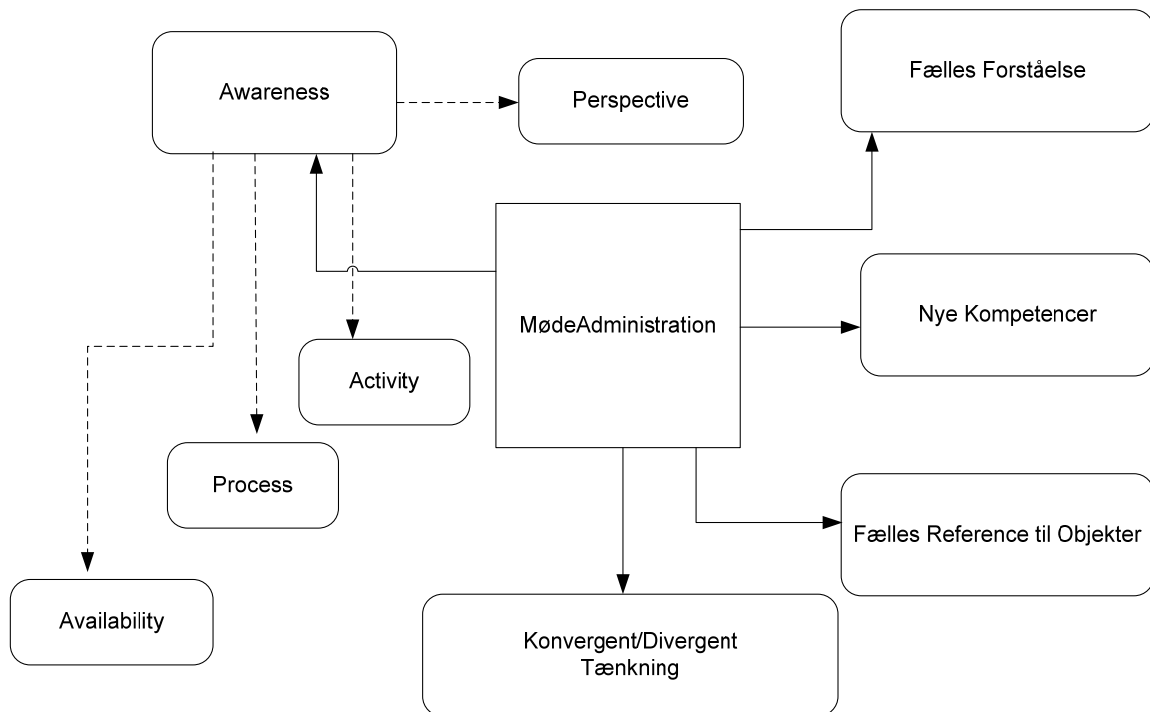


Distribueret Samarbejde

Design af værktøj til distribuerede gruppemøder



Projektgruppe: Anders Lorentz Hansen, Anders Lykke Nielsen,
Martin Schultz & Kim Sonne

Vejledere: Pernille Bjørn og Mads Rosendahl
Datalogi, efterår 2006

Institut for Kommunikation, virksomhed og informationsteknologier

Resumé

Den teknologiske udvikling muliggør i større og større grad geografisk distribueret samarbejde, som i trin med den øgede globalisering forbedrer mulighederne for at udnytte de kompetencer, som er at finde ikke kun på tværs af landet, men i hele verden. Dette projekt fokuserer særligt på de problemstillinger en dansk universitetsgruppe kan opleve ved distribuerede møder, og projektet vil give et bud på hvordan man forbedrer de teknologiske muligheder for at afholde synkrone gruppemøder mellem geografisk adskilte studerende på universiteter.

På baggrund af teoretiske overvejelser og to eksperimenter leverer projektet således en kvalificeret diskussion og en række designovervejelser til hvilken funktionaliteter et ”distribueret-gruppemøde-værktøj” bør besidde, for at forbedre de synkrone gruppemøder. Samtidig har projektgruppen udviklet en prototype på et sådan mødeadministrationsværktøj, for at opnå den erkendelse det giver at gå fra design til implementering. Derudover vil prototype, i forbindelse med eksperiment 2, blive brugt til at teste i hvor stor grad værktøjet er i stand til at tilføje en ny dimension til distribuerede møder.

Rapporten afsluttes med en konklusion på i hvilke grad og hvordan projektgruppen ser distribuerede gruppemøder forbedret ved hjælp af det udviklede design.

Indholdsfortegnelse

<u>1</u>	<u>PROBLEMFELT</u>	5
1.1	PROBLEMFORMULERING.....	7
<u>2</u>	<u>CASE BESKRIVELSE OG AFGRÆNSNING</u>	7
2.1	GRUPPEPROCES OG RESULTAT	9
2.2	METODER	11
2.3	OMGIVELSER/MILJØ.....	11
2.4	AFGRÆNSNING	11
<u>3</u>	<u>METODE</u>	13
<u>4</u>	<u>UDFORSKENDE EKSPERIMENT (EKSPERIMENT 1)</u>	16
4.1	FORSØGSRAMMEN	16
4.2	MØDEFORLØB	17
4.3	MØDEPRODUKT OG EVALUERING AF MØDET	18
4.3.1	RESULTAT AF FORSØGET	19
<u>5</u>	<u>TEORI</u>	20
5.1	PROBLEMATIK VED DISTRIBUTUEDE GRUPPEMØDER	20
5.1.1	FÆLLES MÅLSÆTNINGER.....	20
5.1.2	FÆLLES FORSTÅELSE.....	21
5.2	HØJ OG LAV KOBLING	23
5.2.1	KREATIVE KOMPLICEREDE ARBEJDSPROCESSER MED HØJ KOBLING	23
5.2.2	DIVERGENT & KONVERGENT TÆNKNING	24
5.2.3	REFLEKSIVITET	25
5.3	DOUBLE LEVEL LANGUAGE – FORMEL OG KULTUREL KOMMUNIKATION	25
5.4	AWARENESS – INFORMATIONSTRØMNINGER I SYSTEMET	26
5.4.1	FIRE FORSKELLIGE AWARENESS TYPER	26
5.4.2	PASSIV ELLER AKTIV AWARENESS	27
5.4.3	EXPLICIT VS. EMBEDDED	28
5.4.4	ACCESS ANYWHERE	28
<u>6</u>	<u>OPSAMLING</u>	30
6.1	ERFARINGER I FORHOLD TIL TEORIEN.....	30
6.2	FÆLLES REFERENCER I OBJEKTER.....	31
6.3	MØDEADMINISTRATIONSVÆRKTØJET SOM EN ”NY KOMPETENCE”	32

