reagere på endring enn å følge en plan. Det er også verdi i elementene på høyre side, men de verdsetter elementene på venstre side mer (vår oversettelse fra Beck m fl. 2001).

2.6.2 Ekstremprogrammering (XP)

XP har blitt utviklet på bakgrunn av problemene som kom av de lange utviklingssyklusene til tradisjonelle utviklingsmodeller (Beck 1999a). I starten var dette egentlig bare metoder man hadde funnet på de siste tiårene, som hadde vist seg å være nyttig i en programutviklingsprosess, for å få jobben gjort (Beck 1999b). Selv om de individuelle metodene i XP ikke er ny i seg selv, har de blitt samlet opp og systematisert. Resultatet er at de nå fungerer med hverandre på nye måter. På denne måten har det blitt skapt en ny metodologi for programutvikling. Det har fått ”ekstrem” i navnet fordi man tar disse fornuftige prinsippene, og praktiske fremgangsmåtene, til et ekstremt nivå (Beck 1999b). Nedenfor vil prosessen og teknikkene bli presentert, men fordi det i denne studien fokuseres på rollene i Scrum vil ikke rollene fra XP bli beskrevet.

2.6.2.1 Prosess



Figur 3 - XP livssyklus (Abrahamsson, Salo, Ronkainen & Warsta 2002, s. 19)

XP er delt inn i seks faser som vist i figur 3, og beskrevet i henhold til Kent Beck (1999b, s. 89-96):

I **utforskingsfasen** beskriver kunden ulik funksjonalitet som er ønsket i første versjon av programmet. I denne fasen gjør også utviklingsteamet seg kjent med verktøy, teknologi og ulike praksiser som vil bli benyttet i prosjektet. Utforskningsfasen kan gå fra noen uker til noen måneder.

Planleggingsfasen setter prioritet på de ulike oppgavene som skal utføres og man estimerer timer på de ulike oppgavene. Den første versjonen av programmet blir normalt ikke levert mer enn to måneder etter denne fasen, som i seg selv tar noen dager å gjennomføre.

Den neste fasen omhandler iterasjonene som skal være med i leveransen[14]. Disse gjennomføres i flere runder, hvor fremdriftsplanen som ble satt opp i planleggingsfasen, blir brutt ned i flere iterasjoner som gjennomføres i løpet av to til fire uker. Den første iterasjonen bygger systemets arkitektur. Dette gjennomføres ved at man velger oppgaver som bidrar til å bygge strukturen til hele systemet. Kunden bestemmer hvilke oppgaver som skal gjennomføres i de ulike iterasjonene. De

14 Eng.: Iterations to release

funksjonelle testene som ble spesifisert av brukerne blir kjørt etter hver iterasjon. Når siste iterasjon er gjennomført er systemet klart for produksjon.

I **produksjonsfasen** blir det utført grundig testing av systemet før det overleveres til kunden. I denne fasen kan man også oppdage nye ting som må gjøres og ta stilling til om det skal fikses nå, eller tas i en senere versjon.

Fordi XP leverer et system til kunden som er under kontinuerlig utvikling er det nødvendig med en **vedlikeholdsfase** for å støtte kunden under arbeidet med systemet. Brukerstøtte inngår i denne fasen, og denne fasen kan kreve at man knytter til seg nye personer inn i teamet og endrer team sammensetningen.

Sluttfasen av prosjektet tar til når kunden ikke lengre har flere funksjoner den ønsker implementert. Systemet blir da dokumentert og ingen nye endringer blir utført på arkitektur, kode eller design (Beck 1999b, s. 89-96).

2.6.2.2 Fremgangsmåter og teknikker

Det vil nedenfor bli gitt en oversikt over de ulike fremgangsmåtene og teknikkene som benyttes i XP, slik presentert av Beck (vår oversettelse fra Beck 1999a, s. 71).

Tabell 3 – Fremgangsmåter og teknikker i ekstremprogrammering

Planleggingsspill
Tett interaksjon mellom kunde og programmerere. Programmererne estimerer innsatsen som kreves for implementering av kundens behov og kunden bestemmer seg for omfang og tidspunkt for leveransene.
Små/korte leveranser
Et enkelt system blir raskt satt i produksjon (se figur 3) – i hvert fall en gang hver 2 til 3 måned. Nye versjoner blir levert minimum en gang i måneden.
Metafor
Systemet blir definert av en metafor/sett med metaforer mellom kunden og programmereren. Denne ”delte historien” guider all utvikling ved å beskrive hvordan systemet fungerer.
Enkel design
Fokus på å designe den enkleste løsningen som mulig som kan implementeres. Unødig kompleksitet og ekstra kode blir fjernet umiddelbart.
Testing
Programutvikling blir drevet frem ved hjelp av tester. Enhetstester blir implementert før koden og blir kjørt kontinuerlig. Kunden skriver de funksjonelle testene.
Refaktorering
Omstrukturerer systemet ved å fjerne duplikater, forbedrer kommunikasjon, forenkler og gir fleksibilitet.
Parprogrammering
To personer skriver kode på en datamaskin. Det viktige her er ikke at det er en felles datamaskin, men at man kan se den samme koden, og kan kommunisere med hverandre. Det veksles mellom å skrive kode (driver) og se over koden som den andre skriver (observer).

Felles eierskap
Alle i teamet kan når som helst endre koden.
Kontinuerlig integrasjon
Nye deler med kode integreres inn i kodebasen så snart den er ferdig. På denne måten blir systemet integrert og bygget mange ganger pr dag. Alle tester blir kjørt, og må bli godkjent, før endringene på koden blir akseptert.
40 timers uke
Det er ikke tillatt å arbeide mer en 40 timer i uken. Det er ikke lov å arbeide overtid to uker på rad. Hvis dette skjer, skal det bli sett på som et problem som må løses.
Tilstedeværelse av kunde
Kunden må være tilstede og tilgjengelig hele tiden for teamet.
Kode standard
Det finnes koderegler som må følges av programmererne. Det må fremheves at kommunikasjon gjennom kode er viktig.
Åpent workspace
Det foretrekkes et stort rom med små avlukker. Parprogrammerere bør bli passert i senter av rommet.
Egne regler
Teamet har egne regler som må følges, men som kan bli endret til enhver tid. Alle parter må bli enig om endringene og effekten av de må bli vurdert.

Den raske, inkrementelle utviklingen i XP har sine røtter i ideene bak Scrum (Takeuchi & Nonaka 1986), en agil utviklingsmetodologi som blir beskrevet i neste avsnitt.

2.6.3 Scrum

Den første akademiske referansen til Scrum var i en artikkel skrevet av Takeuchi og Nonaka (1986) hvor en tilpasningsdyktig, rask, og selvorganiserende produktutviklingsprosess med opphav fra Japan ble presentert (Schwaber & Beedle 2002). Scrum er basert på best practice (Sutherland, Viktorov, Blount & Puntikov 2007). I henhold til det agile manifest (Beck m fl. 2001) ligger styrken til Scrum i hvordan medlemmene samhandler gjennom en iterativ prosess. Det handler om raske beslutninger og samarbeid, ikke tunge prosesser og mengder med informasjon. Metodologien fikk sitt navn fra en strategi i rugby, hvor målet er å få en "ball som er ute av spill, inn i spillet igjen" ved hjelp av teamarbeid (Schwaber & Beedle 2002).

Scrum har blitt utviklet for å håndtere systemutviklingsprosessen. Dette er en fremgangsmåte som benytter seg av ideer fra kontrollteori for industriprosess og tilpasser disse til systemutvikling. Resultatet av dette er en fremgangsmåte som reintroduserer tanker om fleksibilitet, tilpasningsdyktighet og produktivitet (Schwaber & Beedle 2002). Scrum, i motsetning til andre fremgangsmåter, definerer ikke noen spesifikke systemutviklingsteknikker for implementeringsfasen. Scrum fokuserer på

hvordan gruppemedlemmer skal fungere for å kunne fremstille systemer fleksibelt, selv om de omkringliggende forutsetningene endres kontinuerlig.

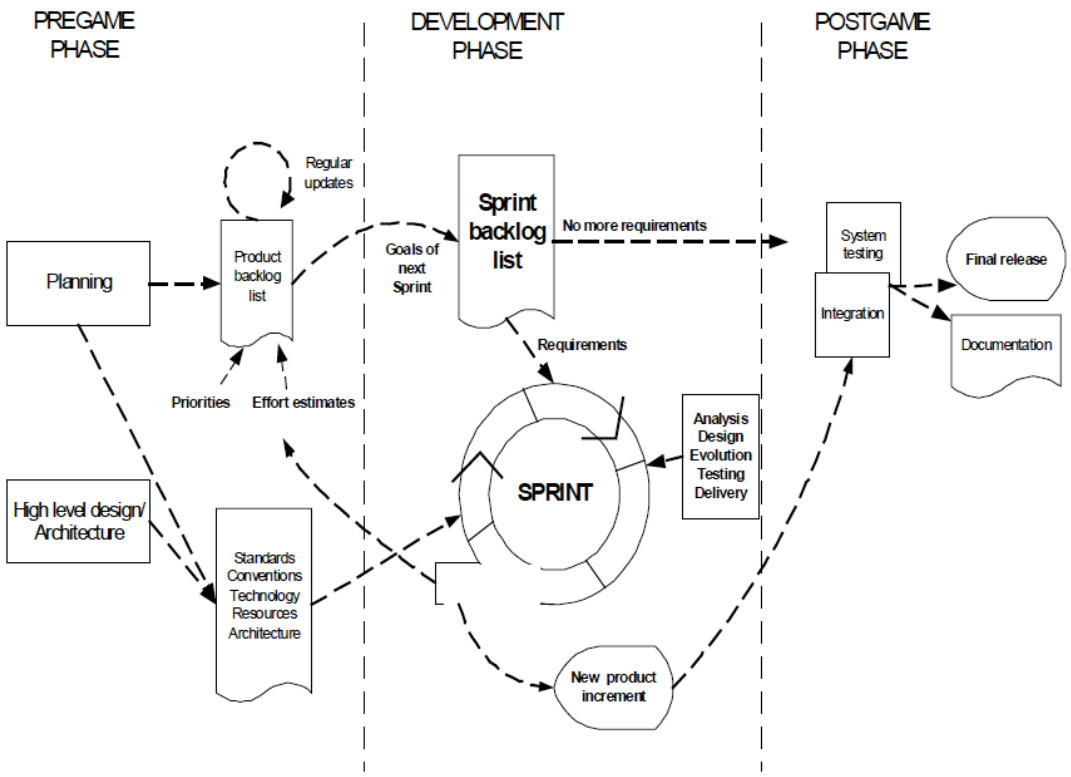
Hovedtanken bak Scrum er at systemutvikling involverer flere miljø og tekniske variabler (for eksempel krav, tidshorisont, ressurser, og teknologi) som det er sannsynlig vil endres underveis i prosessen. Utviklingsprosessen blir dermed uforutsigbar og kompleks, og for å håndtere dette kreves det fleksibilitet i systemutviklingsprosessen. Resultatet av en slik utviklingsprosess er et system som er nyttig når det blir levert (Schwaber 1995).

Scrum har til hensikt å forbedre eksisterende arbeidspraksis (for eksempel test metodikk) i en organisasjon. Dette blir oppnådd ved at den inneholder hyppige ledelsesoppgaver som skal identifisere mangler eller hindre i utviklingsprosessen og de fremgangsmåter som benyttes (Schwaber 1995).

Et element av denne formen for samarbeid, i likhet med rugby, er at deltagerne bør være til stede til samme tid på samme sted. Hele Scrum teamet sitter ifølge metodologien ideelt i det samme rommet. Åpne landskapsløsninger har vært svaret på dette. Bjerrum og Bødker (2003) har studert effekten av å arbeide i åpne kontorlandskaper, og hvordan dette påvirker samarbeid og læring på arbeidsplassen. Her stiller hun spørsmål til om dette gjøres på grunn av læring, effektivitet eller i et rent økonomisk hensyn. Fordi mange selskaper benytter arbeidskraft som er spredt på flere ulike steder, er det ikke lenger mulig for alle Scrum team å ha alle programmererne i et felles kontorlokale. Dette har bakgrunn i utstrakt bruk av utkontraktering av utviklingstjenester til lavkostland, og fremgangen til multinasjonale selskaper som ønsker å benytte alle sine ressurser uavhengig av hvor de måtte befinne seg. Når dette alternativet ikke er tilgjengelig må man se på andre løsninger for kommunikasjon og samarbeid. I avsnitt 3.1.1 blir det sett på forskning på distribuerte Scrum team.

2.6.3.1 Prosess

Scrum prosessen består av en innledende fase (pregame), utviklingsfase (development), og sluttfase (postgame). Dette ser man igjen i figur 4. Nedenfor vil de ulike fasene bli introdusert i henhold til Ken Schwaber (1995), og Schwaber og Beedle (2002). Den innledende fasen inkluderer to delfaser - planlegging og arkitektur/høynivå design.



Figur 4 - Illustrasjon av Scrum prosessen (Abrahamsson, Salo, Ronkainen & Warsta 2002, s. 28)

Planleggingsfasen inneholder definisjonen av systemet under utvikling. En Product Backlog liste **(**se avsnitt 2.6.3.3), som inneholder alle kjente krav**,** blir opprettet. Kravene kan komme fra kunden, salg og markedsføringsavdelingen, kundestøtte og programvareutviklerne. De blir gitt ulik prioritet og man estimerer hvor mye som kreves for å få kravene implementert. Product Backlog listen er kontinuerlig oppdatert med nye og mer detaljerte punkter, samt mer nøyaktige estimater og endrede prioriteter. Planleggingsfasen inneholder også definering av prosjektteam, verktøy og andre ressurser, kursbehov, samt at ledelsen er inne og verifiserer og godkjenner. I hver iterasjon blir den oppdaterte Product Backlogen gjennomgått av Scrum teamet. Dette gjennomføres for å få deres aksept for den neste iterasjonen.

I **arkitekturfasen** blir høynivådesign av systemet, samt arkitekturen, planlagt på bakgrunn av punktene i Product Backlog listen. Hvis man er i ferd med å endre på et eksisterende system, må man identifisere de endringer som er nødvendige for å implementere Backlog punktene, samt problemene disse kan forårsake. Man holder et design gjennomgangsmøte for å se på forslag til gjennomføring, og på bakgrunn av dette møtet blir man bedre rustet til å ta gode beslutninger. I tillegg blir det skissert planer for innholdet i de ulike leveransepakkene.

Utviklingsfasen (også kalt spillfasen) er den agile delen av fremgangsmåten Scrum. Denne fasen blir behandlet som en "svart boks", hvor det uventede er ventet. De ulike miljø og tekniske variablene (slik som tidsramme, kvalitet, krav, ressurser, implementeringsteknologier og verktøy, og utviklingsverktøy) som er identifisert i Scrum, og som kan endres i underveis i prosessen, blir observert og kontrollert gjennom ulike Scrum fremgangsmåter i utviklingsfasens sprinter. I stedet for kun å ta hensyn til disse faktorene i begynnelsen av programvareutviklingsprosjektet, har Scrum som mål å kontrollere disse kontinuerlig for å fleksibelt kunne tilpasse seg endringene.

Systemet er i utviklingsfasen utviklet i sprinter som man ser i figur 4. Se også avsnitt 2.6.3.3. Sprinter er iterative sykluser hvor funksjonaliteten er utviklet eller forbedret for å produsere nye del-leveranser**.** Hver sprint inkluderer de tradisjonelle fasene i programvareutvikling - kravstilling, analyse, design, utvikling og leveranse (se figur 4). Arkitekturen og systemets design utvikles gjennom sprint utviklingen. En sprint planlegges å vare fra en uke til en måned. Det kan for eksempel være tre til åtte sprinter i en systemutviklingsprosess, før systemet er klart for distribusjon. Det kan også være mer enn et team som produserer delleveransen.

Sluttfasen inneholder avslutningen av leveransen. Man går inn i denne fasen når en har kommet til enighet om at miljøvariablene, slik som kravene til systemet, har blitt fullført. Etter denne beslutningen er det ikke mulig å få inn flere punkter eller innvendinger, og man kan ikke finne på nye. Systemet er nå klart for leveranse og det er i denne fasen at forberedelsene til dette tar sted, samt integrasjon, systemtesting og dokumentasjon, slik vist i figur 4.

2.6.3.2 Roller og ansvar

I Scrum er det seks roller som alle har ulike oppgaver og formål gjennom prosessen - Scrum Mester, Produkteier, Scrum Team, Kunde, Bruker og Ledelse.

Scrum Mester

Scrum Mester er en ny ledelsesrolle introdusert i Scrum. Scrum Mester er ansvarlig for at prosjektet blir gjennomført i henhold til fremgangsmåtene, verdiene og reglene til Scrum og at prosjektet skrider frem som planlagt. Scrum Mester interagerer både med prosjektteamet, kunden og ledelsen i løpet av prosjektet. Han eller hun er også ansvarlig for å sørge for at hinder som oppstår i løpet av prosjektet blir fjernet og endret i løpet av prosessen, slik at teamet arbeider så effektivt som mulig.

Produkteier

Produkteier er offisielt ansvarlig for prosjektet, samt å lede, kontrollere og gjøre synlig Product Backlog listen. Scrum Mester, kunden og ledelsen velger hvem som skal være Produkteier. Han eller hun har det siste ordet i avgjørelser angående oppgaver i forbindelse med Product Backlog (se avsnitt 2.6.3.3), deltar i å estimere hvor mye innsats de ulike Backlog punktene vil ta, og gjør de ulike punktene i Backlogen om til funksjonalitet som skal utvikles.

Scrum team

Scrum team er det prosjektteamet som har autoritet til å organisere seg selv, og ta de grep som er nødvendig for å oppnå målene til hver sprint. Scrum teamet er for eksempel involvert i å lage Sprint Backlogen, estimere innsats, og foreslå hindre som må fjernes i fra prosjektet.

Kunde

Kunden deltar i oppgaver som relaterer seg til produktets Backlog punkter (se avsnitt 2.6.3.3) for systemet som blir utviklet eller forbedret.

Ledelse

Ledelsen har ansvaret for å ta de endelige avgjørelsene, så vel som charter**,** standarder og konvensjoner som skal følges i prosjektet. Ledelsen deltar også i å sette mål og krav. For eksempel er ledelsen involvert i å velge Produkteier, se an hvordan prosjektet skrider frem, og redusere Backlogen sammen med Scrum Mester.

2.6.3.3 Fremgangsmåter

Scrum verken krever eller tilgjengeliggjør spesifikke programvareutviklingsmetoder eller fremgangsmåter som man kan benytte. Det Scrum krever i stedet er et sett med ledelsespraksiser og verktøy i de ulike Scrum fasene, slik at man unngår kaoset som blir skapt av usikkerhet og kompleksitet (Schwaber 1995).

Beskrivelsen av Scrum metodene er gitt på bakgrunn av Schwaber og Beedle (2002), to av arkitektene bak det agile manifest (Beck m fl. 2001), og illustrert i figur 5.

Produkt Backlog

Produkt Backlog definerer alt som man antar, basert på nåværende kunnskap, må være med i det ferdige produktet. Dermed er det produkt Backlog som definerer hva som må utføres i prosjektet. Den innehar en kontinuerlig oppdatert prioritetsliste over foretnings- og tekniske krav for systemet som er under nyutvikling eller forbedring. Backlog punkter kan bestå av karakteristiske egenskaper, funksjoner, feilrettelser, feil, ønskede forbedringer og teknologiske oppgraderinger. Elementer som må være i orden før Backlog punkter kan utføres er også med i listen. Flere aktører kan delta i å generere Backlog punkter, som for eksempel kunden, prosjektgruppen, salg og markedsføring, ledelsen og kundestøtte.

Dette punktet inneholder oppgavene som må til for å lage produkt Backlog listen og de mekanismene som kontrollerer den gjennom prosessen, ved at man legger til, fjerner, spesifiserer, oppdaterer og prioriterer produkt Backlog punkter. Produkteieren er ansvarlig for å vedlikeholde produkt Backlogen.

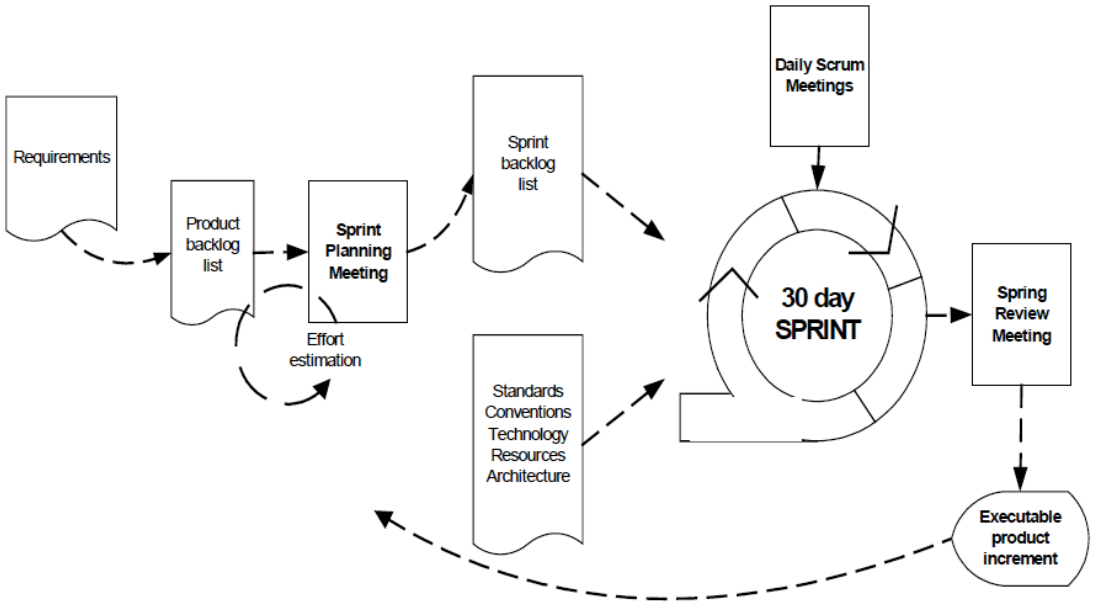
Innsatsestimering

Innsatsestimering er en iterativ prosess hvor Backlog punktestimering er fokusert på et mer nøyaktig nivå, når mer informasjon blir tilgjengelig for et spesifikt Backlog punkt. Produkteieren sammen med Scrum team er ansvarlig for å utføre innsatsestimering.

Sprint

Sprint er den prosedyren man benytter for å tilpasse seg endrede miljøvariabler (krav, tid, resurser, kunnskap, teknologi, med mer). Scrum teamet organiserer seg på en slik måte at de kan produsere et nytt eksekverbart produktnivå i løpet av 30 dager.

Arbeidsverktøyene for teamet er sprint planleggingsmøter, Sprint Backlog og daglige Scrum møter, og andre møter. Sprinten, med dens praksis og inputs er illustrert i figur 5.



Figur 5 - Praksis og input til en sprint (Abrahamsson, Salo, Ronkainen & Warsta 2002, s. 33)

Sprint planleggingsmøte

Sprint planleggingsmøte er et todelt møte som er organisert av Scrum Mester. Kunden, brukerne, ledelsen, produkteier og Scrum teamet deltar i den første delen for å avgjøre hvilke mål og funksjonalitet man skal ha med i den neste sprinten. Den andre delen av møtet, holdt av Scrum Mester og Scrum teamet, fokuserer på hvordan del leveransen vil bli implementert igjennom sprinten.

Sprint Backlog

Sprint Backlogen er startpunktet for hver sprint. Dette er listen over de Product Backlog punktene som har blitt utvalgt til å bli implementert i den neste sprinten. Punktene er valgt av Scrum teamet sammen med Scrum Mester og Produkteier i sprint planleggingsmøte, på bakgrunn av de prioriterte punktene, og mål satt for sprinten. Sprint Backlogen er, i motsetning til Product Backlog, stabil til sprinten (vanligvis 30 dager) er fullført. Når alle punktene i Sprint Backlog er fullført, er en ny iterasjon av systemet levert.

Daglig Scrum møte
Det daglige Scrum møte er organisert slik at man kan holde en løpende oversikt over fremgangen til Scrum Team, samt være planleggingsmøter. Det er tre spørsmål som er sentrale under et slikt møte. For det første må man besvare hva som har blitt utført siden forrige møte. For det andre må man skissere hva som må gjøres før det neste møte, og for det tredje må man identifisere om noe hindrer utviklingsprosessen. Problemer og andre tema blir diskutert og kontrollert i dette korte daglige møte (cirka 15 minutter). Man er på utkikk etter svakheter og/eller hindringer i systemutviklingsprosessen og eksisterende arbeidspraksis, slik at disse kan bli identifisert og fjernet for å forbedre prosessen. Det er Scrum Mesteren som leder et slikt møte. Foruten Scrum teamet kan ledelsen være representert på møte.

Sprint gjennomgangsmøte
På sprintens siste dag presenterer Scrum Mester og Scrum teamet resultatene (altså den fungerende delleveransen) til ledelsen, kundene, brukerne og produkteierne i et uformelt møte. Deltagerne vurderer delleveransen og beslutter oppfølgingsaktiviteter. Sprint gjennomgangsmøte gir styring til hvordan systemet skal utvikles videre, noe som kan medføre en strategiendring, og kan gi nye Backlog punkter.

2.6.3.4 Bruksområde
Scrum er en metode som passer for små team på mindre enn ti ingeniører. Schwaber og Beedle (2002) anbefaler at teamet består av mellom fem til ni prosjektmedlemmer. Hvis man har flere ingeniører tilgjengelig, bør en opprette flere team.

I neste kapittel blir det gitt en oversikt over tidligere forskning som er relevant for denne studien. På denne måten får man også belyst hvor denne studien posisjonerer seg i forhold til denne forskningen.

3 Tidligere Forsking

Studien baserer seg på bidrag fra forskning innen fagfeltet CSCW, programvareutvikling og forskning på MUVE. Mye av den litteraturen som ble funnet har blitt aktivt benyttet i løpet av utformingen av studien og det vil derfor bli gitt en oversikt over hovedtrekkene ved forskningen under. Dette har også til hensikt å belyse forskningsbidraget til studien. Til slutt i kapittelet blir studien posisjonert.

3.1 Programvareutvikling

Når store arbeidsoppgaver skal utføres er det en fordel å dele arbeidet mellom teammedlemmer for å kunne utføre større og komplekse prosjekter. Programvareutvikling befinner seg i denne kategorien av kompleks problemløsning, og samarbeidsteknikker har på bakgrunn av dette blitt utviklet for å løse disse utfordringene. Av særlig relevans for oppgavens problemstilling, og et mye forsket på tema, er fokuset på distribuert arbeidssamarbeid[15]. En god oversikt over disse teknikkene er å finne i Jim Whiteheads artikkel **"**Collaboration in Software Engineering: A Roadmap" (2007). Mer spesifikk forskning på best practice for globalt distribuerte agile team, hevder at distribuerte team og utkontrakterte team kan være like produktive som et lite team som befinner seg på samme sted (Sutherland, Viktorov, Blount & Puntikov 2007). Det er dermed ikke overraskende at man også finner forskning på distribuerte Scrum team.

3.1.1 Distribuert programvareutvikling med Scrum

Hossain, Babar og Paik presenterte våren 2009 en systematisk litteraturgjennomgang forskning på global programvareutvikling med Scrum (2009). I deres gjennomgang tar de kun for seg agile praksiser i forbindelse med ledelse av programvareutviklingsprosjekter. De velger å fokusere på Scrum på bakgrunn av de samme utvalgskriteriene som denne studien, nemlig at Scrum har fokus på dag til dag prosjektledelse, og er den mest utbredte agile prosjektledelsesmetodikken.

Artikkelen åpner med å påpeke at det er en trend at industrien for programvareutvikling blir mer globalisert. Dette blir drevet frem av flere ulike faktorer, slik som forbedret nettverksinfrastruktur, komponentbasert arkitektur og økt leveransepress. Agile utviklingsmetoder oppsto som et svar på disse utfordringene,

15 Eng.: Distributed work collaboration

men artikkelen påpeker at spørsmålet om "which agile practices are effective for [Global System Development] GSD under which circumstances?" (vår ordforklaring Hossain, Babar & Paik 2009, s. 175) ikke har blitt grundig forsket på enda (Hossain, Babar & Paik 2009). Artikkelen påpeker videre at agile metoder for programvareutvikling har fått mye oppmerksomhet på grunn av den fleksible måten endrede krav blir håndtert[16,] og dets fokus på utstrakt samarbeid mellom kunder og utviklere. Videre har de observert at et økende antall GDS prosjektledere vurderer å introdusere agile praksiser.

Denne trenden stemmer godt overrens med tallene fra den tredje årlige undersøkelsen til Version One, "The State of Agile Development", hvor over tre tusen representanter fra IKT industrien i 82 land deltok i løpet av juni og juli, 2008. De deltagende firmaene svarte at 57 prosent av deres agile team var distribuert på nåværende tidspunkt. På spørsmål om hvilken agil metodologi som de benyttet mest, svarte 22 prosent en Scrum/XP hybrid, mens resten svarte at de benyttet andre metodologier (Version One 2008).

Fordi filosofien til agil programvareutvikling er basert på tett og hyppig samarbeid på samme sted kan det by på utfordringer å benytte disse i GSD, men det finnes flere rapporter fra GSD utøvere som viser til suksess (Hossain, Babar & Paik 2009). Andre forskere hevder derimot at det fremdeles er åpent til debatt om agile praksiser i det hele tatt kan benyttes i en distribuert setting (se for eksempel Abbattista, Calefato, Gendarmi & Lanubile 2008). En mulig arena for å studere dette videre er distribuert programutvikling ved hjelp av et MUVE.

3.1.2 Programvareutvikling ved hjelp av et MUVE

Det er akademisk interesse for Second Life, men forskning på programvareutvikling kombinert med et MUVE, slik denne studien er et eksempel på, finnes det lite av. Likevel finnes det forskning på MUVE som med fordel kan benyttes i denne studien, som for eksempel forskning på teamsamarbeid (Kahai, Carroll & Jestice 2007) og databasert læring i et MUVE (Dieterle & Clarke 2009).

16 Eng.: Requirement volatility

Ved å kombinere team samarbeid og databasert læring har Pullen, Norris og Fix (2000) studert hvordan man kan lære seg programmeringsspråket C++ ved hjelp av et MUVE:

> ”The MUVE lends itself well to the combination of structure, object-oriented technology and unstructured interpersonal communication needed to teach C++” (Pullen, Norris & Fix 2000, s. 70).

Hvis man kan lære seg å programmere i et MUVE er det ikke urimelig å anta at en også kan benytte et MUVE for å programmere. Et relevant argument fremlagt av Purbrick og Lentczner (2007) fra Linden Lab, er at Second Life i seg selv er verdens største programutviklingsmiljø. De argumenterer for at Second Life representerer et sosialt miljø hvor man kan programmere sammen med andre. Linden Lab benytter Second Life som en integrert del av deres utviklingsmetodologi selv når de videreutvikler kodefundamentet til Second Life: ”These experiences point toward a re-imagining of programming as a globally immersive collaborative experience” (Purbrick & Lentczner 2007, s. 720).

Denne studien benytter seg av prototyping for å demonstrere hvilke muligheter et MUVE kan tilby. Programvareutviklingsteam lager ofte slike prototyper når et system skal utvikles. Prototyper kan være alt fra enkle papirmodeller til mer avanserte modeller med tilnærmet full funksjonalitet (Kyng 1991). Colorado Technical University har undersøkt hvordan slike prototyper kan designes i Second Life (Calongne, Endorf, Frankovich & Sandaire 2008). Deres erfaringer var positive, og fikk illustrert hvor viktig brukerinteraksjon er for groupware:

> “The backend processing is important, but the user interaction can make or break a software system. Usability, from a developer’s standpoint, is making the user interface support the function and activities that the user performs. The user is most important and ease of use is often the user’s biggest concern” (Calongne, Endorf, Frankovich & Sandaire 2008, s. 194).

Denne studien benytter altså et MUVE for å lage en slik prototype, og det har blitt forsket på hvordan team kan arbeide mer effektivt sammen i Second Life. Dette forskningsarbeidet har blitt utført ved hjelp av prototyping.

3.2 MUVE og Second Life

Lucia, Francese, Passero og Tortora (2008) undersøkte hvordan team kan arbeide mer effektivt sammen i Second Life. Fordi det ikke har innebygget møtefunksjonalitet, foreslår de et system som har fått navnet SLMeeting. Dette systemet ble laget for å bedre funksjonaliteten man behøver for å håndtere og kontrollere møter. Resultatene av den preliminære undersøkelsen var lovende, men de forteller at:

> " [We] need to compare the effectiveness of SLMeeting with other meeting modalities, such as audio meeting, face to face meeting or with the support provided by commercial meeting tools" (Lucia, Francese, Passero & Tortora 2008, s. 304).

Deres fremgangsmåte har flere likhetstrekk med denne studien, men er ikke relatert til et spesifikt arbeidsdomene. Sitatet gjelder likevel for denne studien, som for deres, fordi det vil gå utover rammene for denne studien å gjennomføre en slik komparativ analyse. Når man så har klart å organisere et arbeidsmøte i Second Life, hvordan får en så til et effektivt samarbeid?

Hvis man skal utføre arbeid sammen med andre i et virtuelt miljø er det viktig å ha en måte å koordinere dette samarbeidet. Et MUVE har en rekke egenskaper som gir "Real-World Opportunities for Virtual-World Project Management" ifølge Owens, Davis, Murphy, Khazanchi & Zigurs (2009). Dette er en artikkel hvor det blir fremhevet at et MUVE kan gi bedre teamdynamikk enn tradisjonelle verktøy for virtuelle team ved å fjerne barrierer for interaksjon. Dette er ifølge Ilze Zigurs (2003) viktig fordi virtuelle team oppfører seg annerledes enn tradisjonelle team. Ifølge Owens, Davis, Murphy, Khazanchi og Zigurs (2009) oppstår disse barrierene på grunn av temporal avstand, geografisk avstand og begrenset ansikt til ansikt interaksjon. Som et svar på dette argumenterer de for at geografisk avstand ikke lenger er et hinder i et MUVE fordi avatarer kommer sammen på en felles virtuell plass for å interagere med hverandre. Kulturbarrierer kan minimeres ved at deltagerne kan lage seg en avatar med et generisk utseende uavhengig av rase og kulturvariasjoner. Hvorfor er det viktig å forsere slike barrierer? Man er avhengig av god tillitt mellom deltagerne i et team for at samarbeidet skal fungere optimalt og dette er vanskelig å få til i et tradisjonelt virtuelt team fordi medlemmene ikke kan ha direkte interaksjon med hverandre. Et MUVE kan gi et bedre fundament for tillit ved

at man kan benytte, kontrollere og kombinere verbale og ikke verbale kommunikasjonssignaler i multiple kommunikasjonskanaler, sammen med et lekende miljø som lar brukerne sosialisere og utvikle gruppefellesskap (Owens, Davis, Murphy, Khazanchi & Zigurs 2009, s. 38). Disse egenskapene gjør et MUVE egnet som arena for læring.

Simulation Linked Object Oriented Dyamic Learing Environment (SLOODLE) er et forskningsprosjekt basert på åpne kildekode som har til hensikt å "[..] integrate learning experiences with institutional VLEs [Virtual Learning Environments] and to share data with academic information systems" (vår ordforklaring, Livingstone & Kemp 2008, s. 8). Dette gjøres ved å kombinere Moodle læringshåndteringssystemet med Second Life. Forskningen viser mulighetene for samhandling som ligger i et MUVE, og prototypen i studien inkluderer dermed et rom for kurs og opplæring.

Hvordan kan så studien posisjoneres i forhold til den forskningen som har blitt presentert? Svaret på dette blir gitt i avsnittet under.

3.3 Posisjonering av studien

For å få et bedre bilde av hvor denne studien kan plasseres, er det nyttig å se til tidligere forskning som kan gi svar på følgende spørsmål: Hva har forskningstendensen vært innenfor MUVE sett i sammenheng med distribuert programvareutvikling?

I avsnitt 2.1.1 ble det vist hvordan prototypen kan klassifiseres i forhold til tid og sted ved bruk av tid/sted matrisen til Ellis, Gibbs og Rein (1991). Primært er prototypen designet for å støtte samarbeid på ulikt sted til samme tid, selv om den som et Common Information Space også støtter andre former for samhandling. Forskningen det har blitt sett på i dette kapittelet befinner seg primært i denne kvadranten av tid/sted matrisen. Det har vært noe forskning på MUVE innenfor databasert læring, og noe forskning ved hjelp av prototypeutvikling i et MUVE, men lite på distribuert programvareutvikling ved hjelp av et MUVE. Derimot er det forsket en del på distibuert programvareutvikling og hvordan dette kan bli sett på som cooperative work. Denne studien befinner seg altså i grenselandet mellom disse forskningsgrenene, og har basert seg på resultatene fra dem.

Articulation work, som beskrevet i avsnitt 2.3, er et viktig element i organisering av kodebasen[17] som benyttes for å skape en applikasjon. Oppgavene som må til for å samarbeide mot en felles kodebase innbefatter tilretteleggelse, planlegging, koordinasjon, og overvåkning av de nødvendige stegene for å utføre en samarbeidsoppgave. Stegene kan i denne sammenhengen innebære overlevering av kode til kodebasen eller diskusjoner om ulike deler av programkode (som for eksempel Backlog Items). Samarbeidsoppgaven kan være å skape et nytt dataprogram. Organisering av koden er en vesentlig del av programutvikling når flere aktører er engasjert i et samarbeid, og i denne studien er det viktig å støtte aktivitetene som er relatert til organisering av koden mellom samarbeidende programmerere. Aktivitetene kan støttes ved å tilby muligheter for at samarbeidsdeltagerne kan utføre articulation work. Det finnes mange metodologier for å strukturere programmeringssammarbeid, slik at man utvikler god kode effectively og efficiently, med den hensikt å leverer det kunden har bedt om innenfor gjeldende økonomiske rammer; dette gir satisfaction [18]. I denne studien har det blitt sett på Scrum metodologien. Det har vært viktig å forutse de eventualiteter som kan inntreffe under samarbeidet mellom flere programutviklere, slik at brukerne av prototypen kan artikulere samarbeidsaktivitetene sine. For å få klarhet i disse eventualitetene har det blitt utført en kvalitativ undersøkelse for å finne ut av dagens arbeidspraksis, som beskrevet i kapittel 6. I kapittel 7 blir det vist hvordan designet av prototypen reflekterer dette.

I neste kapittel presenteres forskningsspørsmålene som er relevant for studien. Videre vil det blir redegjort for forskningstilnærming og metodevalg, før det blir sett på hvordan datainnsamling og metode har blitt utført i kapittel 5.

17 Eng.: Repository
18 Effectively, efficiently og satisfaction er forklart i avsnitt 4.6.1

4 Forskningstilnærming og metodevalg

Dette kapittelet vil først gi en oversikt over studiens forskningsspørsmål. Deretter vil det bli redegjort for designforskning, valg av datainnsamlingsmetoder og evalueringsmetoder for det videre arbeidet. En metode er en fremgangsmåte for hvordan man går frem for å fremskaffe og etterprøve kunnskap (Kvale 1997). Formålet er altså å illustrere hvilke perspektiver og fremgangsmåter som er valgt for å besvare forskningsspørsmålene i studien. Hensikten med studiens forskningsdesign kan oppsummeres slik:

> ”A good design for a study, like a good design for a ship, will help it safely and efficiently reach its destination. A poor design, one in which the components are not well integrated or are incompatible, will at best be inefficient, and at worst will fail to achieve its goals” (Maxwell 2005, s. 10).

4.1 Forskningsdesign

I en studie som denne er det viktig å ha et mål for det man ønsker å finne ut av. Hvilke forskningsspørsmål skal besvares? I kapittel 2 har relevant teorien blitt gjennomgått og i kapittel 3 har det blitt redegjort for tidligere forskning. Dette har gitt en dypere forståelse av problemområdet. Ut i fra disse to kapitlene er det mulig å innsnevre mot det feltet som det fokuser på i studien. Dette området er identifisert som *distribuert programvareutvikling med Scrum i et virtuelt miljø*. Denne betegnelsen har kun blitt benyttet for å posisjonere studien i forhold til tidligere forskning, og fremheve kjerneområdet i studien, slik som vist i forrige kapittel.

Følgende forskningsspørsmål skal det arbeides videre med:

1. Hvordan samarbeider distribuerte programvareutviklingsteam og hvilke verktøy benytter de i sin arbeidspraksis?
2. Hvordan kan en slik praksis støttes av et MUVE?

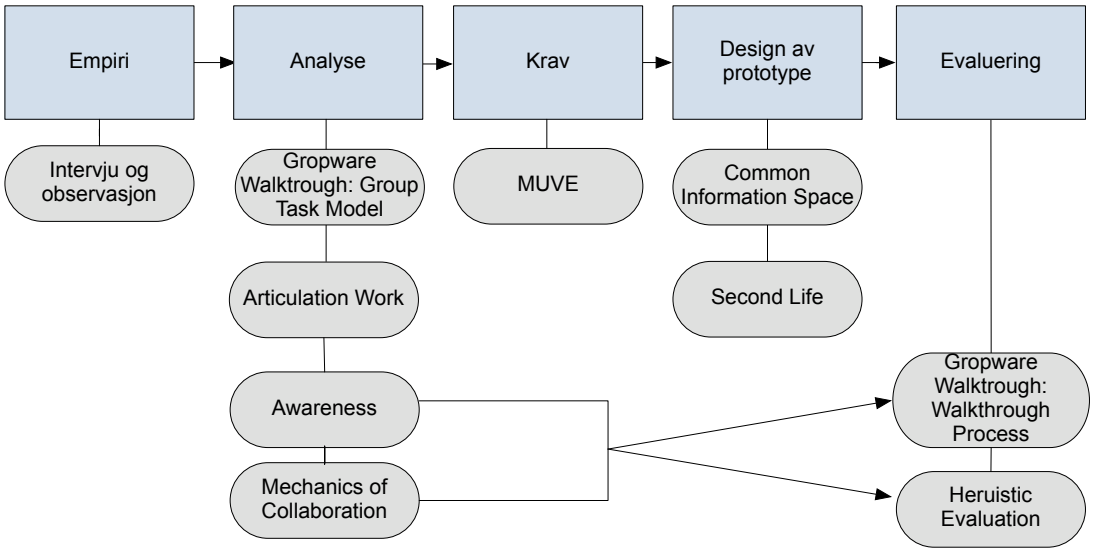
For å besvare forskningsspørsmålene har det blitt utført følgende forskningsaktiviteter:

- Intervju og observasjon.
- Utvikling av et virtuelt workspace i et MUVE som kan forbedre og forenkle den eksisterende måten å drive distribuert programvareutvikling.
- Evaluering for å finne ut hvorvidt workspacet fungerer som et verktøy for å bistå i distribuert programvareutvikling.

Ved å svare på det første spørsmålet søker man å få et innblikk i hvordan den eksisterende arbeidskonteksten fungerer. For å svare på dette har det blitt gjennomført en kvalitativ undersøkelse med intervju og observasjon som er beskrevet i kapittel 5. Resultatene har blitt analysert og presentert i kapittel 6. Det er svært viktig at man har god forståelse av arbeidskonteksten og hvordan samarbeidet fungerer, før man kan lage en groupware løsning. Dette betyr at man kan bruke det første forskningsspørsmålet som en inngangsportal til det andre spørsmålet, hvor en ser på hvordan et MUVE kan benyttes for å støtte eksisterende praksis for programvareutviklingsteam.

Teori fra CSCW-fagfeltet ble presentert i kapittel 2 og 3, og fremgangsmåter for studien blir presentert i dette kapittelet. Dette blir benyttet for å svare på det andre forskningsspørsmålet, ved at det utvikles en groupware løsning som presenteres i kapittel 7 og evalueres i kapittel 8. Figur 6 viser hvordan studiens forskningsdesign er bygget opp.

Formålet med prototypen er å forbedre og/eller forenkle arbeidsmåtene som har vært i bruk tidligere. I denne sammenhengen innebærer det å fjerne eller minske problemene som ble identifisert i analysen av den kvalitative undersøkelsen. Studien har til hensikt å bidra til mer kunnskap, ikke å presentere en endelig groupware løsning for hvordan man skal løse denne type utfordringer. Resultatene fra evalueringen kan, for eksempel, være med på å bygge et fundament for videre arbeid på denne typen forskningsspørsmål.



Figur 6 - Studiens forskningsdesign

4.2 Designforskning

Målet med designforskning[19] er å utvide horisonten til mennesker og organisasjoner ved å designe nye innovative løsninger. Artikkelen til Hevner, March, Park and Ram (2004) beskriver et rammeverk og retningslinjer for å utføre god designforskning. Artikkelen fokuserer primært på teknologibasert design og setter opp syv retningslinjer for å designe gode løsninger som vist nedenfor.

19 Eng.: Design-Science Research

Tabell 4 – Retningslinjer for designforskning (vår oversettelse fra Hevner, March, Park & Ram 2004, s.83)

Retningslinje	Beskrivelse
1: Design som et artefakt	Designforskingen må lage et synlig artefakt representert via en modell, metode eller eksempel.
2: Problemrelevans	Målet for designforsking er å utarbeide teknologibaserte løsninger for å forbedre ulike behov.
3: Design evaluering	Nytten, kvaliteten og verdien av artefaktet må testes grundig gjennom velprøvde metoder. Slike metoder kan være observasjon, ulike analyser, testing og lage ulike brukssenarioer.
4: Forskningsbidrag	Resultatet av forskingen må være et bidrag innen det fagområdet man han utviklet artefaktet.
5: Grundighet[20]	Forskingen skal være basert på rigide metoder både når det kommer til konstruksjon og evaluering av det utviklede artefaktet.
6: Design som en utforskningsprosess	Under letingen etter den optimale løsningen må man hele tiden være sikker på at man følger de lover og regler som gjelder innen det gitte forskingsfeltet.
7: Forskningsformidling	Resultatene av forskningen må presenteres på en god måte, både for teknologiorienterte og ledelsesorienterte personer.

Forskning kan sies å være en aktivitet som bidrar til forståelse av et fenomen. Fenomenene som studeres er i mange tilfeller naturlig, enten som del av samfunnet eller naturen. Forskere forsøker ofte å forstå fenomener uten å påvirke dem direkte. Ifølge Association for Information Systems (AIS) er designforskning en tilnærming som derimot baserer seg på forskerens påvirkning av fenomenene som skal undersøkes (Vaishnavi & Kuechler 2004/5). Designforskning handler ifølge AIS om:

> "[..] the analysis of the use and performance of designed artifacts to understand, explain and very frequently to improve on the behavior of aspects of Information Systems. Such artifacts include - but certainly are not limited to - algorithms (e.g. for information retrieval), human/computer interfaces and system design methodologies or languages" (Vaishnavi & Kuechler 2004/5, fra nettside).

Forskerens rolle er altså først å designe ved hjelp av artefakter[21], for deretter å forstå, forklare og eventuelt forbedre aspekter ved informasjonssystemer. En av grunnleggerne av informasjonsbehandling, Herbert Simon, drar et tydelig skille

20 Eng.: Research Rigor
21 Et informasjonssystem kan bestå av mange artefakter

mellom vitenskapen om det kunstige[22] og naturvitenskapene[23] (1996). Førstnevnte er et av opphavene for designforskning. Han beskriver naturvitenskapene som:

> "[…] a body of knowledge about some class of things - objects or phenomena - in the world: about the characteristics and properties they have; about how they behave and interact with each other" (Simon 1996, s. 1).

Vitenskapen om det kunstige omfatter på den annen side menneskeskapte objekter eller fenomener som er fremstilt for å tilfredsstille noen forhåndsdefinerte mål. Herbert Simon (1996) beskriver dette forholdet som en dikotomi, der man skiller mellom det *normative*, hvordan noe bør være og det *deskriptive*, hvordan noe er. Inndelingen er ikke plassert innenfor en spesiell forskningstradisjon, det være seg samfunnsvitenskap eller naturvitenskap; dikotomien er et forsøk på å skille mellom [1] det menneskeskapte og hvordan det bør være (for å nå forhåndsdefinerte mål), og [2] det naturlige og hvordan det er. Det er også mulig, ifølge Herbert Simon, å anvende deskriptive metoder fra naturvitenskapene for å forstå det menneskeskapte artefaktet (1996).

Dette vil med andre ord si at veien mot forståelse starter ved å designe det som skal forstås. Hva er så design? Flere av de populære forklaringene på hva design er, definerer design i forhold til dets hensikt i stedet for karakteristikkene til artefaktet som blir designet. Bruce Archer forklarer at "design is that area of human experience, skill and knowledge which is concerned with man's ability to mould his environment to suit his material and spiritual needs" (Archer 1973, sitert i; Banks 1994, s. 26). Design er altså en prosess hvor man tar noe fra dets nåværende tilstand til en ønsket tilstand.

Det er i dag vanlig å se på design som bestående av et *produkt* og en *prosess*. Artefaktet som skal bli realisert er selve produktet. Prosessen er selve planleggingen, slik at alle kravene til artefaktet blir tilfredstilt (Walls, Widmeyer & Sawy 1992).

Begrepet *design* er isolert sett klart normativt; man har en plan for å få et fenomen eller objektet fra slik det *er* til slik det *bør* være. Hvordan kan så design bli en del av

22 Eng.: Sciences of the artificial
23 Eng.: Natural sciences

forskning? Dette får man ved å benytte deskriptive aktiviteter, der målet er å forstå hvordan artefaktet *er* eller *fungerer*. Walls, Widmeyer og Sawy (1992) hevder at en designer kan kalles en forsker hvis han konstruerer redskaper for å teste teorier (Walls, Widmeyer & Sawy 1992, s. 38).

Designet av prototypen blir i denne studien den normative aktiviteten, og evalueringen av den utgjør den deskriptive aktiviteten. Empirien som forteller hvordan applikasjonen *bør* designes, er basert på tidligere forskning, intervjuer med aktører innen programvareutvikling og observasjon hos et programvareutviklingsfirma. Ved å evaluere prototypen søker man å forstå hvordan applikasjonen *er*, altså måler man hvordan den fungerer (ved hjelp av et sett med måleegenskaper) når flere aktører anvender den.

4.2.1 Designforskning på informasjonssystemer

Innenfor forskning på informasjonssystemer (IS) finnes det, ifølge Hevner, March, Park og Ram (2004), to distinkte forskningsparadigmer som kan illustrere designets innflytelse på forskning. I deres artikkel om designvitenskap i IS forskning, forklares det hvordan de to perspektivene illustrerer design sin innflytelse på forskning. Paradigmene er atferdsforskningsparadigmet og designforskningsparadigmet. Designforskning er fokusert på å utvide grensene for menneskelige og organisatoriske evner ved å skape *nytteredskaper*[24]. *Atferdsforskning*[25] søker, på den annen side, å utvikle eller bekrefte teorier som har til hensikt å forstå eller forutsi menneskelige eller organisatoriske fenomener innenfor analyse, design, implementering, ledelse og bruk av IS. Dette paradigmet er altså opptatt av *sannferdige*[26] fenomensforklaringer (Hevner, March, Park & Ram 2004).

Nå er det ikke slik at disse paradigmene er isolert fra hverandre; evaluering av nytteredskaper bidrar til sannferdig teori (og fenomensforklaringer) som kan videreutvikles og forbedres, og sannferdige fenomensforklaringer gir informasjon om hvordan man best kan designe nytteredskaper (Hevner, March, Park & Ram 2004, s76-80).

24 Eng.: Utility
25 Eng.: Behavioral Science
26 Eng.: Justified theory

Hvorfor har det blitt valgt å benytte prinsipper fra designforskning i denne studien? Schmidt og Bannon (1992, s. 12-13) fastslår at etablerte forsknings- og utviklingsmiljøer, som for eksempel innenfor datastøttet programvareutvikling[27], design[28], fremstilling[29] og kontorstøttesystemer[30], er alle gyldige og nødvendige for å få innsikt i kravene og karakteristikkene til samarbeid. Det å benytte designforskning passer derfor godt i sammenhenger der forskning innenfor CSCW utgjør det teoretiske rammeverket (Schmidt & Bannon 1992). Designforskning kan også være en nyttig fremgangsmåte for å løse grunnleggende problemer man møter når en skal ta i bruk informasjonsteknologi. Utgangspunktet er ofte at man har identifisert et problem innenfor sosial eller organisatorisk arena, og man ønsker å skape en løsning som vil være verdiskapende (Hevner, March, Park & Ram 2004, s. 77). Videre har det blitt presentert en kontekst med utfordringer og problemer. En av flere fremgangsmåter for å løse disse utfordringene er å designe et artefakt. Dette passer godt overens med måten forskningsspørsmålene har blitt formulert. Det første forskningsspørsmålet søker å forstå hvordan en aktivitet forløper seg i eksisterende praksis, og identifisere problemer. Det andre spør hvordan en slik praksis kan støttes av et MUVE, og først her blir det aktuelt å snakke om design. En måte man kan forbedre det eksisterende samarbeidet er nettopp ved å benytte et artefakt. Artefaktet har da til hensikt å gi oss svaret på ”hvordan”. Hvordan kan man så få klarhet i om dette artefaktet har vært med på å forbedre eller forenkle? Fordi artefaktet ikke snakker for seg selv, må man som nevnt, teste et eller flere aspekter av artefaktet gjennom en evaluering. Valg av evalueringsmetoder blir drøftet senere.

Et viktig spørsmål er om det egentlig er noen forskjell mellom designforskning og alminnelig design/utvikling av et informasjonssystem. Alminnelig design/utvikling av et system går ut på å benytte eksisterende kunnskap til å løse et sosialt eller organisatorisk problem. Kunnskapen som blir benyttet for å skape et slikt system vil allerede være en del av den felles IS-kunnskapsbasen og dermed allment kjent innenfor IS-forskning. Denne består av frukter av tidligere IS-forskning, som består av teorier, rammeverk, verktøy, metoder, modeller, med mer, som gjerne blir samlet i egne metodologier, og som er med på å skape et fundament for videre forskning. I

27 Eng.: Computer-Aided Software Engineering (CASE)
28 Eng.: Computer-Aided Design (CAD)
29 Eng.: Computer Integrated Manufacturing (CIM)
30 Eng.: Office Information Systems (OIS)

denne forbindelse streber designforskning etter å tilføye noe nytt, eller med andre ord, å løse ubesvarte problemstillinger på en mer effektiv eller innovativ måte, og på denne måten gi et kunnskapsbidrag til IS-kunnskapsbasen. En felle man må være klar over er at en velbegrunnet teori som ikke er nyttig, bidrar like lite til IS litteraturen, som et artefakt som løser et ikke-eksisterende problem (Hevner, March, Park & Ram 2004, s. 80-81). Vesensforskjellen finner man altså i forholdet mellom det å benytte kunnskap fra IS-kunnskapsbasen, og det å tilføye kunnskap til den.

4.3 Kvalitative metoder

Vitenskapelig metode er teknikker eller fremgangsmåter som man benytter for å finne svar på forskningsspørsmål (Ringdal 2007). Studien har benyttet kvalitativ metode fordi et overordnet mål er her å utvikle forståelsen av enkelte fenomener knyttet til personer og situasjoner i den sosiale virkelighet (Dalen 2001, s. 16). Kvalitativ metode gir en forståelse for menneskelig oppførsel og årsakene til oppførselen. Den utforsker hvorfor og hvordan mennesker gjør sine valg, i motsetning til en kvantitativ tilnærming som utforsker hva, hvor og når mennesker gjør sine valg (Yin 2003). Studien benytter observasjon og intervju for å få frem hvorfor Scrum benyttes, hvordan Scrum team samarbeider, og for å få innspill til hvordan de kan samarbeide i et MUVE.

Observasjonen har vært todelt. For det første har det blitt observert hvordan man samhandler i Second Life, samt i andre MUVEs, med den intensjon å få frem hvilke løsninger som fungerer bedre enn andre. For det andre har det blitt observert hvordan et distribuert Scrum team samarbeider i en IKT bedrift i Bergen. Det har også blitt hentet inn systemlogger, samtalelogger, med mer. Her er det viktig at man forholder seg til gjeldende juridiske regler, å være klar over at en samtale som blir logget, nødvendigvis ikke blir den samme som hvis man ikke hadde gjort dette. Det er et skille mellom det å ha en formell, og det å ha en uformell samtale. I tillegg er det viktig å tenke på hvilke former for data som er av interesse, og hvordan man kan få samlet inn dette. Det har i løpet av datainnsamlingsfasen blitt intervjuet ni informanter. Oversikt over disse er gitt i tabell 6. Det er viktig å være bevist på den etiske siden til forskning i en slik prosess.

4.4 Etikk

Steinar Kvale forklarer at ”etiske avgjørelser hører ikke til noen enkelt del av intervjuundersøkelsen, men må foretas gjennom hele forskningsprosessen” (Kvale 1997, s. 65). Når man foretar en kvalitativ undersøkelse står man hele tiden ovenfor valg. Mange av disse valgene har en etisk side. For å ivareta anonymiteten til informantene har det ikke blitt benyttet navn på personer eller firma i denne studien. Dette for å ivareta den enkeltes anonymitet så godt som mulig. Det å være med har vært helt frivillig og ikke en arbeidsoppgave pålagt av arbeidsgiver.

4.5 Prototyping

Second Life, og tilsvarende MUVE, har gode muligheter for å lage ny funksjonalitet. Målet for studien har vært å utvikle et produkt som løser noen av de utfordringer en står ovenfor når man skal benytte distribuert Scrum. Dette har blitt utforsket ved hjelp av en prototype som beskrevet i kapittel 7. En prototype kan være:

> ”[…] anything from a paper-based storyboard through to a complex piece of software [konseptuel], and from a cardboard mockup to a molded or pressed piece of metal [fysisk]” (Sharp, Rogers & Preece 2007, s. 241) [vår fremheving av hva som er fysisk og konseptuel prototype, i henhold til deres egen inndeling].

I henhold til Sharp, Rogers og Preece (2007) er det to ulike måter å klassifisere en prototype, slik som vist i sitatet deres. Prototyper som blir utviklet i et MUVE vil være konseptuelle av natur, fordi man ikke kan ta og føle på løsningen. Når man designer en prototype er det to alternative fremgangsmåter. Man kan enten velge å designe en low-fidelity eller en high-fidelity prototype. Hvis studien hadde valgt førstenevnt, kunne det blitt laget en papirmodell over møteromsløsningen, slik man hadde forestilt seg at den skulle være i et MUVE. Når studien nå har valgt å lage en high-fidelity prototype, er det fordi Second Life inneholder mye basisfunksjonalitet, slik at kostnadene og tidsbruken for å designe en komplett løsning, ikke blir uforholdsmessig høy. Prototypen vil også fungere som en levende spesifikasjon for hvordan en slik løsning kan være, noe som vil kunne gi et bidrag til IS-kunnskapsbasen. Løsningen vil også være mulig å teste i bruk, slik det har blitt gjort i denne studien.

Ved å skape virtuelle områder, bygninger, med mer, kan man demonstrere samarbeid i en virtuell verden kombinert med scriptet funksjonalitet. Fordi Second Life er konstruert av objekter, er det formålstjenelig å ha en objektorientert tankegang når man skal skape en prototype. Objekter har mange egenskaper, som bredde, høyde, farge, metoder for å utføre handlinger, med mer. Objekter kan også inneholde andre objekter, hvor hver av disse igjen vil ha sine egne egenskaper og metoder (Braadland 2002). Dette er en av grunntankene i designfilosofien til Second Life.

4.6 Evaluering

Gutwin og Greenbergs (2002) rammeverk for "Workspace Awareness of Real-Time Groupware") er som nevnt i kapittel 2 benyttet i denne studien. I kombinasjon med dette vil det være hensiktsmessig å benytte Pinelle og Gutwin sin "Groupware Walkthrough" (2002) sammen med heruistikker basert på "Heuristic Evaluation of Groupware based on the Mechanics of Collaboration" utarbeidet av Baker, Greenberg og Gutwin (2001) i analysen. Dette fordi man da kan få frem mesteparten av de mekaniske brukervennlighetsproblemene. En svakhet med denne fremgangsmåten er imidlertid at den ikke fokuserer på andre elementer, slik som brukerens følelse av tilstedeværelse i et MUVE, men dette blir dekket inn ved å benytte rammeverket for workspace awareness. For å avdekke de sosiale omkringliggende aspekter har det som nevnt blitt holdt en kvalitativ studie, fordi groupware walkthrough ikke er sterk på dette område.

4.6.1 Målekriterier for usability

Samarbeidsmekanismene, som ble presentert i kapittel 2, representerer basisaktiviteter i et delt workspace. Hensikten med dette avsnittet er å beskrive hvordan man måler usability for groupware. Et groupwares usability, definerer Gutwin og Greenberg (2000), som den graden et groupware system støtter samarbeidsmekanismene for: "a particular set of users and a particular set of tasks" (Gutwin & Greenberg 2000, s. 100).

Når man nå har dette klart definert er det mulig å måle hvorvidt disse gjenspeiler god usability, men man må først definere noen målekriterier. Gutwin og Greenberg (2000) mener at man kan teste for samarbeidsmekanismene i et groupware, ved å undersøke hvorvidt en gruppe kan utføre de *effectively*, *efficiently* og *pleasantly*.

Punktene under er oversatt fra engelsk (Gutwin & Greenberg s 100):

- **Effectiveness** vurderer hvorvidt aktiviteten ble vellykket utført. Her må man også ta hensyn til antall og alvorlighetsgrad av feilene som inntraff mens man gjennomførte aktiviteten. Et bruksvennlig groupware system vil ikke forhindre at samarbeidsmekanismene blir benyttet, og vil ikke forårsake at gruppemedlemmene gjør for mange feil i løpet av disse aktivitetene. Desto mindre feil, dess bedre effectiveness.

- **Efficiency** vurderer hvor mye ressurser (som tid og anstrengelse) som er krevd av brukerne for å gjennomføre en aktivitet. Samarbeidsaktivitetene kan utføres med mindre tid og anstrengelse i et godt groupware system, enn et system med dårlig usability. Her er det viktig å fokusere på arbeidsoppgavene, fordi grupper gjerne deltar i andre aktiviteter som ikke er negativt for det delte arbeidet.

- **Satisfaction** vurderer hvorvidt gruppemedlemmene var fornøyd nok med prosessene og utfallet av hver av gruppeoppgavene. Dette punktet kan noen ganger overlappe med effectiveness og efficiency. Dette fordi problemer med disse sannsynligvis vil redusere tilfredsstillelsen.

Resultatet fra en Groupware Walk Through analyse kan benyttes iterativt til å utbedre kravspesifikasjon, design eller kode, men dette har vært utenfor rekkevidde og tid for denne studien. På den annen side er det et mål med denne studien å gi et bidrag til IS-kunnskapsbasen, som ble beskrevet i avsnittet om designforskning (se avsnitt 4.2). Resultatene fra evalueringen i kapittel 8 blir derfor registrert og dokumentert med den hensikt at eventuelle effektiviseringer eller forbedringer av svakheter kan gjennomføres i en etterfølgende studie.

4.6.2 Rimelig usability evaluering

I avsnittene under presenteres de rimelige usability evalueringsteknikkene som blir benyttet i denne studien. Disse teknikkene er svært annerledes enn tradisjonelle feltbaserte fremgangsmåter for å evaluere flerbrukersystemer. Fordelen er at de kan bli benyttet tidlig i designfasen, er billig, og de trenger ikke ekte brukere eller fullt fungerende prototyper. Nå er det et problem at de ikke benyttes på en arbeidsplass, og dermed ikke tar høyde for de komplekse kulturelle og organisatoriske aspektene ved gruppearbeid. Dette har til konsekvens at disse faktorene blir overlatt til andre teknikker. Årsaken er at de i stedet fokuserer på aktiviteter, som en interagerende gruppe benytter for samarbeid, som er på et lavere nivå (Gutwin & Greenberg 2000).

4.6.2.1 Heuristisk evaluering

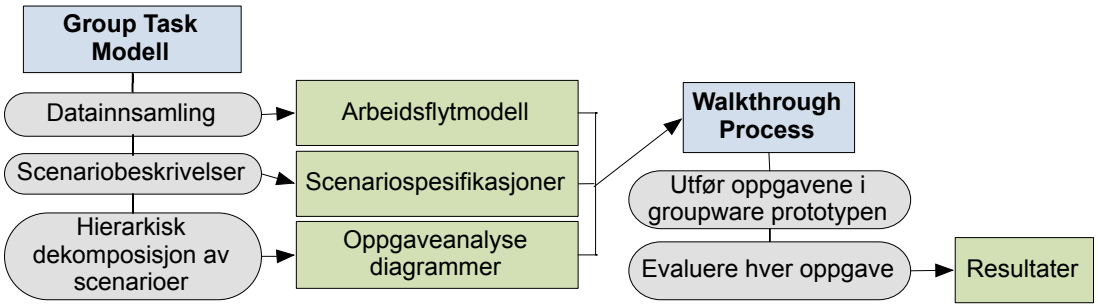
Dette er en bredt akseptert og rimelig evalueringsmetode for å diagnostisere potensielle usability problemer i et brukergrensesnitt (Nielsen & Molich 1990). Denne metodologien benytter et lite sett med usability eksperter som visuelt inspiserer et brukergrensesnitt, hvorpå de vurderer om det oppfyller usability prinsippene (heuristikkene) til for eksempel 'Systemstatus skal vises' eller 'Samsvar mellom systemet og den virkelige verden'. Heuristikker er generelle regler som blir benyttet for å beskrive vanlige egenskaper ved et grensesnitt (usable interfaces), eller sagt på en annen måte; heuristikker er anvendte designprinsipper. Heuristisk evaluering har følgende fordeler; [1] tidseffektiv og kan fullføres i løpet av noen timer; [2] trenger ikke deltagelse fra sluttbrukere; [3] godt dokumentert, slik at de blir enkle å benytte, [4] behøver ikke å være bruksevalueringsekspert for å få et bra resultat; [5] populær blant forskere og industri; [6] tre til fem inspektører vil til sammen vanligvis finne 75-80% av alle bruksproblemene (Sharp, Rogers & Preece 2007, vår oppsummering).

Heuristisk evaluering av groupware (Baker, Greenberg & Gutwin 2001) modifiserer heuristisk evaluering ved å utvikle gruppespesifikke heuristikker basert på samarbeidsmekanismene. Hensikten til Baker, Greenberg og Gutwin (2001) er altså å benytte seg av fordelene til heuristisk evaluering for å evaluere datastøttet samarbeid. Gruppeheuristikkene blir presentert og benyttet kapittel 8, hvor prototypen blir evaluert.

4.6.2.2 Groupware Walkthrough

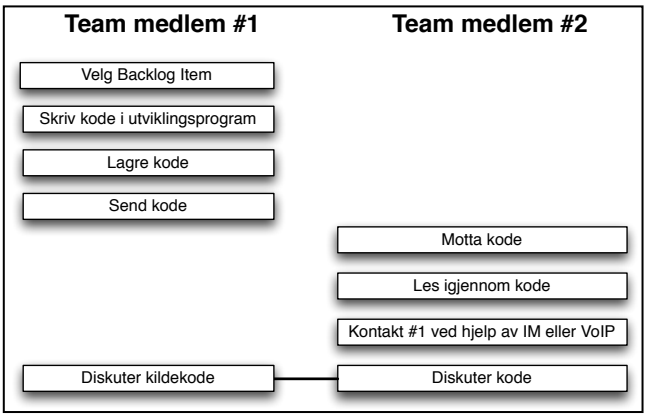
Groupware Walkthrough (GWW) er en rimelig teknikk som kan identifisere samarbeidsspesifikke usability problemer og kan identifisere problemer man ikke ville fått frem ved hjelp av andre inspeksjonsmetoder. Den består av to faser, som er en *Group Task Model*[31] (GTM) for å identifisere og analysere samarbeidsoppgaver og en *Walkthrough Process*[32] for å vurdere hvor godt systemet som designes for å støtte disse oppgavene, fungerer. GTM består, som figur 7 viser, i hovedsak av tre analyseaktiviteter, hvor datainnsamling er den første (Gutwin & Pinelle 2002).

31 Nor.: Gruppeoppgavemodell
32 Nor.: Gjennomgangsprosess



Figur 7 - Groupware Walkthrough oversikt, basert på (Gutwin & Pinelle 2002)

I denne studien har datainnsamlingen bestått av intervju av programvareutviklere og observasjon hos et IKT firma. Aktivitetens hensikt er å kunne skille mellom individuelt arbeid og samarbeid fra den eksisterende konteksten. Det kan være hensiktmessig å utarbeide en arbeidsflytmodell for å identifisere hvor samarbeidsaktivitetene foregår mellom to eller flere brukere. GWW er ikke interessert i individuelle aktiviteter, fordi disse kan evalueres ved hjelp av rammeverk tilpasset enkeltapplikasjoner (Gutwin & Pinelle 2002). Figur 8 viser et eksempel på en arbeidsflytmodell av aktivitetene fra arbeidet til to programmerere. I modellen kan man identifisere hvor det foregår samarbeid. Dette er vist med en strek mellom teammedlemmene der de diskuterer kildekode.



Figur 8 – Arbeidsflytmodell

Den andre aktiviteten i GTM er scenariobeskriveskrivelser, hvor man lager en deskriptiv formalisering av arbeidet, hvem som inngår i arbeidet, og den kunnskap de besitter. Disse scenariospesifikasjonene bør i følge Pinelle og Gutwin

(2001) inneholde beskrivelse av samarbeidsaktivitetene og:

> "[…] a description of the desired outcome (the group goal), a description of the group of likely users, and a description of the circumstances under which the scenario is commonly performed" (s. 105).

Dette gir viktig kontekstuell informasjon når GWW blir gjennomført.

Fokuset er på samarbeid, men det er viktig at man ikke neglisjerer de individuelle handlingene som leder opp til samarbeidet. Scenariobeskrivelsen, som vist i tabell 5, er basert på samarbeidet som ble identifisert i arbeidsflytmodellen (se figur 8). #1 og #2 uttrykker forskjellige brukerroller, og i følgende scenarioet er brukerne begge programmerere.

Tabell 5 – Scenariospesifikasjon (Gutwin & Pinelle 2002)

Scenario: Diskutere kildekode

Aktivitetsbeskrivelse. Team medlem #1 ønsker å diskutere kildekode med et annet team medlem . Han må først få tak i team medlem #2, og høre om han eller hun har tid til å bistå. Deretter må han spesifisere hvilken kode han ønsker å diskutere. Når dette har blitt klargjort, diskuterer #1 og #2 den aktuelle koden.
Brukerspesifikasjon. Teammedlemmer er ansvarlig for at programutviklingsarbeidet går så smidig som mulig. En faktor for å oppnå dette er å bistå hverandre ved å dele kunnskap.
Tilsiktet utfall og resultat. Meningsutveksling om koden.
Forhold/omstendigheter. De to teammedlemmene arbeider fra ulike steder. De benytter e-post til asynkron kommunikasjon, samt IM, telefon og VoIP for synkron kommunikasjon. De utveksler kode via programutviklingsverktøy til kodebasen.

Scenarioene danner grunnlag for en hierarkisk dekomposisjon, som er den tredje analyseaktiviteten i GTM. Målet er her å dekomponere scenarioene til oppgaver. Her uttaler man hva som skjer i scenarioet, hvorpå man deler disse opp i individuelle og samarbeidende deloppgaver. Dette er illustrert i figur 9, hvor hovedoppgavene beskriver *hva* som skjer i et scenario, mens underoppgavene beskriver *hvordan* en aktivitet blir utført.

- Scenario
  - Hovedoppgave
    - Individuell deloppgave
    - Samarbeidende deloppgave

Figur 9 - Hierarkisk inndeling av oppgavene, oversatt fra Gutwin & Pinelle (2002, s. 457)

Nå er det slik at fremgangsmåten ikke har til hensikt å:

> ”[…] determine the absolute sequence of tasks that occur in the scenario, but to identify activities that should be supported in order to cover a reasonable range of possible alternatives” (Gutwin & Pinelle 2002, s. 458).

Gutwin og Pinelle (2002) forklarer at det er hensiktsmessig å vise oppgavene i et analysediagram[33]. I et slikt diagram kan man vise forholdet mellom hovedoppgaver og deloppgaver, oppgaveflyt, og alternative fremgangsmåter som man kan benytte for å løse en gitt oppgave. Man spesifiserer deloppgaver ved å vurdere hvordan disse blir utført i eksisterende praksis. Med andre ord identifiserer man hvilke samarbeidsmekanismer som blir benyttet. Når analysediagrammene er ferdig utformet, kan GWW prosessen gjennomføres for en bestemt prototype.

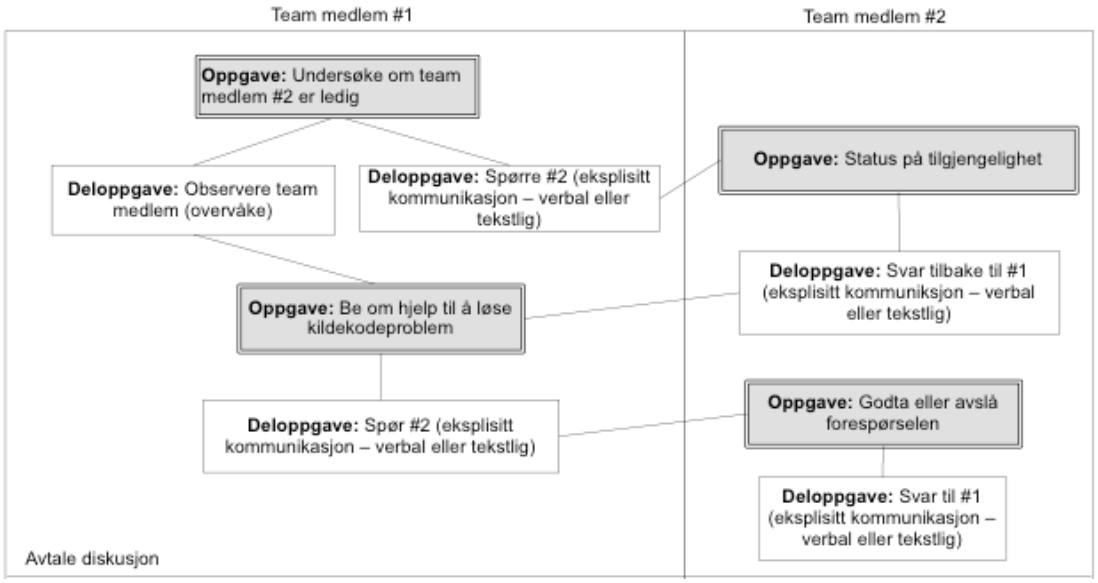
I walkthrough prosessen går evaluererne[34] gjennom oppgavene for å vurdere hvor godt grensesnittet støtter gruppen i å arbeide mot, og oppnå et bestemt mål. Det er viktig å være klar over at alternative løsningsmetoder er mulig, og at alle resultater må registreres. Teknikken fungerer på alt fra low-fidelity prototyper (se avsnitt 4.5) til fungerende applikasjoner, men er ”intended to be formative, where results are used as redesign information in an iterative design cycle” (Gutwin & Pinelle 2002, s. 458). I avsnitt 7.4.1 blir det forklart hvordan dette er tatt hensyn til i denne studien.

I denne studien har evalueringen blitt gjennomgått med et begrenset antall evaluerere. Dette på bakgrunn av at Gutwin og Pinelle (2002) påpeker at flere evaluerere kompliserer evalueringen og overvåkningen av samarbeidsmekanismene. Videre fremhever de at få evaluerere kan være tilstrekkelig for å avdekke mangler ved usability. Det kan likevel være hensiktsmessig å benytte flere evaluerere for å håndtere ulike brukerspesifikasjoner og kompleksiteten til oppgaver som skal utføres samtidig. Når man har flere evalurere tilgjengelig kan walkthrough prosessen utføres på to måter. Enten kan man gå igjennom alle oppgavene sammen for hver bruker, og så diskutere egenskapene til systemet. Alternativt kan hver evaluerer ta på seg rollen til en av brukerne og utføre oppgavene som er spesifisert i analysediagrammet.

33 Eng.: Task analysis diagram
34 Eng.: Usability expert

Sistnevnte har også blitt benyttet i denne studien.

Et slikt analysediagram blir illustrert i figur 10. Dette diagrammet viser oppgaver som ble identifisert gjennom den hierarkiske dekomposisjonen av scenariobeskrivelsen fra tabell 5. Scenarioet skildrer hvordan to teammedlemmer kan diskutere kildekode. Hovedoppgavene illustreres i rektangler med ramme rundt og deloppgavene vises i vanlige rektangler. Hver underoppgave samsvarer med en av samarbeidsmekanismene som beskrevet i Gutwin & Greenberg (2000). Dette vises i parentes.



Figur 10 – Analysediagram av scenarioet diskutere kildekode

Figur 10 viser et scenarioutdrag som har blitt organisert i en naturlig sammenhengende sekvens. Sekvensen har fått navnet «avtale diskusjon», og tar til før gjennomføringen av diskusjonen kan ta sted. Analysediagrammet viser to forskjellige fremgangsmåter for å finne ut om teammedlem #2 er tilgjengelig. Teammedlem #1 kan enten observere #2 eller spørre ham om han er tilgjengelig. Dette scenarioet blir benyttet under evalueringen av prototypen og får en grundigere gjennomgang der. I avsnitt 8.2 blir gjennomføringen av GWW presentert. Dette innebærer en presentasjon av brukerrollene, forskernes rolle og hvordan den ble gjennomført.

4.6.2.3 Kritikk til de rimelige evalueringsteknikkene

Human Computer Interaction (HCI) modeller som primært er basert på informasjonsprosesseringspsykologi (se det som har fått betegnelsen Nielsens Heuristikker (Nielsen & Molich 1990) kan ved første øyekast virke som de skal studere det samme som aktivitetsteori, altså på interaksjonen mellom mennesker (brukere) og objekter (interaktive systemer). Selv om bruker-system interaksjon kan sees på som en delaktivitet, hevder Kaptelinin og Nardi (2006) at "the purposeful interaction with the world cannot be limited to interaction with the user interface of an interactive system" (s. 34). HCI handler altså om interaksjon på et lavere nivå, som er begrenset til oppgaver og fokuserer på funksjonaliteten til et system, ikke dets mening. Under designet av prototypen har det blitt tatt hensyn til denne formen for oppgaver og funksjonalitet. I en tidlig versjon av prototypen var det vanskelig å forstå hva som var dører, og hva som var vegger. I tillegg var det vanskelig å forstå hvordan de kunne åpnes. Det som skulle være dører hadde dårlig affordance[35], slik at man ikke forsto hvordan de kunne åpnes, selv etter at man hadde identifisert de som dører. Etter at prototypen ble redesignet, kan man klart se hva som er dører, og hvordan de åpnes.

Forskningstilnærming, og valg av innsamlings- og evalueringsmetoder, har blitt presentert i dette kapittelet. Intensjonen har vært å beskrive fremgangsmåtene som har blitt benyttet for å gi svar på forskningsspørsmålene. I neste kapittel vil det bli redegjort for den kvalitative undersøkelsen, som skal svare på det første forskningsspørsmålet. Her blir det forklart hvilke hensyn som måtte tas, samt hvilken fremgangsmåte som blir benyttet for å analysere empirien. I kapittel 8 blir de rimelige bruksevalueringsteknikkene benyttet for å evaluere prototypen, som blir designet som svar på det andre forskningsspørsmålet.

35 En egenskap ved et objekt, eller et miljø, som lar individer utføre en handling

5 Datainnsamling og metode

I dette kapittelet blir det vist hvordan fremgangsmåtene og teknikkene som har blitt valgt for å få svar på studiens første forskingsspørsmål har blitt benyttet. En kvalitativ studie har blitt gjennomført hvor det inngikk intervju og observasjon som kartla hvordan en programutvikler arbeider og samarbeider med sine kolleger, hvilke verktøy og metodologier som benyttes, og hvordan dette samspillet fungerer i dag. Videre belyser intervjuene hvordan prosessene kunne vært bedre og kandidatens synspunkter på om et MUVE kan understøtte måten de arbeider på. Observasjonen og tre av intervjuene inngår i en case for et IKT firma i bergensregionen, heretter anonymisert som IKTBergen. Den innsamlede empirien har blitt benyttet som grunnlag for å designe en prototype, som vist i kapittel 7, hvor hensikten var å demonstrere hvordan denne formen for samarbeid kan utføres i et MUVE.

5.1 Kvalitativ studie

Den kvalitative studien hadde til hensikt å finne ut hvordan utviklingsteam arbeider sammen, slik at dette empiriske materialet kunne bli benyttet for å lage en prototype som kan støtte distribuert programutvikling i et virtuelt miljø. Kvalitative intervju ble valgt for å kartlegge hvordan utviklerne benytter de agile metodene og hvordan de samarbeider med andre utviklere i sin hverdag. Den kvalitative intervjuformen gir mulighet til å følge opp og utdype det intervjupersonen sier for å få en forbedret forståelse (Kvale 1997). I forkant av intervjuene ble det holdt et prøveintervju for å se hvordan intervjuguiden (se appendiks A) fungerte i praksis, slik at den kunne forbedres før den kvalitative undersøkelsen tok til.

I en intervjusituasjon er det viktig at man har klart for seg hva det skal spørres om og hvorfor. Dette fordi man i intervjustadiet får klargjort meninger som er relevante for studien og fjerner tvetydighet i svarene. Dette bidrar til å skape et mer pålitelig utgangspunkt for det senere analysestadiet (Kvale 1997).

Den kvalitative undersøkelsen var ikke direkte knyttet til et spesifikt MUVE, fordi målet var at intervjuobjektene skulle beskrive sine ønsker for en løsning som ville støtte deres behov og ikke bare beskrive begrensningene i Second Life.

Studiens kvalitative observasjon var todelt. Observasjon ble benyttet hos IKTBergen for å undersøke hvordan de benytter distribuert Scrum i sin hverdag. Beskrivelse og refleksjoner rundt deres hverdag blir gjengitt i kapittel 6. I det virtuelle miljøet ble det observert hvordan team samarbeider i MUVE som World of Warcraft. Shaowen Bardzell, Jeffrey Bardzell, Tyler Pace og Kayce Reed (2008) har studert dette fenomenet. Samarbeidsdynamikken her har paralleller til hvordan team arbeider sammen i den virkelige verden, og gruppestørrelsene er ofte lik den man ser i Scrum team. I artikkelen ser de på grupper på fem personer hvor medlemmene tar på seg ulike roller, hvor alle har sine egne arbeidsoppgaver, som har som hensikt at gruppen kan utføre oppgaver som de ikke kan klare alene. Sammen er de sterkere og smidigere enn de er når de står alene. Dette ser man igjen i Scrum, fordi teammedlemmene har høyere produktivitet sammen enn de ville hatt alene, og man påtar seg roller som er definert fra før. Videre ble det observert hvordan man samarbeider i et MUVE som Second Life, og på denne måten synliggjort begrensninger og muligheter for design av en prototype i dette miljøet.

5.2 Utvalg

Utvalget av de som ble intervjuet bestod av programvareutviklere som hadde erfaring med agile metoder og Scrum, masterstudenter i informasjonsvitenskap, og en doktorgradsstipendiat med programmering som spesialfelt. Informantene, deres alder og utdanningsnivå er vist i tabell 6, sammen med dato og sted for intervju. Deltagelse i programvareutviklingsmiljøet i Bergen har lettet arbeidet med å få kontakt med gode kandidater. Dette fordi man ellers kan møte motstand når man skal få innpass, ved at portvaktene, altså de som kontrollerer tilgang inn i organisasjonen, sperrer en ute (Maxwell 2004).

Tabell 6 – Informantoversikt

Kategori	Alder	Utdanningsnivå	Dato for intervju	Sted for intervju
Universitetet i Bergen				
Informant 1	24år	Masterstudent	02.06.2009	Universitetet i Bergen
Informant 2	22år	Masterstudent	03.06.2009	Universitetet i Bergen
Informant 3	30år	Doktorgradskandidat	10.09.2009	Universitetet i Bergen
Programutviklere				
Informant 4	33år	Bachelorgrad	29.06.2009	Hans arbeidsplass
Informant 5	29år	Høyskoleingeniør	01.07.2009	Hjemme
Informant 6	25år	Mastergrad	07.08.2009	Universitetet i Bergen
IKTBergen				
Informant 7	38år	Hovedfag	26.08.2009	IKTBergen
Informant 8	37år	Høyskoleingeniør	26.08.2009	IKTBergen
Informant 9	32år	Mastergrad	26.08.2009	IKTBergen

Det kan være fristende å intervjue noen fra egen omgangskrets, men å intervjue personer man står i et forhold til, for eksempel egne kolleger, kan by på utfordringer. Desto lettere tilgang, jo mer komplisert blir intervjuet (Kvale 1997).

I utgangspunktet var intensjonen å intervjue både menn og kvinner, men dette viste seg å bli vanskelig, fordi programmeringsmiljøene i Bergen er sterkt mannsdominerte. Kandidatene som ble valgt til intervju har Scrum erfaring, enten fra akademia, eller arbeidslivet. Dermed har det ikke blitt foretatt et tilfeldig utvalg, men valgt etter bestemte kriterier. Utvalget har dermed blitt styrt, noe som kan være en fordel, fordi det ikke har blitt foretatt en kvantitativ undersøkelse i forkant til hjelp for å foreta et valg. ”Det er også slik at hensikten med et kvalitativt intervju tradisjonelt sett ikke har vært å frembringe statistisk generaliserbar kunnskap” (Ryen 2002, s. 84). Videre forklarer Anne Ryen at det derfor ikke har ”noe for seg å lage et sannsynlighetsutvalg, der utvalgets sammensetning styres etter prinsipp for å lage et «mini univers»” (Ryen 2002, s. 84). Det er bedre at sammensetningen styres av at man leter etter variabler som er relevant i forhold til problemstilling eller teori (Ryen 2002). Det vil uansett ikke være mulig å få et representativt utvalg med så få intervjukandidater og dermed har utvalget blitt styrt for å få et så godt empirisk grunnlag som mulig innenfor studiens rammer. Det er lite hensiktsmessig å få et så stort datamateriale at det blir uhåndterlig, da man likevel ikke vil få et representativt utvalg. Det er bedre å prøve å lage et variert utvalg langs de variablene man ønsker å finne ut noe om (Kvale 1997).

I tillegg ble det på bakgrunn av dette besluttet å intervjue tre personer fra IKTBergen for å få frem et så godt bilde av deres hverdag som mulig innenfor studiens rammer.

Det ble opprettet kontakt med kandidatene på telefon, e-post og personlig oppmøte, hvor det ble informert om studiens formål. Det å stille opp til intervju var helt frivillig og det ble informert om at man når som helst kunne trekke seg fra intervjuet, og at intervjuene ville bli anonymisert.

5.3 Intervju og observasjon

Det kvalitative intervjuet gir muligheter, men også fallgruver. Et intervju kan ikke planlegges i detalj på forhånd og derfor blir det viktig med en intervjuguide (se appendiks A). Dette er en rettesnor under intervjuet som har til hensikt å sørge for at man kommer innom de ulike emnene man ønsker å ta opp. Et intervju bygger på samhandling mellom to eller flere mennesker, og det kan oppstå kommunikasjonsproblemer under intervjuet.

Det er også slik at det ikke er sikkert de svarer det som er rett for dem selv, men svarer det man tror den andre ønsker å høre. Før hvert intervju ble det forklart at det ikke fantes rette og gale svar på spørsmål, men at det viktige var å få frem deres meninger og hvordan de følte eller syntes. Når et intervju skal transkriberes er det viktig at dette meningsinnholdet ikke forsvinner.

Transkripsjon av intervju ble skrevet på bokmål, uavhengig av dialekt. Dette var et velbegrunnet valg fordi målet var å bevare informantenes anonymitet best mulig, men denne omskrivingen til bokmål har ikke ført til at poengene i det som er sagt har gått tapt. Etter intervjuene ble det skrevet et kort sammendrag ut fra notater som ble tatt under intervjuene, hvor beskrivelser av konteksten, stemninger og inntrykk ble tatt med. Således har vært en støtte i arbeidet med å bearbeide materialet i ettertid.

Det første intervjuet var å anse som et prøveintervju. Dette visste informanten ingenting om, fordi ønsket var å ha en så reell intervjusituasjon som mulig, slik at intervjuet kunne beholdes hvis intervjuguiden fungerte tilfredsstillende. Alle intervjuene blir tatt opp på digital diktafon som ble lånt av Universitetet i Bergen og ble gjennomført sommer og høst 2009

5.3.1 Universitet i Bergen

Alle intervjuene ble holdt i lokaler som ble stilt til disposisjon av universitetet fordi intervjuobjektene følte seg komfortable der (se tabell 6). De første intervjuene varte rundt 45 minutter, men ble gradvis lenger, slik at de siste intervjuene varte rundt 1 time 15min. Dette gikk på intervjuerfaring, fordi det var lettere å stille gode oppfølgingsspørsmål når man har fått forsøkt seg noen ganger tidligere.