<u>7</u>	<u>SAMLET EMS DESIGN.....</u>	<u>33</u>
7.1	SKYPE	35
7.2	BSCW.....	36
7.3	MESSENGER	37
7.4	DIAGRAMMERINGSVÆRKTØJET	37
7.5	PLANLÆGNINGSVÆRKTØJET	38
<u>8</u>	<u>MØDEADMINISTRATIONS DESIGN</u>	<u>40</u>
8.1	GENNEMGANG AF MØDEADMINISTRATIONSVÆRKTØJET	41
8.2	INTEGRATIONEN MED PLANLÆGNINGSVÆRKTØJET.....	47
8.3	SYSTEMBESKRIVELSE.....	48
<u>9</u>	<u>FOKUSERET EKSPERIMENT (EKSPERIMENT 2)</u>	<u>50</u>
9.1	FORSØGSRAMMEN	50
9.2	MØDEFORLØB	51
9.3	EVALUERING AF DET FOKUSEREDE EKSPERIMENT	52
9.3.1	OPLEVELSE AF DET FOKUSEREDE EKSPERIMENT	52
9.3.2	PROBLEMER.....	52
9.3.3	FORDELE	53
9.3.4	KONKLUSION PÅ DET FOKUSEREDE EKSPERIMENT	53
<u>10</u>	<u>KONKLUSION.....</u>	<u>54</u>
<u>11</u>	<u>LITTERATURLISTE</u>	<u>57</u>
<u>12</u>	<u>BILAG.....</u>	<u>59</u>

1 Problemfelt

Der har i mange år været forsket i, hvordan grupper kan samarbejde på trods af at gruppemedlemmerne er fysisk adskilt. Problemstillingen er mere og mere aktuell set i forhold til den nuværende samfundsudvikling, hvor globaliseringen og den øgede brug af eksperter nødvendiggør samarbejde på tværs af geografiske placeringer. Udviklingen inden for IT og Internettet har åbnet op for nye muligheder for at understøtte sådanne typer af distribuerede samarbejder, blandt andet virtuelle teams.

Både på uddannelsesinstitutioner, i erhvervslivet og på Internettet ser man mange eksempler på sådanne virtuelle teams, der samarbejder selvom de er fysisk adskilt – og måske aldrig har mødt hinanden.

På trods af den teknologiske udvikling er der dog stadig en række problemer med at få distribuerede samarbejder og virtuelle møder til at fungere optimalt, og derfor er det ofte på grund af nød og ikke af lyst, at man vælger at afholde distribuerede gruppemøder.

Kommunikationen i distribuerede samarbejde kan opdeles i to typer ud fra tidsmæssige forskelligheder. De har hver deres fordele og ulemper.

- Synkron, hvor gruppemedlemmerne kommunikerer simultant – ligesom i en ”face-to-face” samtale. Det kunne f.eks. være ved ”instant messaging”, lyd eller video konferencer.
- Asynkron, hvor gruppemedlemmerne kommunikerer med en tidsforskel. Det kunne f.eks. være på e-mail eller internetforum, hvor man skiftes til at skrive til hinanden.

Der findes en række værktøjer til at understøtte samarbejde ved hjælp af synkron og asynkron kommunikation, og disse værktøjer har i høj grad vist sig effektive i mange situationer. Hvem kan f.eks. forestille sig et distribueret samarbejde uden brug af email eller fildeling, der begge er asynkrone? I mange tilfælde har den asynkrone digitale kommunikation endda vist sig mere effektiv end den gamle ”manuelle” kommunikationsform. Endnu engang er e-mail et godt eksempel. Men asynkron kommunikation er langt fra optimal i alle situationer. I mere komplekse og kreative

situationer, hvor en gruppe skal forhandle omkring vigtige beslutninger, blive enige om konklusioner eller brainstorme, er asynkron kommunikation uegnet, da informationsflowet går for langsomt, og den kreative proces dermed risikerer at gå i stå. Synkron kommunikation er derfor en klar forudsætning for at undgå dette. Af alment kendte synkrone samarbejdsværktøjer i dag har vi "instant messaging" samt video- og lydkonferencer, men der er stadig en række problemer ved brugen af disse værktøjer.

I samarbejdssituationer vil synkrone og asynkrone værktøjer oftest være kombineret, hvor man måske har videokonferencer og instant messaging kombineret med email og versionsstyringsværktøjer. Den slags samarbejdsværktøjer kendes blandt andet som Elektroniske Møde Systemer (EMS). Elektroniske mødesystemer understøtter virtuelle beslutningsprocesser og samarbejder med både synkrone og asynkrone værktøjer, men har også mere produktorienterede gruppe arbejdsfunktioner understøttet [Dennis, *et al.* 1988]. Vi vil derfor bruge betegnelsen EMS som dækkende for en programpakke, der understøtter det virtuelle gruppearbejde ved netop at dække et antal kommunikative kanaler, for eksempel instant messaging.