5.3.2 Programutviklere

Det første intervjuet ble holdt på arbeidsplassen til informant 4 (se tabell 6). Han var tidligere Scrum Mester hos IKTBergen, og hadde i dermed mye kjennskap til Scrum. Intervju nummer to ble holdt hjemme, fordi han følte seg komfortabel der. Det er ikke alle som har mulighet til å benytte arbeidstiden til å bli intervjuet. Det tredje intervjuet ble holdt ved Universitetet i Bergen, dette fordi han nylig hadde fullført mastergrad i informasjonsvitenskap, og dermed følte seg hjemme der.

5.3.3 IKTBergen

IKTBergen er mellomstor bedrift som har kontorer i Bergen, Oslo og Stockholm. Programvaren deres har blitt installert i over 20 land, og de regner seg selv som markedsledere innen sitt markedsområde. Kontaktperson hos dem var informant 9, R&D Manager (se tabell 6). Via telefon og e-post ble det avtalt at de skulle stille tre utviklere til disposisjon som ønsket å la seg intervjue og at deres arbeidsmetoder ville bli observert. På denne måten kunne man få frem styrker og svakheter ved dagens måter å arbeide. IKTBergen har et Scrum Team med deltagere fra Bergen og Ukraina. Det daglige Scrum møte skulle starte 09:30 og det ble derfor avtalt at observasjonen kunne starte fra kl 09:00 26. August 2009. Intervjuene ble holdt samme dag i lokalene til IKTBergen. Hvert intervju ble holdt for seg selv inne på et møterom.

5.4 Datainnsamling

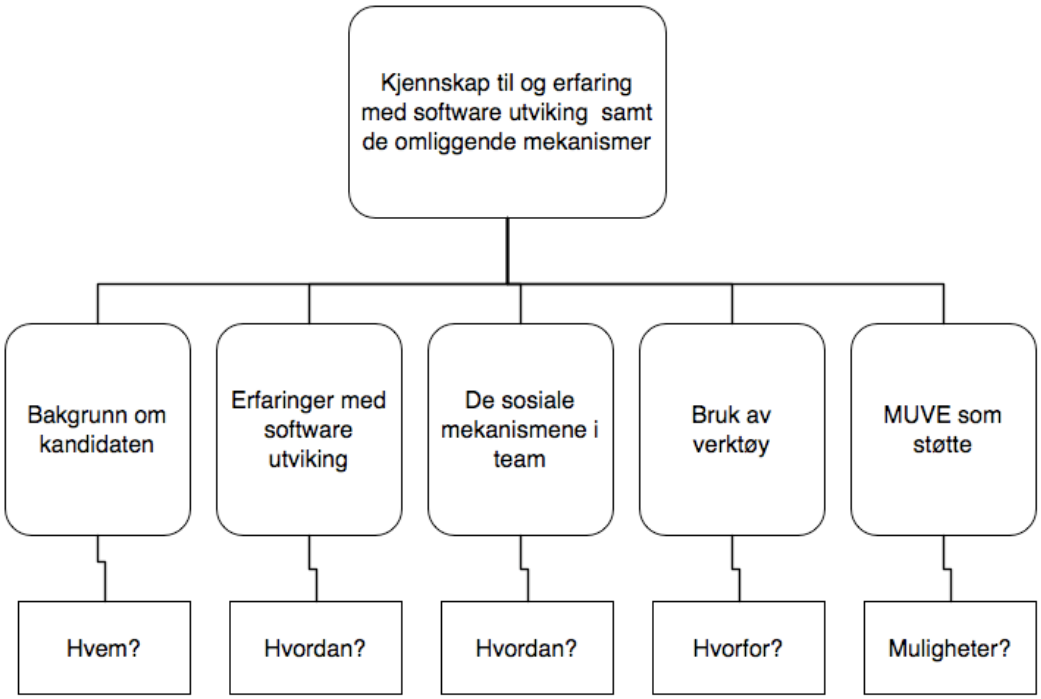
Forarbeidet før en kvalitativ undersøkelse er viktig. Det at man har satt seg godt inn i det en skal undersøke i forkant av intervjuet, er nyttig når man skal ha fokus på den man intervjuer (Widerberg & Bolstad 2001). I forkant av intervju og observasjon var det dermed en fordel å sette seg inn i hvordan programutviklere benytter Scrum, slik at man kunne stille oppfølgingsspørsmål dersom det var nødvendig.

Loggen (se appendiks A), fra det daglige Scrum møte som ble holdt under observasjonen hos IKTBergen, ble innhentet. Denne illustrerer hvordan et slikt møte blir holdt over IM. Det var en fordel å være to observatører, slik at man fikk mest mulig detaljer med seg. Beskrivelsen fra denne observasjonen blir gjengitt i avsnitt 6.2.

Intervjukandidatene, på bakgrunn av deres utdannelse og erfaring, kom med flere perspektiv på det å arbeide i et Scrum team og hvordan man kunne benytte et MUVE for å støtte denne arbeidsmetodikken. Dette ga et godt grunnlag for videre bearbeiding og analyse som er gjengitt i avsnitt 6.3.

5.5 Bearbeiding og analyse

Det innsamlede intervjumaterialet trenger å bli fortolket. I en analyse prøver man å ordne data slik at de får struktur og blir lettere å tolke (Kvale 1997). Det kan være fornuftig å sortere det informantene har fortalt, og da gjerne etter tema. Ved å ordne data slik at de får struktur kan dette gjøre tolkingsarbeidet i analysen lettere. Når en skal bestemme seg for hvilken analysetype man skal benytte, kan det være lurt å tenke gjennom hvem man skriver for. Hvem er leseren av studien? I forbindelse med utarbeidelsen av en intervjueguide (appendiks A) kan det være funksjonelt å sette opp en analyseskisse som sier noe om hva det er man ønsker å finne ut noe om. Analyseskissen for denne studien er vist i figur 11. Dette gjorde arbeidet med å gå gjennom intervjumaterialet, for å ta ut viktig informasjon og sortere det etter tema, enklere (Widerberg & Bolstad 2001).



Figur 11 - Analyseskisse som utgangspunkt for utarbeidelse av intervjueguide

Intervjuet startet med en kategori spørsmål som dekker inn utdanning og tidligere arbeidserfaring. Så ble det stilt spørsmål som handlet om deres erfaringer med og Scrum samarbeid. Her var det deres egen opplevelse, motivasjon og hvordan de opplevde de andre deltagernes motivasjon og arbeidsmestring i forbindelse med Scrum samarbeid, som var i fokus. Videre ble det spurt om hvordan Scrum samarbeidet var organisert, med særlig fokus på erfaring med de ulike rollene innenfor Scrum. Til slutt fikk de se et videoklipp på fem minutter om MUVEet MPK20, Sun sitt virtuelle arbeidssted. Dette ble gjort for at intervjueobjektene skulle forstå konseptene som resten av intervjuet handlet om. Etter det korte klippet var ferdig fortsatte samtalen omkring hvordan de så for seg at Scrum samarbeid kunne støttes i en slik virtuell virkelighet.

Etter hvert som de ferdige intervjuene ble gjennomgått ble det utformet en tabell, som vist i tabell 7, hvor utsagnene fra intervjukandidatene om de ulike temaene ble satt opp mot hverandre.

Tabell 7 – Tema for intervju

Tema for intervju
Kandidatens faglige bakgrunn
Erfaring fra programvareutvikling
Kjennskap til utviklingsmetodologier
Kjennskap og brukererfaring fra Scrum
Samarbeid i samlokaliserte programvareutvikling
Sosiale mekanismer i samlokalisert programvareutvikling
Erfaring med samlokaliserte samarbeidsverktøy
Samarbeid i distribuert programvareutvikling
Sosiale mekanismer i distribuert programvareutvikling
Nyttige verktøy ved distribuert samarbeid
MUVE som støtte for distribuert programvareutvikling

I dette kapittelet har det blitt redegjort for hvordan det var å benytte de kvalitative metodene i praksis. Her har det vært mange potensielle fallgruver, men ved nøye overveielse av faktorene i forskningsdesignen, har disse blitt unngått. I neste kapittel blir det innsamlede empiriske materialet analysert.

6 Resultater fra den kvalitative undersøkelsen

For å besvare det første forskningsspørsmålet blir det i dette kapittelet beskrevet hvordan distribuerte programutviklingsteam samarbeider, og hvilke verktøy de benytter i sin arbeidspraksis. For å få innspill til hvordan en slik praksis kan støttes i et virtuellt workspace, ble intervjukandidatene spurt om hvordan de mente et MUVE kunne støtte deres arbeid. Resultatene presenteres ved først å beskrive de observasjoner som ble gjort på bedriftsbesøk hos IKTBergen. Videre vil relevante tema fra intervjuene bli gjennomgått. Fra tema og observasjon blir et Scrum teams behov for en groupware løsning hentet ut som krav. I kapittel 7 blir kravene benyttet for å fremstille en prototype for distribuert prosjektarbeid i et MUVE.

6.1 Endring av arbeidspraksis hos IKTBergen

Scrum ble for noen år siden introdusert som arbeidsmetodikk hos IKTBergen. I løpet av denne perioden har de forsøkt ulike måter å organisere sine team. På et tidspunkt var det tre separate team i Bergen og ett i Ukraina. De endrede økonomiske rammebetingelsene som følge av, blant annet ”den såkalte finanskrisen”, har resultert i en gradvis reduksjon av bemanning. I dag har de ett team med medlemmer fra både Ukraina og Bergen. Dette har skjedd gradvis, ved at de som har sluttet ikke har blitt erstattet.

For denne studien var det viktig å studere et firma som benyttet seg av distribuert programvareutvikling, noe IKTBergen har sett seg nødt til å benytte på grunn av, blant annet, finansielle rammer. Dette fordi det er en vesensforskjell å benytte distribuert Scrum, hvor man har et sammensatt team fordelt på flere ulike lokasjoner i forhold til Scrum of Scrums, hvor flere ulike team arbeider på det samme produktet (se avsnitt 2.6.3). Forskjellen består i hvordan team medlemmene samarbeider, og hvordan de kommuniserer med hverandre. Det kan argumenteres for at distribuerte Scrum team har den største utfordringen når det gjelder å få til et godt samarbeid, fordi de er mye mer avhengig av groupware og andre IKT løsninger (Dittrich, Randall & Singer 2009).

6.2 Scrum hos IKTBergen

Den 26, august 2009 ble arbeidspraksisen IKTBergen observert. Ved ankomst ble vi ønsket velkommen, hilst på alle medlemmene i teamet som arbeidet i Bergen, og introdusert oss selv. Før møtet begynte hadde vi en hyggelig samtale der vi fikk kaffe og hvor vi fortalte litt rundt prosjektet vårt. Vi forklarte hva vi mente de kunne bistå oss med, samt hva vi kunne tilby dem, og de fortalte litt om firmaets historie.

Bakgrunn

IKTBergen ble etablert i 1989 og er ifølge dem selv Europaledende innen sitt fagfelt. Kunder av deres programvare er olje og gass/energi industrien, finans, ingeniørtjenester, samt tjeneste og offentlig sektor.

Selv om det var en uformell samtale, kom det frem en del nyttig informasjon, som også ble bekreftet i intervjuene. IKTBergen gikk for en del år tilbake fra en hverdag hvor mye var ad hoc til å benytte seg av fossefallsprinsipper. For fem år siden bestemte firmaet seg for å forkaste den gamle kodebasen sin, og utvikle alt på nytt, slik at de kunne dra nytte av nyutvikling innen web, programvareutvikling og databaser.

Dette viste seg å være en monumental oppgave, hvor fossefallsmodellen kom til kort og firmaet var nær ved å gå konkurs. Dette skjedde fordi firmaet i en to års periode ikke hadde noe nytt produkt å vise kundene. Denne perioden var også en stor belastning for de ansatte, fordi arbeidsmengden på den enkelte ble veldig høy. I samtalene våre kom det frem at grunnen til dette var at man ikke klarte å beregne hvor mye arbeid en slik programmeringsoppgave krevde, fordi prosjektet hadde en stor leveranse, og ikke mange små.

På bakgrunn av denne erfaringen ble det bestemt at noe måtte gjøres. Det er et kjent faktum at hver kodelinje som inneholder feil koster mye penger å få rettet, og at utstrakt overtidsbruk øker sannsynligheten for å introdusere slike feil. Dessuten er overtidsbruk kostbart for firmaet i seg selv. Ifølge våre samtaler med team medlemmene i IKTBergen, måtte man skrive mye av koden på nytt igjen når man kom tilbake på jobb dagen etter en lang økt utover kvelden. Dette var rett og slett fordi koden som var skrevet på kveldstid var av dårlig kvalitet. I tillegg gikk dette utover arbeidsgleden til den enkelte, slik at alle følte seg stresset og mindre motivert. I

dag benytter firmaet Scrum som styringsverktøy for utviklingen av sin programvareporteføljе for å forhindre at en slik situasjon oppstår i fremtiden.

Ved å engasjere seg i hvordan andre firma løste tilsvarende utfordringer, ble det bestemt at en fra Bergensteamet skulle delta på et Scrum Mester Kurs. Dette var fordi de ønsket at utviklingen skulle være mer agil. De neste tre årene benyttet IKTBergen til å forankre Scrum i organisasjonen. Det har tatt tid å tilpasse organisasjonens prosesser til Scrum metodikken, men dette har, ifølge dem vi snakket med, vært en stor suksess.

Hvorfor var så dette en suksess? Dette var, ifølge dem selv, fordi de nå hadde bedre kontroll, både når det gjaldt arbeidsmengde, men også kvalitet og evnen til å levere til rett tid. De forklarte at de nå hadde mulighet for å vise både ledelse og kunde hvordan utviklingen skred frem. Dette ble gjort ved å benytte to ukers lange sprinter, hvor teamet lovet å utføre[36] en viss arbeidsmengde, som ut ifra deres erfaring var gjennomførbar i forhold til teammedlemmer og sammensetningen av dem. De forklarte at etter hver sprint er gjennomført, har IKTBergen et ”review” møte hvor kunde og ledelse kan gi tilbakemelding til Scrum teamet.

Når de innførte Scrum valgte de å innføre hele prosessen, ikke bare deler av den. Dette var utfordrende, men siden Scrum er basert på mange års erfaring fra dyktige utviklere (best practice), mente de at dette var en god fremgangsmåte. Når IKTBergen fikk mer erfaring med prosessen hadde de en bedre forutsetning for å fjerne elementer de ikke følte behov for, slik som Scrum Mester, og legge til elementer fra andre metodologier, slik som parprogrammering fra XP.

Daglig Scrum møte

Litt etter skjema, klokken 09:40, startet det daglige Scrum møte. Det første vi reagerte litt på, var at ikke alle deltagerne reiste seg opp, og at vi ikke kunne se deltagerne i Ukraina. Møtet ble nemlig utført ved hjelp av IM, hvor alle deltagerne, både i Norge og Ukraina kunne skrive. Dette var vi ikke klar over før vi ankom bedriften. Nettopp det at alle kunne skrive samtidig var både en fordel, men også en ulempe, fordi det lett ble veldig uoversiktelig når alle ”snakket” i munnen på hverandre.

36 Eng.: Commit

Alle i Bergen beskrev kort hva de hadde gjort dagen før, og hva de hadde til intensjon å gjøre denne dagen. Engelsk ble benyttet som språk, slik at alle parter kunne forstå hva som ble skrevet. Etter møte ble vi forklart at engelsk nivået på konsulentene i Ukraina varierte veldig, både skriftlig og muntlig. Dette var en av årsakene til at de hadde gått bort fra å ha daglig Scrum møte via video.

Det var veldig vanskelig å vite om deltagerne fra Ukraina fulgte med på det som ble skrevet under møtet, fordi man ikke kunne se hva deltagerne i andre enden gjorde. Dette fordi IM ikke støtter denne typen awareness. Dette var ganske frustrerende for IKTBergens deltagere, fordi man ventet en stund, før det plutselig kom mye tekst på skjermen.

Møtet varte cirka femten minutter, men det var vanskelig å si når det egentlig var slutt. Både de i Ukrania og vi i Bergen, ventet på at motparten skulle skrive mer, og det var ikke før Bergenskontoret spurte om Ukraina hadde noe mer, at man kunne avslutte møtet. For å bøte på at man gjerne fikk en informasjonsoverbelastning, ble loggen fra samtalen sendt til hver deltager (se log i appendiks A). På denne måte kunne deltagerne raskt gå over ting igjen dersom de hadde gått glipp av noe, men hadde da ikke anledning til å stille oppfølgingsspørsmål til teamet, uten å innkalle til nytt møte.

Etter Daily Standup møtet var ferdig, ble vi forklart alle de andre variantene av et slikt møte de hadde forsøkt. Når de hadde Scrum of Scrums var denne delen av team-samarbeid en del enklere. De hadde da et slikt møte for hvert team, hvor alle var lokalisert på samme plass. I Bergen ble møtene holdt ved en tavle hvor Sprint Backlog listen ble vist frem med gule lapper (se figur 16), slik at denne kunne bli korrigert i forhold til hva som hadde blitt utført av arbeid dagen før.

Databaserte verktøy

Flere av våre intervjuobjekter har arbeidet på samme måte, men IKTBergen er de eneste vi har vært i kontakt med som nå har gått helt over til et databasert verktøy. Dette forklarte de at tvang seg frem etter at man ikke lenger hadde nok teammedlemmer i Ukraina og Bergen til å ha mer enn ett team. Dermed var det ikke lenger fysisk mulig for alle å stå ved siden av samme tavle. Selv om den beste løsningen gjerne er når alle er til stede på samme sted til samme tid, var ikke dette

lenger mulig, og man måtte se seg om etter alternative løsninger. Datastøttet samarbeid ble da en forutsetning for å arbeide effektivt i team.

IKTBergen er ikke alene om å måtte tenke utradisjonelt. I dagens internasjonale marked, hvor marginene er små, er det helt avgjørende at hvert konkurransefortrinn benyttes til det fulle. Arbeidskraft er billigere i andre land enn Norge, men der man før kunne benytte seg av flyreiser for å administrere og lede utenlandsk arbeidskraft, har de fleste firma i dag kraftig kutt i sine reisebudsjett (Kaspersen 2009). Databaserte verktøy blir dermed et viktig alternativ for å samarbeide. Dette byr på nye utfordringer fordi ingen av dagens løsninger er perfekte.

I starten av samarbeidet med konsulentene fra Ukraina hadde IKTBergen forsøkt å holde regelmessige videokonferanser, men Ukraina kontoret har ikke så gode tekniske løsninger som Bergens kontoret, både når det gjelder datamaskiner, men også Internett linjer. I Bergen hadde de gått til anskaffelse av et Microsoft Round Table kamera (Human Productivity Lab 2006) som de var veldig fornøyd med. Dette kameraet fokuserer på den som snakker, slik at opplevelsen blir mye bedre. Problemet var at Ukraina kontoret kun hadde et web kamera tilgjengelig, og denne løsningen fungerer best når begge ender har det samme kameraet. Dette kom også frem i intervjuet med en av programmererne. Kapasitetsproblemer gjorde også at bildet ofte hakket, og når det var slik at flere av Ukraina konsulentene ikke turde å si noe på grunn av sin dårlige engelsk, ble det til slutt besluttet at man primært skulle benytte IM.

Teambygging

For å bidra til at man fikk bukt med språkproblemene fortalte IKTBergen at de har sendt konsulentene i Ukraina på engelske språkkurs. Dette har hjulpet, men det gjenstår enda mye. Videre fortalte de at Ukraina konsulentene var i Bergen en uke for team-building for to år siden. Dette ønsket de å gjenta, både ved at konsulentene i Bergen får dra til Ukraina, og at de fra Ukraina får komme på et nytt besøk. De fortalte også at Daily Standup via IM trolig ikke hadde fungert like godt, hvis man ikke hadde kjent hverandre på forhånd.

IKTBergen var fullt klar over utfordringene de hadde gitt seg ut på når de nå hadde et distribuert Scrum team. Dette er noe de startet med like før sommeren 2009 på grunn

av at de ikke lenger er nok ansatte til at det er hensiktmessig med to team, spesielt under ferieavvikling. Ekstremprogrammering introduserer parprogrammering som en metode for å forbedre kvaliteten på produsert kode (se tabell 3). Den kan også benyttes på en god måte i opplæringsøyemed av nye konsulenter. I utgangspunktet var det tiltenkt at begge programmererne skulle befinne seg på samme sted til samme tid, men det har vist seg at par programmering også kan fungere når en befinner seg på ulikt sted, til samme tid. Dette har IKTBergen eksperimentert en del med sammen med kontoret i Ukraina, og det har vist seg at dette fungerer veldig bra. Dette mener de selv bidrar til økt team følelse, og man lærer den andre å kjenne på en ny måte. Håpet er at denne samarbeidsmåten kan forsere kulturbarrierer, og i tillegg heve det engelske språket til begge parter.

På kontoret i Bergen arbeider det en som kjenner kulturen og språket til teamet i Ukraina. Han ble ikke ansatt fordi han kom derfra, men fordi han var en dyktig programmerer som bor i Bergen. Dette har likevel vært heldig for Scrum teamet, fordi man har en som kjenner begge kulturer, og som kan oppklare misforståelser på grunn av språk og kultur. Alle var enig om at dette var veldig positivt og spesielt nyttig nå når de skulle være et sammensveiset team.

Planen var i utgangspunktet å holde to intervju før lunsj, men alle var så engasjerte og ønsket å vise oss rundt og forklare hvordan de arbeidet sammen, at det bare ble tid til et intervju. Med tanke på at vi bare hadde en ren teoretisk forståelse av hvordan Scrum team arbeidet sammen, var dette veldig verdifull tid for oss, fordi det ga oss en mer helhetlig forståelse av prosessen.

Kontorutforming

Tidligere var IKTBergen lokalisert i sentrum av Bergen, men hadde nå flyttet litt utenfor sentrumskjernen. Dette var fordi de her hadde fått mye større lokaler til samme pris, og det var praktisk i forhold til hvor de ansatte bodde. Lokalene hadde et åpent landskap, som vi ble fortalt var veldig viktig når man jobbet i et Scrum team. Tidligere hadde de hatt skillevegger mellom pultene, men selv dette var tatt ned, fordi de ble oppfattet som kommunikasjonsbarrierer. Vi ble fortalt at kommunikasjon var avgjørende for et godt programmeringsteam, og enda viktigere når en benyttet Scrum. Figur 12 viser hvordan lokalet hos IKTBergen var satt opp.

Figur 12 - Åpent kontorlandskap hos IKTBergen

Det så for oss ut som om de ansatte hadde faste plasser i det åpne landskapet, noe som understøtter læring på arbeidsplassen. Bjerrum og Bødker (2003) forklarer at det å ha felles informasjon liggende fremme om pågående prosjekter, slik vi observerte at det var hos IKTBergen, er med på å fremme læring. IKTBergen hadde noen kontorer til disposisjon hvis man hadde behov for å ta samtaler som de andre i teamet ikke trengte å høre, men kun samtaler som ikke var viktige for teamet skulle tas i eget rom. Bjerrum og Bødker (2003) forklarer hvor viktig disse aspektene er for et godt og effektivt arbeidsmiljø.

I inngangspartiet var det satt opp en elektronisk tavle som viste status på prosjektet og hvem som var tilstede i lokalet. Dette kan være nyttig awareness informasjon for deltagerne og besøkende. Bjerrum og Bødker forklarer at ”[…] at a glance‘ visibility of a permanent record of group activity and decisions is essential for teams working in shared large workspace” (2003, s. 202). Navnene på tavlen er blitt tatt bort i bildet som vist i figur 13.

Figur 13 - Statustavle i inngangspartiet hos IKTBergen.

Sosial aktivitet

Etter lunsj gjorde vi oss klar til neste intervju, mens fire andre utviklere gikk for å spille Fussball (se figur 15). Dette var noe de hadde gått til innkjøp av da Scrum ble innført med den hensikt å skape bedre teamfølelse blant de ansatte. Det ble fortalt at dette ble benyttet for avstressing og teambygging, og var på en måte et høydepunkt i løpet av dagen. De hadde også laget et IKT system (se figur 14) for å holde rede på liga spill. Etter de var ferdig å spille så vi at dette ga dem energi og giv til å arbeide konsentrert resten av dagen Bødker og Christiansen (2006) forklarer hvor nyttig slike avbrekk er i løpet av en arbeidsdag.

Figur 15 - Fussballbord hos IKTBergen

Figur 14 - Fussballstatistikk hos IKTBergen

De neste to intervjuene ble holdt like etter lunsj og varte frem til klokken var halv fem. Vi var slitne etter en lang dag med mange inntrykk som vi hadde behov for å få bearbeidet inn i notatform så snart som mulig. Dette avsnittet er en del av resultatet fra denne prosessen.

For å få frem designkrav til prototype er det blitt valgt ut en del sentrale tema fra intervjuene som vil bli gjennomgått under.

6.3 Gjennomgang av intervjutema

Det har blitt innhentet mye empiri som har bidratt til å svare på det første forskningsspørsmålet, som var å få frem hvordan distribuerte programvareutviklingsteam arbeider, og hvilke verktøy de benytter i sin arbeidspraksis. Informantene har blitt delt opp i tre kategorier, slik vist i tabell 6. Empirien har videre blitt benyttet for å besvare hvordan en slik praksis kan støttes i et MUVE. Nedenfor har empirien blitt strukturert i henhold til viktige samarbeidstema. Disse temaene og de betraktningene som ble gjort under besøket hos IKTBergen, har dannet grunnlaget for design av en prototype. Hensikten med denne er å understøtte distribuert Scrum samarbeid i et MUVE.

6.3.1 Sosiale mekanismer og relasjoner for samarbeid

I dette avsnittet vil det bli sett på; kommunikasjon, nåværende arbeidspraksis med dets sosiale mekanismer, hvordan man kan arbeide i et MUVE, og sosiale sammenkomster både her og i et MUVE.

Dagens arbeidspraksis
Det er hos IKTBergen forskjellige oppfatninger av det å sitte i et åpent landskap når man jobber. Tidligere Scrum Mester hos IKTBergen er helt overbevist om at dette er best. Denne holdingen deles også av han som er manager der i dag. Det kan illustreres av følgende påstand:

> ”Igjen, som sagt, dette vil variere fra team til team og person til person. Vi har hatt en veldig naturlig blanding. Vi har delt, eller spredt personligheter i teamene våre, veldig forskjellige. Noen på ene siden ønsker å jobbe masse i par og ser fordeler, mens noen ønsker å sitte alene og noen ønsker kanskje kontor sant. Vi har vært ganske åpen de siste årene at vi kjører åpent landskap, det gir best kommunikasjon for oss eller i allefall en kommunikasjonsform for oss som funker. Vi jobber stort sett på de samme prosjektene og da er det nyttig informasjon som faktisk går og summer litt rundt i rommet og du plukker opp en del ting, og du plukker opp om hvis det er samtaler du bør være med på osv” (Programmerer).

Som nevnt tidligere forklarer Bjerrum og Bødker (2003) nytten av åpne landskaper. Programmereren never videre at det selvsagt er motstand mot å organisere teamet slik, men at dette er noe som funker bra. En annen informant fra IKTBergen er ikke enig i dette. Han mener det stort sett er bra, men under aktiviteter som parprogrammering, ville det vært bedre for teamet om de, for eksempel, satt seg inn på et kontor. Dette illustreres godt i det han forklarer her:

> ”Det varierer jo da, når vi har jobbet i parprogrammering da, så blir volumet ganske høyt. Det er så voldsomt lett å bli forstyrret når man sitter hver for seg og jobber. Hvis du ikke har headset på deg så er det voldsomt lett om noen sier noe så blander du deg inn i den diskusjonen, så går det et kvarter og så mister du. . Det syns jeg er viktig når man programmerer, å komme inn i en flow, jeg vet ikke helt hvilket uttrykk jeg skal bruke, du kommer inn i en tankegang, og da jobber du og så plutselig kikker du opp” (IKTBergen).

Dette er faktorer som det blir tatt høyde for i de virtuelle arbeidsrommene som blir designet i studien.

Arbeidspraksis i et MUVE
Når en informant fra Universitet ble spurt om hvordan han tror det ville være å samarbeide med andre via et MUVE, forteller han at han er skeptisk til å jobbe i en virtuell plattform. Dette fordi han tror at det vil innebære at man må stenge ute den virkelige verden for å jobbe der, slik Yankelovich (2007) forklarer dette er hensikten til MPK20 prosjektet. Informanten forklarer dette slik:

> ”Du kan si at mitt hovedankepunkt mot virtuell virkelighet er at du setter som forutsetning at man på en måte må blokkere ut den ordentlige virkeligheten for å tre inn i den, den virtuelle, at du på en måte skal erstatte alt som er rundt deg. Det er jeg ikke sikker på at er hensiktsmessig. La oss si at vi to sitter på et kontor et sted, så skal vi snakke sammen med en som sitter et annet sted, en tredje person. Det at vi to sitter i samme rommet, er egentlig en stor ressurs for den kommunikasjonen som skal skje for vi har jo direkte kontakt. Hvis vi må ta på oss et ”headset” og må stirre inn i skjermen og snakke inn den, og egentlig miste, vi bryter den kontakten mellom oss to for å bringe oss sammen i en virtuell virkelighet. Så er det ikke sikkert at det i seg selv er verdt det” (Universitetet i Bergen).

For å vite om det er ”verdt det” som han sier, må man undersøke dette nærmere, slik det blir gjort i denne studien. Det er også mulig at teknologien i dag setter begrensninger, noe som betyr at man må blokkere ute vår virkelighet for å klare å konsentrere seg godt i en virtuell virkelighet (Yankelovich 2007). Teknologisk utvikling skjer i dag så fort, at dette er en begrensning som trolig kan løses i fremtiden.

Sosiale sammenkomster

En informant fra IKTBergen forteller at de ved noen anledninger har invitert de eksterne teammedlemmene til å komme til Bergen med det formål å få et bedre sosialt miljø:

> ”Absolutt, bare det å ha en daglig kontakt er positivt. Som sagt, vi har jo invitert det teamet over her et par ganger, ene og alene, på en måte, for å bli kjent og ha et ansikt å relatere seg til, osv. Så det er klart, det er kjempeviktig” (IKTBergen).

Det å møtes ansikt til ansikt er, ifølge informanten, bra for teambygging. På denne måten kan man få praktisert språkkunnskap og muligens bryte ned kulturbarrierer.

Det å spille spill er, som nevnt i avsnitt 6.2, noe IKTBergen benytter lokalt. Deres tidligere Scrum Mester kjøpte inn et Fussball bord (se figur 15) som er mye benyttet til adspredelse. En annen informant fra IKTBergen sier følgende om det å spille Fussball:

> ”For teamet sin del føler jeg at det har vært kjempebra. Når vi spiller 5 eller 10 minutter, får du plutselig beveget kroppen din litt fysisk og ropt og kjeftet litt og slike ting. Det er greit å få ut litt aggressivitet, få en utblåsning rett og slett” (IKTBergen).

Videre mener han at det å ha det mer sosialt på jobben er viktig for teambygging. Dette har gjort at både utviklere og ledelsen har fått det mer sosialt på jobben.

Sosiale sammenkomster i et MUVE
En informant fra Universitet forteller følgende om det å ha det sosialt i en virtuell virkelighet:

> ”Så hvis du skulle gjort teambuilding, så ville jeg heller gjort det med et spill. Ønsker du bare å ha en sosial sammenkomst, så vil jeg si at du har sannsynligvis, fremfor å kjøre ti sosiale sammenkomster på virtuell virkelighet, heller sette opp ett stort møte hvor du faktisk bringer folk inn. Det vil være mye mer nyttig” (Universitetet i Bergen).

Han ser altså for seg muligheten for å spille spill som et sosialt tiltak på arbeidsplassen, og dette er noe man kan gjøre online også. Han påpeker at den sosiale gevinsten i en vanlig sosial sammenkomst, er størst ved å møtes i den virkelige verden, men at en kan ha stor faglig nytte av å arbeide sammen i en virtuell virkelighet:

> ”Jeg vet ikke, men jeg vil ihvertfall tro at den sosiale gevinsten ut i fra et sosialt møte inne på virtuell virkelighet er mindre, men den faglige gevinsten kan være ganske stor” (Universitetet i Bergen).

Det er et viktig poeng at den faglige gevinsten er stor, men informantene er litt sprikende når det gjelder hvor godt det sosiale vil fungere i et MUVE. Dette har nok med at de færreste av dem noen gang faktisk har forsøkt dette.

6.3.2 Awareness
En informant fra Universitetet i Bergen påpeker at om man skal kunne dele workspace er det viktig å kunne skille ut den private delen:

> ”Så kunne du på en måte tatt vekk disse her *privacy*, privatlivets fred hadde ikke blitt så inngrepet da, fordi kodene er jo offentlig uansett. Da kunne du ha sett på hva som faktisk skjer og gjort veldig mye i det virtuelle miljøet” (Universitetet i Bergen).

Han mener at det å kunne følge skjermene til de andre i teamet vil bidra til en god awareness. Det med privat skjerm og offentlig skjerm er noe som også IKTBergen har

sett behovet for når de jobber med parprogrammering. Det er et behov for å kunne lese e-post og kunne svare på henvendelser i løpet av arbeidsdagen.

Informanten forsøker så å tenke på hvordan awareness hadde blitt påvirket hvis man delte skjermbildet sitt inn i et MUVE. Han ser det som fordelaktig at andre som er i MUVEet da kan se hva han arbeider med. Hvis han ikke følger med i MUVEet oppstår det derimot et problem:

> ”En annen ting er jo hvis jeg går innpå ditt virtuelle kontor, hvordan skal, og det da er en slags analog til ditt fysisk kontor på ett eller annet vis, så hvordan i alle dager skal man da fysisk vise at jeg er til stede, det er jo umulig” (Universitetet i Bergen).

Det er en utfordring om man er fokusert på sin egen datamaskin og samtidig skulle være tilstede i en virtuell verden med en avatar. Her foreslår han at man kan montere høytalere i kontorene, slik at en kan overføre stemmen til brukeren i MUVEet, til den virkelige verden. Dette syns han var en ubehagelig tanke, og ikke noe han ville ønsket. Nå er det slik at så lenge en er pålogget MUVEet, selv om en ikke har brukt det på en stund, fremdeles kan få overført lydene fra MUVEet. Dette kan så klart være forstyrrende hvis en arbeider med helt andre ting.

En informant fra IKTBergen har en klar oppfattelse av at det å være representert ved en avatar i et MUVE kan gi en sterkere følelse av tilstedeværelse enn hvis en kun ser et navn på IM:

> ”Du sitter kanskje på MSN [form for IM] og tenker, ”åå nå var den og den med der, men hvor er han?” Hadde du hatt et virtuelt space så hadde du på en måte oppdaget de personene som ikke var der, og du ser de som er der på en mye enklere måte enn med et vanlig bilde” (IKTBergen).

En kan altså si at man er mer ”tilstede” ved å være representert av en avatar, enn om man bare er tilstede på IM, slik Olivier og Pinkwart hevder i sin artikkel (2007).

Hvordan en lykkes med å innføre Scrum

I løpet av intervjuene ble det pekt på hva som er den beste måten å implementere Scrum. Ifølge Scrum metodologien, og bekreftet av erfaringene til IKTBergen, er det best å implementere alt fra Scrum på en gang, ikke en del om gangen. Dette illustreres godt av en informant fra IKTBergen:

> ”Det hjelper ikke å adoptere enten det er “Scrum” eller parprogrammering sånn halvveis, du må på en måte gå hele veien. Så kan du adopterer deler av det etterpå. Hvis du ikke har prøvd hele filosofien, så blir det til at det henger igjen de gamle syndene likevel. Så det har hvertfall vært filosofien vår, å adoptere fullt ut og så tilpasse seg etterpå. Det har vi hatt god erfaring med” (IKTBergen).

Tidligere Scrum Mester bekreftet også dette, når han hevdet at det er bedre å ta bort elementer som man erfarer ikke fungerer slik man ønsker, etter at en har fått erfaring med hele prosessen. Hos IKTBergen har dette gitt seg utspill i at de har gått bort ifra å ha en dedikert Scrum Mester, slik nevnt her:

> “Når vi først begynte med [Scrum], tok vi det mye mer *strict* enn vi er nå. Han skulle være tester, han skulle være SCRUM Master, så han skulle hele tiden springe ute og passe på, innkalle møter i hytt å pine… Vi har ingen SCRUM Master nå, det bare går litt av seg selv føler vi i hvert fall” (IKTBergen).

Selv om de i dag ikke har en dedikert Scrum Mester observerte vi og tolket ut fra intervjuene, at rollens oppgaver blir ivaretatt av teamet. Måten dette gjøres på er at de bytter på å ta på seg denne rollen. Manager hos IKTBergen forteller at han ønsker at noen skal stå frem for å ta på seg denne rollen.

For prototypen er det viktig å være klar over at Scrum metodologien vil bli tilpasset den enkelte arbeidskontekst. Det er ikke slik at man kan lage en prototype basert på metodologien alene, men man må ta høyde for det erfaringsgrunnlaget som den enkelte bedrift har opparbeidet seg, slik at en tilpasser løsning til bedriftens prosesser. For å identifisere hvordan Scrum teamet hos IKTBergen gjennomfører sine daglige Scrum møter, har det blitt laget en task modell basert på intervju og observasjon ved hjelp av Groupware Walkthrough som er lagt ved i appendiks C.

Det er ulike faktorer som spiller inn når man skal få til et godt samarbeid blant programmerere i et team. Det er kanskje enklere å få til et godt samarbeid når en er samlokalisert, men det betyr ikke at det ikke finnes utfordringer også her.

6.3.3 Samlokaliserte team

Når man skal studere samarbeid i et MUVE er det slik at dimensjoner av tid og sted gjerne ikke oppfattes på samme måte som man er vant til (Olivier & Pinkwart 2007). Som nevnt i avsnitt 2.1.1 er Second Life en applikasjon som etterstreber å støtte eksisterende arbeidspraksis. Det er derfor nyttig å se på de verktøy som benyttes

lokalt, fordi dette er verktøy som også er nødvendig for å samarbeide i et MUVE, som vist i tabell 1.

Møter og samarbeidsverktøy
Hovedinntrykket fra intervju og observasjon hos IKTBergen er at det var tavler og gule lapper som var det beste og mest brukte kommunikasjonsmiddelet for å gjennomføre møter når alle var tilstede. En informant fra IKTBergen forteller at:

> ”Det er noe med, det enkle er ofte det beste når det gjelder sånne ting. Der ser vi, vi har masse verktøy. Likevel ser vi at det beste er ofte å henge en gul sticker på tavlen liksom. Det er mye “Scrum” har vært om. Vi har masse små kort eller post-it lapper, de flytter vi på og de jobber vi med” (IKTBergen).

Dette understøttes også av to av informantene fra Universitet, men de benyttet også projektor for å vise Scrum Works under møtene. Scrum team benytter altså mye direkte kommunikasjon for å løse felles problemer på arbeidsplassen. Den ene programmereren fremhever viktigheten av direkte kommunikasjon i en samarbeidssituasjon, noe som også støttes av en av informantene fra IKTBergen:

> ”Vi sitter jo da i et landskap så vi driver jo bare og spør hverandre når vi trenger ett eller annet. Jeg er vel en av dem som har lettest for å spørre, det må jeg innrømme. Det er rett og slett fordi det er en veldig lett tilgjengelig måte å finne informasjon på, i stedet for å undersøke selv, så er det mye enklere å spørre en som du vet har jobbet med det” (IKTBergen).

Dette fremhever igjen behovet for vanlige tavler som man kan skrive meldinger på, og videre hvor viktig det er for teammedlemmene å kunne henvende seg til sine kolleger for hjelp.

Det kom frem i løpet av intervjuene at det daglige møtet kun skal vare 15 minutter og at det derfor ikke er mye tid til å sette opp en kommunikasjonsløsning. En informant fra IKTBergen sier:

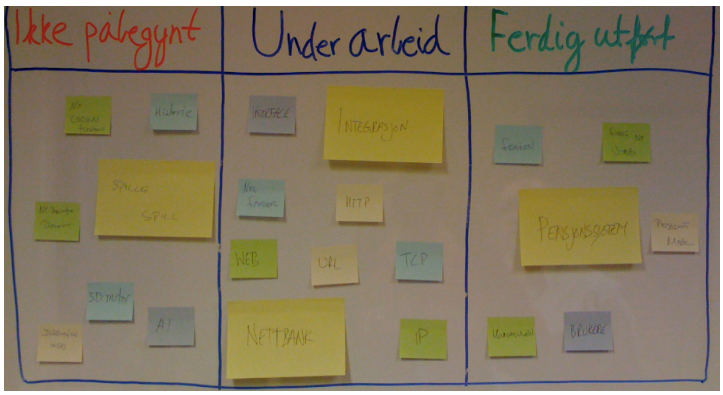
> ”Det møtet der har vi veldig fokus på at skal vare i 15 minutter, vi bruker ikke mer tid på det. Bruker en mer tid på det, så er det spesifikke diskusjoner som skal foregå blant utvalget” (IKTBergen).

Det er derfor et krav til prototypen at det ikke skal ta for lang tid å sette opp møtet og gjennomføre møtet. Selve utformingen av møterommet kan ha en innvirkning på tidsbruken på møter.

Møterom designet for å støtte Scrum

Når man skal lage et virtuelt møterom for å avholde Scrum møter, er det viktig å sette seg inn i hvordan et slikt møterom ville blitt utviklet i den virkelige verden for å tilegne seg den nødvendige kunnskapen om nødvendige funksjoner og best mulig tilrettelegging. Det er ikke slik at å tilrettelegge et møterom i den virtuelle verden vil bli best mulig om en bare kopierer inn et standard møterom fra vår verden, fordi en da ikke vil ta i bruk de nyvinningene som en virtuell arbeidsplass vil kunne gi. Videre er det også måter å gjøre ting på i den virkelige verden, som ikke er hensiktsmessig i et MUVE. Ved å tolke input fra intervjukandidatene og observasjon hos IKTBergen var intensjonen å få frem designkrav til et ideelt møterom.

Hos IKTBergen ble Scrum møtene tidligere holdt stående rundt en tavle (se figur 16 for et eksempel), som ble manuelt oppdatert fra dag til dag. Dette var noe som fungerte tilfredsstillende for å holde oversikten over fremdriften. Tidligere Scrum Mester hos IKTBergen, mener at alle steder hvor det er plass til det, bør det være tavler som en kan skrive opp ideer og tanker. Tilgang til IKT verktøy som ScrumWorks (Denniston 2009), og lignende, er også nevnt som en fordel. I et optimalt Scrum møterom bør man altså ha tavler for å dele ideer, og i tillegg ha tilgang til de ulike systemene som viser oppdatert status i prosjektet.



Figur 16 – Eksempel på manuell Scrum Backlog liste

Når en skal avholde Scrum møter er det viktig at alt involvert personell er tilgjengelig når møtene skal holdes. Denne forutsetningen bekreftes også i intervjuene, hvor det blir hevdet at det å ha alle tilstede i samme rom når man gjennomfører møter er den beste løsningen.

En informant fra IKTBergen sier følgende om kommunikasjon:

> ”Men kommunikasjon, det er klart, det er *key*. Vi har liksom den kjente pyramiden, direkte kommunikasjon er klart bedre enn skrevet kommunikasjon osv.” (IKTBergen).

Dette bekreftes også av en programmerer, som med sin tidligere erfaring som Scrum Mester hos IKTBergen, understreker viktigheten av å ha alle til stede. Dette betyr at det er viktig å tilrettelegge for en løsning hvor alle kan delta på møtene. Han påpekte videre at det er viktig at man har støtte for de verktøy som et Scrum team benytter i sin arbeidspraksis, slik figur 16 er et eksempel på.

Møterom i et MUVE
Når man først har samlet alle på møtet er det viktig, slik tidligere Scrum Mester påpeker, at riktig informasjon er tilgjengelig i det virtuelle arbeidsrommet. Han fokuserer på at løsningen utviklet i denne studien må være enkel å bruke og kun inneholde de elementene som er nødvendig. Han vil ha Product Backlog, Sprint Backlog og buglister tilgjengelig, og et rom som er utformet slik at en kan snakke og samarbeide på en effektiv måte. Et annet viktig innspill fra han og en annen av programmererne er at det ikke må være nødvendig å bevege seg igjennom store områder for å komme til de forskjellige fasilitetene.

6.3.4 Distribuerte team
Dette avsnittet vil synliggjøre utfordringer med distribuert arbeid i eksisterende praksis, og hvordan en slik form for cooperative work kan støttes i et MUVE.