For distribuerede møder generelt forekommer der et problem i og med at samarbejdspartnerne har sværere ved at læse hinandens signaler end i face-to-face situationer. Det er f.eks. kropssprog og gestikulation der mangler. Det betyder, at samarbejdspartnerne ikke har optimale kår for at skabe tillid til hinanden, og misforståelser opstår nemmere. I en diskussion i et virtuelt team kan det også være sværere at styre samtalen, vide hvornår en person er færdig med at tale, hvornår en anden gerne vil tilføje noget osv.

Alle disse faktorer betyder tilsammen, at diskussioner i distribuerede grupper oftest ikke er lige så effektive, som diskussioner hvor personerne sidder fysisk samme sted. Effekten af dette er, at en fælles forståelse i forhold til problemerne er sværere at opnå og produktet derfor typisk lider under dette.

Som studerende har vi ofte oplevet behovet for at afholde distribuerede møder, men er oftest nået til den erkendelse at softwareudvalget har været for begrænset til at understøtte det tilstrækkeligt.

Disse problemstillinger har ledt os frem til følgende problemformulering.

1.1 Problemformulering

Hvordan kan man forbedre de teknologiske muligheder for at afholde synkron gruppemøder mellem geografisk adskilte studerende på universiteter?

2 Case beskrivelse og afgrænsning

For at holde fokus gennem vores arbejde i projektet, har vi valgt at arbejde ud fra en case. Denne case har primært det formål at beskrive projektets målgruppe og sekundært at fungere som empiri. Casens mål er derfor at være en reference for vores produkt, således at vi hele tiden kan visualisere vores målgruppe og holde vores beslutninger op imod denne.

Vores tænkte case består af en gruppe universitetsstuderende fordelt på forskellige universiteter i Danmark. Vi forventer at gruppen har en nogenlunde ens faglig baggrund, men ikke nødvendigvis kender hinanden i forvejen. Gruppen kunne for eksempel bestå af datalogistuderende fra RUC, SDU og Århus universitet der samarbejder i et projekt. Vi har valgt dette grundlag ud fra en tese om, at når casen består af universitetsstuderende fra Danmark, med nogenlunde samme faglighed og kulturelle baggrund, har vi begrænset os fra også at skulle overveje de problematikker og udfordringer, der særligt opstår hvis gruppemedlemmerne er meget forskellige – kulturelt såvel som fagligt og sprogligt.

I forhold til vores konkrete projekt ser vi, at der må være behov for en programsammensætning, som indeholder de fornødne værktøjer til distribueret kommunikation, da det formodes at de studerende ikke har mulighed for at mødes fysisk ved alle møder igennem deres projektforsløb. Selv hvis muligheden er der, vil en optimal

programsammensætning kunne spare de involverede parter for unødigt og tidskrævende transport.

Det er tiltænkt i så vid en udstrækning som muligt, at anvende teknologier og programmer som de involverede parter allerede kender. Dette tilstræbes for at gøre programpakken lettere at tilgå, samt at spare udviklingsomkostninger.

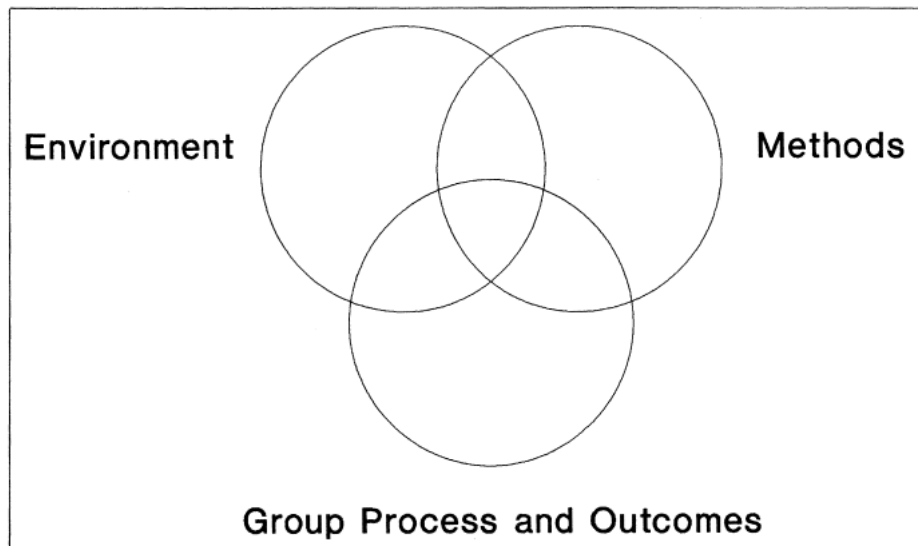
Vores teoretiske fundament for programsammensætningen er at finde i bl.a. [Dennis, *et al.* 1988] der anvender betegnelsen elektronisk mødesystem (EMS) for denne type systemer.

I denne artikel påpeger forfatterne at der i et EMS er tre forskellige koncepter der skal tages stilling til. Vi vil præsentere disse koncepter og bruge dem i den udstrækning de kan være med til at forme vores case. Det er dog ikke vores hensigt at beskrive den bagvedliggende teori, men blot at benytte den til at kvalificere vores valg. Dette gør vi i høj grad for at præcisere i hvilken kontekst vores projekt skal læses, og derigennem at lave en naturlig afgrænsning for projektet.

De tre områder er:

- Gruppeproces og resultat
- Metoder
- Omgivelser/miljø

Disse områder hænger sammen og påvirker hinanden. Hvis der vælges en særlig proces og en særlig metode, har det betydning for hvad der er af mulige omgivelser – og omvendt. Det vil sige, at man ifølge modellen, ved at specificere to af områderne, automatisk begrænser det tredje. Det er altså ikke muligt at undgå at forholde sig til alle områderne.