Kommunikasjon
Det ble spurt om hvordan det sosiale blir påvirket av avstandsarbeid. En informant fra Universitetet i Bergen hadde mye kunnskap omkring dette. Han forklarte at den sosiale kontrakten fungerer bedre når en møtes ansikt til ansikt, enn når man snakker sammen via ulike medier, slik som telefon, VoIP, IM, videokonferanse osv. Når man møtes via slike løsninger mener han at det fort blir mye informasjon og at den ikke tilegnes like godt som når en møtes ansikt til ansikt. Slik tenker han om dette:

> ”Det er ett eller annet som mangler for den kommunikasjon, hvertfall fra min personlige erfaring, som gjør at det er utrolig vanskelig faktisk. For det første så føler jeg, det er jo min personlige erfaring da, så føler jeg på en måte at de tankene, det som blir sagt, det sitter mye dårligere i hukommelsen. Du har selv vært på en telefonkonferanse og snakket om en million ting, to dager etterpå

> så bare *eh, vi sa ett eller annet om det, men jeg husker ikke helt hva*" (Universitetet i Bergen).

Her kommer det frem at informanten mener at møter og samtaler som holdes tradisjonelt i et møterom hvor alle er tilstede, er den optimale løsningen for ham. Videre sier han at terskelen for å avvise og nedprioritere møter på telefon, VoIP eller IM er lavere enn om man har et vanlig møte:

> "De har andre arbeidsoppgaver som de skal gjøre, så da blir det nødvendigvis en sånn kamp om ressurser. Så er det med at ringer du folk, så er det ikke alltid de kan ta telefonen. Det erfarer jeg også veldig ofte. Prøver å få til en Skype samtale rundt et tema, nei ikke nå jeg skal til lunsj, jeg skal, sant, nei nå kom den inn, huff, nå må jeg gjøre det, nå skal jeg i møte. Så liksom det er vanskelig å koordinere det på langt hold, men med en gang noen skal møtes så blir timeplanen ryddet uansett hva som skjer" (Universitetet i Bergen).

Her forklarer informanten hvordan han opplever den sosiale kontrakten når man tar kontakt på IM eller VoIP. Slike medier er lavterskel kommunikasjon og enkle å bruke for å etablere kontakt og for å sjekke om en person er tilgjengelig. Disse mediene er også enklere å overse om man har det travelt. Personen i andre enden kan jo ikke se om man er tilstede (Nardi, Whittaker & Bradner 2000).

Videre understreker informanten dette argumentet ved følgende påstand:

> "Det er ett eller annet, når du møtes "face-to-face" så ligger det en mye sterkere sosial kontrakt i det. Hvis jeg bare la være å svare på den telefonen nå så vet ikke de om jeg ikke var der, eller om jeg faktisk bare lot være å være der" (Universitetet i Bergen).

Dette er viktig å ta hensyn til når man skal designe en groupware løsning for cooperative work.

Distribuert arbeid

Det var IKTBergen og en informant fra Universitetet i Bergen som hadde erfaring med distribuert arbeid i team. IKTBergen har forsøkt en rekke hjelpemidler, og vært på utkikk etter løsninger som kan hjelpe dem til å forbedre deres kommunikasjon med de eksterne teammedlemmene fra Ukraina.

Gjennom intervju, observasjon og notater har vi avdekket at de fleste bruker IM som grunnlag for kommunikasjon med medlemmer som ikke er lokalisert på samme sted. En informant fra IKTBergen forteller om møtene på IM:

> ”Alle skriver ….hva han gjorde i dag og i går om han hadde noen problemer osv og så neste mann . Her er det med at alle skriver det de har gjort samtidig og så eventuelt spør noen spørsmål om noen har noe problem så dukker det opp på slutten av møtet” (IKTBergen).

Videre benyttes også IM til daglig kommunikasjon blant de samlokaliserte teammedlemmene og med de i Ukraina. IM er en effektiv kommunikasjonskanal og det er derfor den blir benyttet i så stor grad (Nardi, Whittaker & Bradner 2000). Et krav til prototypen vil være å ha støtte for slik tekstlig kommunikasjon. Utover ren tekstlig kommunikasjon har det vært prøvd ut ulike løsninger for å få til lyd, video og overføring av tavle informasjon på samarbeidsmøtene. En informant fra Universitetet beskriver hvordan han har erfart dette:

> ”Man kan bruke gratis verktøy som Skype eller så kan du på en måte kjøpe hundre tusen, millioner kroner i utstyr for å få kameraer og alt slags dilldall, delte ”blackboards” og sånt som det der. Så i utgangspunktet er det løst. Det har gått fra å være et teoretisk til teknisk til at det nå bare er et pengeproblem, hvor mye penger bruker du på det” (Universitetet i Bergen).

Videre snakker han om andre muligheter hvor det finnes billige løsninger så lenge man har Internett linje, slik som et webkamera. IKTBergen har også prøvd å benytte dette men har møtt en del utfordringer med denne løsningen. De føler at webkamera løsninger er best på møter med få deltagere. En informant fra IKTBergen sier:

> ”En-til-en så har vi nok verktøy per i dag føler jeg, men det er det å kunne koble flere sammen som jeg ser på som en stor styrke” (IKTBergen).

Han etterlyser løsninger som er tilpasset team fordi de rimelige løsningene er best på punkt til punkt samband. Behovet er løsninger som er bedre tilpasset, og at man kan delta uavhengig av lokasjon og som kostnadsmessig er gjennomførbare. Dette inngår som et krav til prototypen.

IKTBergen har forsøkt å kjøre parprogrammering mellom deltagerne på det distribuerte teamet, hvor de har benyttet en løsning med delt skjerm hvor begge jobbet sammen, og satt opp en audiokanal som et supplement for å kommunisere mens de

jobbet. Dette har de hatt stort utbytte av. En informant fra IKTBergen beskriver det slik:

> ”Og det med delt skjerm funket ganske greit. Så har vi funnet ut at det er viktig at en hele veien, så har vi et delt område og et privat område, så du ikke føler deg overvåket, at du kan lese en e-post innimellom uten at den andre kan se det. Så tror jeg det at vi har kommet frem til at du må ha en slags kjernetid med parprogrammering. Du kommer ikke på jobb klokken 8 og sitter på parprogrammering til klokken fire. Da er du sliten. Da er du ikke responsiv til andre ting som kommer utenom heller” (IKTBergen).

IKTBergen er fornøyd med hvordan de parprogrammerer distribuert, og vil fortsette å arbeide på denne måten. Det vil være en stor fordel om prototypen har støtte for dette.

Når medlemmene fra Ukraina får litt tid til å forberede seg før det daglige Scrum møte, fungerer det bedre. En informant fra IKTBergen sier:

> ”Men det som en ser, hvis en tenker litt bak i tid, så har det vært sånn at vi har hatt bedre suksess der på en måte der de eksterne teamet vårt har fått fem minutter på forhånd for å forbedre seg, på en måte har litt klart til møtet er i gang. Det negative er at du får ikke på en måte den naturlige interaksjonen, sant, du får på en måte at folk sitter og *poster* inn med sitt, så blir det ikke en dynamikk i det, da mister du litt av effekten” (IKTBergen).

Når samtalene på IM er forberedt på forhånd kan dette gå utover den naturlig flyten og kan bli noe rotete, noe som ble observert hos IKTBergen (se avsnitt 6.2).

Utfordringer

En av utfordringene som kom klart frem i intervju og observasjon var knyttet til språk og kultur, som man støtet på når man skulle samarbeide på tvers av landegrenser. En informant fra IKTBergen beskriver det slik:

> ”Språk har definitivt vært en barriere og utfordring for oss. Som sagt, i begynnelsen så var det enkelte som ikke klarte å ha en god samtale med på engelsk. Vi har investert i det. Vi hjalp de med kursing og fulgte opp på den måten da. Litt, vi ser nå, en av de som ikke ønsker å gjøre parprogrammering, gjør det mer eller mindre på grunn av at det er vanskelig å snakke engelsk, sånn direkte” (IKTBergen).

IKTBergen hadde her en fordel ved at en av de ansatte var fra Ukraina. Hans gode innsikt i norsk og ukrainsk kultur, i tillegg til å ha god beherskelse av engelsk og norsk, var til god hjelp for teamet. Språk og kulturproblemene bekreftes også av deres tidligere Scrum Mester, og han vil på bakgrunn av disse erfaringene ikke anbefale

team å ha deltagere fra ulike land med ulikt språk og kultur. Dette er utfordringene andre firma støter på, fordi lavkostland som brukes til utkontraktering av arbeid, gjerne har et annet språk og kultur enn de i Vesten. Hvis man tilrettelegger for sosialisering og opplæring i en groupware løsning, slik det er gjort i denne studien, kan dette være med på å overkomme slike hindre for samarbeid.

En annen utfordring for IKTBergen var at linjekapasiteten hos det eksterne kontoret var så liten at dette skapte problemer med videooverføring. Linjekapasitet er altså noe prototypen må ta hensyn til hvis den skal kunne benyttes av flest mulig.

Samlet plattform for samarbeid ved distribuert prosjektarbeid i et MUVE

IKTBergen har forsøkt å sette sammen en portefølje av produkter for å oppnå et effektivt samarbeid med sine eksterne teammedlemmer. Her har de forsøkt ulike løsninger med varierende hell, og deres erfaringer tilsier at det ville vært et stort fremskritt om man hadde en portal eller et grensesnitt hvor flere av tjenestene disse programmene utfører, kunne samles under en paraply.

Det fremheves av en systemutvikler hos IKTBergen at det er fordelaktig å ha muligheten for å samle flere ulike funksjoner i samme verktøy. Dette gjelder både vanlig bruk og i en opplæringssammenheng:

> ”Hvis man har fem forskjellige verktøy og kan gjøre alt i ett, så er jo det mye bedre. For det merker jo vi, jo flere forskjellige verktøy man bruker, jo mer strev er det. Hvis du kan gjøre to ting i ett verktøy så er det ofte bedre da, for du skal lære folk opp og da” (IKTBergen).

Hvis løsningen som designes for distribuert Scrum samler funksjonaliteten, ville dette bli sett på som en fordel. Videre forteller han at det er viktig at de løsningene som utvikles er enkle og intuitive å benytte, fordi dette gjør opplæring og tilrettelegging av dem mye enklere for brukerne. Dette illustreres også godt i intervjuet med han som er manager for programmererne hos IKTBergen:

> ”Det er viktig. Det skal være, det må være så enkelt så mulig. Det må være intuitivt og det må være enkelt å bruke. Og det ser jeg med de fleste verktøyene. Det er ikke de mest avanserte verktøyene alltid som vinner frem, det er de som er lette og bruke. Det er det du vinner på i sånne situasjoner” (IKTBergen).

Prototypen som designes i studien har som mål å være både enkel og intuitiv, samtidig som den innehar funksjonalitet for å støtte deler av Scrum prosessen. Prototypen har ikke som mål å støtte selve programmeringsarbeidet, fordi dette ville gått utover studiens rammer.

Prosjektstyring

Studiens intervju og observasjon gir et bilde over hvor komplisert det kan være å drive prosjektstyring i et land som Norge, hvor en har spredt bosetning og ofte ikke tilgang til alle ressurspersoner i samme geografiske lokasjon. Det er derfor behov for et verktøy for å utføre distribuert prosjektstyring, både for å styre teamet og for å holde kunder oppdatert på status.

En informant fra Universitetet i Bergen beskriver i intervjuet et oppdrag han hadde for et firma i Oslo, som hadde engasjert et firma i Stavanger for å gjøre et stykke utviklingsarbeid. Han var engasjert for å kravstille produktet som skulle utvikles. Oslofirmaet, som skulle ha denne programvaren utviklet, inngikk avtale med en selger fra firmaet i Stavanger om hvilket arbeid som skulle utføres i henhold til kravspesifikasjonen. Dette arbeidet gikk ikke som planlagt og en av hovedårsakene var vanskeligheten med å holde en tett kontakt mellom kunde og leverandør etter hvert som produktet ble utviklet. Han forsøker å konkretisere problemet slik:

> ”[…] vet ikke om det er et spesielt problem med landet vi bor i, eller hva det er for noe… Men problemet er avstanden” (Universitetet i Bergen).

Videre mener informanten at det ikke er enkelt for små og mellomstore firma å finne seg samarbeidspartnere på samme geografiske sted i Norge. Han sier følgende om geografien i Norge:

> ”Så avstanden er et problem, at du har et firma i forskjellige geografiske lokasjoner hvor det ene skal utføre et oppdrag for det andre. Så det er et problem, for du skal ha tett… hele forutsetningen for de fleste agile metoder er at du har tett kommunikasjon” (Universitetet i Bergen).

Dette vil med andre ord si at det kanskje er større behov for distribuerte groupware løsninger i Norge. I andre land kan det være lettere å opprette lokalt samarbeid på grunn av befolkningstetthet og topografi.

En annen problemstilling når interessentene er spredt over ulike lokasjoner er mulighet til å vise frem produktet man utvikler. En informant fra IKTBergen sier at det er sjeldent toppledelse og kunder er tilstede på gjennomgangsmøtene. Han er usikker på hvorfor, men mener at lettere tilgang til møtene kan gjøre det enklere for dem å delta:

> ”Siden vi har kontor i Oslo som er selger og Top Management og sånne ting. Det er mer for å vise de og litt for oss, hvis vi har sittet for oss selv og programmert er det voldsomt greit å få med seg hva de andre har gjort. Få med seg hva som skjer. Det har ikke vært så mange kunder som har kommet inn. Tror det har vært to eller tre ganger i perioder” (IKTBergen).

Det er viktig å ha en god måte å kommunisere med personer som sitter på andre steder, slik at man får en god flyt i kommunikasjonen som kan holde alle oppdatert på de ulike prosjektenes fremdrift. Reising tar mye tid, koster mange penger og er ikke bra for miljøet (Kaspersen 2009). En mulig løsning er distribuert prosjektstyring, slik IKTBergen gjør det i dag.

Frihet til å arbeide distribuert

Et annet tema som er aktuelt for denne studien er muligheten til å kunne arbeide distribuert hvis en ønsker. I dagens arbeidssituasjon kan det være vanskelig å samle alle arbeidstakere på samme sted, dag etter dag. Det kommer ofte situasjoner som krever at de ansatte må kunne arbeide både hjemmefra og på reiser. Tanken bak SUN MPK20 er, ifølge Nicole Yankelovich (2007), at de ansatte skal ha mulighet for å jobbe på ulike steder, men fremdeles ha tilgang til nødvendige verktøy og sitt eget sosiale nettverk Tanken er altså at man skal altså være en del av teamet uansett hvor man måtte befinne seg.

En informant fra Universitetet i Bergen er opptatt av friheten til utviklerne og han mener at utviklerne må bli gitt romslige rammer for å kunne utføre sitt arbeid. Han beskriver utviklernes arbeid på denne måten:

> ”Men, programmering er jo en kunst, det er jo ikke noe annet å si. Det er i sin simpleste form så er det et håndverk, og i sin mest elegante form så er det på linje med kunst i mine øyne. Hvis du har en haug med kunstnere, så nytter det ikke å fortelle de hva de skal male. Du kan kun bare gi de motiver og spille på kreativiteten deres og se hva som kommer ut. Og stort sett er det bra, selv om det ikke akkurat er hva du forestilte deg på forhånd” (Universitetet).

Dette gjennspeiler hvor utfordrende det kan være å tilfredstille alle behov som kreative skjeler har behov for i sitt arbeid, og hvor viktig det er å ikke kvele kreativiteten som er nødvendig for å oppnå et bra resultat. Likevel er det viktig for ledelsen å ha styring på prosjektet, noe som skaper en hårfin balanse mellom frihet og kontroll.

IKTBergen har merket at etter at det daglige Scrum møtet ble flyttet til IM var det flere som benyttet seg av hjemmekontor. Når de hadde møtene i det åpne landskapet var det nødvendig å være mye tilstede slik at teamet fungerte optimalt. Dette illustreres i det som en informant fra IKTBergen forklarer:

> ”Ja, det er ganske mange her som bruker hjemmekontor. Det er vel hvertfall ei som er hjemme i dag. Det er en del som bruker hjemmekontor ganske ofte. Det fungerer ganske greit. Vi har opplevd, spesielt når vi hadde rene geografiske team, så merket vi at du falt litt ut hvis du var hjemme noen dager på rad, så fikk du ikke med deg alt. Du gikk glipp av Stand Up Scrummen da for eksempel og da etter hvert så går du glipp av informasjon” (IKTBergen).

Flere hos IKTBergen ønsket å jobbe hjemmefra noen dager i uken for å få litt arbeidsro. Når man var på jobben ble en ofte forstyrret av henvendelser, og det kunne ødelegge den gode programmeringsflyten. Teamet har vært igjennom en tilvenningsperiode. Dette har resultert i at de nå er vant til at deler av teamet ikke er fysisk tilstede. Fordelen med dagens organisering er at man får med seg det daglige Scrum møtet uansett hvor en jobber. Som et eksempel på at dette kan fungere forteller en informant fra IKTBergen at han arbeidet fra England i et helt år. Dette var fordi kjæresten studerte der. Dette var en positiv erfaring, hvor han følte han bidro på en god måte, selv om han mener at dette ble i lengste laget. Han fremhever at han synes at det er en fordel at IKTBergen lar sine medarbeidere jobbe eksternt.

6.3.5 Skepsis til MUVE som samarbeidsløsning

I den kvalitative undersøkelsen kom det frem synspunkter og en viss skepsis mot å bruke et MUVE som samarbeidsplattform. Et argument som går igjen, er hva som er nytten av å gjøre ting tredimensjonalt i forhold til todimensjonalt. Dette er et godt argument som man må ha i tankene når en benytter seg av tredimensjonale miljøer. Det er ikke noe mer hensiktsmessig å lage et søylediagram i 3D, enn det er å lage det samme i 2D. Faktisk blir den tredimensjonale modellen lett rotete. Det er derfor viktig å undersøke hvor det er hensiktmessig å benytte et MUVE. Det at et MUVE er en

tredimensjonal representasjon av et miljø, som man kan tilpasse så mye en ønsker, uavhengig av vår verdens fysiske lover, åpner opp mange muligheter.

Et tenkt scenario er trafikk optimalisering, hvor det vil være mulig å lage en digitalisert, tredimensjonal versjon av Bergen by, hvor man kan hente inn bil og trafikk data fra Statens vegvesen (2009). Når man så har lagt inn hele trafikknettet, med skilter, signallys, togoverganger osv, kan en simulere hvordan byggearbeider vil påvirke trafikkmønsteret. Videre kan en så via matematiske formler få frem effekten av ulike alternative planer for trafikk traseer. I tillegg kan det argumenteres for at ren todimensjonal grafikk ville vært bra nok, men at en får et helt annet perspektiv når man kan se noe så komplekst som trafikk i tre dimensjoner. Veiingeniøren kan simulere at han kjører en bil langs den nye traséen, og kan i detalj se hvordan det fungerer. Det at han kan ”se” er et godt stikkord her, for det er viktig at trafikanter har god sikt til det som skjer i trafikken.

En informant fra Universitetet i Bergen er først veldig skeptisk til å arbeide sammen i en virtuell verden, men når han tenker litt på saken kommer han frem til følgende:

> ”Selvfølgelig i selve utviklingen vil det ikke være hensiktsmessig at folk sitter og koder på hvert sitt virtuelle kontor, eller kanskje, vent litt nå, hvis du kunne gått fra ditt virtuell kontor til et annet sitt virtuelle kontor og sett nøyaktig hva han holdt på med” (Universitetet i Bergen).

Her ser han at det vil være en fordel om han kan se hva de andre i teamet gjør på, ved at en har tilgang til deres programmeringsverktøy og ser hva de programmerer på. Han ser for seg muligheten til å integrere et bilde fra programmeringsverktøyet sitt inn i et virtuelt miljø for bedre å kunne samarbeide med distribuerte medarbeidere.

En informant fra IKTBergen funderer omkring mulighetene til å jobbe i en virtuell verden etter å ha sett demonstrasjon av Sun sin virtuelle arbeidsplass:

> ”Her er en versjon, du har sikkert mange andre muligheter, og det er klart at det er på en måte å kunne knyttet et ansikt til og ha en dialog på tvers av kontorene. Selv om jeg må si at jeg er en skeptiker i utgangspunktet” (IKTBergen).

En annen informant fra IKTBergen er i utgangspunktet skeptisk han også, men:

> ”Så hvis det foregår på en voldsomt knirkefri måte så tror jeg det kunne vært kjempebra. Jeg har fulgt andre programmer, å ja… Så lenge det praktiske problemet blir løst så har jeg voldsomt tro på at det hadde passet utrolig bra for IKTBergen” (IKTBergen).

Selv om flere er skeptiske til MUVE som løsning for distribuert prosjektarbeid, er det altså flere som ser nytten hvis de praktiske utfordringene blir løst på en god måte.

6.4 Oppsummering av hovedfunn

Det har i løpet av den kvalitative studien blitt fremhevet en del momenter som er viktig når man skal designe en groupwareløsning for å støtte eksisterende arbeidspraksis. Informasjonen som har kommet fram behøver ikke bli benyttet for å designe en løsning i et MUVE, men kan med fordel også bli benyttet i andre sammenhenger.

Som grunnprinsipp kom det frem at løsninger utviklet for å støtte Scrum praksis, må være *effective*, *efficient* og *pleasantly* for brukerne. Det vil være utfordrende å lage en løsning som tilfredsstiller disse tre målekriteriene for usability, men dette er noe man må strekke seg etter, slik vist i avsnitt 4.6.1. Konkret vil det si at løsningen må være lett å benytte, gi god tilbakemelding på handling, og tilby nødvendige kommunikasjonsmuligheter for å arbeide distribuert. I tillegg har de distribuerte medarbeidere sett behovet for sosialisering, teambygging og mulighet for å gjennomføre kurs og opplæring i et felles miljø. Dette ble også påpekt under observasjonen hos IKTBergen.

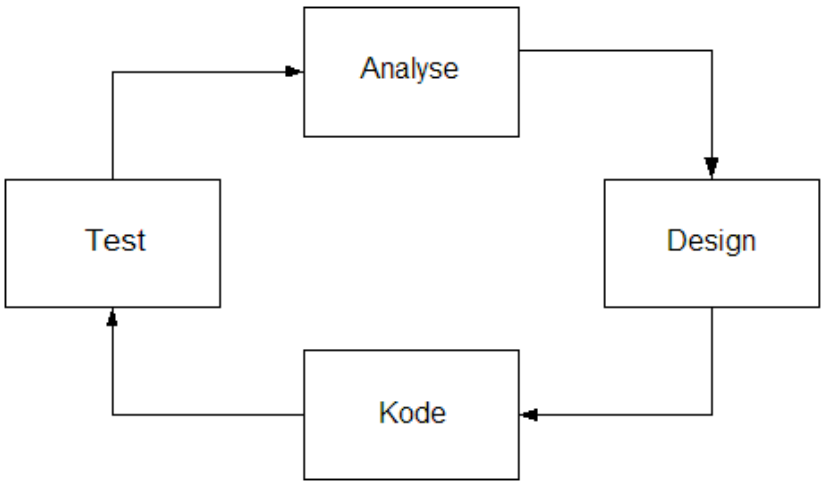
De funnene som kom fram gjennom analyse av observasjon og intervju har gått inn som krav til prototypen som presenteres i neste kapittel. Evaluering av denne vil bli gjennomgått i kapittel 8.

7 Prototypedesign i Second Life

I dette kapittelet vil det bli beskrevet hvordan prototypen i Second Life, som har til hensikt å støtte Scrum prosessen, har blitt designet på bakgrunn av den kvalitative undersøkelsen. Samarbeidsplattformen lar teammedlemmer møtes, sosialisere og arbeide sammen. Kapittelet er en gjennomgang av de valg som ble tatt i designfasen, og utfordringene med å sette seg inn i de teknikker, verktøy og valg som var nødvendig for å kunne bygge en prototype i Second Life (se appendiks B for mer utfyllende informasjon). Prototypen blir deretter gjennomgått, og funksjonaliteten forklart.

7.1 Valg av rammeverk

Utvikling av prototypen har fulgt kjerneaktivitetene til et utviklingsprosjekt, med fire faser; analyse, design, kode og testing, som vist i figur 17.



Figur 17 - Analyse-design-kode-test modell, etter Hoffer, George og Valacich (2005, s.16)

Analysefasen beskriver kravene som kom frem under den kvalitative undersøkelsen som input til design fasen. I designfasen ble prototypen utformet, før den ble implementert gjennom koding/utvikling. Underveis ble den testet oppimot designkrav (se tabell 8). Ved å se prototypen som er under utvikling oppimot kravene i en ny analysefase, kan den forbedres (se figur 17). Den ferdigstilte prototypen blir til slutt evaluert i kapittel 8.

7.2 Forberedelser

Designet av prototypen startet med å laste ned klienten til Second Life, lage en avatar, og logge på. I appendiks B blir det gitt en gjennomgang av dette, og hvordan man

designer objekter. Det som kan være noe underlig, er at man ikke kan velge det etternavnet man ønsker på avataren, men må velge fra en forhåndsdefinert liste. Dette er en begrensning i Second Life som forhindrer brukerne i å ha sitt eget etternavn.

Alle begrep som er benyttet under, og vist i hermetegn ved førstegangsbruk, er forklart i Wiki og kunnskapsdatabasen til Second Life (Linden Labs 2009b).

Det er to typer land i Second Life, disse går under betegnelsene ”mainland” og ”private regioner”. Den største forskjellen på dem er at mainland er eid av Linden Labs. Man kan kjøpe land hos dem på auksjoner eller direkte fra privatpersoner som selger. Private regioner derimot, er det gjerne eiendomsmeglere eller privatpersoner som kjøper og leier ut, eller bruker selv. De kan igjen drive med utleie av det landet de eier. En annen forskjell er at om man skal kjøpe land av Linden Labs så må man ha en ”oppgradert konto”[37]. Å lage seg en konto i Second Life er gratis, men det koster penger å eie land. Det å opprette en oppgradert konto har ulik pris, avhengig av type betalingsplan. Fra $72[38] dersom man betaler årlig, til $9,95[38] om en betaler pr måned. Man trenger ikke en oppgradert konto for å leie land hos en megler, kun når man skal eie mainland. Det å kjøpe land fra Linden Labs på auksjon koster fra noen kroner oppover til tusenvis av kroner avhengig av størrelse og plassering. I tillegg til selve anskaffelseskostnaden løper det en månedlig landleiekostnad til Linden Labs som skal betales.

Muligheten for gratis land ble sjekket opp, men det er ikke enkelt å finne noe sted en kan bruke over tid. Det er en del områder som er definert som ”sandbox”. Her kan alle bygge, men det er ofte mange restriksjoner på byggeaktiviteten og tingene man lager der blir sendt tilbake etter en stund, slik at det er vanskelig å bruke området til en prototype.

Anskaffelseskostnaden på land kjøpt på auksjon var høy, så selv om leien var billigere var det mer lønnsomt å leie. Når man leier betaler man vanligvis en litt høyere sum pr måned, men man slipper anskaffelseskostnaden. Det som bestemmer prisen på et landområde er størrelsen, typen land og antall ”prims” det støtter. Prims er den

37 Eng.: Premium account
38 Priser pr oktober 2009

betegnelsen som brukes på hvert objekt man lager. Dersom man skal bygge et bord (se appendiks B), vil dette typisk bestå av en bordflate med fire bein. Dette bordet vil da bestå av fem prims når det bygges.

Valget falt på å leie land til prototypen, og da gjenstod det å velge hvilken type vi ønsket. Det er tre typer land man kan velge å leie. Den første typen er en full region som er 65536 kvadratmeter og støtter 15000 prims. Leien for en slik øy er normalt på ca $295[38] dollar i måneden. Videre kan man leie et "homestead" som er 65536 kvadratmeter, men som kun støtter 3750 prims, med en månedlig leie på $125[38]. Den siste typen land man kan leie er en øy som heter "void". Den har veldig mange begrensinger i bruk og støtter kun 750 prims og koster $75[38] i måneden. Agentene øker leiekostnaden for å kunne ha fortjeneste, men fordelen er at man slipper anskaffelseskostnaden. De fleste agenter tilbyr å leie ut større og mindre deler etter behov. Prosjektet trengte ikke en hel øy, og dermed falt valget på å leie 1/4 av en homestead for å kunne lage prototypen. Denne er på 16384 kvadratmeter og støtter 750 prims. Lenken for å komme til prototypen er:
http://slurl.com/secondlife/Shuffle%20Customs/204/210/502

7.3 Analyse

Analysefasen vil basere seg på de krav og innspill som kom frem i kapittel 6, hvor data fra den kvalitative undersøkelsen ble gjennomgått. De innspillene som ble mottatt er brutt ytterligere ned til håndterbare krav, som man kan se i tabell 8. Intervju og observasjoner går inn som studiedelen av designforskingsrammeverket (Hevner, March, Park & Ram 2004). Valget på å benytte Second Life som plattform legger mange føringer og begrensninger på utviklingen av prototypen. Dette er det tatt hensyn til i kravene.

Mange av kravene og problemstillingene som kom frem fra intervju og observasjon var ikke enkle å oversette direkte til konkrete krav. Tema som skepsis, ledelse og best practice for Scrum, er med i det videre arbeidet med prototypen, selv om det ikke er konkrete krav i tabell 8.

Tabell 8 – Krav fra den kvalitativ undersøkelsen

Krav	Beskrivelse
Enkelt å bruke	• Løsningen må kunne settes opp raskt, Scrum møtet varer bare 15 min. • Man må enkelt kunne arbeide og delta på møter og samlinger uavhengig av fysisk lokasjon. • Ingen unødig funksjonalitet i møterommet utover det som trengs for å gjennomføre møtene. • Enkle og intuitive[39] løsninger vil også gjøre brukerterskelen lavere og behovet for opplæring mindre.
Bevegelse og synlighet	• Ønsker at man ikke må bevege for mye rundt, ting innen rekkevidde. • Forflytning må skje hurtig og greit. • Oversiktlige arbeidsområder, lett å få oversikt uten å bevege seg for mye.
Awareness	• Mest mulig integrert i samme bilde, færrest mulig skjermbilder. • Behov for å skille ut et privat workspace. • Når teamene er samlokalisert er det vanlig at man spør hverandre om hjelp for å løse ulike oppgaver, dette er noe en bør ha støtte for i prototypen også. Videre et ønske om å se hva de andre jobber med.

Verktøy og kommunikasjon	• Støtte både tekstlig og muntlig kommunikasjon.
	• Manuelle tavler man kan skrive meldinger på.
	• Funksjonalitet tilsvarende gule lapper.
	• Ressurser som må være tilgjengelig under et Scrum møte:
	  o Backloggen
	  o Buglister
	  o Programmet Scrum Works
	  o Andre nyttige verktøy og ressurser etter behov
	• Mulighet for å drive parprogrammering

Sosialisering og teambygging	• På den virtuelle arbeidsplassen bør man ha mulighet til å kunne utføre opplæring og utdanning av teamet. • Mulighet for å spille spill og sosialisere. o Når man bygger et møterom i et MUVE må en se på de fordelene det gir for å sosialisere og teambygging. Dette er spesielt viktig når de ulike miljøene som samarbeider ikke har muligheten til å møtes fysisk.
Andre krav	• Brukerne skal kunne benytte løsningen fra egen arbeidsplass • Lav kostnad og høy tilgjengelighet • Målet med å utvikle en prototype må være at den kan samle funksjonalitet i en paraply • Viktig at løsningen ikke krever for mye av internettbåndbredde. • Skal kunne benyttes på normale kontormaskiner.

7.4 Design

I dette avsnittet blir det gjennomgått de arkitektoniske designvalg som ble foretatt under konstruksjonen av prototypen. Valgene er basert på teorien som har blitt gjennomgått, og funn fra den kvalitative undersøkelsen som er oppsummert i tabell 8. Således skal det vises hvordan viktige aspekter ved Common Information Space (CIS) er gjenspeilet, hvordan awareness er ivaretatt gjennom designet, og hvordan

39 Intuitiv: umiddelbart oppfattende, som skyldes intuisjon og ikke erfaring eller refleksjon (fra ordnett.no).

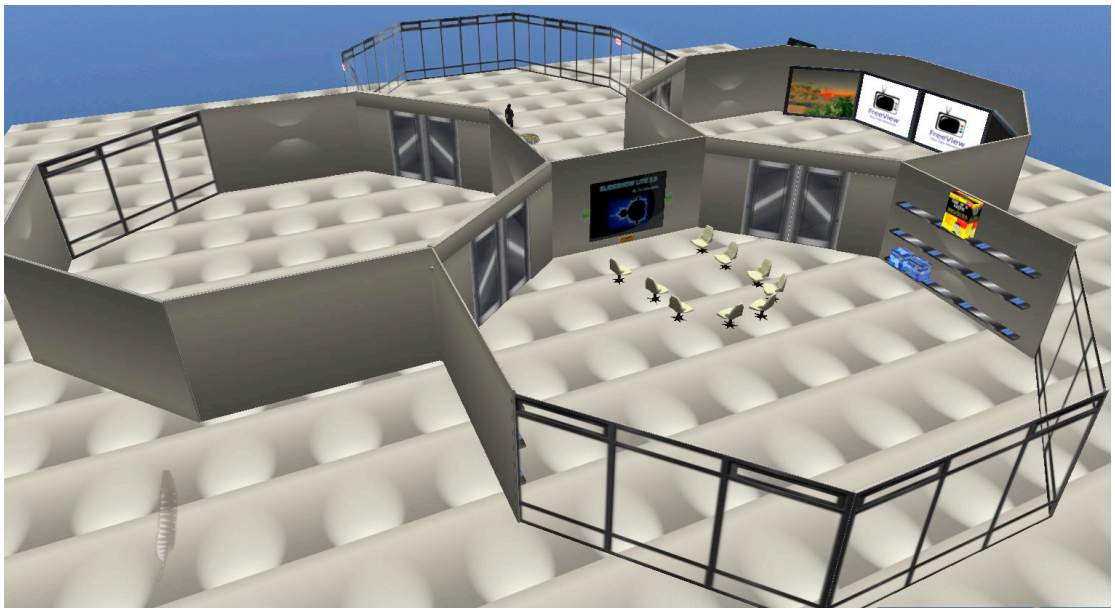
samarbeidsmekanismene er støttet i prototypen. Det å støtte samarbeidsmekanismene, som nevnt i avsnitt 2.4, er viktig for at basisaktivitetene i et shared workspace er mulig å gjennomføre. Det er som nevnt i avsnitt 4.6 samarbeidsmekanismene som utgjør grunnlag for evalueringen.

Arbeidet med designet hadde som mål å lage en prototype som hadde nok funksjonalitet for å svare på om en slik løsning kunne understøtte møter i et MUVE. Det ble bestemt at det skulle utvikles en konseptuell prototype i Second Life. Denne prototypen skule være en High-Fidelity prototype, fordi den skal inneholde funksjonalitet som er å finne i den endelige versjonen av produktet (Sharp, Rogers & Preece 2007). Arbeidet ble startet opp med å lage enkle skisser på papir av hvordan prototypen burde se ut, men dette arbeidet ble raskt flyttet inn i Second Life. Grunnen var at noe av styrken til Second Life er å manipulere objekter. I praksis ble hele arbeidet med å designe prototypen utført der, fordi dette ga muligheten til å se resultatet av designvalg umiddelbart. En annen stor fordel med å bygge her, var at man ikke trengte å være på samme sted for å arbeide på prototypen. Den ene kunne designe når han hadde tid, slik at den andre kunne logge seg på senere, se hva som hadde blitt laget, endre på det, og gi tilbakemelding.

7.4.1 Fra ide til designvalg

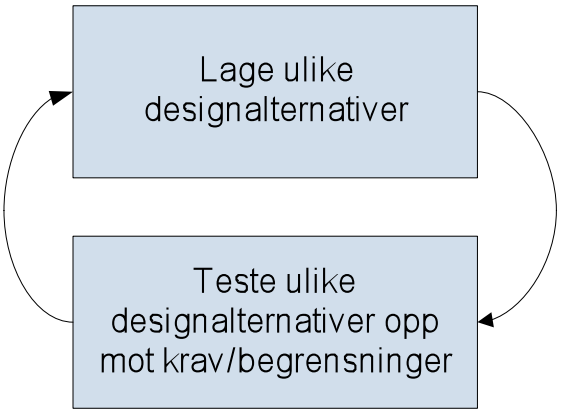
Det overordnede designet for prototypen hadde som målsetning at det skulle være en følelse av plass rundt avatarene, dette for å unngå kameravinkler der bildet til brukeren ender opp utenfor rommet som handlingen skal foregå i. Videre måtte designet være av en slik karakter at det ikke var behov for brukernes avatarer å bevege seg over store avstander for å komme mellom de ulike stedene. Dette ble gjenspeilet i designvalget hvor rommene ble bygget som bikuber, med åtte vegger i hvert rom. Sistnevnte var valgt for å stimulere brukernes awareness om hvor de andre teammedlemmene befinner seg. Selv om informasjon om hvor brukerne befinner seg blir present på et kart, er det bedre å kunne se de andre avatarene som arbeider workspacet. Når man satt sammen fire rom på denne måten var det kort vei å gå mellom de ulike rommene, og veggene var godt tilpasset for å vise ulik informasjon som vist i figur 18. Det er korte avstander mellom de ulike rommene, fordi dette kom frem som et krav fra den kvalitative undersøkelsen. Rommene har blitt designet for å støtte opplæring, sosialisering, samarbeide, og Scrum arbeidsmøter. Til slutt ble det

også utviklet et større område for å kunne møtes og sosialisere ytterligere.



Figur 18 – Prototypens arkitektur

Arbeidet med å utvikle prototypen har blitt gjennomført i henhold til en design/test fremgangsmåte (Hevner, March, Park & Ram 2004) slik at man kunne forsøke alternative ideer i praksis, som vist i figur 19. Dette har ført til at designet har blitt omgjort flere ganger underveis for å lage en så god løsning som mulig.



Figur 19 – The Generate/Test Cycle, oversatt til norsk fra Hevner, March, Park & Ram (2004, s. 89)

7.4.2 Tekniske løsninger for møter og samarbeid

Når man skal forbrede et møte i det virtuelle møterommet har man behov for å kunne sette opp de ulike systemene og presentasjonene man skal benytte for å gjennomføre møtet. Kommunikasjonen inn og ut av Second Life er styrt av mulighetene og

begrensningene som ligger i plattformen. MUVEet går mot mer åpne standarder som kan løse noen av disse begrensningene.

Når et møte skal klargjøres i Second Life er det to løsninger som virker tilfredsstillende. Den første er å ta skjermbilder av det man ønsker å presentere og laste dem opp via grensesittet. Da kan disse bildene presenteres på skjermer i møterommet. Dette fungerer bra for presentasjoner, grafer og andre statusbilder. Et annet alternativ er å bruke programvare som kan sende skrivebordet, eller et av programvinduene på datamaskinen, inn i Second Life. Denne prosessen har fått betegnelsen ”streame” på engelsk. Fordelen med å streame kun ett vindu er at man kan ha andre programmer åpne slik som e-post, uten at det er synlig for de andre deltagerne. Dersom man sender ut denne strømmen, på en IP-adresse og –port, kan Second Life koble til denne strømmen og vise den på en skjerm. Dette fungerer bra for vising av filmer, podkast, webkamera, og ulike dataskjermer live.

Videre kan man benytte den innebygde nettleseren for å benytte Internett ressurser. Man kan også vise nettsider på skjermer som er i møterommet. Dette kan være til nytte om man ønsker å vise en teknisk oppdatering eller annet innhold for Scrum teamet.

Den mest brukte måten å kommunisere med andre avatarer er å bruke den innebygde IM løsningen, som gjør at man tekstlig kan kommunisere med enkeltpersoner og grupper. IM understøtter *eksplisitt kommunikasjon* i henhold til Gutwin og Greenberg (2000), forklart i tabell 2.

En annen god løsning er å bruke VoIP. Da kan man snakke med de avatarene som befinner seg rundt en, eller man kan opprette private samtaler eller gruppesamtaler med to eller flere personer.

7.4.3 Rettigheter til objekter og land

Når man lager et objekt blir en både eier og skaper av objektet. Dette er de to rollene som avatarer kan ha til et objekt. Dersom et objekt er både laget, og eid av samme person, er det han som har generert det første gang. Eier av et objekt kan sette rettigheter på de objektene han eller hun lager. Disse rettighetene arves når objektet

går videre til andre. De tre rettighetene man kan sette på et objekt er kopiere, endre og gi videre til andre. Den som eier objektet kan ikke oppheve begrensninger som er satt av den som laget eller som eide objektet tidligere, men man kan sette nye begrensninger på det. Eksempel på dette er at man har et objekt som er mulig å kopiere, samt å gi videre. Da kan eier si at det skal være mulig å gi videre, men ikke kopiere for neste eier. Brukerautentiseringen for pålogging støttes for å beskytte opphavsretten til objekter en avatar lager. Denne måten å forvalte rettigheter på er gjennomgående enten man snakker om et objekt, en tekstil, scripts, med mer. Dette støtter oppunder Gutwin og Greenbergs (2000) samarbeidsmekanisme som kalles *beskyttelse*. Denne går ut på at brukerne av et gitt system skal være beskyttet mot ødeleggelse av sitt arbeidet.

I Second Life er det mulig for landeiere å sette rettigheter til hva andre innbyggere kan gjøre på land eieren kontrollerer. Man kan velge om alle, ingen, eller personer tilhørende en bestemt gruppe kan bygge objekter. Dette gjøres via ”about land” menyen.

7.4.4 Design av Scrum møterom

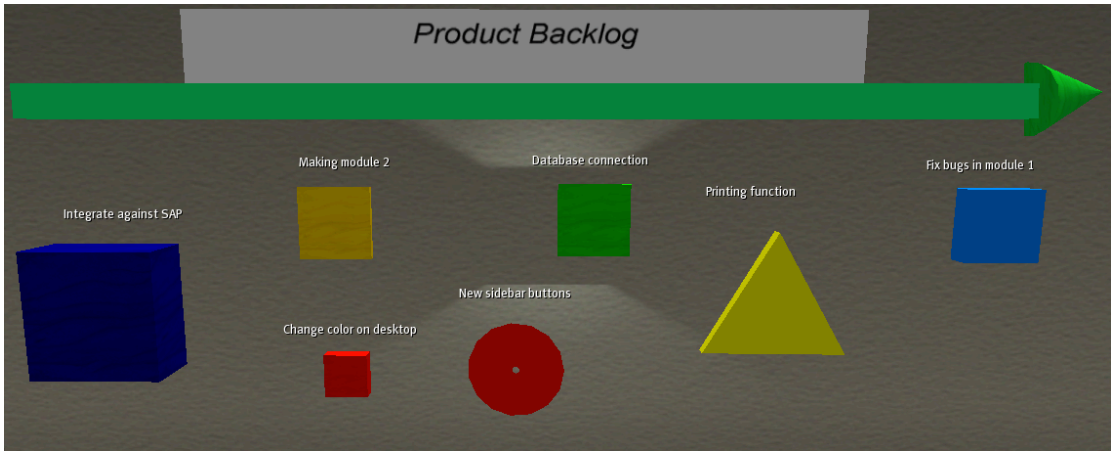
De krav til et møterom for daglige Scrum møter, som ble fanget opp, er at rommet bør ha tilgjengeliggjort Product Backlog, buglister og Scrum Works for å kunne vise ulike informasjonselementer. Videre ser man behovet for at det er oversiktelig, inneholder kun nødvendig funksjonalitet, og er enkelt å bruke. Innenfor workspace awareness er det viktig å kunne ha oversikten over hvem som er aktiv i et delt workspace. Konseptet påpeker hvor viktig det er at systemet gir informasjon om menneskene som eksisterer rundt en. Dette blir ivaretatt ved hjelp av avatarer, hvor man også kan definere rollen til brukeren. Som det ble nevnt i avsnitt 2.2 kan mennesker med ulik bakgrunn og kompetanse, problematisere hvordan de oppnår en felles forståelse for informasjonsobjektene i et CIS. Denne tilleggsinformasjonen (om rollen til hver bruker) kan påvirke hvordan de forholder seg til hverandre. Her vil de ulike rollene i Scrum, som forklart i avsnitt 2.6.3.2, være med å bidra til at man kan orientere seg om hvilken bakgrunn og kompetanse meddeltagerne har.