Figur 1: Konceptuel model [Dennis, *et al.* 1988]

Alle disse faktorer har betydning for hvordan vores system skal designes, for derigennem bedst muligt at kunne understøtte det beskrevne samarbejde.

2.1 Gruppeproces og resultat

Forfatterne opstiller en række områder der skal tages stilling til i forhold til gruppen. Vi vil nu benytte den foreslåede oversigt som en skabelon for at fastsætte detaljerne i vores case og vælger punkter indenfor hvert område, som er relevante for vores projekt og derfor vores case. Af samme årsag vælger vi kun at benytte de tre første kategorier.

- **Gruppen**

- *Gruppestørrelse:*

En gruppe på universitetet der skal lave et projekt vil typisk være på tre til seks personer.

- *Formel/uformel:*

Mødesituationerne er uformelle med en planlagt dagsorden som i videst mulig udstrækning bliver fulgt.

- *Samarbejdsperiode:*

Projektet varer et semester, men casen strækker sig over et gruppemøde.

- *Tidligere erfaringer:*

Gruppen har ikke nødvendigvis arbejdet sammen før, men alle er bekendt med den arbejdsform der benyttes.

- *Samhørighedsforhold:*

De er bekendt med hinanden inden mødet, men har ikke nødvendigvis mødt hinanden fysisk. Jo bedre de kender hinanden jo nemmere vil det være for dem at benytte værktøjet.

• **Opgaven**

- *Opgavetype:*

Gruppens endelige mål er at lave en projektrapport. Til det enkelte møde arbejdes i retning af produktet som er en del af selve projektrapporten.

- *Emne:*

Projektet opstiller vi som et datalogisk projekt Det vil sige at projektgruppen er vant til at arbejde med computere.

• **Konteksten**

- *Incitament:*

Gruppens incitament er at denne skal bestå projekteksamen. De enkelte gruppemedlemmer kan godt have forskellige holdninger til hvordan arbejdet skal foregå, men generelt har alle gruppemedlemmerne samme mål.

- *Organisations struktur:*

I en gruppe er de studerende formelt lige. Det betyder, at der altså ikke er nogen der bestemmer mere end andre.

- *Miljø:*

På trods af at de er placeret på forskellige universiteter, forudsætter vi at deres metoder og arbejdsform er forenelig, ligeledes er deres kulturelle baggrund overvejende ens, således at der ikke opstår komplikationer på baggrund af kulturelle forskelle og sproglige barrierer.

2.2 Metoder

En gruppes metoder påvirkes af den opsætning og arbejdsform gruppen benytter. I vores case er gruppen uden en valgt mødeleder og der er altså ingen til eksplicit at facilitere arbejdet. Deres proces kan skifte mellem at være parallelt eller sekventielt opdelt og de arbejder kun med en session ad gangen [Dennis, *et al.* 1988]. Det vi vil sige med denne opsætning er, at gruppen arbejder uden nogle projektledere og vi vil primært undersøge den synkrone proces. Der er også tale om asynkront arbejde i projektet, men i casen, som jo er et møde, taler vi om en synkron arbejdsform. Det drejer sig her om lyd, video og 'instant messaging'. Til selve mødet foregår dialogen synkront over lyd, dvs. den er parallel, hvorimod deres brug af 'instant messaging' vil være sekventiel, altså turbaseret. Udenfor møderne er der udelukkende tale om sekventielt, asynkront arbejde med fildelingsværktøjer og e-mail værktøjer.

2.3 Omgivelser/miljø

Der er i denne case ikke tale om en ikke face-to-face situation, men det som [Dennis, *et al.* 1988] kalder for 'multi-individual sites' og det vi kalder for et geografisk distribueret samarbejde. Det betyder at gruppen sidder geografisk spredt ved hver deres computer, uden mulighed for at mødes. Gruppen er således bundet af de tilgængelige værktøjer, såsom Skype, MSN messenger, BSCW¹ og en kalender.² Disse værktøjer tilføjer nogle restriktioner for arbejdet og det er blandt andet disse restriktioner der fører til vores valg af prototype – som beskrives nærmere i 'Metode'.

2.4 Afgrænsning

I dette afsnit vil vi, på baggrund af den fremlagte case, afgrænse vores projekt til kun at omhandle de elementer som er relevante for vores problemstilling. Da vi har begrænset os til kun at se på synkrone møder, afgrænser vi os derfor fra at evaluere på de asynkrone samarbejdsværktøjer, og deres relevans for gruppearbejdet.

¹ Se <http://bscw.fit.fraunhofer.de/about.html> for informationer om værktøjet.

² De nævnte værktøjer er eksempler, vi som projektgruppe har haft til rådighed, de repræsenterer udvalgte funktioner, som lyd, video, fildeling og kalender. Vi siger intet om kvaliteten af disse værktøjer frem for andre af samme stil.

De asynkrone værktøjer er dog stadig en nødvendig del og skal derfor indgå i programpakken, selvom de ikke bliver anvendt i casen.

Det er vores ønske at se på den verbale kommunikation og især den tekstlige fastholdelse af resultatet fra det enkelte møde. I henhold til casen har vi ved at vælge en gruppe danskere med nogenlunde samme kulturelle og faglige baggrund forsøgt at lave en naturlig afgrænsning i forhold til de socioemotionelle signaler – det værende eksempelvis kropssprog, gestikulation og lignende.

Vi ønsker ikke i dette projekt at gå særligt ind i de problemstillinger, fraværet af disse signaler giver ved distribueret samarbejde. I vores valg af case, er problematikkerne ved at gruppemedlemmer ikke kan opfange disse socioemotionelle signaler under mødet, så vidt muligt minimeret.