Rettighetene til objekter kan også gjenspeile hvilken rolle brukeren innehar. Tanken er at møtedeltagerne stiller seg opp i en halvsirkel, og ser mot skjermene, hvor det

over ”hodet” på hver avatar, går klart fram om man for eksempel er Scrum Mester, Produkt Eier, eller team medlem. Her vil Scrum Mester kunne lede møtet, vise Product Backlog, lister med feil, og grafer fra Scrum Works under møtet. Alle deltagerne vil da kunne følge med på skjermene og få den samme informasjonen som vist i figur 20.

Figur 20 – Scrum møterom.

Det har også blitt utviklet en Produkt Backlog liste i møterommet (se figur 21), som hver deltager kan manipulere. Dette blir en tavle tilsvarende den vist i figur 16. Her kan man manipulere størrelse, farge og beskrivelse av objektene. Dette bidrar til å *planlegge* aktivitetene som skal gjennomføres i prosjektet. Dette er også relatert til en av samarbeidsmekanismene til Greenberg og Gutwin (2000). Bjerrum & Bødker (2003) fremhever at når prosjektinformasjonen kan stå fremme mellom møtene, kan det bidra til økt kunnskapsdeling mellom deltagerne.



Figur 21 – Product Backlog

Det å kunne koordinere det daglige programmeringsarbeidet er med på å bedre teamdynamikken og forbedre samarbeidet internt i teamet, og i tillegg unngå at flere jobber med samme oppgave. Videre kan det å kunne delta på møter hjelpe med planleggingen og organiseringen av programutviklingen som teamet skal utføre. Dette underbygger samarbeidsmekanismen *koordinasjon av handling* (Gutwin & Greenberg 2000). Koordinasjon og planlegging er også viktige momenter i articulation work-konseptet (se avsnitt 2.3).

7.4.5 Design av rom for opplæring og møter

Det kom frem i intervju at det å drive opplæring og faglige samlinger i et virtuelt miljø var noe som kunne gi store gevinster. Rommet er enkelt utstyrt med en fremviser, og noen enkle stoler som avatarene kan benytte. Her er det bare for møteleder å laste opp presentasjonen, legge inn filene i fremviseren og starte. Møteleder kan velge om møtet skal holdes via IM eller VoIP. Oppbyggingen av rommet er vist i figur 22.

Figur 22 – Rom for opplæring og møter

Det har også blitt tilrettelagt for et online bibliotek i møterommet. Her kan man stille ut ressurser, og linke til nettsider med aktuell informasjon for pågående prosjekter. Bjerrum & Bødker (2003) forklarer som nevnt tidligere at det å ha prosjektinformasjon tilgjengelig for alle medlemmene, bidrar til læring og

kunnskapsdeling. Her kan man linke til ressurser internt i Second Life, både steder og bøker, og man kan linke til Internett referanser (se figur 23).

Figur 23 – Ressursbibliotek.

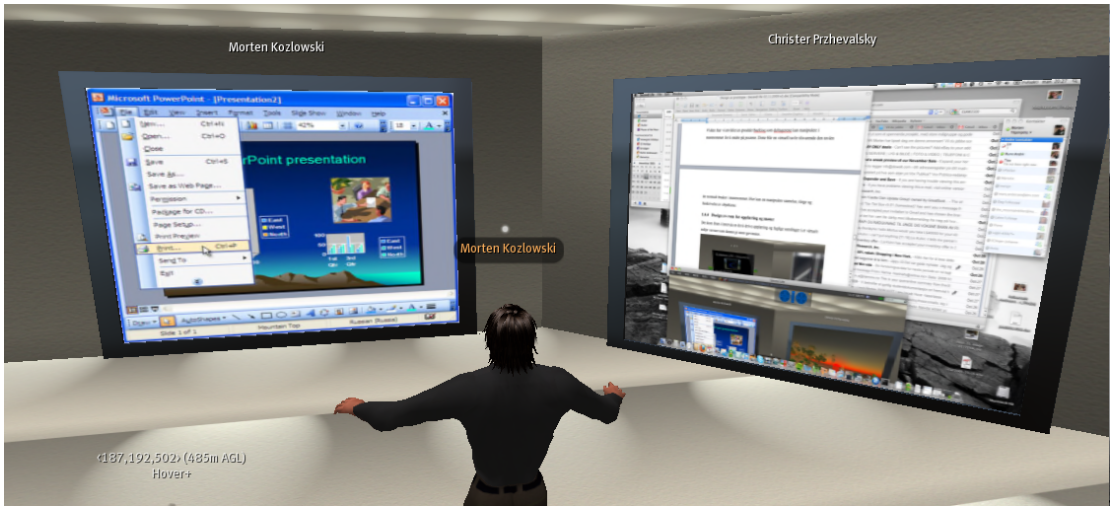
7.4.6 Design av samarbeidsrom

Et betydningsfullt konsept under designet av dette rommet har vært Workspace Awareness (WA). Det er sentralt innenfor WA å vise hva brukerne foretar seg på nåværende tidspunkt. Dette kom også fram som et ønske i den kvalitative undersøkelsen. Dette gjennspeiler seg i kravet om å kunne benytte parprogrammering. Ved å overføre[40] skrivebord, eller ønsket programvindu, kan man vise medarbeiderne hva en holder på med. Ved at navnene settes over skjermene klarer man å skille dem fra hverandre (se figur 24).

Ved å arbeide sammen i et delt workspace kan man overhøre hva andre holder på med. Dette er *indirekte kommunikasjon* som gir tilgang på informasjon man ellers ikke ville fått, og dette hjelper skaper en bedre *koordinering av handling*. Videre kan man spørre sine kolleger om *hjelp,* ved å benytte *eksplisitt kommunikasjon,* for å løse ulike oppgaver, noe som kan være til stor nytte når en arbeider. Dette omtaler Gutwin og Greenberg (2000) som samarbeidsmekanismer. De erfarne programmererne kan også observere arbeidet til de mindre erfarne teammedlemmene, slik at de kan tilby

40 Eng.: Streaming

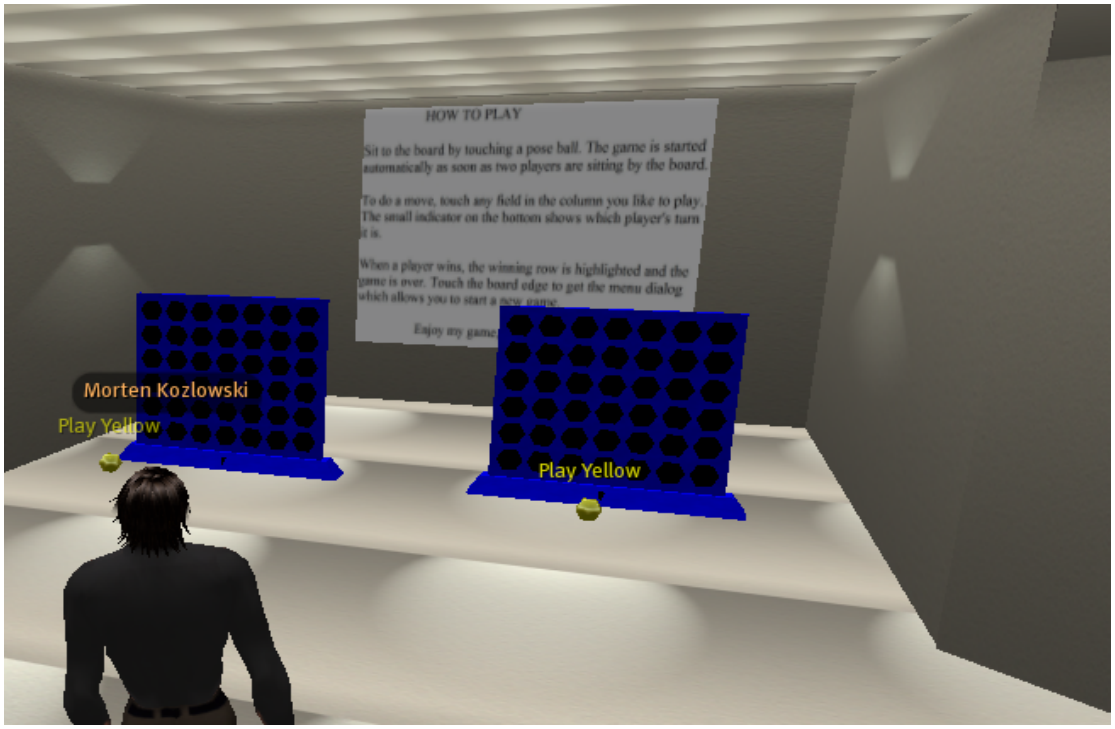
sin hjelp hvis det er nødvendig. Her benyttes med andre ord samarbeidsmekanismene *overvåking* og *hjelp*.



Figur 24 – Samarbeidsrom

7.4.7 Rom designet for sosiale aktiviteter

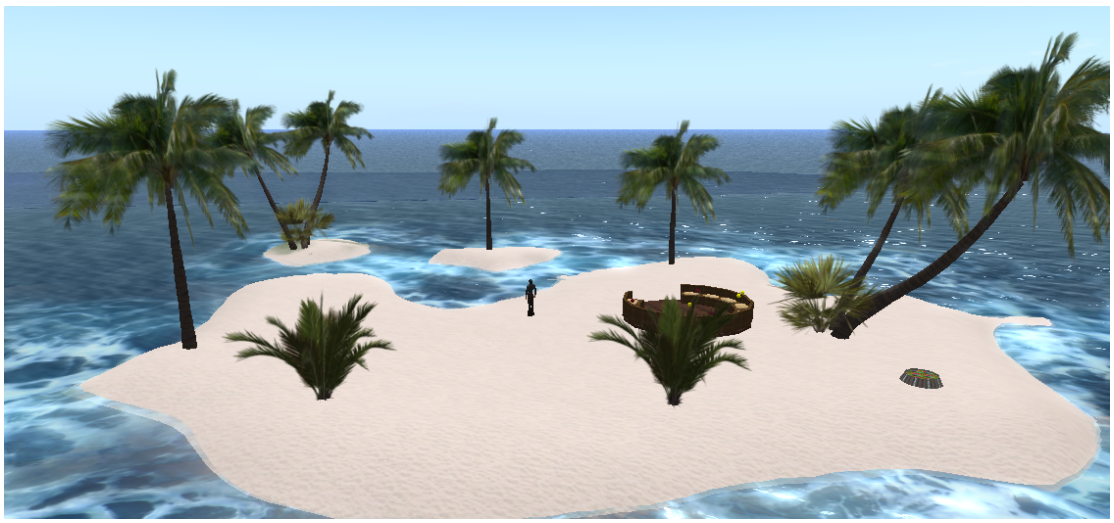
Sosialisering via det å spille spill var noe som hadde stor suksess i IKTBergen. Det ble, på bakgrunn av dette, laget et rom i prototypen hvor tilsvarende aktiviteter kan gjennomføres. Det å kunne ta små avbrekk i en hektisk programmeringssituasjon ble av IKTBergen sine medarbeidere fremhevet som en av suksessfaktorene for å lykkes. Kombinasjonen med å kunne gjøre ulike sosiale aktiviteter sammen med teammedlemmer som ikke er samlokalisert, kan bidra til å skape en bedre teamfølelse i det distribuerte teamet. Dette kan også være en døråpner for å bedre kommunikasjonen mellom distribuerte team (se figur 25).



Figur 25 – Rom for sosiale aktiviteter

7.4.8 Design av rekreasjonsområde

Prototypen har fått tilført et rekreasjonsområde uten for selve arbeidsområdet. Dette området kan man nå ved å bruke teleporteringssystemet som er blitt satt opp mellom rekreasjonsområdet og arbeidsområdet. Tanken med dette er at man kan trekke seg tilbake og snakke mer uformelt i avslappede omgivelser som man ser i figur 26.



Figur 26 – Rekreasjonsområde.

På rekreasjonsområdet, som er designet som en tropisk øy, er det blitt laget til en sittegruppe som vist i figur 27.

Figur 27 – Sittegruppe.

7.4.9 Design av nettside

Det har blitt utviklet en støttenettside til prototypen i Second Life på www.heppy.org. Dette er en low-fidelity nettside som er ment å være et supplement til prototypen. Her kan man lagre samtalelogger fra daglige Scrum møter hvis disse blir gjennomført ved tekstlig kommunikasjon. Her er kan man legge inn en lenke som man kan benytte for å komme direkte til prototypen. Tanken er at man i Second Life kun skal trenge å forholde seg til det som befinner seg i prototypen.

7.5 Koding og videreutvikling

Mesteparten av arbeidet med å lage prototypen ble utført i designfasen. Når noe er designet i Second Life er det klart til bruk med en gang. Dette medfører at det er en flytende overgang mellom prototypen og et endelig produkt. I kodefasen av prosjektet ble nettsiden utviklet, med tilhørende scrips. Videre ble de ulike scripts som var nødvendig for å få prototypen til å virke. For at en avatar skal kunne gjøre ulike bevegelser slik som å sitte på en stol, trenger objektet, det vil si stolen, å inneholde et script som gjør dette (se appendiks B). I prototypen har vi implementert script for å åpne og lukke dører. Mulighet for avatarer til å teleportere seg fra en lokasjon til en

annen, det vil si flytte seg uten å gå mellom ulike lokasjoner. Videre har vi laget Script for å sitte på objekter slik som benken i figur 27.

7.6 Tilleggsmomenter om designvalg

Det var ikke alle designkrav (se tabell 8) som var like enkle å implementere. For eksempel det å utvikle en løsning som fungerer for møter som ikke skal vare lengre enn 15 minutter. Dette var utfordrende fordi denne tiden skal inkludere alle de tekniske forberedelsene. Etter prototypen var fullført ble dette testet. Det viste seg at det ikke var vanskelig å starte opp, og gjøre seg klar i prototypen, på under et minutt. Dette forutsetter at man har litt erfaring med bruk av den.

Et mål med prototypen er at den skal brukes mer i den daglige arbeidssituasjonen også, slik at brukerne blir mer fortrolig med den. Det er først da den kommer til sin fulle nytte. Det har i denne studien ikke vært fokusert på å utvikle et nytt grensesnitt til Second Life, slik at man kan integrere flere tjenester sammen til en plattform. Derimot har det blitt laget en løsning som gjør det mulig å gjennomføre møter på en samlet plattform. Second Life kjører fint på selv enkle bærbare datamaskiner, dersom grafikken ikke skrus for høyt opp. Når det kommer til linjer og båndbredde, har våre målinger vist at dersom avatarer befinner seg på samme sted, trengs ikke like kraftig forbindelse, som om avatarene flyr rundt for å oppdage nye steder. Dette vil med andre ord si at hvis bedriften etablerer er fast møtested, slik som i studiens prototype, skal ikke linjekapasiteten være den største utfordringen.

7.7 Oppsummering

I dette kapittelet har det blitt beskrevet hvordan prototypen har blitt utviklet. Dette innebar valg av teknologi, brukerkrav, og arkitektur i Second Life. I tillegg har designvalgene blitt gjennomgått og det har blitt beskrevet hvordan teori fra CSCW-feltet blir gjenspeilet i prototypen.

Ved å benytte teori fra CSCW-feltet til å designe en prototype som er ment å forbedre, forenkle og støtte distribuert programvareutvikling som cooperative work har dette kapittelet vist en løsning på hvordan dette kan gjøres. I neste kapittel vil prototypen bli evaluert, med hensyn til teori fra CSCW, for å se om dette er en løsning som fungerer i henhold til kravene som kom frem i den kvalitative undersøkelsen.

8 Evaluering av prototype

Studien har benyttet Heuristic Evaluation of Groupware (Baker, Greenberg & Gutwin 2001) og Groupware Walkthrough (Gutwin & Pinelle 2002) for å evaluere prototypen (se avsnitt 4.6.2 for en oversikt). Det er åtte heuristikker som har blitt vurdert, ved hjelp av fire masterstudenter, i henhold til prototypen. Etter gjennomgangen av hver heuristikk blir denne evalueringsteknikken oppsummert i avsnitt 8.1.9. Groupware Walkthrough blir så gjennomgått, hvor det har blitt valgt ut fem scenarioer for teamsamarbeid. I denne gjennomgangen har det deltatt opp til fem personer med programmeringskjennskap, for å teste hvorvidt man klarer å gjennomføre scenarioene. Kjennskapen til Second Life varierte fra ingen erfaring til mye. Etter scenarioene var gjennomført ble disse evaluert for å vurdere hvor god usability prototypen besitter. Etter de to evalueringsteknikkene er ferdig beskrevet, har helheten blitt vurdert i et eget oppsummeringsavsnitt.

8.1 Heuristic Evaluation of Groupware

De engelske titlene til heuristikkene er beholdt fra artikkelen til Baker, Greenberg og Gutwin (2001), fordi man står i fare for å miste forfatternes intensjon ved å oversette til norsk. Evalueringen har tatt hensyn til et Scrum teams behov, på bakgrunn av den kvalitative undersøkelsen.

8.1.1 Heuristic 1 - Provide the means for intentional and appropriate verbal communication

Second Life tilbyr både IM og audio kommunikasjon via VoIP. Digital lyd har blitt en moden teknologi, og i dag har man ikke de samme problemene en hadde på grunn av dårlig båndbredde, ventetid og kvalitet. Det er ingen mulighet for videooverføring, men det er ikke noe vi har savnet, da avatarene gir en god tilstedeværelse. Med IM og VoIP har man mulighet til å snakke med alle som er i området rundt en, altså lokal IM kanal, til en spesifikk gruppe, eller man kan velge å føre en privat samtale. Når en skriver tekst til noen, så blir ikke det man skriver sendt før man trykker enter. Dette blir betegnet som pakkepost, i motsetning til sanntids IM, hvor en og en bokstav blir sendt til mottaker.

Saul Gutwin oppsummerer tre måter å motta informasjon fra verbal utveksling slik (Baker, Greenberg & Gutwin 2001, s. 126):

Talk explicitly: Dette er godt støttet. Hvis noen på teamet lurer på hva du holder på med, kan en forklare både via IM eller via VoIP.

Overhearing: Dette fungerer kun når man snakker i lokal IM vindu. Prater man derimot direkte med en annen person, så er det ikke mulig for andre å overhøre hva de snakker om, uansett hvor nærmt man står. Dette gjelder både IM og VoIP.

Running commentary: Dette er ikke naturlig å gjøre hvis man benytter IM, da man bevisst tenker over det man skriver, og slike uttalelser er noe man gjerne gjør uten å tenke over det. Men hvis man benytter VoIP, så fungerer dette på en god måte, selv om det kanskje ikke føles like naturlig som i den virkelige verden.

8.1.2 Heuristic 2 - Provide the means for intentional and appropriate gestural communication

Eksplisitte fakter og andre visuelle handlinger blir brukt ved siden av muntlige utvekslinger for å utføre intensjonal kommunikasjon. Denne formen for kommunikasjon er noe groupware-systemet må ta høyde for, siden de er usynlige for de ulike partene på grunn av at man er avstandsseparert. Hvis man har med slik støtte, så løses dette som oftest med en form for legemliggjøring. I Second Life tar dette form av avatarer, og selv om avatarene er relativt enkle i sin utforming, kan man gjøre ulike former for hånd og kroppsbevegelser. Fordelen i Second Life, er at det nærmest ikke finnes noen begrensning for hva som kan scriptes av bevegelser. Dette betyr at man kan få laget et stort sortiment av intensjonale fakter. Er man på en virtuell forelesning, så er det ikke noe i veien for at en kan få avataren til å rekke opp hånden.

8.1.3 Heuristic 3 - Provide consequential communication of an individual's embodiment

Ved å observere at en persons kropp interagerer i workspace kan en hente inn mye informasjon, fordi kroppshandlinger som posisjon, holdning, og bevegelser av hode, armer, hender og øyne avgir uintenderte signaler som andre kan ta til seg. Dette er nyttig, da det å observere andres arbeid er en av primærmekanismene for å hente inn awareness informasjon om hva som foregår, altså hvem som er i workspace, hvor de er, og hva de holder på med. Et viktig poeng, som skiller dette fra eksplisitte fakter

som i heuristikk 2, er at skaperen av informasjonen ikke med viten utfører noen handlinger for å informere den andre part, og at den som observerer kun tar til seg det som er tilgjengelig. Uintendert kroppsspråk kan deles i to kategorier, som til sammen gir oss samtale awareness (Baker, Greenberg & Gutwin 2001, s. 128):

Handlinger knyttet til workspace:

- Gaze awareness (altså at man vet hvor en annen person ser).
- Mulighet for å se at en annen person beveger seg mot et objekt.
- Kan høre gjenkjennbare lyder etter hvert som deltagere utfører sine gjøremål.

Handlinger knyttet til samtale:

- Subtile hint en plukker opp fra samtalepartner, slik at man kan justere sin språklige oppførsel.
- Visuelle: Ansiktsuttrykk, kroppsspråk, øye kontakt og fakter.
- Verbale: Intonasjon, pauser, eller det å benytte spesifikke ord.

Målet for et groupware-system er å fange opp og overføre, den eksplisitte og den subtile dynamikken som oppstår mellom samarbeidende deltagere. Dette er ingen enkel oppgave, og selv om man har benyttet teknikkene for legemliggjøring som er nevnt i heuristikk 2, så er de svært begrenset. Når det gjelder Second Life, har man knyttet synsretningen til avataren til brukers synsfelt. Dermed kan andre til en viss grad observere hva en annen avatar ser på. Selv om dette hjelper, så betyr den svake koblingen mellom en persons faktiske kropp og avatarens, at mange betydningsfulle signaler ikke blir fanget opp og sendt til de andre teammedlemmene. Second Life sin VoIP mulighet gir et godt samtale supplement, som kan bistå til awareness. Kvaliteten på lyden er like god som andre former for VoIP. Man kan snakke samtidig, og i studien har det med hell blitt testet ut samtaler mellom Norge og USA. Stemmen er retningsgivende, noe som betyr at man til en viss grad kan vite hvor i rommet den man snakker med befinner seg, uten å se etter avataren.

8.1.4 Heuristic 4 - Provide consequential communication of shared artifacts (i.e. artifact feedthrough)

Når man har en ansikt-til-ansikt situasjon, vil betydningsfull kommunikasjon ofte involvere uintendert informasjon som gjenstander avgir, når de blir manipulert av

individer. Når den som manipulerer et objekt får tilbakemelding, kalles denne informasjonen *feedback*. Hvis noen observerer at objektet blir manipulert, får de *feedthrough*, som er den uintenderte informasjonen objektet avgir. En annen ressurs for ansikt-til-ansikt interaksjon er evnen til å identifisere den som manipulerer en gjenstand. Hvis man vet hvem som har utført en handling har man kontekstuell informasjon som hjelper en å forstå handlingen. Siden aktør og handling befinner seg på ulikt sted, er feedthrough vanligvis den eneste måten systemet har for å dele gjenstandsinformasjon med andre deltagere.

I et groupware-system, er det viktig at *feedback* blir delt med alle deltagere. Hvis et objekt blir manipulert, må systemet vise at objektet blir valgt, og hva som skjer med det frem til dets endelige posisjon. Dette kan støttes av *handlings feedthrough*. Tidligere var det vanlig at groupware-systemer hadde støtte for WYSIWIS (Hva du ser, er hva jeg ser). I Second Life kan en se hvilke objekter andre avatarer manipulerer i den virtuelle verden, ved at det kommer lysende punkter i luften fra avatar til gjenstand. Gjenstander kan også ha lyder knyttet til seg, som de vil avgi når de blir manipulert. Hver gjenstand har også historikk knyttet til seg. Her kan man få informasjon om hvem som har laget det, og hvem som er nåværende eier. Fordi gjenstander ikke blir slitt, eller blir eldre, gir ikke dette den samme forståelsen av hva som har skjedd med en gjenstand, som vi har i det virkelige liv.

Det er i dag vanskelig å reprodusere den store rekkevidden av subtile lyder som finnes i et naturlig workspace, og da alle brukerne i Second Life kan knytte sine egne lyder til gjenstander, kan man risikere at lydene gjenstandene avgir, er helt annerledes enn i vår egen verden.

8.1.5 Heuristic 5 - Provide protection

Når man samhandler ansikt-til-ansikt i et delt workspace, vil en ha fysiske begrensninger som forhindrer flere deltagere i å interagere på samme tid. Groupware-systemer på sin side gir mulighet for at flere kan arbeide parallelt i samme workspace, og samtidig manipulere det samme objektet. Dette er svært nyttig, men kan skape konflikt ved at man kan forstyrre hverandres arbeid, eller ødelegge noe som allerede er ferdig. Et groupware-system må beskytte brukerne fra slike situasjoner.

Det er to viktige elementer som skaper en slik beskyttelse, ifølge Baker, Greenberg og Gutwin (2001, s. 131) :

- Mulighet for å kunne forvente hva de andre aktørene har tenkt å foreta seg.
- Sosiale protokoller for samhandling.

I Second Life har man mulighet for å begrense tilgang til objekter. Dette er helt vesentlig, da alle gjenstander har en viss pengeverdi, og systemet hadde brutt sammen hvis man kunne manipulere andres objekter, uten at man kunne forhindret dette på noen måte. Det er også mulig å gi tilgang til sine objekter til andre, slik at en kan arbeide på dem sammen (se avsnitt 7.4.3). Det er derimot vanskelig å vite hva en annen har tenkt å foreta seg, uten at den andre part eksplisitt uttaler sine intensjoner. Det er altså slik at ved de lysende prikkene som kommer frem, slik forklart i heuristikk 4, får man vite hvilket objekt som skal bli manipulert og av hvem, men en får ikke vite hva som skal skje med det.

8.1.6 Heuristic 6 - Management of tightly and loosely-coupled collaboration

Med sammenkobling mener vi i hvor stor grad aktører samarbeider. Altså snakker vi om hvor mye arbeid en person kan utføre, før det krever diskusjon, instruksjon og konsultasjon med andre. Man bytter mellom løs og tett sammenkobling, hvor en altså flyter mellom individuelt arbeid og teamarbeid. For å håndtere slike overganger, må man kunne se på ulike deler av workspace når man utfører individuelt arbeid, slik at en kan følge med på hva de andre holder på med. For å hjelpe noen med deres oppgaver, må en vite hva de gjør, hva deres mål er, hvor langt de har kommet med oppgaven og status på arbeidsområdet.

I prototypen kan man for eksempel bygge objekter sammen, enten ved at man samarbeider om å bygge enkelte deler, eller at en setter sammen hver sine objekter til et større byggverk. Ved at en kan se hva den andre bygger, har man mulighet for å gi hjelp og råd til hverandre, mens en jobber med sine egne objekter. Man har også mulighet for å be om hjelp, enten via VoIP eller IM. Ellers har man full mulighet til å styre kameravinkel, slik at en kan få et overblikksbilde over hvor de andre avatarene befinner seg, og en kan få, slik vi har vist i de andre heuristikkene, delvis informasjon over hvilke handlinger de utfører. Årsaken til at dette bare blir delvis, er at man kan

utføre handlinger i Second Life, som ikke blir gjort tydelig for andre, slik som for eksempel å prate uten at andre kan overhøre samtalen. Nå vises det riktignok i IM vinduet at noen prater sammen, og dette hjelper for å bedre awareness.

8.1.7 Heuristic 7 - Allow people to coordinate their actions

En vesentlig del av ansikt-til-ansikt samarbeid er hvordan teammedlemmene medierer interaksjonen seg i mellom ved å vente på tur og forhandle om bruk av den delte workspace. Dette gjør man for å unngå konflikter med andre, og for å kunne effektivt fullføre det man skal gjøre. Koordinering av handlinger er nødvendig for at oppgaver skal skje i riktig rekkefølge, til rett tid, og samtidig skje innenfor oppgavens begrensninger.

Når man skal støtte koordinering av samhandling i et groupware-system, er det viktig at en ikke skaper et system som har til hensikt å håndtere dette for brukerne. Dette fordi vi som aktører vanligvis er dyktige til å koordinere kommunikasjon og interaksjon selv, og dermed må verktøy som lages støtte deltagernes egne evner. Workspace awareness må gi den informasjonen en trenger for å forstå hvorvidt andres oppførsel eller hendelser passer inn med det man forventer fra tidligere erfaringer. Groupware må dermed gi mulighet for at man kan ha oversikt over andre i det delte arbeidsområdet, og se hva de gjør på. De visuelle teknikkene fra Heuristikk 2 til 5 vil være en hjelp til å få dette på plass, og som det ble påpekt over har Second Life bra støtte for dette, selv om de subtile aspektene til dels faller utenom.

MUVE tilbyr en annen form for awareness enn andre CSCW-systemer, fordi at en har avatarer som man kan benyttes i samhandling med andre. En virtuell verden overgår på mange måter begrensningene en har ved at man er på ulikt sted til samme tid. Ved at brukerne kan befinne seg på samme virtuelle sted, til samme tid, har aktører bedre mulighet for å koordinere samhandling. I tillegg til å måtte støtte de ulike elementene over, må et groupware-system støtte verbal kommunikasjon for å støtte samarbeid. Dette, som vi så over, har Second Life støtte for ved bruk av VoIP.

8.1.8 Heuristic 8 - Facilitate finding collaborators and establishing contact

Vanlige groupware-systemer har ikke god støtte for uformelle møter, og det er et stort hinder for samarbeid, da de færreste møter er formelle. Formelle møter er gjerne

avtalt via IM, e-post, telefon, og så videre. Uformelle møter blir lagt til rette for ved at man befinner seg i fysisk nærhet til hverandre, slik at en kan vedlikeholde awareness over hvem som befinner seg omkring der man selv er. Dermed oppstår gjerne de uformelle møtene nettopp på grunn av denne nærheten, som for eksempel når to personer møtes på gangen. Selv om samtalene ikke trenger å være lange, kan man få koordinert handlinger, utveksle informasjon, eller tilby muligheter. Vellykkede team er avhengig av regelmessig, uformell og uplanlagt kontakt mellom medlemmene. Det er ekstra vanskelig å støtte denne typen uformelle groupware møter, da flaskehalsen for rik spontan interaksjon er avstand.

Det er på dette punktet Second Life virkelig viser fordelene med å benytte et MUVE som plattform for CSCW. Som nevnt i heuristikk 7, gir koblingen virtuelt sted på samme tid, mange av de mulighetene for å treffe på andre og ha uformelle møter, som man har i den virkelige verden. Det er lett for andre å se at man er tilgjengelig, ved at en kan jobbe med enkeltoppgaver og samarbeidsoppgaver i samme delte workspace. Det finnes også flere muligheter for å vite hvem som befinner seg i område rundt, ved at en kan se de andre aktørene, få opp kart over hvor alle befinner seg, eller at man kan bevege seg i det virtuelle landskapet for å se hvem som er der. Andre hjelpemidler er brukerlister, statusmenyer, og det at man kan hente fram informasjon om den enkelte avatar. Som i vår verden, kan en vidt åpen dør indikere at man er klar for å motta besøk, mens en lukket dør indikerer at man er opptatt. Ved at en har muligheter for å sende IM meldinger til andre, som enten mottaker får umiddelbart, eller blir sendt på e-post, spørre noen direkte over VoIP, har man gode muligheter for opprette kontakt. Fordi en har et virtuelt workspace, er arbeidskonteksten lett tilgjengelig. Alle verktøy og objekter har en for hånden, og nye verktøy kan bli lagt til.

8.1.9 Oppsummering

Ved å benytte heuristikker og awareness rammeverket som grunnlag for evaluering av prototypen beskrevet tidligere, har flere sider av et MUVE blitt fremhevet. Dette har blitt oppsummert i tabell 9 (nåtid) og tabell 10 (fortid). Utgangspunktet for denne gjennomgangen har hatt som forutsetning at deltagerne arbeider som et Scrum team, og dermed har gitt de andre medlemmene fulle rettigheter til informasjon om hverandre i prototypen.

Tabell 9 – Elementer for workspace awareness i relasjon til nåtid (Tabell fra Gutwin & Greenberg 2002 oppimot denne studiens empiri)

Kategori	Element	Svar for prototypen i Second Life
	Tilstedeværelse (Awareness) (Er det noen i workspace?)	Et kart viser alle som er pålogget Second Life. Venner kommer opp i en venneliste, hvor påloggingsstatus vises. Avatarer, som er pålogget Second Life og befinner seg i workspace, indikerer at de er tilgjengelig, vet at man kan se dem.
Hvem	Identitet (Hvem deltar? Hvem er det?)	Deltagere identifiseres ved en avatar som har navn stående over "hodet". Man kan få frem mer informasjon om en person ved å ta frem "profilen" til en bruker. Man kan også kjenne igjen avaterer man har møtt tidligere.
	Eierskap (Authorship) (Hvem utfører handlingen?)	I Second Life vises hvem som utfører handlingen ved at det går en hvit prikkete linje bort til objektet brukeren utfører handlingen på. En kan også til enhver tid se hvem som er eier og hvem som har laget et objekt.
	Handling (Action) (Skjer det noe?, Hva gjør de?)	Meningsfull kommunikasjon vises gjennom avatarens tilstedeværelse. Et eksempel på dette er når en bruker skriver en beskjed som kan oppfattes av andre brukere innenfor en begrenset virtuell lokasjon. Dette manifesteres gjennom avataren ved at avataren beveger hendene i tastetrykk bevegelser og lyd av tasting.
Hva	Intensjon (Hvilket mål er handlingen del av?, Hva skal de oppnå?)	Man kan kommunisere handlinger en skal utføre. Handlinger avataren utfører blir med en gang synlig i Second Life. Eksempelvis at man kan se og høre at en person skriver en melding, eller at man kan se at en annen bruker flytter avataren sin i retning av et objekt.
	Artefakt (Hvilke objekter jobber de med?)	Når en bruker manipulerer eller gjør nytte av et objekt, markeres dette med hvit prikket linje fra brukerens avatar til det aktuelle objektet. Manipulering og bruk av objekter blir med en gang synlig for andre brukere innenfor samme virtuelle lokasjon. Objekter kan avgi lyd ved bruk.
	Lokasjon (Hvor utføres handlingen?)	Man kan se på kartet hvor de andre avatarene befinner seg. Avstand til det som manipuleres kan manifesteres ved lyd. Desto lenger avstand jo lavere lyd. Denne funksjonaliteten avhenger av at det er lagt til rette for lyd, både på objektet som blir manipulert og hos brukeren. Dette vil med andre ord si at det ikke nødvendigvis er logisk hvilke lyder som kommer, og hvor kraftig lyden er. Dette har løsningen i studien tatt hensyn til, og laget logiske lyder. Dører lager knirkelyder når de åpnes, ikke en 17 mai fanfare.
	Blikk (Gaze) (Hvor ser andre brukere?)	Brukere kan observere hvor andre brukere retter blikket gjennom bevegelsen til avataren og hvilken retning den vender. Fordi en kan løsrive blikk fra avataren, med 360 graders kameraet, kan en få et feilt inntrykk av hvor den andre ser.
Hvor	Perspektiv (Hvor kan brukerne se?)	Brukere ser Second Life verden gjennom et "kamera" plassert bak avatarens rygg. En kan også velge å se nærmere på objekter, ved å flytte rundt på "kamera" slik man ønsker.
	Rekkevidde (Hva kan brukerne manipulere?)	Det er vanskelig å se hvem som kan manipulere hva. Dette er fordi all manipulering blir styrt av rettighetstilgang til objekter. Fordi en eier av et objekt har mulighet for å gi tilgang til å endre objektet til sine venner, er det ikke mulig å vite hvem som har rettigheter til objektene utover eier. For å manipulere et objekt må man for det første kunne observere objektet i workspace. For det andre kan man ikke være for langt unna. I prototypen er den av en slik størrelse at man er innenfor rekkevidde til alle objekter.

Tabell 10 – Elementer for workspace awareness i relasjon til fortid (Tabell fra Gutwin & Greenberg 2002 oppimot denne studiens empiri)

Kategori	Element	Svar for prototype i Second Life
Hvordan	Handlingshistorie (Hvordan skjedde denne handlingen?)	Det har ikke blitt funnet støtte for dette.
	Artefakthistorie (Hvordan fikk dette artefaktet denne tilstanden?)	Man kan se hvem som har laget objektet og hvem som eier objektet. En kan også se når en mottok det, om noen har gitt objektet til deg. En får kun opp siste byttet, ikke historikk over alle eierskiftene, og man kan ikke se navn på forrige eier. Hvordan et eksisterende objekt har blitt modifisert blir ikke logget. Man kan huske hva som har skjedd, hvis en var tilstede, eller noen har manuelt dokumentert hendelsen.
Når	Hendelseshistorie (Når skjedde denne hendelsen?)	Man kan se når et objekt ble laget og når en har kjøpt eller solgt et objekt, men ikke når et objekt sist ble manipulert. Sistnevnte må manuelt dokumenteres, hvis en ønsker å bevare slik historikk.
Hvem (fortid)	Tilstedeværelseshistorie (Hvem var her, og når?)	En kan vite hvem som har vært innlogget i workspace og tidspunkt for siste innlogging, hvis man er medlem av samme team, slik som denne prototypen forutsetter.
Hvor (fortid)	Lokasjonshistorie (Hvor har en person vært?)	Vi har ikke funnet støtte for dette, utover at man blir logget på der man avsluttet sist. Dette forutsetter at man ikke endrer standard innstillinger.
Hva (fortid)	Handlingshistorie (Hva har en person gjort?)	Man kan se at det har kommet til nye objekter i workspace, og hvem som har laget dem. Hvis objekter er fjernet eller flyttet, må man huske hvor de var tidligere, slik som i vår verden. Egen hukommelse er et viktig verktøy utover funksjonaliteten i et MUVE.
		Endringer siden sist må dokumenteres manuelt hvis dette er ønskelig, for det er ikke noen automatikk i dette i prototypen. Man kan med andre ord ikke se hvem som har endret et objekt, når, og hvordan det ble endret.

Resultatene fra denne evalueringsteknikken vil bli sett oppimot funnene fra Groupware Walkthrough (GWW) i slutten av dette kapittelet.

8.2 Groupware Walkthrough

For å identifisere samarbeidsoppgaver for et Scrum team har det blitt gjennomført en GWW oppgaveanalyse. Denne består, som nevnt i avsnitt 4.6.2.2, av to hovedkomponenter: en Group Task Model[41] for å modellere arbeidet som finner sted,

41 Nor: Gruppeoppgavemodell

og en Walkthrough Process[42] av systemet for å evaluere systemets grensesnitt (Gutwin & Pinelle 2002, s. 455).

På bakgrunn av samarbeidsoppgavene som ble identifisert og analysert i kapittel 6, og deretter oppsummert i kravtabellen i kapittel 7, har det blitt laget seks bruksscenarioer. I appendiks C er elementer fra Group Task Modell lagt ved. Denne består som nevnt av en aktivitetsbeskrivelse, beskrivelse av brukerne og den kunnskap det forventes at de besitter, et forventet utfall, og et sett med omstendigheter som kan påvirke gjennomføringen. I tillegg er det lagt ved analysediagram som identifiserer alle oppgaver og deloppgaver i hvert bruksscenario. Det har så blitt utført en walkthrough prosess i henhold til GWW, som hadde til hensikt å analysere hvor godt disse blir støttet av prototypen. Alle evaluererne ble spurt spørsmålene som skal besvares for hver arbeidsoppgave i walkthrough prosessen (se appendiks C), og totalinntrykket fra evaluererne er oppsummert i tabell 11.

8.2.1 Walkthrough prosess for scenario 1 - Daglig Scrum møte

Første scenario består i å holde et daglig Scrum møte. Som vist i analysediagrammet for scenarioet (se appendiks C), er det tre ulike aktiviteter som utføres for å avholde et slikt møte. For det første må det kalles inn til møtet. Second Life har ikke integrert støtte for elektronisk møteinnkalling. Dette er altså noe man må foreta på utsiden av Second Life. Hvis Scrum Mester, ved hjelp av samarbeidsmekanismen *overvåking*, ser at teammedlemmer er i Second Life, kan han eller hun, ved hjelp av samarbeidsmekanismen *eksplisitt kommunikasjon* spørre medlemmet om han eller hun er tilgjengelig, uten å benytte møteinnkalling. Her kan man benytte VoIP eller IM. Videre må Scrum Mester tilrettelegge for møtet og tilgjengeliggjøre den informasjon som trengs for å gjennomføre det. Til slutt holdes møtet med de innkalte deltagerne.

8.2.1.1 Analyse av resultater

Det ble gjennomført en walkthrough prosess av et daglig Scrum møte med fem evaluerere, som vist i figur 28. En hadde rollen som Scrum Mester, mens de andre var teammedlemmer. Utgangspunktet for testene var at alle var pålogget Second Life.

42 Eng: Gjennomgangsprosess

Figur 28 – Scenario walkthrough av daglig Scrum møte

Aktivitet 1: Planlegge daglig Scrum møte

Testen startet med at Scrum Mester (#1) ønsket å planlegge det daglige Scrum møtet. For å gjøre dette måtte han først undersøke om team medlem (#2) var tilgjengelig. Her hadde han to alternative deloppgaver, hvor en ene var å observere (*overvåke*) om #2 var tilgjengelig. Man kan sette status på avataren til opptatt, hvis man ikke ønsker å bli forstyrret. Det virket som om #2 hadde ledig tid, så #1 spurte da #2 via VoIP om han hadde tid til å delta på møtet. #1 svarte tilbake at han hadde tid til å delta på møtet. Den andre måten å undersøke om medlemmer var tilgjengelig, var å spørre de direkte ved hjelp av *eksplisitt kommunikasjon.* #2 svarer da tilbake om han eller hun er tilgjengelig for møtet. Dette gjentok #1 for de andre team medlemmene.

Delkonklusjon

Det var litt frem og tilbake for å finne et tidspunkt som passet for alle evaluererne. Denne oppgaven ble gjennomført uten tekniske problemer. Således var det tid for å forbedrede møte. Her var det tre oppgaver for Scrum Mester, og en for team medlemmene.

Aktivitet 2: Forbedrede daglig Scrum møte
Først måtte Scrum Mester orientere seg om situasjonen. Her valgte evaluereren å sette opp en møteagenda i henhold til Scrum metodologien. Neste oppgave var å dele dokumenter, hvor det ble lastet opp en Microsoft Powerpoint presentasjon som skulle holdes under møtet. Til slutt ble Sprint Backlogen klargjort i møteromslokalet, slik at denne kunne bli diskutert under møtet.

Evaluererne som hadde rollen som teammedlemmer måtte gjøre seg opp en status om hva som ble gjort i går, hva som skal gjøres i dag, og om det har vært noe som stopper fremgang. Her forsøkte evaluererne å sette seg inn i rollen som programmerere.

Delkonklusjon
Det var ingen problemer med å lage et fiktivt scenario for hva møtet skulle handle om. Her ble det benyttet inspirasjon fra IKTBergen. Det viste seg også at Microsoft Powerpoint har funksjonalitet for å konvertere en presentasjon til grafikkfiler, som lot seg lett laste inn i prototypen. Sprint Backlogen ble tilpasset dagens situasjon, tilsvarende det man ville gjort ved hjelp av gule lapper (se figur 16). Dette gikk lettere enn forventet.

Aktivitet 3: Holde daglig Scrum møte
Den første aktiviteten er at Scrum Mester Morten ber om status fra team medlemmene. Her benytter de samarbeidsmekanismen *eksplisitt kommunikasjon* via IM. I henhold til Scrum metodologien ber han teammedlemmene om status. Nedenfor vises et utdrag fra IM loggen fra evalueringen, hvor Alita gir status på eget arbeid:

[3:15] Morten Kozlowski: Alita, hva er din status?
[3:15] Alita Shinn*:* I går laget jeg den nye modulen for å kunne printe PDF filer.
[3:15] Morten Kozlowski: Bra :-)
[3:16] Tias Krautrauch: Den har jeg savnet, awesome.
[3:16] Christer Przhevalsky: Den ventet jeg også på. Supert :)
[3:16] Christobal Humburg: Pleier å printe doc, jeg
[3:16] Christer Przhevalsky: Det er lett for at det blir feil når man skriver ut word dokumenter, så derfor liker jeg å skrive de ut som PDF :)
[3:16] Morten Kozlowski: Hva planlegger du i dag?
[3:16] Alita Shinn: I dag skal jeg teste modulen jeg laget i går og så har jeg et møte etter lunsj.
[3:16] Morten Kozlowski: Bra. Noe som hindrer deg i å oppnå framdrift?
[3:17] Alita Shinn: Arbeidet mitt går som planlagt.
[3:17] Morten Kozlowski: Bra :-)
[3:17] Alita Shinn: :-)
[3:17] Morten Kozlowski: Godt jobbet :-)

Denne oppgavesekvensen gjentok seg for de andre team medlemmene.

I IM loggen blir det vist hvordan Scrum Mester Morten tar opp en problemstilling, og orienterer om status på denne, ved hjelp av Product Backlog. Her både forteller han om en spesifikk region og peker til denne. Videre klargjør han problemstillingen for team medlemmene. Det blir så en liten diskusjon omkring dette problemet, før Scrum Mester forteller hva som skal skje videre. Han avslutter så møtet.

[3:23] Morten Kozlowski: Jeg mener vi får problemer på grunn av manglende integrasjon mot SAP. La oss se på product backloggen :-)
(Morten peker (eksplisitt kommunikasjon) til den delen av Product Backlogen hvor SAP integrasjon er utestående)
[3:23] Morten Kozlowski: Uten den får vi ikke penger fra finansfolkene. Noe som bør få mesteparten av fokus denne uken
[3:23] Christer Przhevalsky: Dessuten tar SAP alle konsulentene vi ønsker å ansette i Ukraina
[3:24] Morten Kozlowski: Hvis vi bruker første del av uken på SAP integrasjon vil nok mye bli bedre
[3:25] Tias Krautrauch: Jeg tror ikke en halv uke er nok til å ferdigstille SAP-integrasjon. Det er et veldig omfattende system
[3:25] Christer Przhevalsky: Vi få satse på at vi får samlet ressursene i løpet av uken. Det er utrolig hva vi får til.
[3:25] Christobal Humburg: De burde vært litt mer SOA
[3:25] Tias Krautrauch: Ja, SAP er for lite SOA, helt klart.
[3:26] Christobal Humburg: Sømløs integrasjon hadde vært tingen
[3:26] Morten Kozlowski: Jeg mener det viktigste nå er gjennomgått og jeg tar ressursdialogen med management etter møtet. Takker for oppmøtet :-)

Konklusjon
Møtet ble gjennomført uten problemer, selv om to av dem ikke hadde erfaring med Second Life fra tidligere. Eneste bistand som ble gitt dem, var at de fikk en kopi av guiden til Second Life, som er med i appendiks B, og lenken til prototypen.

Alle var enig i at dette var noe de hadde benytte i arbeidssammenheng hvis Scrum Mester ønsket dette, og at det ikke var arbeidskrevende å benytte prototypen. Christobal som har erfaring fra daglige Scrum møter fra tidligere, men ikke MUVE, gjør seg opp følgende betrakting etter at møtet var gjennomført: “Kan se for meg at det er større awareness over second life enn det er over f.eks skype som jeg har brukt til scrum-møter tidligere” Christobal Humburg.

De andre evaluererne mente også at man fikk en god oversikt over hva som foregikk på møtet, selv om det hadde vært en fordel om man startet en setning med å si hvem man snakket til. Dette fordi det lett ble uoversiktelig når mange snakket samtidig. Dette er det samme problemet som IKTBergen hadde når man benyttet IM (se avsnitt

6.2). Det ble bemerket at møtet sikkert hadde vært med oversiktelig hvis alle hadde benyttet VoIP.

8.2.2 Walkthrough prosess for scenario 2 - Holde kurs

Som vist i analysediagrammet for scenarioet i appendiks C, er det tre ulike aktiviteter som utføres for å holde et kurs. For det første må det kalles inn til kurset. Second Life har ikke integrert støtte for elektronisk møteinnkalling, slik nevnt i scenario 1. Dette er altså noe man må foreta på utsiden av Second Life. Videre må kursholder tilgjengeliggjøre presentasjonen. Til slutt gjennomføres kurset.

8.2.2.1 Analyse av resultater

Det ble gjennomført en walkthrough prosess av det å holde et kurs, med fem evaluerere, som vist i figur 29. Morten hadde rollen som kursholder, mens de andre var kursdeltagere. Utgangspunktet for testene var at alle var pålogget Second Life.

Figur 29 – Scenario walkthrough av å holde et kurs

Aktivitet 1: Planlegge opplæring

Kursholder undersøkte tilgjengelighet til kursdeltagerne ved å spørre direkte. Kursdeltagerne bekreftet tilgjengelighet tilbake til kursholder. Testen startet med at kursholder (#1) ønsket å planlegge opplæringen. For å gjøre dette måtte han først undersøke om kursdeltager (#2) var tilgjengelig. For å gjøre dette spurte han direkte ved hjelp av *eksplisitt kommunikasjon.* #2 svarer da tilbake om han eller hun er tilgjengelig for møtet. Dette gjentok #1 for de andre team medlemmene.

Delkonklusjon
Det gikk fint å finne et tidspunkt som passet for alle kursdeltagerne. Denne oppgaven ble gjennomført uten tekniske problemer. Således var det tid for å forbedrede møte. Her skulle kursholder laste opp presentasjon.

Aktivitet 2: Forberede opplæring
Kursholder lastet opp en presentasjonen han skulle benytte under kurset.

Delkonklusjon
Microsoft Powerpoint har funksjonalitet for å konvertere en presentasjon til grafikkfiler, som lot seg lett laste inn i prototypen. Det oppsto ingen problemer med å gjennomføre denne aktiviteten. Fra start av oppgaven, til presentasjonen var lastet opp, og klargjort for visning, gikk det cirka 1 minutt.

Aktivitet 3: Gjennomføre opplæring
Den første aktiviteten var at kursholder Morten begynte å presentere kursmateriale. Her benyttet han samarbeidsmekanismen *eksplisitt kommunikasjon* via IM. Nedenfor vises et utdrag fra IM loggen fra evalueringen, hvor kursdeltager Christer stiller et spørsmål:

[3:30] Morten Kozlowski: Velkommen. Tenkte jeg skulle holde en liten presentasjon om å holde kurs i Second Life og om mulighetene dette gir :-)
[3:31] Morten Kozlowski: Som sliden viser er Second Life godt egnet til å gi presentasjoner. Det kan også brukes som en felles opplæringsplattform for internopplæring, kurs på ulike produkter, med mer.
[3:33] Tias Krautrauch: Praktisk. Det virker som et veldig fleksibelt system.
[3:33] Christer Przhevalsky: Hey! *(eksplisitt kommunikasjon - gjør tegn til at han har et spørsmål ved å ta opp armen)*
[3:33] Morten Kozlowski: Hei, Christer. Hva tenker du på?
[3:33] Christer Przhevalsky: Jeg har et spørsmål
[3:33] Morten Kozlowski: ok?
[3:33] Christer Przhevalsky: Har du tid nå?
[3:33] Morten Kozlowski: Ja
[3:33] Christer Przhevalsky: Hvordan kan man få laste opp en presentasjon til den skjermen der? *(eksplisitt kommunikasjon - peker til den skjermen hvor presentasjonen holdes)*
[3:34] Morten Kozlowski: Det er enkelt, du tar Powerpoint filen din,velger å lagre slidene som bilder, laster dem opp med "bulk upload", og drar dem over til prosjektoren. Dermed er det bare å starte presentasjonen.Totalt tar hele prosessen ca 1 min.
[3:34] Christobal Humburg: Wow! Sømløs integrasjon!
[3:35] Morten Kozlowski: Sant
[3:36] Christer Przhevalsky: Takk for svaret ☺

Konklusjon
Kurset ble gjennomført uten problemer, selv om ikke alle evaluererne hadde erfaring med Second Life. Kursholder fikk tilbakemelding underveis i kurset, og følte at deltagerne fulgte med. Det at deltagerne ikke trengte å følge med var en bekymring som kom fram fra en av evaluererne etter kurset var ferdig: "Jeg skeptisk til om man får en god awareness, men man kan jo ikke vite om personen bak en avatar faktisk er til stede eller ikke" (Tias Krautrauch). I Second Life er det slik at avatarene gir en klar indikasjon på at man ikke er tilstede, hvis man ikke har utført noen handlinger i grensesnittet de siste minuttene. Hodet på avataren bøyer seg ned, og meldingen "away" kommer til synet.

Videre fungerte det bra å holde kurset med evaluerere som befant seg på ulike steder i Norge. En av evaluererne meldte tilbake at kurset gikk greit, men at det var vanskelig å opprettholde konsentrasjonen. De andre evaluererne kommenterte ikke dette som et problem.

8.2.3 Walkthrough prosess for scenario 3 - Diskutere kildekode
Slik det kom frem i intervjuundersøkelsen (se punkt 6.3.3) vil det oppstå situasjoner hvor man trenger hjelp fra de andre i teamet. Et konkret eksempel på dette er hjelp til å forstå hvordan man skal løse et programmeringsproblem. Som analysediagrammet for scenarioet (se appendiks C) viser, er det tre ulike aktiviteter som utføres i en slik situasjon. Dette er å avtale, forberede og ha en diskusjon. Det teammedlemmet som ber om hjelp har fått benevnelsen #1 og den som blir spurt har fått #2.

8.2.3.1 Analyse av resultater
I dette scenarioet har funksjonaliteten blitt evaluert med to mannlige evaluererer, hvor den ene var teammedlem #1, og den andre var teammedlem #2. Det ble gjennomført to gjennomganger. Dette fordi det er to ulike fremgangsmåter for å dele kildekode. Utgangspunktet for testene var at teammedlem #1 og #2 var pålogget Second Life.

Gjennomgang 1

Aktivitet 1: Avtale diskusjon
Testen startet ved at #1 observerte om #2 var tilgjengelig, ved hjelp av samarbeidsmekanismen *overvåking*. Det virket som om #2 hadde litt tid tilgjengelig, så #1 spurte da #2 via VoIP om han hadde tid til å se raskt over noe kildekode. #1 svarte tilbake at han hadde noen minutter til rådighet.

Delkonklusjon
Som nevnt tidligere var det nyttig at kunne man sette status på avataren til opptatt, hvis man ikke ønsker å bli forstyrret.

Aktivitet 2: Forberede diskusjon
For å kunne se på kildekode sammen, var det nødvendig for #1 å tilgjengeliggjøre denne på en skjerm i rommet for samarbeid. Kildekoden #1 hadde spørsmål om, tok han bilde av, og lastet opp til prototypen. Bildet kom så inn i inventarmappen[43] (se figur 33), og derifra ble det dratt til skjermen han ønsket å vise bildet. #1 trykket så på skjermen, og valgte i menyen at bildet skulle vises.

Delkonklusjon
Dette gikk uten problemer ved å benytte standard Second Life funksjonalitet.

Aktivitet 3: Diskuter kildekode
#1 spurte så #2 om dette var en fornuftig måte å løse problemet. #2 kom så med noen forbedringsforslag. Deretter startet en diskusjon for å komme frem til den beste løsningen, som varte noen få minutter. Diskusjonen foregikk på VoIP, og #1 fikk i løpet av diskusjonen svar på hans problem.

Konklusjon
#2 syns det var uheldig at han ikke kunne bevege seg rundt i koden, men måtte forholde seg til et statisk bilde. Utover dette fikk man gjennomført oppgaven på en god måte.

Gjennomgang 2

Aktivitet 1: Avtale diskusjon
Testen startet ved at #1 spurte via IM om teammedlem #2 var tilgjengelig. Det var han, og #2 svarte dermed positivt tilbake. #1 spurte så #2 om han kunne få hjelp med å løse et problem, og #2 svarte at det kunne han gjerne.

43 Eng.: Inventory folder

Aktivitet 2: Forberede diskusjon
#1 valgte å tilgjengeliggjøre kildekoden ved å streame denne inn i Second Life. Den tekniske løsningen for å gjøre dette er beskrevet i avsnitt 7.4.2. Her hadde man de samme begrensningene som når man skal parprogrammere (se avsnitt 8.2.4).

Konklusjon
Her har man ikke en god løsning i dag, og evaluererne hadde heller foretrukket å vise dette som et statisk bilde i bedre kvalitet.

Aktivitet 3: Diskuter kildekode
#1 forklarte sitt problem, og viste hvor i kildekoden han satt fast. Diskusjonen foregikk via IM, og varte noen minutter. #1 fikk i løpet av diskusjonen svar på sitt problem.

Konklusjon
#2 syns teksten var veldig liten, som resulterte i at han hadde litt problemer med å lese kildekoden, men etter at #1 hadde fokusert litt nærmere på teksten, klarte #2 å lese den. Når #2 hadde behov for å se andre deler av kildekoden, kunne #1 vise frem dette. Dette var noe tungvint, men fungerte tilfredsstillende, selv om det var slitsomt å se på skjermbildet over lengre tid.

8.2.4 Walkthrough prosess for scenario 4 - Parprogrammering
Som vist i avsnitt 2.6.2, er parprogrammering en av teknikkene i XP (se tabell 3). Analysediagrammet for scenarioet (appendiks C) viser at det er fem ulike aktiviteter som utføres i en slik situasjon. Dette er å avtale, sette opp den tekniske løsningen, koordinere arbeidsoppgaver, arbeidsdeling, og til slutt programmere og observere. Det blir bestemt når evaluererne koordinerer arbeidsoppgaver og arbeidsdeling hvem som skal starte som observatør[44] og hvem som skal starte med å skrive kode[45], samt hvilket Backlog Item det skal arbeides med. I løpet av arbeidsøkten byttes det på disse rollene, og man kan arbeide med flere Backlog Items. Det teammedlemmet som initierer parprogrammering har fått benevnelsen #1, og den som blir spurt har fått #2.

44 Eng.: Observer
45 Eng.: Driver

8.2.4.1 Analyse av resultater

I dette scenarioet har to mannlige evaluerere testet ut funksjonaliteten som teammedlem #1 og #2. På forhånd var det satt opp en liste med Backlog Items (se figur 21).

Aktivitet 1: Avtale parprogrammering

Testen startet ved at #1 spurte #2, ved hjelp av IM, om han var tilgjengelig for parprogrammering etter lunsj. #2 kontrollerte kalenderen sin for å se om han hadde noen møter etter lunsj. Dette hadde han ikke, og bekreftet til #1 at han var ledig.

Aktivitet 2: Sette opp parprogrammering

#1 setter så opp parprogrammeringsdatamaskinen ved å koble til to skjermer, to mus og to tastatur, slik som observert hos IKTBergen. Videre tilgjengeliggjorde han skjermbilde fra parprogrammeringsmaskinen til Second Life. #2 kobler seg så til parprogrammeringsmaskinen via Remote Desktop Protocol (RDP)[46].

Delkonklusjon

Dette viste seg å være vanskelig å få til, på grunn av den tekniske løsningen. Det å streame skjermbilde inn i Second Life benytter tredjeparts programvare, hvor #1 benyttet prøve versjoner av programvare. Fordi det var vanskelig å sette opp, med forstyrrende reklame fra programvareleverandør, ga det dårlig *satisfaction*. #1 måtte også ta kontakt med systemansvarlig for å få mappe porter gjennom brannmur og ruter, slik at streamen kom frem til Second Life. Når streamen først var kommet frem til Second Life, gikk det fint å koble den til en skjerm i samarbeidsrommet. Det viste seg at oppløsningen måtte settes til 4*3 format, for ellers passet ikke bildet til skjermen.

Aktivitet 3: Koordinere arbeidsoppgaver

Etter at parprogrammeringsmaskinen var satt opp, og begge parter koblet til denne, var det tid for å koordinere arbeidsoppgavene. #1 spurte #2 via IM hvilket Backlog Item det skulle arbeides med, og pekte (*eksplisitt kommunikasjon*) til riktig del av Backlog Listen.

46 RDP protokollen gjør det mulig å fjernstyre en datamaskin over Internett, slik at det virker som om man arbeider direkte på den

Aktivitet 4: Koordinere arbeidsdeling
Det ble så avtalt hvem som skulle observere og hvem som skulle kode, ved at #1 spurte #2 ved hjelp av IM.

Delkonklusjon
Det oppstå ingen problemer underveis i denne koordineringen. Her var det en fordel at man ikke hadde andre forstyrrende aktiviteter i workspace samtidig som denne koordineringen skulle foretas.

Aktivitet 5: Programmere og observere
I testen kodet #1, mens #2 observerte ved å benytte samarbeidsmekanismen *overvåking.*

Konklusjon
Det var vanskelig å holde oversikt på grunn av dårlig kvalitet og lav oppløsning på streamen inn i Second Life, og en ble ganske trøtt i øynene etter noen minutter. På grunn av sen oppdatering av endringene som ble utført på parprogrammeringsmaskinen, var det vanskelig for observatøren å gi gode tilbakemeldinger til #1 som kodet. Fra #1 endret noe, tok det ca 5 sekunder før #2 kunne se hva som hadde endret seg.

På grunn av problemene med å parprogrammere ville det gjerne vært bedre om man gjorde dette direkte i programvareutviklingsløsningen, i stedet for via prototypen. Det finnes i dag løsninger for å streame i HD kvalitet, men de koster flere tusen dollar, og var utenfor denne studiens budsjett. Tanken var også å lage en lavbudsjettsløsning som ville fungere på liten båndbredde. En mulighet som ikke finnes pr i dag er å parprogrammere direkte i protypen uten å streame dataskjermen. Her ser vi for oss to alternative løsninger. Second Life kunnet hatt en innebygget programmeringsløsning som hadde vært tilfredsstillende for distribuert programutvikling ved hjelp av parprogrammering. Den andre løsningen ville vært at Second Life hadde hatt innebygget støtte for å streame fra lokal datamaskin, og inn i den virtuelle verdenen , med høy kvalitet.

8.2.5 Walkthrough prosess for scenario 5 – Dele biblioteksressurs

Som analysediagrammet for scenarioet (appendiks C) viser, er det tre ulike aktiviteter som utføres. Dette er å dele, finne, og benytte/holde à jour biblioteksressurser. Dette kan foregå asynkront, eller synkront. De to team medlemmene som legger ut og benytter ressurser har fått betegnelsen #1 og #2.

8.2.5.1 Analyse av resultater

I dette scenarioet har to mannlige evaluerere testet ut funksjonaliteten som teammedlem #1 og #2. De befant seg på ulike steder.

Aktivitet 1: Dele ressurs vis biblioteket

Testen startet med at #1 la ut et dokument i biblioteket. Dette gjorde #1 ved å opprette et objekt som vist i appendiks B, la inn dokumentet han ønsket å dele med teammedlem #2, og satt rettigheter på objekter.

Delkonklusjon

Det kreves litt erfaring med prototypen for å få dette til, noe som viste seg i løpet av denne walkthrough prosessen.

Aktivitet 2: Finne biblioteksressurser

Videre lot han #2 få tilgang til objektet, og sendte IM til #2 om at objektet var lagt ut. I dette tilfellet var ikke #2 pålogget prototypen, og fikk dermed automatisk e-posten tilsendt som vist under:

> ”[4:58] #1: Hei #2, jeg har lagt ut en ny ressurs i biblioteket som jeg ønsker at du ser på når du har tid. Fint om du gir meg en tilbakemelding på hva du mener om den. Det er en nybegynner guide for Second Life. Vi snakkes, hilsen #1.”

Hvis #2 hadde vært pålogget ville denne meldingen blitt vist i IM vinduet i protoypen, og ikke sendt pr e-post. Når #2 logget seg på fikk han den samme meldingen i sitt IM vindu, med info om at den var ”Saved Wed Nov 25 13:58:58 2009”. Her benyttes lokal tidssone, og ikke Second Life tid.

Når #2 skulle benytte prototypen var det nødvendig å teleportere til de rette koordinatene. For å finne dette åpnet han på lenken han hadde fått utlevert (se avsnitt 7.2) for å komme seg til de virtuelle kontorlokalene.

For å vite hvite hvilke ressurser som #1 hadde gjort tilgjengelig, testet #2 først om det gikk an å finne dette ut ved å se på egenskapene til informasjonsobjektene i biblioteket. Dette fungerte bra fordi man klart kunne se at #1 var eier av objektene. Andre alternativ var å spørre #1 direkte:

> ”[5:32] #2: Hei #1. Hvilke ressurser har du lagt ut i biblioteket?
> [5:33] #1: Hei #2. jeg har lagt ut nybegynnerguide nå, men tidligere har jeg også lagt ut to manualer for programmering i Second Life.”

Delkonklusjon
Det var nyttig at prototypen sender ut en e-post hvis man ikke er tilstede når man får en forespørsel via samarbeidsmekanismen *direkte kommunikasjon*. Informasjonen blir i stedet gitt til teammedlemmet i form av asynkron kommunikasjon, uten at brukeren trenger å utføre ekstra handlinger for dette. Denne funksjonen er det også mulig å slå av hvis man ikke ønsker at det skal være slik som dette.

Aktivitet 3: Benytte, fjerne og forbedre biblioteksressurser
Teammedlem #2 ønsket så å benytte en av ressursene. Dette forsøkte han å gjøre ved å høyreklikke på nybegynnerguiden, for så å kopiere objektet til sin egen inventar mappe, men dette valget var det ikke mulig å trykke på. Dette hadde årsak i at rettighetene for objektet ikke var slik at #2 kunne benytte det. Rettighetene ble så ordnet, slik at testen kunne fortsette, slik anbefalt av (Gutwin & Pinelle 2002).

For at #2 skulle kunne lese boken, var det egentlig tenkt at man skulle kunne ta en kopi av objektet. Dette fungerte ikke på grunn av manglende rettigheter til objektet. Det viste seg at en løsning var å kjøpe objektet for 0 Linden dollar, slik at man fikk dette i egen inventar mappe. Etter at man fikk objektet i inventar mappen, måtte man dra objektet fra inventarmappen til gulvet i prototypen. Dette vil med andre ord si at andre i rommet kan se hvilken bok #2 har tenkt å lese, ved at det går en prikkete linje fra avatar til objekt (*indirekte kommunikasjon*). Deretter høyreklikket han på objektet, og åpne dette. Når objektet var åpnet, fikk han en liste over innholdet i boken. Således måtte alle kapitlene i boken lastes over igjen til inventar mappen, litt som å pakke ut en komprimert mappe med filer. De fem først kapitlene åpnet seg direkte i skjermbildet, mens alle ble lagt i en folder i inventarmappen.

Etter at #2 hadde lest igjennom det han ønsket av boken, ga han tilbakemelding til #1 om kvaliteten på den:

> ”[5:56] #2: Hei #1. Jeg har lest igjennom noe av begynner guiden du lastet opp til biblioteket. Var en litt tidkrevende prosess å få alt fram, men når det var gjort, gikk det fint. Jeg syns det var altfor omfattende informasjon i en bok. Kan du forbedre den ressursen litt, slik at de andre i teamet kan få en bedre oversikt.
> [5:59] #1: ok #2, skal se om den kan lages litt mindre omfattende.”

Fordi kvaliteten ikke var tilfredsstillende ønsket #1 å forbedre denne. Han åpnet så boken, og slettet noen av kapitlene som ikke var relevante. På grunn av begrensende rettigheter til akkurat dette objektet, måtte #1 flytte kapitlene over i en ny bok, før han kunne editere teksten i hvert kapittel. Deretter kunne han legge den modifiserte boken tilbake i biblioteket.

Konklusjon
Når objekter blir laget, er det viktig at man setter hvem som skal ha tilgang til objektet. Dette er i henhold til samarbeidsmekanismen *beskyttelse.* Hvis man setter dette feil, blir det som i denne gjennomgangen problemer når andre skal benytte objektet man ønsker å dele. Når man først har fått litt erfaring med å benytte prototypen lærer man seg hvordan dette skal gjøres, slik at slike situasjoner ikke oppstår. Prosessen med å lese en bok i prototypen er ikke den beste, fordi alle kapitlene åpner seg opp som egne dokumenter i skjermvinduet. Likevel må evalueringen sies å ha lykkes, fordi teammedlem #1 klarte å dele den ressursen han ønsket med teammedlem #2, og #2 klarte å nyttegjøre seg av denne.

8.2.6 Walkthrough prosess for scenario 6 - Spille spill

Etter en lang arbeidsøkt er det godt med en pause. IKTBergen spilte Fussball (se avsnitt 6.2), mens man i prototypen kan spille fire på rad (se figur 24). Spillet har blitt plassert i rommet for sosiale aktiviteter. Dette er et spill for to personer, hvor teammedlem #1 og teammedlem #2 deltar. Som analysediagrammet for scenarioet (appendiks C) viser, er det to ulike aktiviteter som utføres. Dette er å avtale, og deretter spille fire på rad.

8.2.6.1 Analyse av resultater

I dette scenarioet har fire mannlige evaluerere testet ut funksjonaliteten som teammedlem #1 (rød farge) og teammedlem #2 (gul farge), ved å spille spillet tre ganger etter hverandre. Under blir det forklart hvordan det var å spille fire på rad.

Aktivitet 1: Avtale å spille et spill

Teammedlem #1 undersøkte om #2 var tilgjengelig ved å spørre #2 direkte via IM:

”[6:30] #1: Hei #2, har du tid til å spille 4 på rad med meg nå, trenger en liten pause:-)”.

Teammedlem #2 svarte så tilbake via IM:

”[6:31] #2: Ja, det hadde vært kjekt. Trenger en god pause jeg også :)”

Delkonklusjon

Det var enkelt å få avtalt å spille et spill.

Aktivitet 1: Avtale å spille et spill

Teammedlem #1 trykket på ”Play Red” objektet. Dette inneholder programkode som gjør at man kan spille spillet. Etter at #2 trykket på ”Play Yellow” objektet, var det klart for å spille.

Et problem som oppsto etter at man hadde trykket på objektet, var at man fikk problemer med kameravinkelen. I stedet for å se spillet, endte man opp med å se på baksiden av veggen bak spillet. Dette skjedde for alle evaluererne, men det var enkelt å få flytte kameravinkelen tilbake, slik at spillet kunne fortsette, i henhold til groupware walkthrough prosessen (se avsnitt 4.6.2.2).

Spillet indikerte hvem som skulle starte først, og det var #1, fordi programkoden har definert at rød alltid starter. Ved å trykke på spillbrettet avgjør man hvor den neste brikke faller ned. Den første med fire på rad vinner, og i et av tilfellene var dette #1:

> ”[6:33] #2: Der tok du jammen innersvingen på meg. Det var et smart trekk du gjorde der.
> [6:33] #1: :-)”

Konklusjon
Det å spille spill fungerte veldig greit i de tre gjennomgangene, selv om man hadde et problem med kameravinkelen. Dette kunne ha blitt ordnet i en designendring av prototypen. Det var også nyttig at avatarene satt seg ned oppå objektet man manipulerte, slik at man fikk en del uintendert informasjon (*indirekte kommunikasjon*) som den andre spilleren kunne forstå. Man fikk med andre ord klar indikasjon på når den andre spilleren var klar til å starte.

8.2.7 Oppsummering
Gutwin og Greenberg (2000) forklarer hvordan man kan benytte en tabell for å sette samarbeidsmekanismene oppimot måleegenskapene Effectivness, Efficiency og Satisfaction. På bakgrunn av dette har det blitt utviklet en tabell som oppsummerer resultatene fra studiens Groupware Walkthrough (s. 100).

Tabell 11 – I hvilken grad samarbeidsmekanismene er støttet i Prototypen

Samarbeids-mekanisme	Effectiveness	Efficiency	Satisfaction
Eksplisitt kommunikasjon	Ja	Ja	Ja
Indirekte kommunikasjon	Ja, men ubevisst kroppsspråk er ikke støttet.	Ja, men en må kontinuerlig følge med fordi historikk er dårlig støttet.	Nei, fordi det ikke lagres noe historikk over handlingsforløp og fordi avatarene ikke gir fra seg noe indirekte kroppsspråk.
Koordinasjon av handling	Ja	Ja	Ja, når en benytter eksplisitt kommunikasjon samtidig som en arbeider.
Planlegging	Ja	Ja	Ja
Overvåking	Ja, men endring av Script og andre objekter i inventarmappen er noe en ikke kan overvåke.	Ja	Ja, når en bygger på felles objekter.
Hjelp	Ja	Ja	Ja
Beskyttelse	Ja	Ja	Ja, men noen utfordringer med å se hvilke rettigheter andre har til objekter.

8.3 Oppsummering av evaluering

I dette kapittelet har awareness rammeverket, i kombinasjon med Heuristisk evaluering for groupware, og Groupware Walkthrough, blitt benyttet for å evaluere prototypen designet i denne studien. I dette avsnittet det bli trukket fram hovedfunnene i evalueringen(se tabell 9, 10 og 11), oppimot kravene som kom fram i den kvalitative undersøkelsen.

Det viste seg at det var uproblematisk å holde oversikt over evaluererne så snart de var logget på Second Life. Dette gjenspeiles også i tabell 9, hvor det blir bemerket hvordan dette fungerer i praksis. Prototypen var designet slik at det var lett å holde oversikt over alle avatarene som deltok i walkthrough prosessene. Dette er illustrert i figur 28. Når det gjelder å holde oversikt over hva som har skjedd tidligere i workspacet, var ikke dette like enkelt, som vist i tabell 10. Det var nyttig at man kunne se at det var kommet til nye objekter i workspace, hvem som hadde laget dem, og når. Det er viktig å være klar over at dersom objekter er fjernet eller flyttet, må man huske hvor de var tidligere, slik som i vår verden. Egen hukommelse er dermed et viktig verktøy utover funksjonaliteten i prototypen.

Scrum møte fungerte i henhold til designkrav, uten at det oppsto noen hindre i bruken, men det var litt vanskelig å vite hvem som snakket til hvem. Dette er trolig fordi, slik tabell 11 viser, at ubevisst kroppspråk ikke er støttet. Hvis dette hadde vært støttet, hadde det trolig vært slik at man kunne få oppmerksomheten til den man snakket med på en bedre måte enn tilfellet var i walkthrough prosessen. Prototypen var enkel å benytte for evaluererne, selv om to av dem ikke hadde noen erfaring med Second Life fra tidligere. Det var heller ikke noe problem at evaluererne befant seg på ulike steder under walkthrough prosessen.

Koordinasjonen mellom scenarioene fungerte på en god måte, og evaluererne fant fort tonen, selv om ikke alle kjente hverandre på forhånd. Etter at evalueringen var ferdig, hadde man en lengre uformell samtale på øyen som har blitt designet som rekreasjonsområde, slik vist i figur 27. Sosialiseringen fungerte med andre ord på en god måte, som tilfredstilte de krav som kom fram i den kvalitative undersøkelsen (se tabell 8).

9 Konklusjon

I denne studien har det blitt sett på hvordan et MUVE kan støtte distribuert programvareutvikling. Dette har blitt gjort ved å gjennomgå relevant teori og tidligere forskning. Dette har dannet grunnlaget for en kvalitativ undersøkelse. Intensjonen var å få innblikk i eksisterende arbeidspraksis til Scrum team som arbeider distribuert. Resultatene fra undersøkelsen ble benyttet som grunnlag for å designe en prototype for å støtte denne praksisen ved hjelp av et MUVE. Funksjonaliteten til prototypen ble deretter evaluert for å stadfeste hvorvidt den tilfredstilte behovene fra den kvalitative undersøkelsen.

Studiens forskningsdesign innehar to forskningsspørsmål. I det første ble det spurt om hvordan distribuerte programutviklingsteam samarbeider og hvilke verktøy de benytter i sin arbeidspraksis. Dette har blitt besvart ved å foreta en kvalitativ undersøkelse som har bestått av intervju med programutviklere og observasjon hos et IKT-firma. For det første ble det gjennomgått relevant teori fra CSCW-fagfeltet, herunder Common Information Spaces, articulation work, samarbeidsmekanismer og awareness. Til slutt ble det sett på relevant forskning innen programutvikling, hvor det har blitt forsket på MUVE og distribuert Scrum. Utover dette ble det undersøkt hvordan Second Life har blitt benyttet i databasert læring. Denne studien befinner seg i grenselandet mellom disse forskningsgrenene, og har basert seg på resultatene fra dem.

På bakgrunn av dette ble det gjennomført en kvalitativ undersøkelse bestående av intervju og observasjon. Resultatene fra denne undersøkelse er et empirisk materiale som har blitt analysert på bakgrunn av teori fra CSCW-feltet. Hovedfunnene fra den kvalitative undersøkelsen er en beskrivelse av hvordan Scrum team samarbeider, hvilke verktøy de behøver i sin arbeidspraksis, og hvordan de kan benytte disse for å samarbeide distribuert.

Det andre forskningsspørsmålet hadde som mål å få klarhet i hvordan en slik praksis kan støttes av et MUVE. Det empiriske materialet har vist at utfordringene med dagens praksis er at det er vanskelig å samarbeide distribuert. Det finnes et stort utvalg av punkt-til-punkt løsninger, men disse er ikke tilfredsstillende når man skal

samarbeide som et team. Bjerrum og Bødker (2003) forklarer at årsaken til dette er at man er avhengig av god støtte for alle samarbeidsmekanismene og en mulighet for å kunne holde seg kontinuerlig oppdatert med hva de andre teammedlemmene arbeider med. Dette inkluderer artefakter i workspace og samarbeidsmekanismen implisitt kommunikasjon. Fordi man ikke er samlet i et felles kontorlandskap har man ikke mulighet for å se og høre hva andre holder på med, og på denne måten ha mulighet til å bli en bidragsyter i prosjekter og andre arbeidsområder utover ens eget. Det ble observert hos IKTBergen at den eneste måten for å vite hva det distribuerte teamet i Ukraina arbeidet med, var å spørre dem direkte. Dette ledet til flere problemer som er diskutert i kapittel 6: dobbeltarbeid, manglende felles forståelse for endelig løsning, manglende felles referansebibliotek, lav team følelse ved at en ikke kan ha felles sosiale aktiviteter, kreve ekstraarbeid for å oppdatere deltagere på programmeringsprosjekt, og vanskeliggjøre kulturforståelse.

Det empiriske materialet fra den kvalitative undersøkelsen behøver ikke bli benyttet for å designe en løsning i et MUVE, men kan med fordel også bli benyttet i andre sammenhenger. I denne studien har empirien derimot blitt benyttet for å svare på det andre forskingsspørsmålet, som spurte hvordan en slik praksis kan støttes av et MUVE. Deretter ble empirien analysert for å hente ut krav til prototypen. Grensesnitt og funksjonalitet har blitt designet slik at det reflekterer teori fra CSCW-feltet, og har bestått av et virtuelt workspace i Second Life. Intensjonen har vært å forbedre og forenkle den eksisterende måten Scrum programvareutviklingsteam benytter distribuert prosjektarbeid.

Det er ifølge Dittrich, Randall og Singer (2009) hensiktmessig å benytte teori fra CSCW-fagfeltet for å støtte programvareutvikling. Dette fordi denne arbeidspraksisen, ifølge dem, er cooperative work. Ved å vise hvordan prototypen er et Common Information Space (CIS), og hvordan den underbygger articulation work (Schmidt & Bannon 1992) og samarbeidsmekanismene til Gutwin og Greenberg (2000), har det blitt vist hvordan man kan bruke teori fra CSCW-feltet til å støtte distribuert programutvikling. Alle samarbeidsmekanismene har blitt tatt høyde for ved at studien har benyttet et MUVE, som vist ved presentasjon av designvalg og gjennom funnene fra evalueringen. Samarbeidsmekanismene representerer funksjonalitet som bør være tilgjengelig i et delt workspace. Hensynet til awareness

har blitt gjenspeilet i prototypen gjentatte ganger, ved at man kan se hva andre deltagere gjør på, hvor de er, hva de har gjort, om de er tilgjengelig, og så videre, i det delte workspacet. Schmidt og Bannon (1992) påpeker at det å konstruere et CIS er en god fremgangsmåte for å støtte articulation work. Prototypen tilbyr deltagerne en arena for å diskutere intensjonen til informasjonsobjekter, og på denne måten kan de øke sin forståelse for konteksten til objektet. Denne muligheten er grunnleggende for CIS. Videre er articulation work støttet gjennom tilretteleggelse for kommunikasjon, planlegging, overvåking og koordinasjon i prototypen.

Prototypen, som et CIS, tilrettelegger også for at ulike brukere, med forskjellig bakgrunn og kompetanse, kan samarbeide på kryss av landegrenser, med støtte for flere ulike kommunikasjonsformer. Den innehar mulighet for å presentere informasjon til de andre deltagerne, slik at ulike brukere kan bli bevisst på sine meddeltageres bakgrunn og kompetanse. Grunnlaget for at deltagerne kan oppnå en felles forståelse av informasjonsobjektene har dermed blitt styrket i forhold til eksisterende praksis. Årsaken til dette er at man ikke kan ta i bruk tradisjonelle hjelpemidler når man arbeider distribuert, og er avhengig av computer support for støtte en slik praksis.

Prototypen har blitt evaluert i den hensikt å finne ut hvorvidt den hadde god groupware usability. Dette ble gjennomført i to steg. For det første ble det foretatt en heuristisk evaluering for groupware, og sett på hvor god awareness man har i prototypen. For det andre ble det vurdert hvorvidt applikasjonen støttet samarbeidsmekanismene til Gutwin og Greenberg (2000), ved å gjennomføre en Groupware Walkthrough (Gutwin & Pinelle 2002). Selv om ikke grunnlaget for alle samarbeidsmekanismene fungerte optimalt, viste evalueringen gode resultater. Det har blitt utviket et møterom i prototypen som støtter Scrum møter og som har til intensjon å overkomme utfordringene med dagens praksis. Evalueringen har vist at møterommet fungerer tilfredsstillende ved at deltagerne kan benytte seg av basisfunksjonalitet i et MUVE. Fordelen med dette er at alle interessenter kan delta fra sin egen arbeidsstasjon, uavhengig av fysisk lokasjon. Videre har studien utviklet et rom som er egnet for å holde presentasjoner og opplæring. Dette kan være nyttig når man skal gi felles informasjon til teamet og andre interessenter. I programutviklingsteam er det et kontinuerlig behov for oppdatering og

kompetanseoverføring. Det er derfor nyttig å holde kurs og presentasjoner i et felles workspace. Opplæringsrommet inneholder et felles bibliotek hvor teamet kan samle prosjektinformasjon, kursmateriale og nyttige lenker til ekstern informasjon. Det tredje rommet er satt opp som et samarbeidsrom, hvor de ulike teammedlemmene kan vise hva de arbeider med. Her kan man for eksempel benytte parprogrammering for å samarbeide. Det at man kan observere hva de andre arbeider med gir en bedre awareness enn i dagens praksis. Det siste rommet har til intensjon å støtte sosialisering. I dagens praksis oppstår det gjerne kultur- og språkproblemer fordi man har lite uformell samhandling og fordi man arbeider på ulike fysiske lokasjoner. Evaluering viste at det var lav terskel for å sosialisere med andre. Sist, men ikke minst, har det vist seg at prototypen er enkelt å benytte, selv for helt uerfarne brukere.

På bakgrunn av resonnementet ovenfor kan man betrakte et MUVE som et godt utgangpunkt for å støtte distribuerte programvareutviklere i deres samarbeid. I neste avsnitt vil studien bli evaluert, og avslutningsvis vil det presenteres forslag til videre forskning.

9.1 Evaluering av studien

Prototype evalueringen ble utført i en laboratoriekontekst i Second Life, i den hensikt å finne ut hvorvidt prototypen hadde god groupware usability. Dette ble gjort ved å teste om den understøttet samarbeidsmekanismene. Det er en begrensning at prototypen ikke har blitt testet ut over lengre tid hos IKT-firma som benytter distribuert programutviklingssamarbeid med Scrum. Dermed er det ikke sikkert man vil få kartlagt om, for eksempel, navigering i prototypen fungerer bra, om den generelle forståelsen av MUVEet er tilfredsstillende, eller om den er robust over lengre tid, med last fra mange brukere. Neste naturlige steg er således å gjennomføre en oppfølgende studie hvor dette hadde blitt gjennomført.

Fokuset i denne studien har et av interesseområdene for CSCW-forskningsfeltet, og mye av forskningslitteraturen som har blitt benyttet har blitt publisert i relasjon til dette. I tillegg har det blitt presentert forskningslitteratur med fokus på programmering og MUVE. Det finnes derimot andre forskningsfelt som kunne vært interessant for studiens problemområde, som sosiologiske arbeidsstudier og organisasjonsforskning. Dette kunne muligens bidratt med nyttig informasjon om, for eksempel, forholdet mellom de ulike rollene som er involvert i programvareutvikling.

9.2 Videre forskning og fremtidige utfordringer

Resultatene fra den kvalitative undersøkelsen og observasjonene kan benyttes i fremtidige forsking på groupware- og samarbeidsstudier. Evaluering av prototypen viste at den hadde tilfredsstillende groupware usability, men noen problemer ble imidlertid identifisert. Disse problemene kan bidra til å skape nye problemstillinger innenfor problemområdet til denne studien. Dette kan være aktuelle utgangspunkt for fremtidig forskning.

Second Life støtter tradisjonelle former for kommunikasjon, som IM og VoIP, men det kunne vært hensiktsmessig å undersøke andre former for interaksjon og brukergrensesnitt, som ville forbedret kommunikasjon og awareness mellom brukerne.

Fordi dette er en løsning tilpasset programutviklere, kunne det vært nyttig å inngå et forskningssamarbeid for å videreutvikle prototypen. Dette kan organiseres som et open source-prosjekt, fordi deler av kildekoden til Second Life nå er fritt tilgjengelig. På denne måten kunne man fått designet en skreddersydd løsning, hvor man for eksempel kunne integrert eksisterende utviklingsverktøy med prototypen, som ville tilfredsstilt samarbeidsmekanismene og awareness på en bedre måte.

En aktuell problemstilling for fremtidig forskning, som nevnt i evaluering av studien, er også å undersøke hvorvidt prototypen kan fungere i en reell arbeidskontekst.

Groupware løsninger, slik som den utviklet i studien, er et viktig supplement når bedrifter beveger seg over mot en mer fleksibel og mobil måte å arbeide, som beskrevet av Bødker og Christiansen (2006). Det empiriske materialet som viser hvordan distribuerte Scrum team samarbeider, prototypen som har blitt designet på bakgrunn av denne, samt evalueringen, er nøkkelfunnene i denne studien.

10 Litteraturliste

Abbattista, F., Calefato, F., Gendarmi, D. & Lanubile, F. (2008). *Incorporating social software into distributed agile development environments*. Automated Software Engineering - Workshops, 2008. ASE Workshops 2008. 23rd IEEE/ACM International Conference. 46-51 s.

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). *Agile software development methods. Review and analysis*. Vuorimiehentie, Finland, VTT Publications 478.

Archer, B. (1973). *The Need for Design Education*, Royal College of Art.

Avison, D. & Fitzgerald, G. (2003). *Information Systems Development: Methodologies, Techniques and Tools*, McGraw-Hill Higher Education.

Baker, K., Greenberg, S. & Gutwin, C. (2001). *Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration*. Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction. Springer-Verlag.

Banks, F. (1994). *Teaching technology*. London, Routledge/Open University. 253 s.

Bannon, L. & Bødker, S. (1997). *Constructing common information spaces*. Proceedings of the fifth conference on European Conference on Computer-Supported Cooperative Work, Lancaster, UK. Kluwer Academic Publishers.

Bannon, L. J. & Schmidt, K. (1989, 13-15 September). *CSCW: four characters in search of a context*. ECSCW'89. Proceedings of the first European Conference on Computer Supported Cooperative Work,, Gatwick, London. 358-372 s.

Bannon, L. J. (2000). *Understanding Common Information Spaces in CSCW, Workshop on Cooperative Organisation of Common Information Spaces*. Workshop on Common Information Spaces, Copenhagen, Denmark.

Bardzell, S., Bardzell, J., Pace, T. & Reed, K. (2008). *Blissfully productive: grouping and cooperation in world of warcraft instance runs*. CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work, San Diego, CA, USA. ACM. 357-360 s.

Beck, K. (1999a). Embracing Change with Extreme Programming. *Computer*, 32 (10): 70-77.

Beck, K. (1999b). *Extreme Programming Explained: Embrace Change*, {Addison-Wesley Professional}.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C.,

Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). *Manifesto for Agile Software Development*. Tilgjengelig fra: http://agilemanifesto.org/ (lest 06.11.2009).

Bjerrum, E. & Bødker, S. (2003). *Learning and living in the 'new office'*. Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work, Helsinki, Finland. Kluwer Academic Publishers.

Boden, A., Nett, B. & Wulf, V. (2008). *Articulation work in small-scale offshore software development projects*. Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering, Leipzig, Germany. ACM.

Braadland, T. R. (2002). *Innføring i informasjonsbehandling*. 2nd utg. Oslo, Fagbokforlaget. 381 s.

Bødker, S. & Christiansen, E. (2006). Computer Support for Social Awareness in Flexible Work. *Comput. Supported Coop. Work*, 15 (1): 1-28.

Calongne, C., Endorf, S., Frankovich, D. & Sandaire, J. (2008). A virtual environment for designing user interface prototypes. *J. Comput. Small Coll.*, 24 (1): 188-195.

Cohen, D., Lindvall, M. & Costa, P. (2004). *An Introduction to Agile Methods*. Advances in Computers, New York. Elsevier Science. 1-66 s.