Da vi har valgt en universitetsgruppe som case, hvor gruppemedlemmerne alle har nogenlunde samme formål og mål med arbejdet, vil vi i dette projekt også afgrænse os fra at gå nærmere ind i de problemstillinger og det teoriapparat der omhandler forhandlinger. Vi anerkender at der i kommunikationen ved et gruppemøde kan foregå en vis forhandling, men dette mener vi ikke er tilstrækkeligt til decideret at gå ind i hele forhandlingsperspektivet og den deraf følgende forhandlingsteori.

3 Metode

Vi ønsker en kvalificeret metodetilgang, for at kunne besvare vores problemstilling og afkode den udvidelse af handlerummet vi søger at skabe for geografisk distribueret gruppearbejde. Det drejer sig specifikt om udarbejdelse og fastholdelse af information i synkrone, distribuerede gruppemøder. Vi har valgt at gribe opgaven an, ved at benytte tre forskellige tiltag, som tilsammen skal begrunde og styrke vores besvarelse af problemstillingen. Vi vil her beskrive de tre tiltag og hvad vi forventer de vil kunne bidrage med:

- **Teori:** Gennem videnskabelige skrifter og artikler vil vi identificere de krav og behov, en gruppe kan have til et værktøj som skal understøtte synkront distribueret gruppearbejde. Samtidig vil vi danne et begrebsapparat, som vi kan benytte til at udvikle og kvalificere vores design. Vores case og afgrænsning har dikteret vores litteratursøgning. Det er meningen at teorien skal give indsigt i det valgte problemfelt og bidrage til en løsning på problemstillingen.
- **Prototype:** For at kunne forholde os kritisk til de identificerede behov og den designidé vi har udtænkt, vil vi udarbejde en prototype som en besvarelse af problemformuleringen. Vores prototype er et mødeadministrationsværktøj, som vil blive beskrevet i senere afsnit, men vi vil her i metoden løfte sløret for hvad vi mener prototypen skal indeholde. Den prototype vi vil lave skal kunne varetage:
 - Mødeoprettelse.
 - Registrering af brugere.
 - Dagsordenspunkter med redigerbare referater – synlige for alle også mens der skrives.
 - Opgavepunkter som også er redigerbare.

Vi forudser at det at udvikle et program³ på baggrund af vores design, vil skabe en kritisk og produktiv refleksion over designet, da vi i så fald skal forholde os til detaljer i designet i forbindelse med brugbarheden. Man kunne forestille sig at overse væsentlige

³ Hele koden er dokumenteret og kommenteret i bilaget.

problematikker ved kun at lave et konceptdesign, uden en reel implementering.

Derudover giver det os også muligheden for at kunne teste designet bagefter.

- **Eksperimenter:** Vi vil foretage to forsøg, et først i forløbet og et til sidst som test af produktet.
 - **Udforskende eksperiment:** Det første eksperiment har til formål at afdække de problemer og mangler vi ser i de gængse samarbejdsværktøjer for distribueret kommunikation. Eksperimentet skal foreligge inden designfasen af konceptet, således at resultatet, samt de overvejelser eksperimentet måtte føre til, kan inddrages i designudviklingen. Det vil også være på baggrund af dette eksperiment at, vi vil udvælge hvilket værktøj vi vil udvikle en prototype af.
 - **Fokuseret eksperiment:** Efter udvikling af en funktionel prototype, ønsker vi at teste denne i et forsøg på at fastslå om vi igennem vores design er kommet de mangler, vi identificerede i det første eksperiment, til livs. Ligeledes skal dette forsøg kunne fastslå om de designtiltag vi har identificeret igennem vores teori, har vist sig at være nyttige i forhold til det formål de var tiltænkt.

Testgruppen

Begge eksperimenter vi har udført, er med os selv som testgruppe. Vores projektgruppe svarer næsten eksakt til den vi har skitseret i vores case, så det giver god mening; men rent metodisk ligger der, særligt i det fokuserede eksperiment, en fare i at vi selv er testpersoner. Vi har et indforstået kendskab til designet og dets brugbarhed. Vi overser eller ignorerer måske ting i programmet fordi vi kender det og samtidig kan vi have forudindtagede holdninger til hvad eksperimentet skal give af konklusioner og løsninger. Vores eksperiment ville garanteret have haft en anden konklusion, hvis vi havde udført eksperimentet med en ekstern testgruppe. Dette mener vi dog ikke ville have været hensigtsmæssigt, da en ekstern gruppe sandsynligvis ville have haft svært ved at lade være med at fokusere på implementeringens fejl og mangler i programmet. Det er ikke implementeringen der er i fokus i vores projekt, men derimod designet. En ekstern testgruppe ville yderligere have haft svært ved at lægge mærke til det bagvedliggende design. Derfor mener vi at den bedste løsning var at foretage eksperimenterne med os selv som testpersoner.

Vi erkender altså den begrænsede værdi af vores empiriske data, og vi vil primært bruge det fokuserede eksperiment til at se om vores prototype kan ændre på den opfattelse af et distribueret gruppemøde vi noterede ved det første eksperiment.

De tre metodiske tilgange vil tilsammen give et varieret indtryk af de relevante problematikker vi møder og dermed bidrage til et kvalificeret løsningsforslag. Eksperimenterne vil kunne teste både vores brug af gængse produkter på markedet og vores eget program. Teorien samler forsøg og overvejelser, så vi kan kvalificere vores design yderligere. Det sidste tiltag, prototypen, er tiltænkt at skulle give os de overvejelser og den indsigt man får, når man går fra en designidé til et reelt produkt. Med disse tre tiltag mener vi derfor at være godt udrustet til at kunne besvare vores problemstilling.