Dalen, M. (2001). *Intervju som forskningsmetode*. Oslo, Norway, Universitetsforlaget

Denniston, C. (2009). *Danube Technologies Announces Release of ScrumWorks Pro 4.1 for Improved Program Management of Scrum Projects and Epics*, Reuters. Tilgjengelig fra: http://www.reuters.com/article/pressRelease/idUS94374+12-Oct-2009+PRN20091012 (lest 12.11.2009).

Dieterle, E. & Clarke, J. (2009). Multi-User Virtual Environments for Teaching and Learning. I: *Encyclopedia of multimedia technology and networking (2nd ed)*, s. 146-158, Hershey, PA: Idea Group.

Dittrich, Y., Randall, D. & Singer, J. (2009). Software Engineering as Cooperative Work. *Computer Supported Cooperative Work (CSCW)*, 18 (5): 393-399.

Ellis, C. A., Gibbs, S. J. & Rein, G. (1991). Groupware: some issues and experiences. *Commun. ACM*, 34 (1): 39-58.

Endsley, M. R. (1995). Toward a Theory of Situation Awareness in Dynamic Systems. *37*, 1: 32-64(33).

Gaba, D., Howard, S. & Small, S. (1995). Situation Awareness in Anesthesiology. *Human Factors*, 37 (1): 20-31.

Gerson, E. M. & Star, S. L. (1986). Analyzing due process in the workplace. *ACM Trans. Inf. Syst.*, 4 (3): 257-270.

Greif, I. (red.). (1988). *Computer-supported cooperative work: a book of readings*, Morgan Kaufmann Publishers Inc. 781 s.

Grudin, J. (1994). Computer-supported cooperative work: history and focus. *Computer*, 27 (5): 19-26.

Gutwin, C. & Greenberg, S. (1996). *Workspace awareness for groupware*. CHI '96: Conference companion on Human factors in computing systems. ACM Press. 208-209 s.

Gutwin, C. & Greenberg, S. (2000). *The Mechanics of Collaboration: Developing Low Cost Usability Evaluation Methods for Shared Workspaces*. Proceedings of the 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. IEEE Computer Society.

Gutwin, C. & Greenberg, S. (2002). A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Comput. Supported Coop. Work*, 11 (3): 411-446.

Gutwin, C. & Pinelle, D. (2002). *Groupware walkthrough: adding context to groupware usability evaluation*. Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves, Minneapolis, Minnesota, USA. ACM. 455-462 s.

Heaton, J. (2007). *Introduction to Linden Scripting Language for Second Life*, Heaton Research, Inc. 236 s.

Hevner, A. R., March, S. T., Park, J. & Ram, S. (2004). Design science research in information systems. *MIS Quarterly*, 28 (1): 75.

Hoffer, J. A., George, J. F. & Valacich, J. S. (2005). *Modern Systems Analysis and Design (4th Edition)*, Prentice-Hall, Inc.

Hossain, E., Babar, M. A. & Paik, H.-y. (2009). *Using Scrum in Global Software Development: A Systematic Literature Review*. Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering - Volume 00. IEEE Computer Society.

Human Productivity Lab. (2006). *Microsoft announces RoundTable (Formerly RingCam) a 360 Degree "Super Webcam*, Human Productivity Lab. Tilgjengelig fra: http://www.humanproductivitylab.com/archive_blogs/2006/06/27/microsoft_announces_roundtable.php (lest 23.08.2009).

Kahai, S. S., Carroll, E. & Jestice, R. (2007). Team collaboration in virtual worlds. *SIGMIS Database*, 38 (4): 61-68.

Kaptelinin, V. (2003). *UMEA: translating interaction histories into project contexts*. Proceedings of the SIGCHI conference on Human factors in computing systems, Ft. Lauderdale, Florida, USA. ACM.

Kaptelinin, V. & Nardi, B. A. (2006). *Acting with Technology: Activity Theory and Interaction Design (Acting with Technology)*, The MIT Press.

Kaspersen, L. (2009). *Kutter forretningsreiser - sparer millioner*. Oslo, Dagens Næringsliv (lest 23.09.2009).

Koch, M. & Gross, T. (2006). *Computer-Supported Cooperative Work - Concepts and Trends*. Proceedings of the 111th Conference of the Association Information and Management (AIM), Luxembourg, Luxembourg.

Kvale, S. (1997). *Det kvalitative forskningsintervju*. Oslo, Ad notam Gyldendal. 236 s.

Kyng, M. (1991). Designing for cooperation: cooperating in design. *Commun. ACM*, 34 (12): 65-73.

Linden Labs. (2009a). *Currency Exchange*, Linden Labs. Tilgjengelig fra: http://secondlife.com/whatis/currency.php (lest 06.11.2009).

Linden Labs. (2009b). *Second Life Knowledge Base*. Tilgjengelig fra: http://secondlife.com/kb (lest 19.11.2009).

Livingstone, D. & Kemp, J. (2008). Integrating Web-Based and 3D Learning Environments: Second Life Meets Moodle. *UPGRADE, The European Journal for the Informatics Professional*, IX (3): 8-14.

Lorentzen, H. (2007). *Moraldannende kretsløp: Stat, samfunn og sivilt engasjement*. Trondheim, Norge, Abstrakt forlag.

Lucia, A. D., Francese, R., Passero, I. & Tortora, G. (2008). *SLMeeting: supporting collaborative work in Second Life*. Proceedings of the working conference on Advanced visual interfaces, Napoli, Italy. ACM.

Maxwell, J. (2004). *Qualitative Research Design : An Interactive Approach (Applied Social Research Methods)*, {SAGE Publications}.

Maxwell, J. A. (2005). *Qualitative research design: an interactive approach*. Thousand Oaks, Calif., Sage Publications. XIV, 175 s. s.

Mitchell, K. (2009). *Over 15 Billion Minutes of Voice Have Been Delivered in Second Life*. Tilgjengelig fra: http://lindenlab.com/pressroom/releases/19_05_09 (lest 06.10.2009).

Nardi, B. A., Whittaker, S. & Bradner, E. (2000). *Interaction and outeraction: instant messaging in action*. Proceedings of the 2000 ACM conference on Computer supported cooperative work, Philadelphia, Pennsylvania, United States. ACM.

Neisser, U. (1976). *Cognition and Reality*. San Fransisco, W.H. Freeman.

Nielsen, J. & Molich, R. (1990). *Heuristic evaluation of user interfaces*. CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems. ACM Press. 249-256 s.

Nurminen, M. I., Fjuk, A. & Smørdal, O. (1997). *Taking Articulation Work Seriously - an Activity Theoretical Approach*. International Conference on Software Engineering, Leipzig, Germany. Turku Centre for Computer Science.

Olivier, H. & Pinkwart, N. (2007). Collaborative Virtual Environments – Hype or Hope for CSCW? *Ifl Technical Report Series*, 7 (14).

Omoronyia, I., Ferguson, J., Roper, M. & Wood, M. (2009). Using Developer Activity Data to Enhance Awareness during Collaborative Software Development. *Computer Supported Cooperative Work (CSCW)*, 18 (5): 509-558.

Owens, D., Davis, A., Murphy, J. D., Khazanchi, D. & Zigurs, I. (2009). Real-World Opportunities for Virtual- World Project Management. *IT Professional*, 11 (2): 34-41.

Pinelle, D. & Gutwin, C. (2001). *Group Task Analysis for Groupware Usability Evaluations*. Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. IEEE Computer Society.

Pullen, J. M., Norris, E. & Fix, M. (2000). Teaching C++ in a multi-user virtual environment. *SIGCSE Bull.*, 32 (2): 60-64.

Purbrick, J. & Lentczner, M. (2007). *Second life: the world's biggest programming environment*. Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion, Montreal, Quebec, Canada. ACM.

Ringdal, K. (2007). *Enhet og mangfold: samfunnsvitenskapelig forskning og kvantitativ metode*. Bergen, Fagbokforlaget. 502 s.

Ryen, A. (2002). *Det kvalitative intervjuet: fra vitenskapsteori til feltarbeid*. Bergen, Fagbokforl. 317 s.

Rymaszewski, M., Rosedale, P., Batstone-Cunningham, B., Ondrejka, C., Winters, C., Wallace, M. & Au, W. J. (2008). *Second Life : the official guide*. Indianapolis, Wiley Pub.

Salem, B. & Earle, N. (2000). *Designing a non-verbal language for expressive avatars*. Proceedings of the third international conference on Collaborative virtual environments, San Francisco, California, United States. ACM.

Schmidt, K. & Bannon, L. (1992). Taking CSCW seriously. Supporting articulation work. *Computer-Supported Cooperative Work* (1): 7--40.

Schwaber, K. (1995). *SCRUM development process*. Conference on Object-Oriented Programing Systems, Languages, and Applications (OOPSLA'95) Workshop on Business Object Design and Implementation.

Schwaber, K. & Beedle, M. (2002). *Agile software development with Scrum*. Series in agile software development. Upper Saddle River, NJ, Prentice Hall.

Second Life Quickstart. (2009). Linden Labs. Tilgjengelig fra: http://secondlife.com/support/quickstart/ (lest 17.07.2009).

Sharp, H., Rogers, Y. & Preece, J. (2007). *Interaction Design: Beyond Human-Computer Interaction*, Wiley.

Simon, H. A. (1996). *The sciences of the artificial*. Cambridge, Mass., MIT Press.

Statens vegvesen. (2009). *Om Statens vegvesen*. Tilgjengelig fra: http://www.vegvesen.no/Om+Statens+vegvesen (lest 06.10.2009).

Steuer, J. S. (1995). Defining virtual reality: dimensions determining telepresence. I: *Communication in the age of virtual reality*, s. 33-56, L. Erlbaum Associates Inc.

Strauss, A. (1985). Work and the division of labor. *The Sociological Quarterly*, 26 (1): 1-19.

Strauss, A. L., Fagerhaugh, S., Suczek, B. & Wiener, C. (1985). *Social organization of medical work*. Chicago :, University of Chicago Press.

Strauss, A. L. (1993). *Continual permutations of action*. Communication and social order. New York :, Aldine de Gruyter.

Strauss, A. L. & Corbin, J. M. (1993). The Articulation of Work through Interaction. *The Sociological Quarterly*, 34 (1): 71-83.

Suchman, L. (1994). Working relations of technology production and use. *Computer Supported Cooperative Work (CSCW)*, 2 (1): 21-39.

Sutherland, J., Viktorov, A., Blount, J. & Puntikov, N. (2007). *Distributed Scrum: Agile Project Management with Outsourced Development Teams*. Proceedings of the 40th Annual Hawaii International Conference on System Sciences. IEEE Computer Society.

Takeuchi, H. & Nonaka, I. (1986). The new new product development game. *Harvard Business Review*.

Vaishnavi, V. & Kuechler, W. (2004/5). *Design Research in Information Systems*. January 20, 2004, last updated August 16, 2009 utg. Tilgjengelig fra: http://desrist.org/design-research-in-information-systems.

Version One. (2008). *3rd annual survey: The State of Agile Development*, Version One. Tilgjengelig fra: http://pm.versionone.com/whitepaper_AgileSurvey2008.html (lest 02.10.2009).

Waldo, J. (2008). Scaling in Games & Virtual Worlds. *Queue*, 6 (7): 10-16.

Walls, J., Widmeyer, G. & Sawy, O. E. (1992). Building an Information System Design Theory for Vigilant EIS. *Information Systems Research*, 3 (1): 36-59.

Whitehead, J. (2007). *Collaboration in Software Engineering: A Roadmap*. 2007 Future of Software Engineering. IEEE Computer Society.

Widerberg, K. & Bolstad, K. (2001). *Historien om et kvalitativt forskningsprosjekt: en alternativ lærebok*. Oslo, Universitetsforlaget. 206 s.

Yankelovich, N. (2007). *MPK20: Sun´s Virtual Workplace*. [Digital]. Yankelovich, N. (produsent). Distribuert av Sun Microsystems, Inc.

Yin, R. K. (2003). *Case Study Research: Design and Methods*. Newbury Park, California, Sage Publications.

Zigurs, I. (2003). Leadership in Virtual Teams: - Oxymoron or Opportunity? *Organizational Dynamics*, 31: 339-351.

11 Appendiks A – Kvalitativ undersøkelse

11.1 Log fra daglig Scrum møte hos IKTBergen

Frank says (10:32):
hi
Petter says (10:32):
Good morning!
Charlotte says (10:32):
hello
Jostein says (10:32):
God morgen!
m.teigland@iktbergen.no says (10:33):
hey there
Kristofer says (10:33):
Good morning!
Kristian says (10:33):
Hello!
Kristofer says (10:33):
I am in another meeting, so Petter is PO today
Petter says (10:33):
Lucky me

Kristofer has left the conversation.

Charlotte says (10:34):
Y: prepared for work after vacation, started "Dynamic address field mapping" T: continue.
Frank says (10:34):
Y: Changes in Mozaic import UI, some research how to detect language of Mozaic search page. T: Implementing Mozaic icons in IKTBergen UI, changes in IKTBergen web browser. Done.
Petter says (10:35):
Charlotte: Is everything clear as far as "Dynamic address field mapping" is concerned? There were a lot of comments on comments in the e-mails yesterday
Kristian says (10:35):
Yesterday: reading e-mails after vacation, started Mozaic testing with Frank.
Today: continue with Mozaic.
Frank says (10:35):
From Erik
Y: SPM: Modify import to include DUNS number
T: Fixing and dev. Testing; probably, merging to rel641
No impediments
Done

Roger has been added to the conversation.

m.teigland@iktbergen.no says (10:35):
Yesterday: IPS support on the phone, running some scripts, fixed the problem with emails not being sent and with extra workflow choices
Today: some burning tickets, meet with Ingrid and decide.
Petter says (10:36):
Yesterday: Started on the Mozaic test cases in Testuff, continued with the Total support report ++
Today: Continue on Mozaic test cases and get started with the documentation
Jostein says (10:36):

Y: Modelling and documenting on 'Dynamic address field mapping'. Demo on UI changes from Trygve. T: Meetings with students from Bergen University, start 'Unique Mozaic import', I'm off early today.

Alexei says (10:36):

Morning!
Y: Some work on missing documents ticket, the rest installing new harddisk.
T: Missing documents ticket.

Frank says (10:37):

....

Petter says (10:37):

Any questions or impediments?

Alexei says (10:38):

Has everybody updated SW lately? No change in remaining hrs last 2 days....

Kristian has left the conversation.

Kristian has been added to the conversation.

Kristian has left the conversation.

Jostein says (10:41):

Adjourn?

Frank says (10:42):

Adjourned!

m.teigland@iktbergen.no says (10:42):

yea!

Jostein says (10:42):

Bye

m.teigland@iktbergen.no has left the conversation.

Alexei says (10:42):

Bye

11.2 Intervjuguide

Tema i forbindelse med undersøkelse om programutvikling. Temaene ble ikke nødvendigvis gjennomgått i gitt rekkefølge, fordi man forsøkte å gjennomføre intervjuet som en samtale for best å fange opp informantenes meninger. Temaene er mer et redskap for å sikre at de ulike problemstillingene som er interessante for studien ble gjennomgått.

Bakgrunnsinformasjon om kandidaten;
- Utdanning
- Erfaring (arbeid, hobby)
- Hvordan ser en normal arbeidsuke ut

Forhold til tradisjonelle programvareutviklingsmetoder;
- Erfaringer i bruk

Forhold til Agile utviklingsmetoder;
- Erfaringer i bruk

Forhold til Scrum;
- Erfaringer i bruk
- Roller en har hatt og møter en har deltatt på
- Ulike Roller i Scrum:
  - Scrum Master
  - Product Owner
  - Scrum Team
  - Costumer
  - Management
- Ulike møter i Scrum:
  - Sprint Planning Meeting
  - Sprint Backlog
  - Daily Scrum Meeting
  - Sprint Review Meeting

Praktisk gjennomføring av møter;

Hvilke verktøy og hjelpemidler pleier en å bruke for å avvikle de ulike møtetypene i Scrum møter når man er tilstede i samme rom.

Hvilke verktøy og hjelpemidler ser en for seg en trenger for å understøtte de samme møtene i Scrum når man ikke er tilstede på samme sted.

Se video fra SUN MPK20 – Sun's Virtual Workspace for å gi kandidaten inntrykk av muligheter.
http://research.sun.com/projects/mc/video/MPK20-aug2007.mov?cid=921781

Hvordan ser en for seg dette arbeidet utført i en 3D verden?

Fordeler
Ulemper
Tilstedeværelse / Awareness
Sosiale forhold
Arbeidsmiljø
Tekniske faktorer
Andre faktorer som du mener er viktige i en slik setting

11.3 Intervjuspørsmål

1. Bakgrunn;
    a. Hvor lenge har du hatt din nåværende jobb?
    b. Hva gjorde du før du begynte der?
    c. Fortell litt om utdanningen din.
    d. Hvorfor falt valget på denne utdanningen?

2. Arbeidssituasjon;
    a. Hvordan er en vanlig arbeidsdag?
    b. Fortell litt om arbeidsoppgavene du har i dag?
    c. Hvordan liker du arbeidsoppgavene dine?
    d. Fortell litt om hvordan du samarbeider med dine kollegaer?
    e. Hvordan fungerer samarbeidet med kunder?
    f. Hva føler du kunne vært bedre?
    g. Hvor jobber du fra?

3. Kjennskap til ulike utviklingsmetoder;
    a. Fortell litt om din erfaring med ulike utviklingsmetoder.
    b. Bruker dere Scrum?
    c. Hva bruker dere Scrum til og hvordan?
    d. Har dere benyttet Scrum lenge?
    e. Hvordan fungerer Scrum best?
    f. I Scrum er det definert ulike roller. Fortell litt om din erfaring med disse?
    g. Hvilke erfaringer har du med gjennomføring av Scrum?

4. Samarbeidsformer;
    a. Hvordan samarbeider du med andre som er tilstede på samme sted som deg?
    b. Hvordan samarbeider du med andre som ikke er tilstede på samme sted som deg?
    c. Hvilke verktøy benytter du for å samarbeide med andre?
    d. Føler du at Scrum en motiverende faktor for samarbeid?
    e. Hvilke verktøy ser du for deg hadde vært nyttig når en arbeider i et Scrum team som er samlokalisert?
    f. Hvilke verktøy ser du for deg hadde vært nyttig når en arbeider i et Scrum team som ikke er samlokalisert?

(Se video fra
http://research.sun.com/projects/mc/video/MPK20-aug2007.mov?cid=921781
– Sun's Virtual Workspace for å gi kandidaten en inntrykk av muligheter)

5. Samarbeid i en virtuell verden;
    a. Hvordan tror du at en virtuell verden kan understøtte Scrum samarbeid?
    b. Hvilke verktøy ser du for deg en kan benytte seg av i en virtuell verden for å understøtte Scrum?

c. Er dette noe du kunne tenkt deg å benytte?
d. Hvilke positive sider har en slik arbeidsmåte?
e. Hvordan tror du det sosiale ville bli påvirket hvis en utførte Scrum møter i en virtuell verden?
f. Tror du det er mulig å skape et godt miljø blant ansatte som er spredt på ulike lokasjoner ved hjelp av en slik virtuell virkelighet?

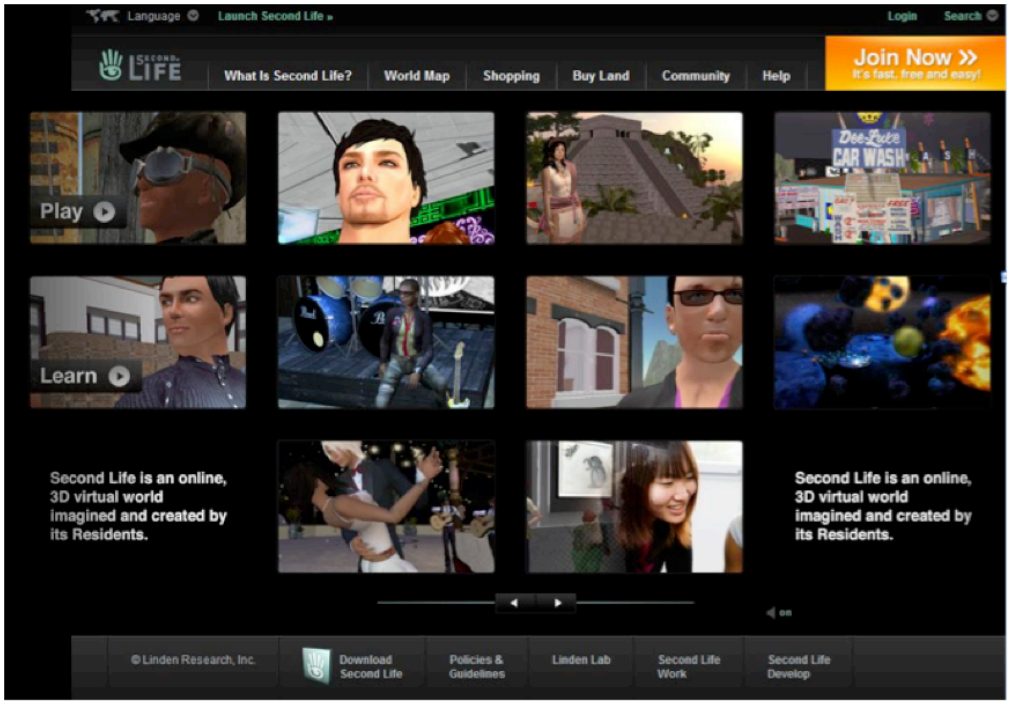
6. Annet;
    a. Noe annet du vil legge til?

12 Appendiks B – Design i Second Life

I dette appendikset blir det beskrevet hvordan man kommer i gang med å bygge i Second Life. Dette er fremgangsmåten som har blitt benyttet for å lage prototypen i studien, og kan være et nyttig bidrag til IS-kunnskapsbasen.

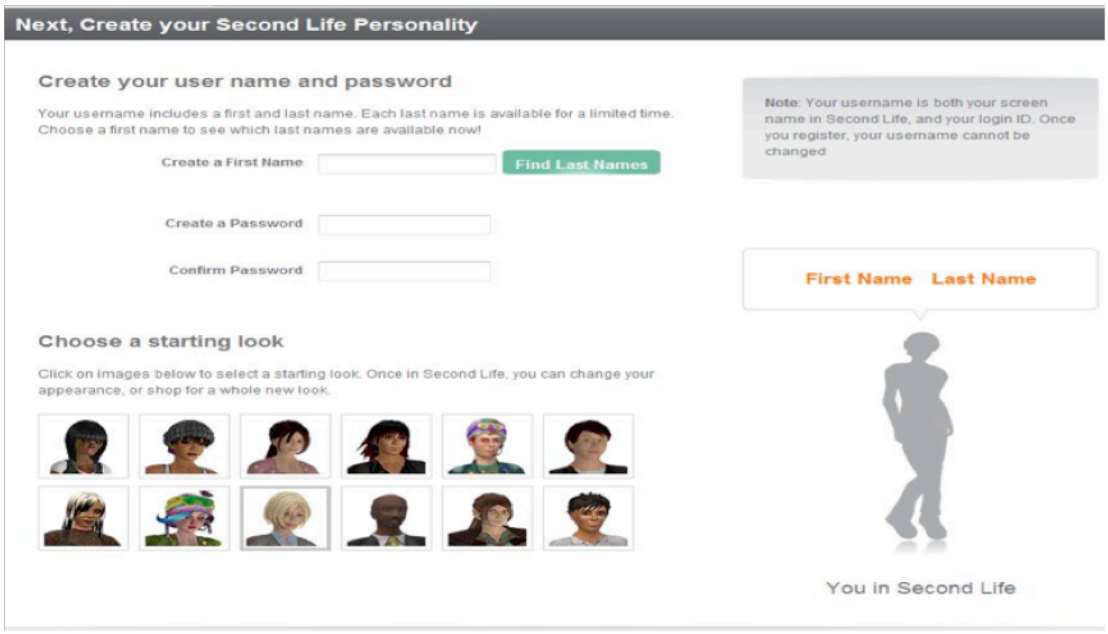
12.1 Lage brukerkonto og logge på

Man starter opp med å gå inn på nettsiden http://www.secondlife.com. Denne siden er vist i figur 30.



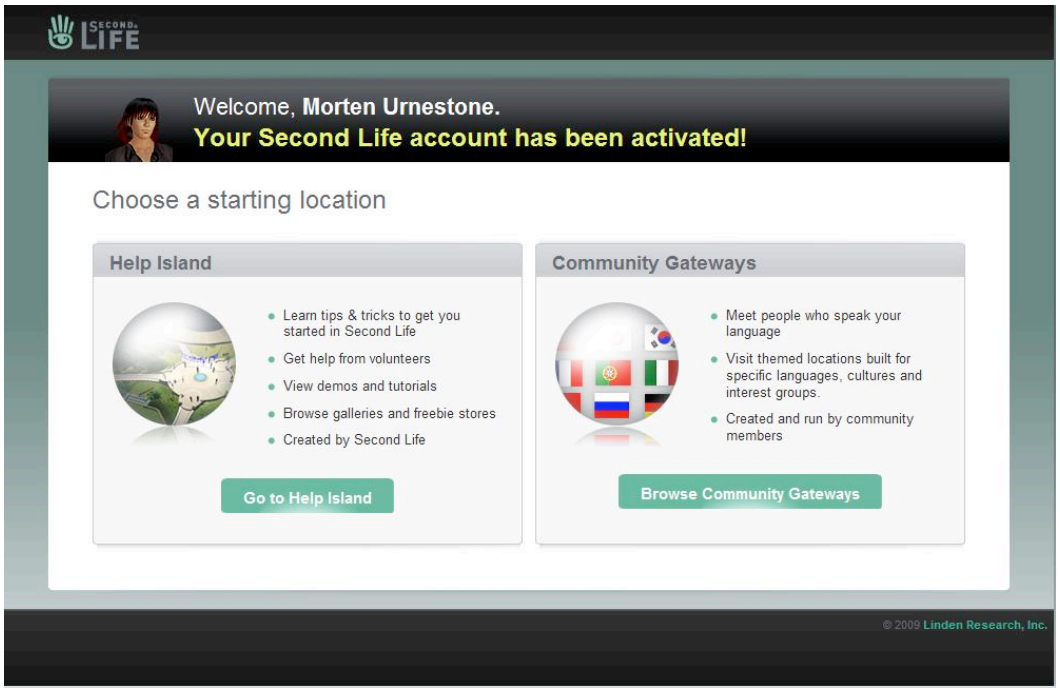
Figur 30 - Startsiden til Second Life

Her trykker man på ”join now” første gangen. Hvis man har bruker fra tidligere velges ”login” i høyre hjørnet. Første del av registreringen krever at kontoen blir knytet til dine persondata som navn, e-post, land man bor i, kjønn og fødselsdato. De objektene som blir kjøpt i Second Life av din avatar må også kunne knyttes til deg som person. I andre del av registreringen får man opp bildet som er vist i figur 31.



Figur 31 – Opprette avatar

Her velger man fornavn på avataren. Her kan man dessverre ikke fritt velge etternavn, men får opp en autogenerert liste avhengig av det fornavnet man valgte. Videre velger man om man vil ha en kvinnelig eller mannlig avatar, og til slutt hvordan den skal se ut. Siste del av registreringen er å skrive inn et sikkerhetsspørsmål for at Linden Labs skal kunne gi telefonsupport, hjelpe med passord og andre henvendelser. Det er også verdt å nevne at man ikke kan endre brukernavnet på avataren i etterkant. Deretter trykker man på ”create account”, og en bekreftelse blir så sendt på e-post. Når man trykker på aktiveringslinken får en opp to valg, som vist i figur 32.



Figur 32 - Velge startområde

Det anbefales å starte i ”Help Island” for å komme i gang. Det andre alternativet er å starte i et miljø tilpasset språk og kultur fra ditt hjemsted. Etter at man velger startsted får man opp en meny for nedlasting av selve klienten til Second Life. På den samme siden er det også link til Second Lifes ”quickstart guide” (Second Life Quickstart 2009). Denne anbefales fordi den gir en rask og grundig innføring i de basiskunnskaper en trenger for å bruke Second Life.

Nå er det bare å logge inn i Second Life via klienten, og deretter akseptere ”terms of Service”. Man ankommer destinasjonen som ble valgt tidligere. På ”Help Island” man lære seg ulike ferdigheter, og snakke med andre nybegynnere. Når man føler at en mestrer menyer og navigering kan man gå videre. Dette gjør man ved hjelp en teleporteringsportal, tilsvarende den i prototypen, hvorpå man kan fortsette å utforske Second Life.

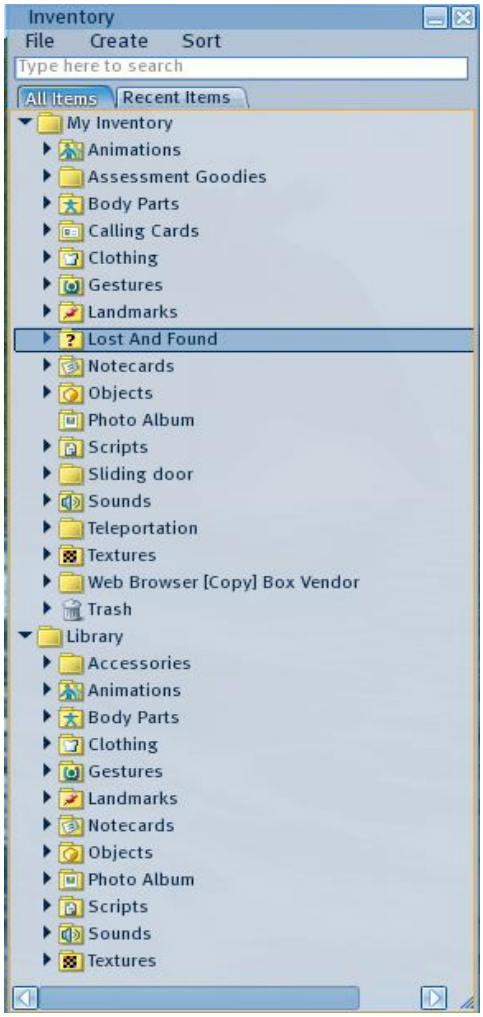
12.2 Byggeprosessen

Når man skal lage objekter, finner man raskt ut at dette er en tiløvd ferdighet. Før det var mulig å starte med å designe prototypen gikk det med mye tid i sandboxer, som er gratis, for å mestre de ulike byggeteknikkene. Her kommer man i kontakt med andre som bygger, og kan lære fra dem. Det sosiale aspektet er viktig i en slik prosess, og

det var derfor ikke ønskelig å gå til anskaffelse av et område for prototypen før disse ferdighetene var på plass. Det virker så enkelt å bygge, men det er mange ferdigheter som må til for å få et bra resultat. Her er det eneste som nytter å gjenta prosessen helt til man mestrer den, men heldigvis er det hjelp å få. Det finnes mange illustrative videoer på YouTube om hvordan man bygger, og mange tips og triks tilgjengelige på et utvalg av nettsider som innbyggerne har lagt ut. I de neste avsnittene blir noen av teknikkene som ble benyttet for å designe prototypen, som beskrevet i kapittel 7, gjennomgått.

12.2.1 Inventar mappen

Inventar mappen, som vist i figur 33, er avatarens personlige oppbevaringssted for de ting han eier, lager eller kjøper. Inventarmappen består av to hovedmapper; personlig inventarmappe og et bibliotek. I personlig inventarmappe ligger alt som brukeren eier. I biblioteket ligger et utvalg ting som man fritt kan benytte.



Figur 33 - Inventarmappen

Det å holde oversikten i denne mappen er viktig for å kunne bruke Second Life optimalt. Alt man bygger kan man ta kopier av, og disse kopiene ender da opp i inventarmappen. Det er derfor viktig å gi tingene man lager gode navn, slik at man kan finne dem igjen. En ulempe er at det kun er navnet som beskriver objektet i mappen. Vil man se hvordan objektet ser ut, må en ta det ut og sette det på bakken.

I figur 33 kan man se de ulike typene ting en avatar kan ha med seg, slik som klær, bilder, scripts, objekter, tekstiler, lyder, også videre.

12.2.2 Bygge via objekter

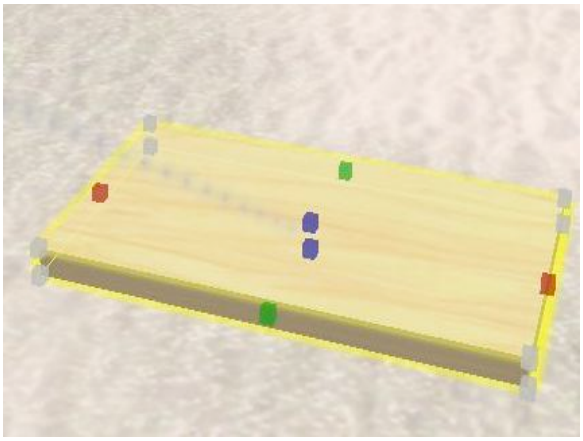
Figur 34 viser hvordan menyen ser ut når man bygger et objekt. Når et objekt er i editeringsmodus vil man få opp piler som viser X-, Y- og Z-koordinater, og man kan flytte objektet med å dra pilene i ønsket retning. Deretter, som man ser i menyen, kan man velge å rotere objektet (se Figur 36), eller strekke og dra i det, for å oppnå ønsket fasong og størrelse (se Figur 35).

Figur 34 - Lage objekt

Således kan man velge mellom ulike fasonger på objektene. Her finnes følgende ferdigdefinerte former: ”box, cylinder, sphere, torus, tube, ring and sculpted”. Alle objekter i Second Life er bygget opp av disse formene. Når man velger å bygge via ”build” knappen, får man opp en tryllestav. Med denne kan man generere en av de ferdig definerte objekttypene.

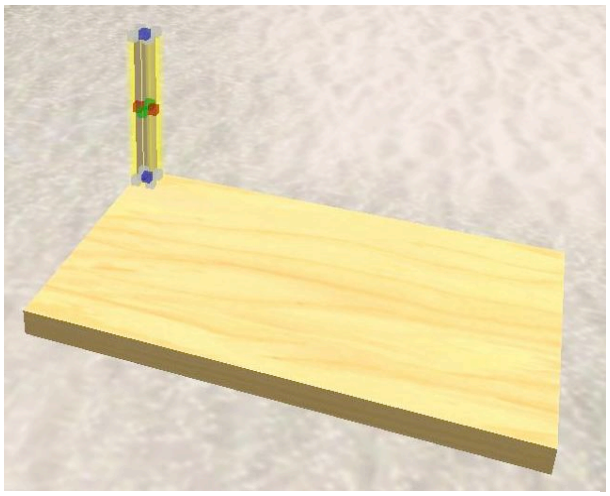
For å demonstrere hvordan man bygger vil det i det følgende bli gitt en gjennomgang av hvordan man går fram for å designe et bord. Årsaken til at et bord er valgt, er at det er lettere å vise teknikken i mindre skala. Teknikkene man bruker for å bygge mindre objekter er direkte overførbare til større konstruksjoner.

Figur 36 - Rotere objekt

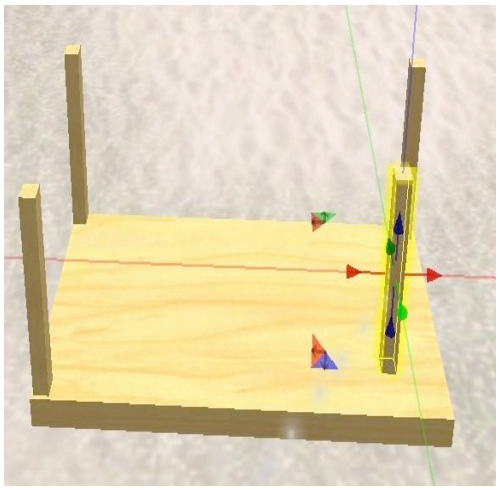


Figur 35 - Strekke og tilpasse objekt

Når man skal bygge et bord starter man med et av de ferdig definerte objektene. Her velger man den som er nærmest den fasongen man ønsker. Her er det fornuftig å starte med en firkant (se figur 34), som kan bli manipulert om til en bordplate (se figur 35). Videre velger man et objekt av samme type for å lage bena til bordet, og tilpasser dette (se figur 37). Fordi det er vanskelig å lage fire like ben, er en god teknikk å ta kopi av benet inn i inventar mappen, og bruke dette om igjen for å lage de andre bena. Dette gjør at man bare trenger å plassere dem (se figur 38). Dermed har man et bord bestående av fem separate objekter.



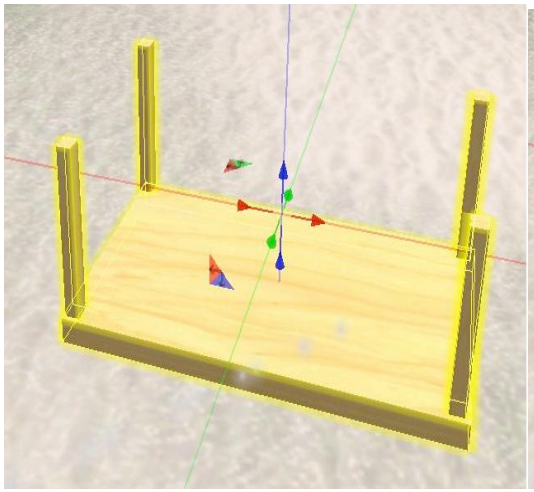
Figur 37 - Lage bordben



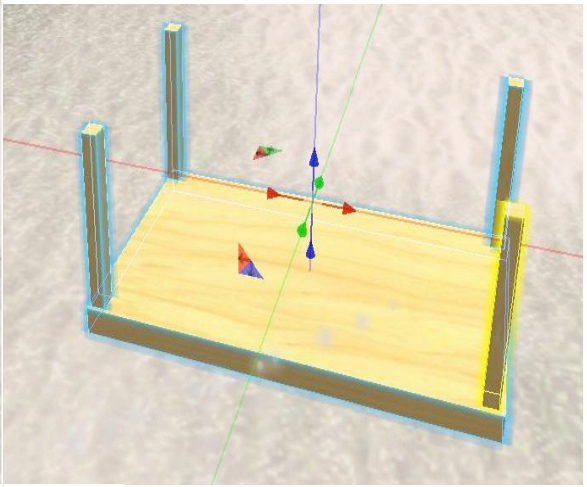
Figur 38 - Plassere bordben

Det tar litt tid før man forstår hvordan man kan sette sammen et objekt med flere separate deler. Det at et objekt består av flere deler, gjør det vanskelig å håndtere det, fordi man må flytte alle delene samtidig. Hvis ikke ødelegger man objektet. Heldivis er det mulig å linke sammen objekter slik at de fremstår som en helhet (et objekt).

Dette gjøres først ved å velge alle objektene (se figur 40), for så å linke dem sammen ved å trykke CTRL+L, eller velge linke fra handlingsmeny. Man kan ta bort denne koblingen igjen ved å trykke CTRL+SHIFT+L, eller avlinke fra handlingsmeny. Når objektet er linket fremstår det som i figur 39. Her kan man se at fire av delene er blå, mens en fremdeles er gul. Den gule er master prim i det sammensatte objektet. Dette betyr at det er det objektet som resten orienterer seg ut i fra. Dette er viktig når man skal lage script. Scriptet må være plassert i master prim for å kunne fungere. Når man skal plassere noe i forhold til dette bordet ved hjelp av et script, er det master prim som bestemmer koordinatene i rommet.

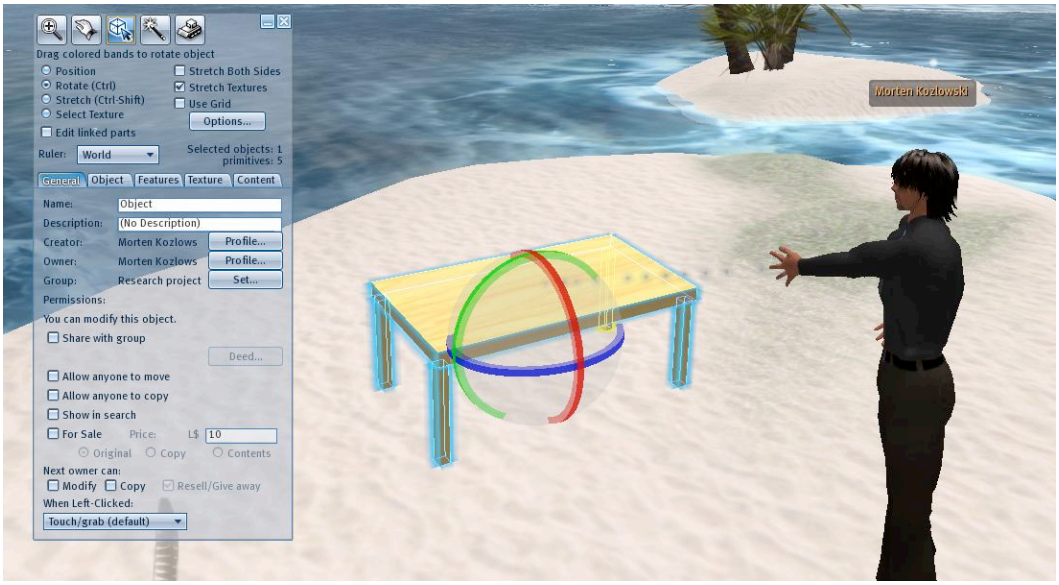


Figur 40 - Velge alle objektene



Figur 39 - Linke objektene

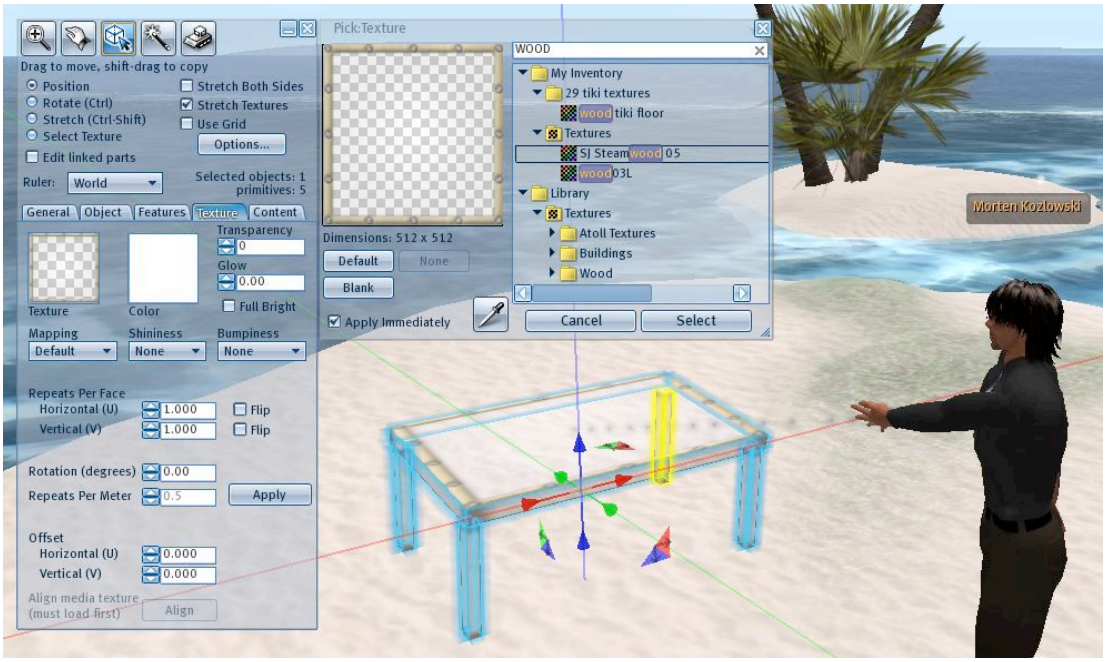
En ting man raskt oppdager når man gjør ting i Second Life, er at det ikke er noen angre mulighet. Sletter man noe, eller gjør noe man angrer på, er det ingen mulighet til å rette dette opp. Det eneste man kan gjøre er å starte på nytt. Det at man ikke kan angre, er noe som man kan savne mye, spesielt når man er nybegynnere. Designet av prototypen hadde gått raskere fram om dette var en mulighet. Etter hvert som ferdighetene utvikler seg blir heldigvis behovet for en angre funksjon. I Figur 41 vises det ferdige bordet.