4 Udforskende eksperiment (eksperiment 1)

For at kunne danne et overblik over de behov der skal dækkes førend en gruppe på optimal vis kan foretage et geografisk distribueret gruppemøde, har vi som nævnt foretaget et udforskende eksperiment, for at se hvilke mangler vi stødte ind i ved et distribueret gruppemøde, som ikke allerede er løst ved brug af almindelige, let tilgængelige programmer.

I dette afsnit vil vi lægge ud med at beskrive de omstændigheder der dannede rammen for vores eksperiment, for derefter at sammenfatte de behov og mangler vi oplevede.

4.1 Forsøgsrammen

Opgave: At gennemføre et gruppemøde med følgende dagsorden, udelukkede ved brug af verbal kommunikation og tekstchat:

Opgave inden selve mødet:

- Alle har sørget for at deres mikrofon fungerer
- Inden d. 17./10. skal alle have skrevet deres artikler på litteraturlisten

Dagsorden:

- Vi fremlægger hver især vores tekster
- Vi taler design
- Vi skal lave en liste over hvilke behov vi mener vi har brug for i en gruppemødesituation
- Vi skal aftale dagsorden for møde med Pernille

Tidspunkt og varighed: Den 17/10/06 kl. 11:00 til kl. 15:00

Deltagere: Anders Lorentz Hansen, Anders Nielsen, Kim Sonne og Martin Schultz

Placering: Alle deltagere sad hjemme og loggede på BSCW, instant messaging og Skype på mødetidspunktet.

Teknologi: Én benyttede Linux, OpenSuse og de andre tre Windows. Skype blev benyttet til den verbale kommunikation, Kopete/Messenger som chat (hvis lyden skulle svigte) og BSCW som fildelingsværktøj. Alle havde en bredbåndsforbindelse.

4.2 Mødeforløb

Gruppen bestod af fire personer og der blev ikke oprettet en talerliste til at holde styr på hvem der næste gang måtte sige noget, i stedet blev der forsøgt en mere flydende dialog, således at kreativiteten ikke blev begrænset af en tur-baseret diskussion.

Inden mødet havde vi problemer med at få lyden i Skype til at fungere for alle gruppemedlemmerne. Der var det givtigt at gruppens medlemmer kunne bruge Instant Messaging til at skrive til hinanden indtil lyden var sat endeligt op.

Mødets primære formål var at gennemgå de læste artikler og diskutere design, med en forventning om at det skulle ende med et produkt, i form af en liste af behov. Disse forventninger gjorde at mødet var præget af megen diskussion.

Efter at have sikret kontakt til alle mødedeltagere, således at alle kunne høre hinanden, blev artiklerne diskuteret – det fungerede ved at alle havde en kopi af dokumentet åben på skærmen imens en artikel blev præsenteret. På dette tidspunkt blev der ikke skrevet referat, men hvert gruppemedlem sørgede selv for at nedskrive de andres kommentarer til sin artikel.

Herefter begyndte en mere kreativ proces, hvor gruppen diskuterede design, i forhold til de værktøjer der var interessante for det enkelte gruppemedlem. Diskussionen var livlig og alle kom til orde. Da vi kendte hinanden i forvejen kunne vi også genkende hinandens stemmer, og blandt andet derfor oplevede vi det ikke som noget problem at vi ikke havde

video. Hvis vi ikke havde kendt hinanden i forvejen ville behovet for video sandsynligvis have været større.

I denne del af mødet blev resultaterne af diskussionerne fastholdt i et referat, da det var her at mødets produkt skulle findes. Referatet blev skrevet ved at et gruppemedlem skrev referatet lokalt på sin maskine, og løbende kopierede ”referatudklip” ind i en fælles gruppechat. På den måde blev gruppen løbende holdt ajour med hensyn til beslutningerne i gruppen.

4.3 Mødeprodukt og evaluering af mødet

Mødet resulterede i et referat og en liste af værktøjer som vi mente var relevante ved et distribueret gruppemøde.

Diskussionen fungerede ret godt. Vi oplevede ikke at have særligt behov for de socioemotionelle tegn under selve mødet. Det var ikke svært at komme til orde og vi havde ikke svært ved at vide, hvornår de andre ville sige noget eller hvornår de var ved at være færdige med at tale. Vi oplevede heller ingen kulturelle eller sproglige misforståelser under diskussionen, på trods af at vi ikke kunne se hinandens kropssprog, gestikuleringer, ansigtsudtryk etc. Alt dette var sandsynligvis blevet lettere af, at vi i forvejen havde kendte hinanden og har nogenlunde samme kulturelle baggrund. Der var dog et mindre awareness problem under mødet. Hvis en person ikke sagde noget i lang tid, var det svært at vide om personen stadig fulgte med i samtalen. Hvis der også havde været visuel kontakt, kunne det måske have løst problemet, men vi opfattede det dog ikke som noget der var afgørende for resultatet og effektiviteten af det distribuerede møde.

Tekstchatten viste sig derimod ikke at være særlig god til at formidle referatet og de designpunkter vi skulle blive enige om. Ved ændringer i et punkt skulle hele listen sendes igen for at forhindre at deltagerne mistede overblikket. Det gjorde det svært at holde tråden og for ikke at få flere udgaver af samme punkt, måtte vi udvælge en til at skrive punkterne, men da alle havde input, skulle der enten dikteres eller redigeres flere gange i punkterne. Det blev hurtigt til, at det værktøj vi ville designe var et bedre værktøj til at

holde dagsorden og referat for mødet. Ligeledes så vi et behov for et diagramværktøj hvor alle kunne designe og redigere i produktet samtidig, men da vi i vores proces så et større behov for at kunne fastholde resultatet af en online møde, faldt vores valg på et mødeadministrations-/referatværktøj.

4.3.1 Resultat af forsøget

Efter dette eksperiment valgte vi at arbejde videre med et mødeadministrationsværktøj, der kan behandle mødereferater som alle kan kigge med i, imens det bliver skrevet. Det skal være muligt for alle at redigere i det, også så to kan skrive samtidig, om end i hvert sit punkt. Detaljer om designet af dette værktøj beskrives i 'Mødeadministrations Design' kapitlet.

5 Teori

I dette kapitel vil vi gennemgå en række teoretiske overvejelser, som har været med til at kvalificere og begrunde vores valg i designet af vores program. Vi begynder med nogle generelle problematikker ved distribuerede gruppemøder, set i forhold til vores case, navnlig den samhørighed en gruppe kan opnå ved at dele fælles forståelse og målsætninger. Derefter dykker vi ned i den processuelle arbejdsform for at afkode hvad det er denne arbejdsform har af konsekvenser for vores case. Til sidst vil vi beskæftige os med den kommunikation der foregår i samarbejde med systemet og udenom systemet, samt de informationsstrømninger som er at finde i systemet.

5.1 Problematik ved distribuerede gruppemøder

I dette afsnit vil vi gennemgå en række teoretiske overvejelser, som er relevante i forhold til vores case og hvad det er vi gerne vil understøtte med vores programpakke - herunder både det program vi har designet og de programmer vi anbefaler at brugerne benytter i samspil med vores. Vi ønsker at afdække det grundlag, vi mener, skal være på plads forud for et distribueret gruppearbejde.

5.1.1 Fælles målsætninger

For at arbejdsprocessen i en gruppe kan være konstruktiv, er det nødvendigt at gruppemedlemmerne er enige om målene. Med fælles mål bliver gruppen mere fokuseret og kan dermed bedre udforske nye ideer samt træffe bedre beslutninger. Derudover er det normalt at hvis gruppemedlemmer føler de har været med til at formulere målene, så er der større ejerskab omkring målene, hvilket betyder at der er større chance for at de bliver ført ud i livet. [Farooq, *et al.* 2005]

Det er altså vigtigt, at systemet hjælper gruppemedlemmerne med at blive enige om målene, og derefter skal systemet understøtte at gruppemedlemmerne holdes fast i de fastsatte mål - dermed ikke sagt at målene ikke kan ændres.

5.1.2 Fælles forståelse

Selvom en gruppe har fået fastsat fælles målsætninger er det mindst lige så vigtigt at den også danner en fælles forståelse. [Olson & Olson, 2000] fremhæver i deres artikel, at dette er helt centralt for at kommunikationen kan fungere i en gruppe. Kort sagt jo bedre den fælles forståelse er, jo bedre er kommunikationen og dermed vil der være en højere produktivitet.

Med fælles forståelse menes der den viden som gruppen har til fælles, og er bevidst om at de har til fælles. En person taler på forskellige måder alt efter hvem han eller hun taler med. Dette gør man på baggrund af hvilken viden man tror den anden person har. En datalogistuderende der skal forklare en teknisk ting til sin medstuderende eller kæreste vil sandsynligvis ikke bruge samme ordvalg og fremgangsmåde i de to situationer. Ligeledes har den kontekst de pågældende personer er i, betydning for hvordan kommunikationen vil forløbe [Olson & Olson, 2000]. Hvis man kender hinandens baggrund, styrker, svagheder og vidensniveau forløber kommunikationen meget bedre, da man derved bedre er klar over hvordan man kan tale – man har en fælles forståelse. [Olson & Olson, 2000]

Samtidig ligger der også i den fælles forståelse, at man ved at se hvordan den pågældende f.eks. reagerer på et udsagn, kan tilpasse sin kommunikation derefter. Det vil sige at det er vigtigt under selve kommunikationen at tilpasse sin fælles forståelse, hvis den oprindelige forståelse ikke var korrekt. [Olson & Olson, 2000]

Den fælles forståelse skabes og omformuleres konstant ved de forskellige “tegn” der kommer i løbet af en samtale. Jo færre “tegn” der er, jo sværere er det at skabe den fælles forståelse og risikoen for misforståelser øges. I [Olson & Olson, 2000] artiklen nævner forfatterne otte forskellige dimensioner, som forskellige typer kommunikationsmetoder kan give. Jo flere af disse dimensioner der opfyldes, jo bedre fælles forståelse kan man opnå, og jo bedre vil kommunikationen være.

- coprecence, når man er i det samme fysiske rum
- visibility, når man kan se hinanden
- audibility, når man kan høre hinanden

- contemporality, når den kommunikation der gives, modtages med det samme
- simultaneity, når alle parter kan både give og modtage beskeder samtidig
- sequentiality, når rækkefølgen af den turbaserede kommunikationen bliver overholdt
- reviewability, når det er muligt at se andres beskeder
- revisability, når man selv kan se sin besked inden den sendes

Forskellige kommunikationsmetoder opfylder disse dimensioner. I nedenstående skema ses det hvilke metoder der opfylder hvilke dimensioner.

<i>Medium</i>	<i>coprecence</i>	<i>visibility</i>	<i>audibility</i>	<i>Contemporality</i>	<i>simultaneity</i>	<i>sequentiality</i>	<i>reviewability</i>	<i>revisability</i>
<i>face to face</i>	x	X	x	x	x	x		
<i>telephone</i>			x	x	x	x		
<i>video-conference</i>		X	x	x	x	x		
<i>two-way chat</i>				x	x	x	x	x
<i>answering-machine</i>			x				x	
<i>e-mail</i>							x	x
<i>Letter</i>							x	x

Tabel 1 [Olson & Olson, 2000]

For at et værktøj skal kunne bruges i distribuerede grupper, er det altså vigtigt at så mange af dimensionerne bliver understøttet. Dette vil vi komme mere ind på i vores designovervejelser af vores programpakke.