Figur 41 - Det ferdigstilte bordet

12.2.3 Tekstiler (Textures)

Nå er bordet bygget, men som man ser fra Figur 41, har det ingen farger ennå. Fordi standard farge, og tekstil på objekter er tregul, ser det ut som et bord i tre. Dersom man hadde bygget noe annet, for eksempel et bilde, vil det se ut som en trekloss før man setter tekstil på objektet. Generelt kan man si at det er tekstilene som får objektene til å se bra ut, og det er scriptene som gir objektene i liv. Scripts blir gjennomgått i avsnitt 12.2.4. For å finne en tekstil som passer, har man mange ulike fremgangsmåter. I utgangspunktet er en tekstil en vanlig grafikkfil i JPG format. Hver avatar har et lite utvalg av standard tekstiler i inventarmappen. Hvis man blir lei av disse, kan man benytte andre. Det er mulig å benytte bilder man har tatt, eller funnet online, og laste dem opp via valgmenyen. Det billigste alternativet er å finne dem ved hjelp av de innebygde søkemulighetene. Noen steder må man betale litt for dem, mens andre steder er de gratis. Prototypen har i hovedsak blitt laget ved hjelp av fritt tilgjengelige tekstiler. Når man vil legge en tekstil, eller farge, på et objekt, velger man objektet, for deretter å trykke på editer. Da får man opp menyen som vist i figur 42.



Figur 42 - Legge på tekstil og farge på objekt

Her kan man se at tekstilen blir lagt på objektet. Den tekstilen som ble valgt her, var en glassflate med treramme. Bordet blir dermed et glassbord med treramme. Her er det bare fantasien som setter grenser. Ved å velge en annen tekstil, som tremønster eller noe i metall, ville bordet sett annerledes ut.

12.2.4 Scripts

Som nevnt tidligere er det scriptene som får liv til objektene. Second Life bruker sitt eget scriptspråk, som de kaller Linden script språk (Heaton 2007). Dette språket må mestres før man kan gi objektene funksjonalitet. Her er det en fordel å gå til anskaffelse av en bok om scripting, slik det ble i denne studien. Det gikk mye tid i studien med til å scripte ulik funksjonalitet til objektene i prototypen. For å komme opp på et nivå der man kan lage avansert funksjonalitet, på lik linje med andre programmeringsspråk, må det øves mye. Ved hjelp av script kan man få en avatar til å danse, rekke opp hånden på et møte, eller plassere avataren på en stol. Det er ganske komplisert å manipulere avatarer ved hjelp av script. En kompliserende faktor, i tillegg til å plassere selve avataren i forhold til X-, Y-, Z- koordinater, må avataren også roteres riktig i forhold til objektet, som eksempelvis kan være en stol.

For å hjelpe innbyggerne, har det blitt etablert et script bibliotek inne i Second Life, hvor man kan få tilgang til ferdige script for ulike formål. Dette er det nyttig å benytte

seg av, men selv med ferdig utviklede script er dette ikke enkelt. Årsaken er at de stort sett er uten dokumentasjon, og trenger mye tilpassning for å fungere optimalt.

For å benytte script må disse plasseres inn i objekter. I sammensatte objekter, er det master prim som inneholder scriptet. Det er under fanen ”content” at man legger inn, og kjører scriptet fra objektet.

Med disse ferdighetene og kunnskapene startet arbeidet med å designe prototypen, som vist i kapittel 7.

13 Appendiks C - Groupware Walkthrough – Group Task Model

I dette appendikset er det lagt ved senariobeskrivelse og analysediagram for hvert scenario. Alle evaluererne hadde før gjennomgangen av hvert scenario lest grunndig igjennom dette. Gjennomgangene ble utført i henhold til Gutwin og Greenbergs (2002) Groupware Walkthrough prosess, som er gjengitt i figur 43.

For hvert samarbeidsscenario:

· Se over scenario beskrivelsen og analysediagram for å bli kjent med brukerne, ønsket resultat, and omkringliggende omstendigheter.

- **For hver oppgave i scenarioet:**
  - Forsøk å utfør hver deloppgave.
  - Noter deg hvordan det gikk å utføre hver deloppgave.
  - Noter problemer, anta deretter at de er løst, og fortsett.

· **Etter hver oppgave, spør følgende spørsmål:**

- Kan oppgaven bli utført *effectively*—gir brukergrensesnittet mulighet for å utføre oppgaven (og på riktig måte)?
- Kan oppgaven bli utført *efficiently*— hadde gruppen brukt nødvendig innsats for å få dette til?
- Kan den bli utført med *satisfaction*—hadde gruppen vært motivert for å utføre denne oppgaven, og ville de vært fornøyd med utfallet?

- **Etter at oppgavene er utført:**
  - Etter at alle oppgavene er gjennomført, må man vurdere hvorvidt grensesnittet gir mulighet for at gruppen kan oppnå det ønskede målet.

Figur 43 – Stegene i groupware walkthrough, oversatt til norsk fra Gutwin & Pinelle (2002, s. 458)

13.1 Scenario 1 - Daglig Scrum møte

Evalueringen har til hensikt å vurdere hvorvidt møtedeltagerne klarer å gjennomføre sine aktiviteter. Nedenfor vil det bli gitt en stegvis liste med oppgaver (se figur 43) som møtedeltagerne, som beskrevet i tabell 12, må gjennomføre for at møte skal fungere slik det er tiltenkt.

Det antas at prototypen blir benyttet for å arrangere og holde daglig Scrum møte. Denne antagelsen blir tatt fordi det er prototypen som skal evalueres.

For å arrangere møtet kan Scrum Mester undersøke om teammedlemmene er tilgjengelige på to måter. Den første måter er å benytte samarbeidsmekanismen *overvåking* for å se om noen er tilgjengelig for møte. Dette er mulig fordi man som bruker av Second Life har mulighet for å sette status til opptatt, som svever over hodet til avataren. Den andre måten er å ta kontakt med medlemmene ved å benytte samarbeidsmekanismen *eksplisitt kommunikasjon* ved å spørre dem direkte. Her kan man benytte VoIP, IM, eller man kan sende et notat (litt à la e-post) til medlemmene. Medlemmene svarer så tilbake om de er tilgjengelige. Når Scrum Mester vet hvem som er tilgjenglig, kan han eller hun be om møte. Her kan hver enkelt akseptere eller avslå forespørselen.

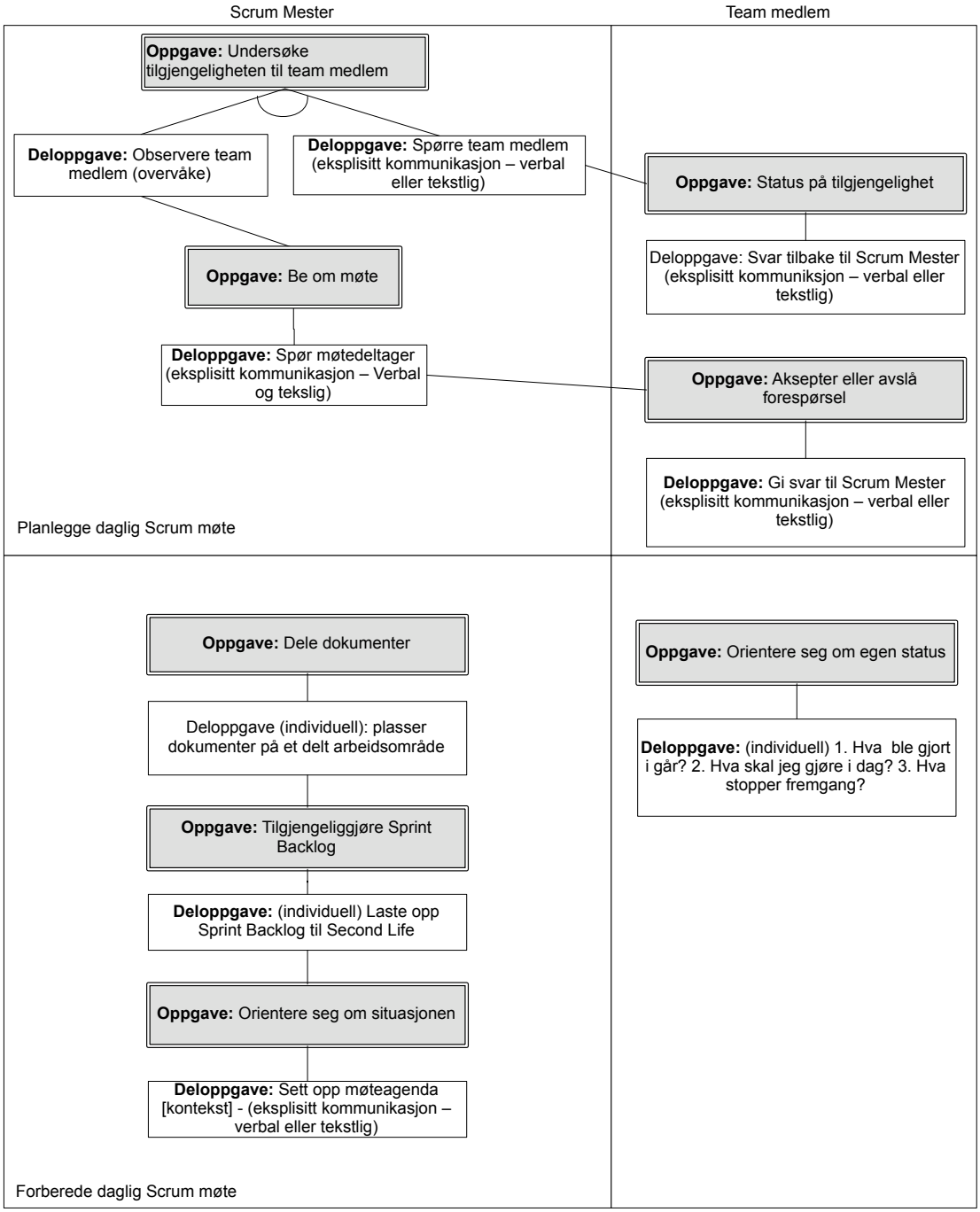
Før møtet starter må Scrum Mester tilrettelegge for møtet. Han eller hun må, i den forbindelse, laste opp Sprint Backlog og presentasjonen han eller hun skal holde til Second Life. Videre orienterer Scrum Mester seg om prosjektstatus. Her kan han eller hun hente inn informasjon både tekstlig og muntlig fra de andre i organisasjonen. Før møte er det også viktig at teammedlemmene orienterer seg om sin egen status, slik at de kan presentere denne på møte.

Når møtet starter i møterommet i prototypen har Scrum Mester fire konkrete oppgaver, hvor rekkefølgen tilpasses etter situasjonen: [1] Innhente status fra møtedeltagerne, hvor de skal fortelle hva som ble gjort i går, hva som skal gjøres i dag, og hva som eventuelt hindrer dem fra å oppnå fremdrift. Møtedeltagerne kan svare verbalt eller tekstlig. [2] Orientere om status på Sprint Backlog (se avsnitt 2.6.3.3). Her kan han eller hun peke til spesifikke regioner (gestikulering), og/eller fortelle om dem (eksplisitt kommunikasjon). [3] Ta opp en problemstilling. Her har også teammedlemmene mulighet for å ta opp ulike tema. [4] De tre overnevnte punktene leder gjerne til en diskusjon som det er Scrum Mesters oppgave å lede. Her kan begge parter argumentere for sitt ståsted, men Scrum Mester har det siste ordet.

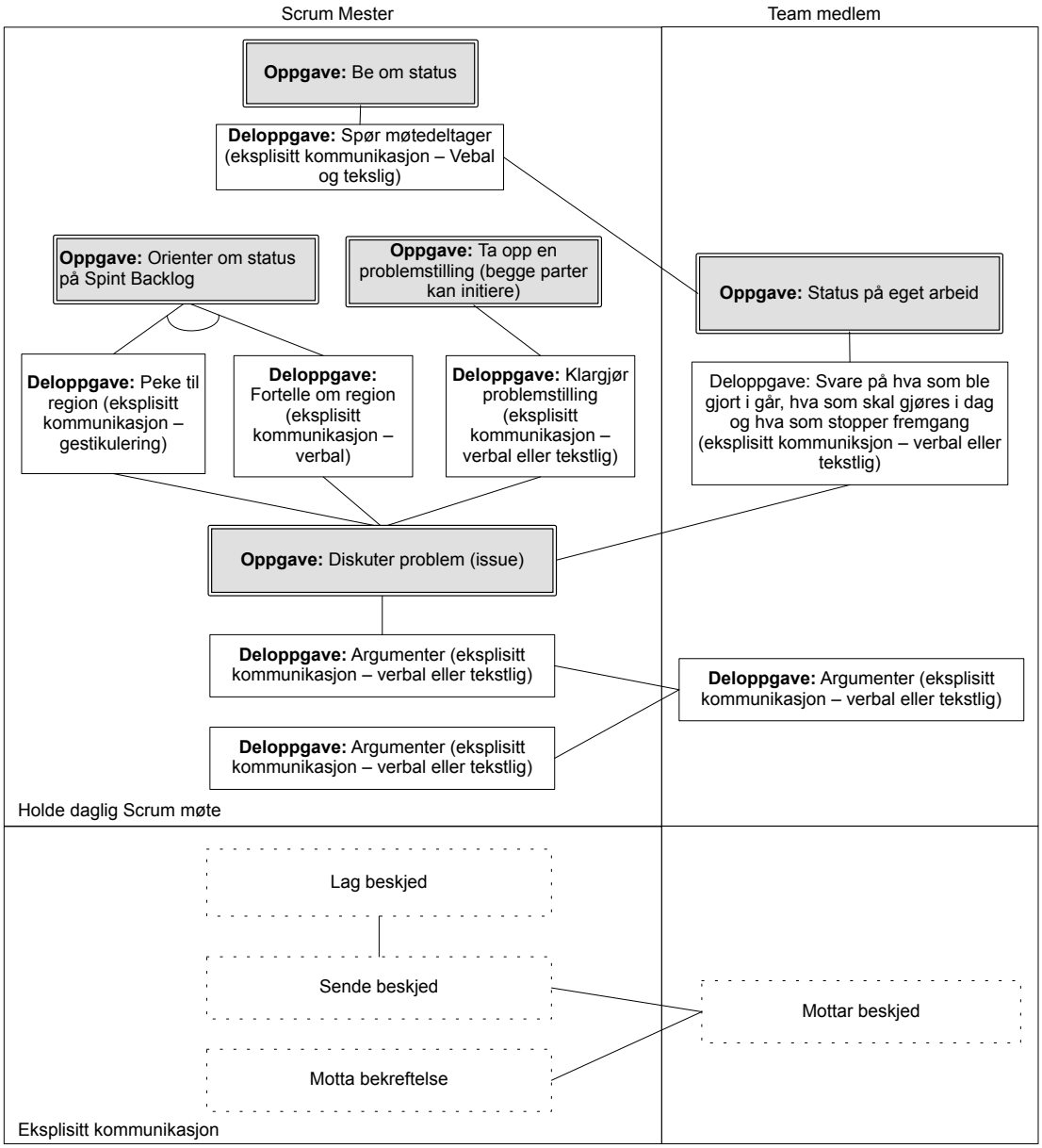
Målet med denne gjennomgangen er å vurdere hvor godt det fungerer å planlegge, forberede, og holde daglig Scrum møte, og hvor godt deltagerne greier å oppfatte hva de andre deltagerne informerer om.

Tabell 12 - Scenariobeskrivelse for daglig Scrum møte

Aktivitetsbeskrivelse. Daglig Standup møtet holdes på IM slik at alle kan delta uavhengig av lokasjon. Møtet holdes i henhold til beskrivelsen gitt i avsnitt 2.6.3, hvor Scrum Mester spør teammedlemmene hva de gjorde i går, hva de planlegger å gjøre i dag, og eventuelt hva som hindrer dem fra å oppnå fremdrift.
Brukerspesifikasjon. Scrum Mester er ansvarlig for å organisere møtet. Han tilrettelegger informasjon, og koordinerer innspillene fra de andre møtedeltagerne.
Brukerspesifikasjon. Teammedlemmer er ansvarlig for å utvikle løsninger i henhold til kravspesifikasjon.
Tilsiktet utfall og resultat. Scrum Mester får oversikt over prosjektstatus, som han kan rapportere til øvrig ledelse og Produkt eier. Han får også synliggjort hindre som kan stoppe prosjektfremdrift. Team medlemmene får større innsikt i helheten til prosjektet, og hvordan deres innsats bidrar til at prosjektet kan lykkes. Ved at hindre blir identifisert, kan disse løses tidlig, før de skaper problemer for leveransen.
Forhold/omstendigheter. Teammedlemmer og Scrum Mester deltar fra ulike steder. Som nevnt tidligere er det vanlig å utkontraktere utviklingstjenester til lavkostland, mens Scrum Mester og øvrig ledelse befinner seg i hjemlandet til firmaet.



Figur 44 – Analysediagram for scenario 1: Holde møte, del 1



Figur 45 – Analysediagram for scenario 1: Holde møte, del 2

I dette avsnittet har de nødvendige elementene for å gjennomføre en groupware walkthrough prosess for et Scrum møte blitt presentert. I neste avsnitt presenteres tilsvarende informasjon for scenario 2, holde kurs.

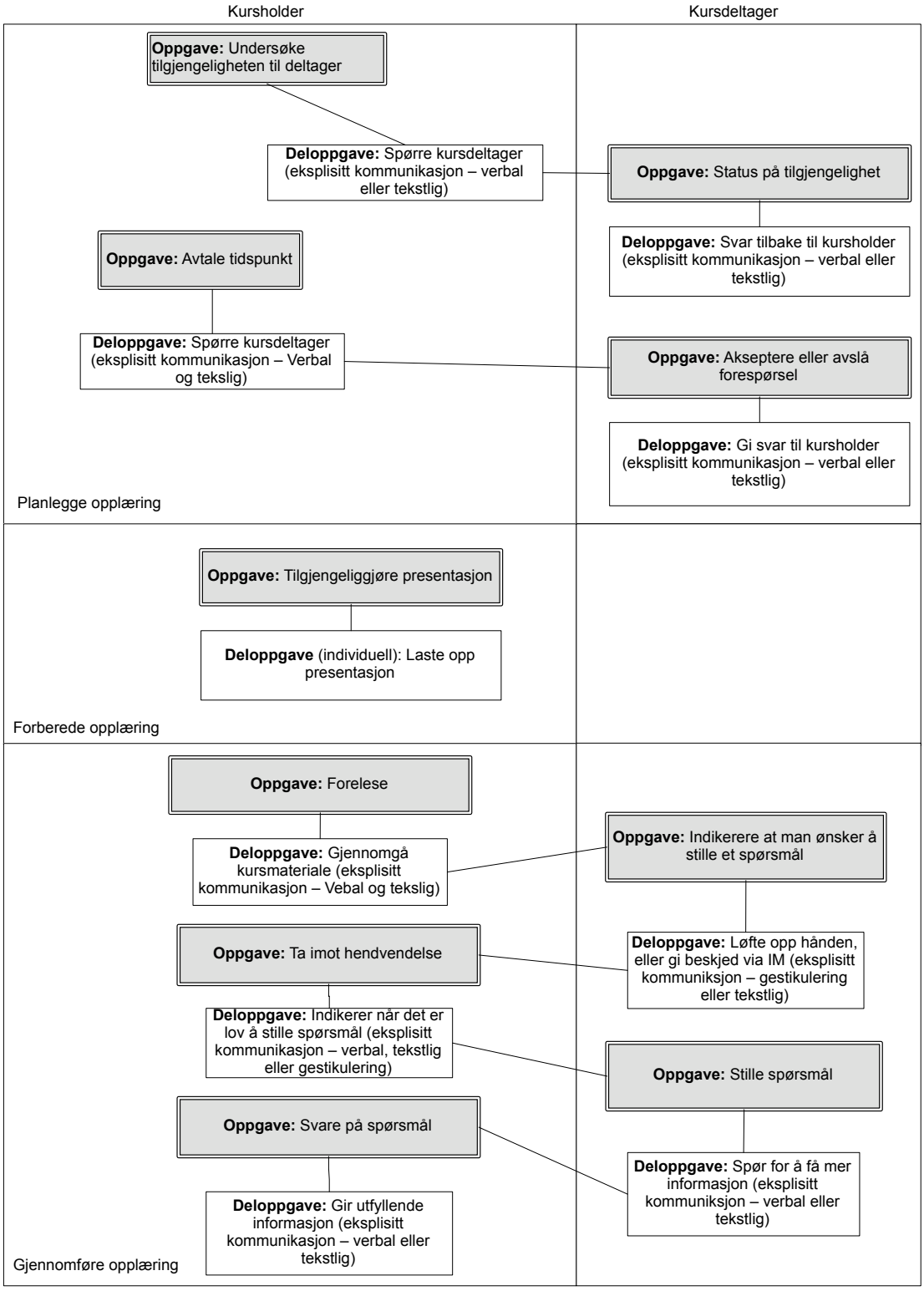
13.2 Scenario 2 - Holde kurs

Selve innkallingen foregår på samme måte som når man holder et møte, men fordi kursholder kanskje ikke er en av teamet baserer han eller hun seg på *eksplisitt kommunikasjon*. Dermed benytter han eller hun ikke samarbeidsmekanismen *overvåking*. I forkant av møtet må han eller hun laste opp presentasjonen som skal holdes. I løpet av selve opplæringen er det mulighet for kursdeltagerne til å stille avklarende spørsmål ved hjelp av samarbeidsmekanismen *eksplisitt kommunikasjon*.

Kursdeltagerne kan indikere at de ønsker å stille spørsmål ved å gestikulere eller sende en IM til foredragsholder. Kursholderen gir tegn til kursdeltager når han eller hun kan stille sitt spørsmål. Kursdeltager stiller spørsmålet sitt verbalt eller tekstlig, og får svar tilbake.

Tabell 13 - Scenariobeskrivelse for å holde et kurs

Aktivitetsbeskrivelse. Kursholder undersøker tilgjengelighet til kursdeltagere. Kursdeltagere svarer tilbake om når han er tilgjengelig for kurs. Kursholder avtaler så tidspunkt med kursdeltagere. Kursholder gjør så presentasjonen tilgjengelig før kurset starter. Han holder så kurset, hvor kursdeltagere kan stille spørsmål underveis.
Brukerspesifikasjon. Kursholder skal formidle kunnskap til teammedlemmene. Han kan være innleid eller en ressursperson fra bedriften.
Brukerspesifikasjon. Kursdeltager kan være medlem av teamet, ledelsen, eller andre samarbeidspartnere.
Tilsiktet utfall og resultat. Kunnskapsformidling til kursdeltagerne.
Forhold/omstendigheter. Kursholder og kursdeltagerne befinner seg i et vanlig kurslokale. Det blir holdt respektive kurs på ulike lokaliteter. Kursholder kommuniserer med kursdeltagerne ved hjelp av IM, VoIP og telefon (synkron) eller e-post (asynkron).



Figur 46 – Analysediagram for scenario 2: Holde kurs

I dette avsnittet har de nødvendige elementene for å gjennomføre en groupware walkthrough prosess for å holde et kurs, blitt presentert. I neste avsnitt presenteres tilsvarende informasjon for scenario 3, diskutere kildekode.

13.3 Scenario 3 - Diskutere kildekode

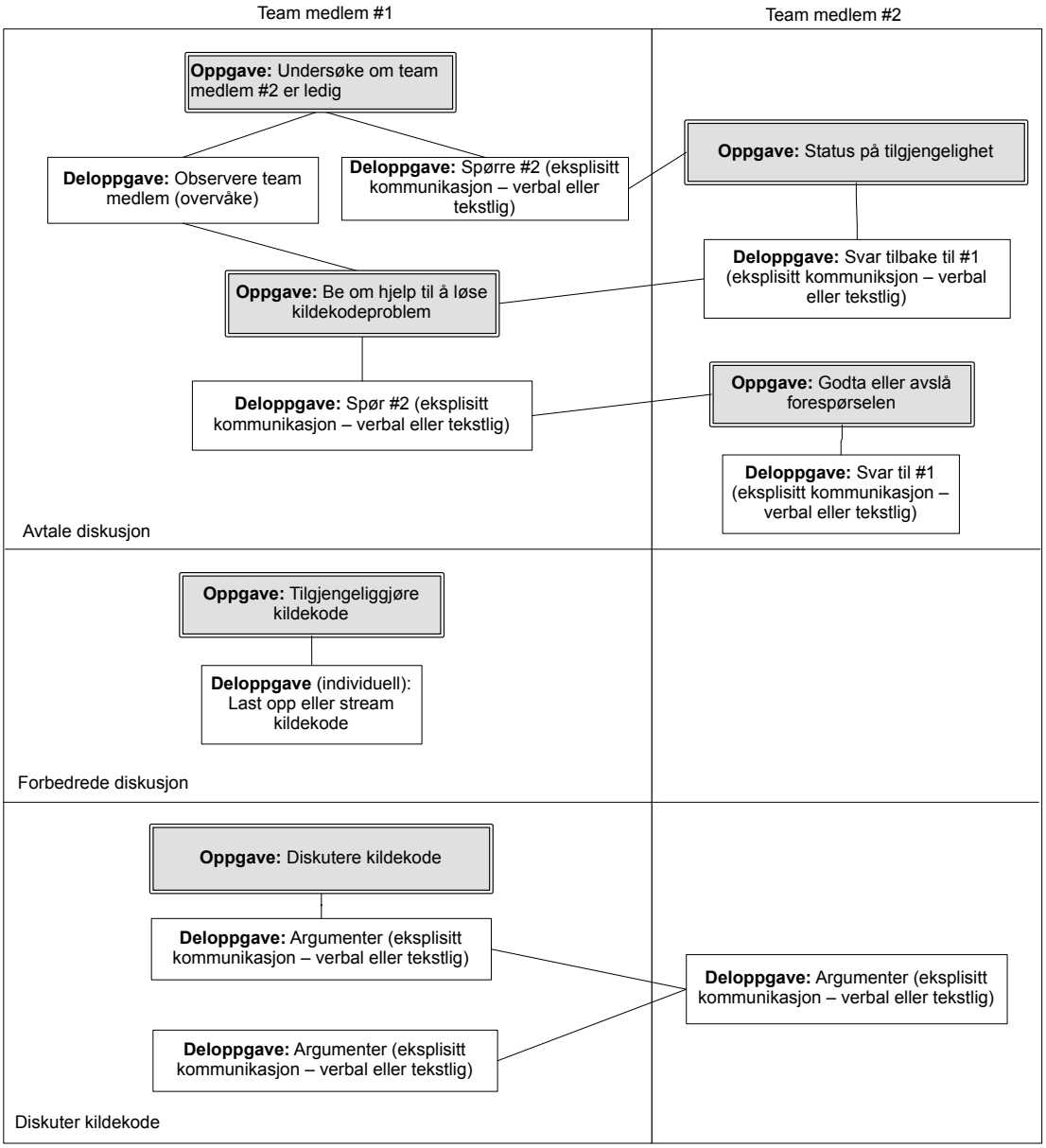
For å arrangere diskusjonen må #1 undersøke om #2 er tilgjengelig. Han kan ta kontakt ved å spørre direkte i Second Life ved hjelp av IM eller VoIP. Et siste alternativ er å sende et notat (litt à la e-post). #1 svarer deretter tilbake om han eller hun er tilgjengelig. Her har man også mulighet for å observere om #2 er tilgjengelig hvis han eller hun er pålogget prototypen. Når #1 vet at #2 er tilgjenglig, kan han eller hun be om møte. Her kan hver enkelt akseptere eller avslå forespørselen.

Før diskusjonen tar til i prototypen må #1 tilgjengeliggjøre kildekoden. Her har man to muligheter for å vise kildekoden til #2. #1 kan enten laste opp kildekoden som et bilde, for så å vise denne til #2, eller streame skjermbilde med koden inn i Second Life (slik som vist i figur 24).

Diskusjonen om kildekoden kan foregå via IM eller VoIP. Det er vanlig at man har flere slike korte diskusjoner i løpet av en dag, slik det kom frem i den kvalitative undersøkelsen (se kapittel 6).

Tabell 14 - Scenariobeskrivelse for å diskutere kildekode

Aktivitetsbeskrivelse. Team medlem #1 ønsker å diskutere kildekode med et annet team medlem . Han må først få tak i team medlem #2, og høre om han eller hun har tid til å bistå. Deretter må han spesifisere hvilken kode han ønsker å diskutere. Når dette har blitt klargjort, diskuterer #1 og #2 den aktuelle koden.
Brukerspesifikasjon. Teammedlemmer er ansvarlig for at programutviklingsarbeidet går så smidig som mulig. En faktor for å oppnå dette er å bistå hverandre ved å dele kunnskap.
Tilsiktet utfall og resultat. Meningsutveksling om koden.
Forhold/omstendigheter. De to teammedlemmene arbeider fra ulike steder. De benytter e-post til asynkron kommunikasjon, samt IM, telefon og VoIP for synkron kommunikasjon. De utveksler kode via programutviklingsverktøy til kodebasen.



Figur 47 – Analysediagram for scenario 3: Diskutere kildekode

I dette avsnittet har de nødvendige elementene for å gjennomføre en groupware walkthrough prosess for diskutere kildekode, blitt presentert. I neste avsnitt presenteres tilsvarende informasjon for scenario 4, parprogrammering.

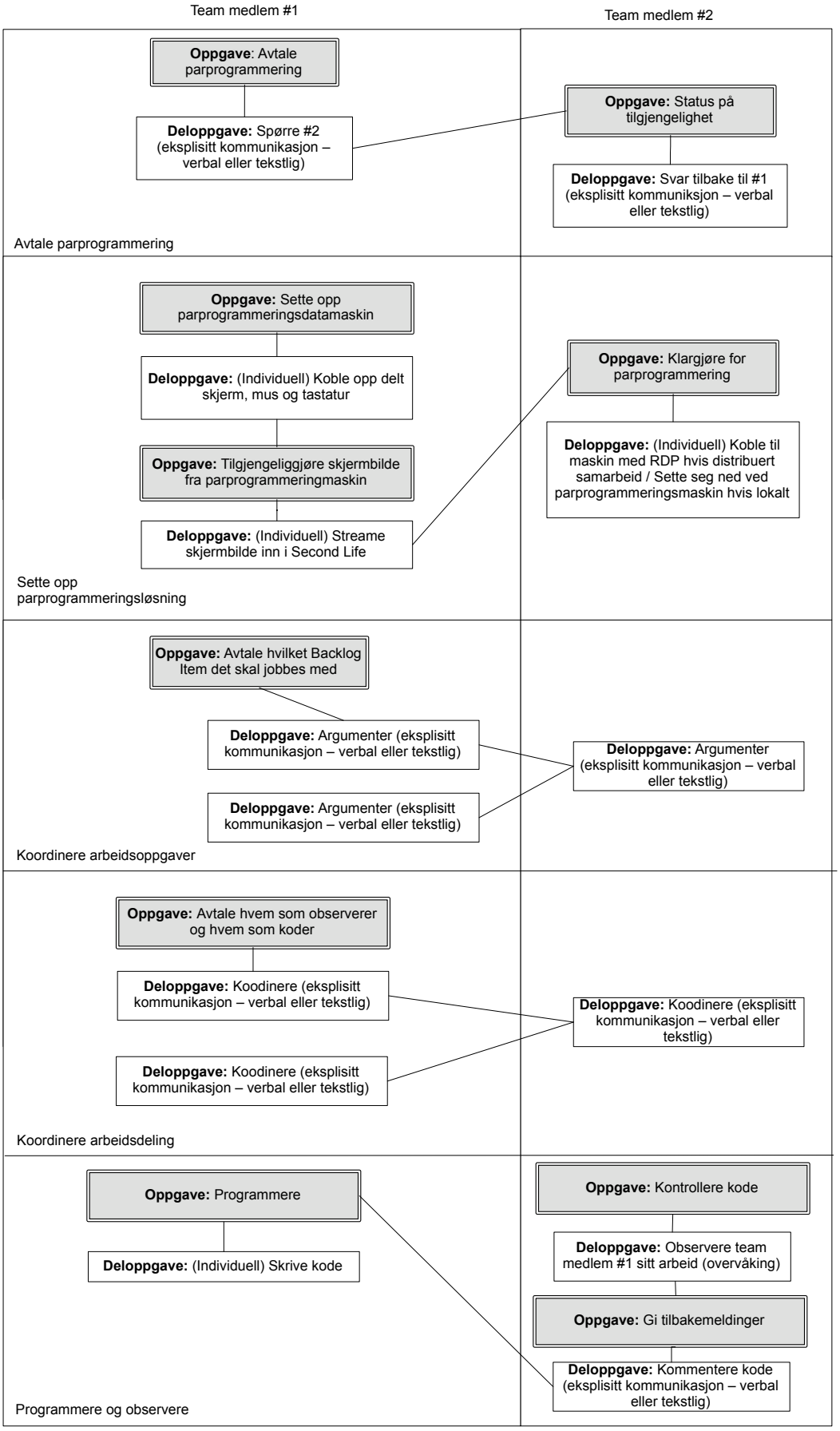
13.4 Scenario 4 – Parprogrammering

#1 undersøker om #2 er tilgjenglig for å parprogrammere, ved å spørre direkte ved IM eller VoIP. #2 gir så svar tilbake om han eller hun er tilgjengelig eller ikke. Videre må den tekniske løsningen for parprogrammering settes opp. Dette kan enten gjøres på samme fysiske maskin med to tastaturer, to mus, og to skjermer, eller det kan settes opp en distribuert løsning med Remote Desktop Protokoll slik IKTBergen hadde gjort med Ukraina kontoret. Etter at maskinen har blitt konfigurert, tilgjengeliggjør #1 skjermbildet i prototypen. #2 må så koble seg til parprogrammeringsmaskinen ved å benytte RDP hvis det skal arbeides distribuert, eller sette seg ned ved parprogrammeringsmaskinen hvis begge befinner seg på samme fysiske lokasjon.

Hvis #2 er tilgjengelig kan #1 avtale med #2 hvilket Backlog Item det skal arbeides med. Her kan begge parter argumentere for hva de mener er viktigst. Før arbeidet starter, ber #1 om å få klargjort hvem som skal ha hvilken rolle. Her diskuterer #1 og #2 hvem som skal kode og hvem som skal være observatør. Det er vanlig at man bytter disse rollene flere ganger i løpet av økten.

Tabell 15 - Scenariobeskrivelse for parprogrammering

Aktivitetsbeskrivelse. Team medlem #1 spør team medlem #2 direkte om han eller hun er tilgjengelig for parprogrammering. #2 svarer så tilbake om han eller hun er tilgjengelig. #1 setter så opp parprogrammeringsmaskin, som #2 kobler seg til. Teammedlemmene må så avtale hvilket Backlog Item det skal arbeides med. Videre må det avtales hvem som skal observere og hvem som skal kode.
Brukerspesifikasjon. Teammedlemmer er ansvarlig for at programutviklingsarbeidet går så smidig som mulig. En faktor for å oppnå dette er å bistå hverandre ved å dele kunnskap.
Tilsiktet utfall og resultat. Forbedret kodekvalitet og kunnskapsdeling.
Forhold/omstendigheter. #1 og #2 arbeider enten på samme eller ulikt sted. De benytter e-post til asynkron kommunikasjon, og IM, VoIP og telefon til synkron kommunikasjon.



Figur 48 – Analysediagram for scenario 4: Parprogrammering

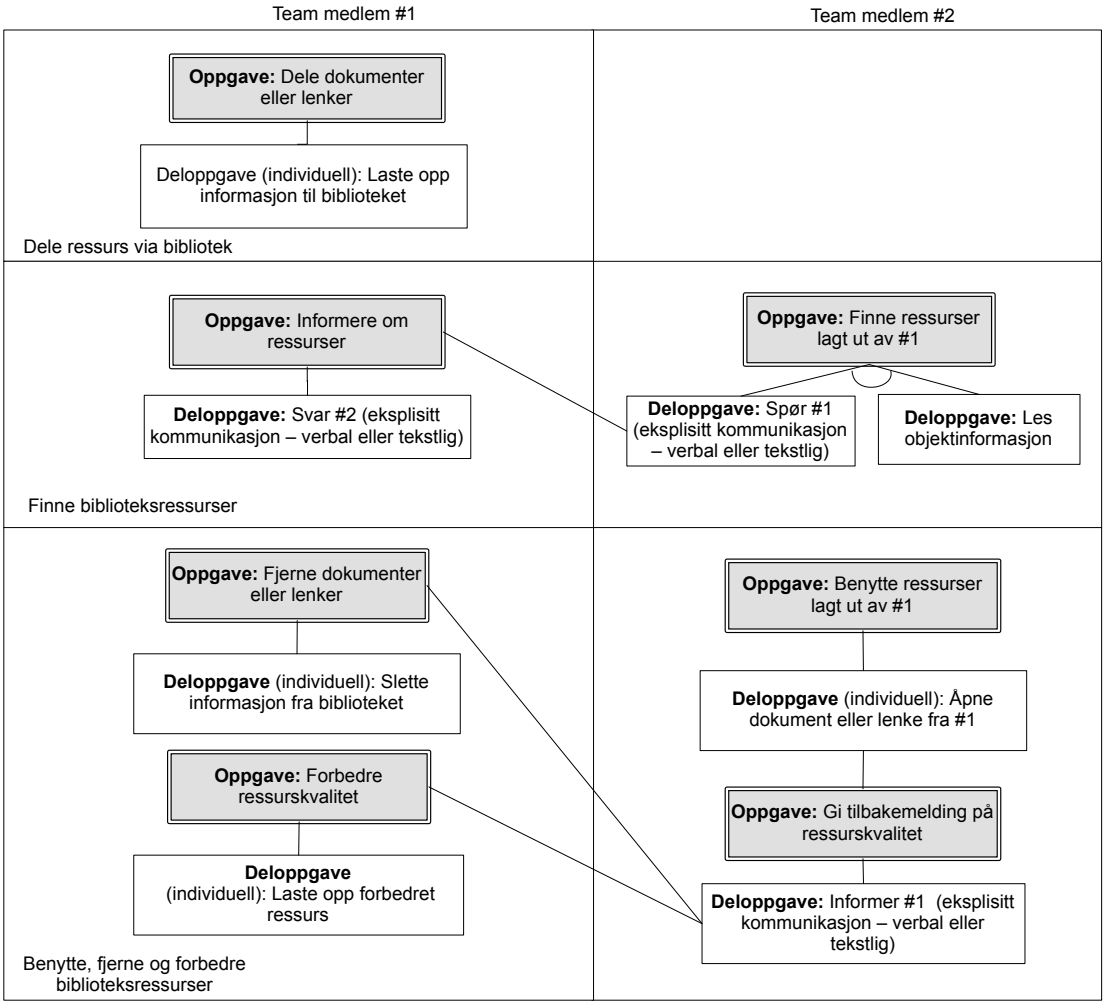
I dette avsnittet har de nødvendige elementene for å gjennomføre en groupware walkthrough prosess for gjennomføre parprogrammeing, blitt presentert. I neste avsnitt presenteres tilsvarende informasjon for scenario 5, dele biblioteksressurs.

13.5 Scenario 5 – Dele biblioteksressurs

Teammedlem #1 deler dokumenter og lenker ved å laste dem opp til biblioteket i samarbeidsrommet. Han eller hun kan så informere #2 om dette hvis det er ønskelig. For å få vite hvilke dokumenter eller lenker som er lastet opp av #1, kan #2 lese informasjonen som er knyttet til hvert informasjonsobjekt, eller spørre #1 hva han eller hun har lagt ut. #2 benytter så dokument eller lenke lagt ut av #1, ved å åpne dette i MUVEet. #2 gir så tilbakemelding til #1 på ressurskvalitet, hvorpå #1 kan velge å forbedre eller fjerne et dokument eller en lenke.

Tabell 16 - Scenariobeskrivelse for å dele biblioteksressurs

Aktivitetsbeskrivelse. Team medlem #1 har vært på et eksternt kurs og fått med seg nyttig kursmateriale tilbake. Dette ønsker han eller hun å dele med #2. Først må #1 ta kontakt med #2 for å finne ut hvordan de skal utveksle informasjonen. På forhånd er det nyttig å vite at #2 ikke har fått denne informasjonen fra før. Informasjonen blir så utvekslet etter avtale.
Brukerspesifikasjon. Teammedlemmer er ansvarlig for at programutviklingsarbeidet går så smidig som mulig. En faktor for å oppnå dette er å bistå hverandre ved å dele kunnskap.
Tilsiktet utfall og resultat. Utveksle informasjonsressurser.
Forhold/omstendigheter. #1 og #2 arbeider enten på samme eller ulik lokasjon. De benytter e-post til asynkron kommunikasjon, og IM, VoIP og telefon til synkron kommunikasjon.



Figur 49 – Analysediagram for scenario 5: Dele biblioteksressurs

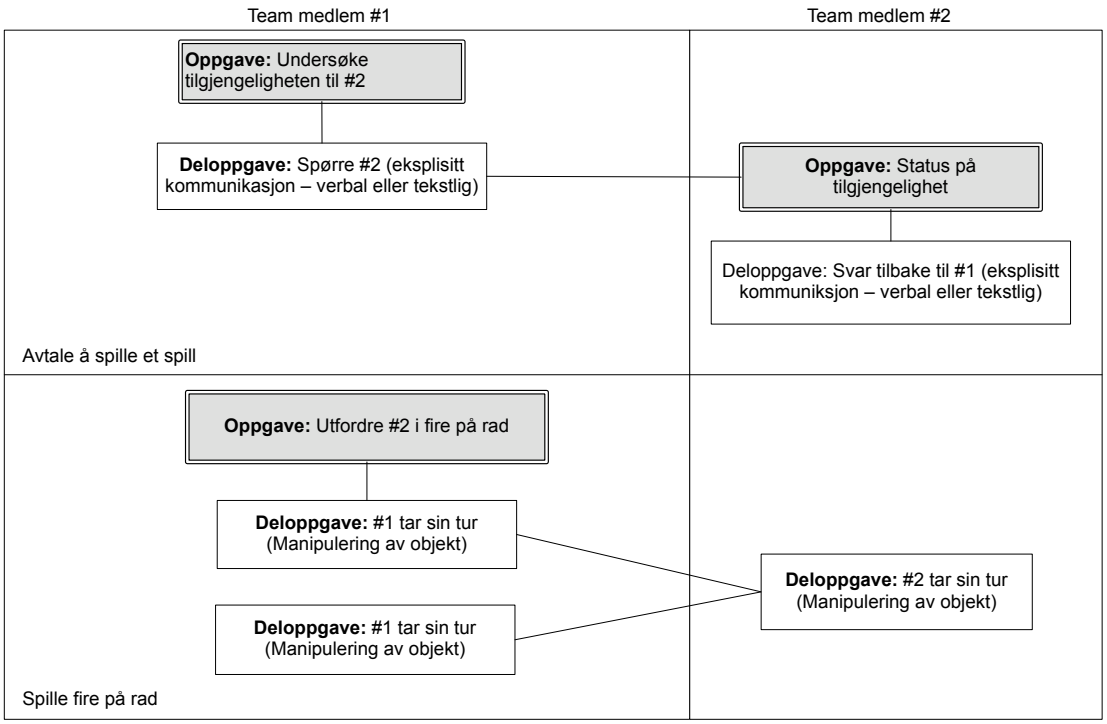
I dette avsnittet har de nødvendige elementene for å gjennomføre en groupware walkthrough prosess for å dele en biblioteksressurs, blitt presentert. I neste avsnitt presenteres tilsvarende informasjon for scenario 6, spille spill.

13.6 Scenario 6 - Spille spill

Teammedlem #1 undersøker om teammedlem #2 er tilgjenglig for å spille en omgang med fire på rad, ved å spørre direkte på IM eller VoIP. #2 gir så svar tilbake om han eller hun er tilgjengelig eller ikke. Hvis #2 er tilgjengelig kan #1 utfordre #2 til å spille fire på rad. Hver spiller tar en tur hver, til spillet er vunnet av #1 eller #2.

Aktivitetsbeskrivelse. Team medlem #1 undersøker om team medlem #2 har tid til å spille et spill, ved å spørre direkte. #2 svarer om han eller hun er tilgjengelig. #1 utfordrer så #2 til å spille et spill. De spiller så spillet til en vinner.
Brukerspesifikasjon. Teammedlemmer er ansvarlig for at programutviklingsarbeidet går så smidig som mulig. En faktor for å oppnå dette er å bistå hverandre ved å dele kunnskap.
Tilsiktet utfall og resultat. Rekreasjon, sosialisering og teambygging
Forhold/omstendigheter. #1 og #2 arbeider enten på samme eller ulik lokasjon. De benytter e-post til asynkron kommunikasjon, og IM, VoIP og telefon til synkron kommunikasjon.

Tabell 17 – Scenariobeskrivelse for å spille et spill



Figur 50 – Analysediagram for scenario 6: Spille spill

I dette avsnittet har de nødvendige elementene for å gjennomføre en groupware walkthrough prosess for å spille spill, blitt presentert.