5.2 Høj og lav kobling

Med en fælles forståelse for arbejdsområdet og en fælles målsætning for resultatet på plads, er man nødt til at forholde sig til arbejdets karakter. I forhold til kommunikations og arbejdsopgaver hvor der er tale om rutine arbejde uden den store form for kompleksitet er der tale om lav kobling. Er der derimod tale om højt intensivt og kreativt arbejde er der tale om høj kobling [Olson & Olson, 2000]. Kobling refererer altså til de arbejdsopgaver gruppen har og den klarhed og fælles forståelse der er knyttet til den enkelte opgave. Jævnfør vores case, vil vi nu afdække nogle af de begreber der gør sig relevant for en forståelse af problemstillingen vedrørende arbejdsprocessen.

5.2.1 Kreative komplicerede arbejdsprocesser med høj kobling

I [Olson & Olson, 2000] fremhæves det, at for at et distribueret samarbejde skal forløbe succesfuldt, er det ifølge forfatterens empiriske undersøgelser nødvendigt at arbejdet har en lav kobling. Arbejde med en høj kobling er afhængigt af de enkelte arbejders specifikke kvalifikationer. Det er arbejde der er komplekst, tvetydigt og usikkert, i og med at det ikke er prøvet før. Ved denne type arbejde betyder det som regel, at gruppemedlemmerne skal mødes jævnligt for at diskutere problemstillingerne igennem igen og at det er strengt nødvendigt med en god og rig kommunikation.

Arbejde med lav kobling er derimod mere rutinepræget. Arbejdsprocessen kan i og for sig godt være kompleks, men hvis alle ved præcis hvad de skal gøre, og de har gjort det mange gange før, så kan det stadig betegnes som arbejde med en lav kobling. Det kan siges på den måde at alle dem der skal udføre arbejdet har en meget høj "fælles forståelse" for arbejdets mål og proces – og jo højere fælles forståelse gruppemedlemmerne har omkring arbejdets mål, jo lavere kobling har arbejdet.

I vores case ser vi specifikt på gruppearbejde mellem studerende. Dette arbejde kan oftest karakteriseres som værende af høj kobling. Dermed mener vi dog ikke, at vi på baggrund

af [Olson & Olson, 2000], uden videre kan forkaste tankerne om et vellykket distribueret samarbejde. Men det er særlig vigtigt at være opmærksom på hvordan man kan understøtte denne form for arbejde som foregår i en universitetsgruppe.

Det kreative arbejde i en universitetsgruppe kan betegnes som værende af høj kobling, da gruppen skal producere et produkt, som ingen fra gruppen sandsynligvis direkte har prøvet før. Det skal søges ny viden, og der kræves mange diskussioner og ideer. I tråd med det påpeger [Farooq, *et al.* 2005] ligeledes, at særligt de kreative processer kan være vanskelige at få til at fungere i et distribueret samarbejde. Derfor er det vigtigt at gøre sig klar, hvad der skal til for specifikt at understøtte denne form for arbejde.

Vi vil i det efterfølgende se på en måde, at styre den højt koblede arbejdsproces ved at foretage en præcisering og afklaring af kernen i konceptet.

5.2.2 Divergent & konvergent tænkning

Divergent tænkning er ens evne til at få ideer, se muligheder, stille åbne spørgsmål etc., og altså at brede sig og udvikle nye tanker. Konvergent tænkning er når man forsøger at snævre alle de foregående tanker ind, for at finde den løsning man ønsker at arbejde med. I en kreativ proces foregår disse to måder at tænke på hele tiden og de supplerer hinanden [Farooq, *et al.* 2005]. Det er helt centralt at et system giver brugerne mulighed for både at brede sig og se nye muligheder (divergent), for derefter at snævre ind, for at finde løsningsforslag og mulige implementeringsmuligheder (konvergent). [Farooq, *et al.* 2005] påpeger ligeledes at hvis der i gruppen er forskellige synspunkter kan der ved diskussionen af disse perspektiver opstå en slags erkendelsesmæssig konflikt – hvilket yderligere kan hjælpe den divergente tænkning. Hvis der er en erkendelsesmæssig konflikt i gruppen, tvinger det gruppemedlemmerne til at tænke problemstillingen igennem en gang til for at kunne argumentere for deres standpunkt og måske derigennem nå til den konklusion at det gamle standpunkt kan forkastes [Farooq, *et al.* 2005]. Et system kan i sig selv være med til at skabe eller gøre gruppen opmærksom på denne erkendelsesmæssige konflikt.

5.2.3 Refleksivitet

Refleksivitet drejer sig om, at gruppens medlemmer reflekterer over gruppens mål, strategi, proces osv. [Farooq, *et al.* 2005] opdeler refleksivitet i tre elementer:

- Refleksion
- Planlægning
- Handling

Refleksion er selve det at være opmærksom, holde øje med og evaluere gruppens arbejde og samtidigt at forholde sig kritisk til arbejdet. Planlægning er en mulig konsekvens af refleksionen, hvor gruppemedlemmerne overvejer hvad der skal gøres og handling er så den fase hvor gruppen beslutter sig for at føre det ud i livet. Hvis systemet understøtter disse tre elementer betyder det at gruppens arbejde konstant kan genforhandles og dermed forbedres. [Farooq, *et al.* 2005]

5.3 Double level language – Formel og kulturel kommunikation

Robinson mener, at der i et distribueret samarbejde bør være to forskellige niveauer for kommunikation [Robinson, 1991]. De to begreber defineres som havende følgende værdi:

- **Formel kommunikation:** er essentiel da den skaber fælles referencer og klarhed. Den er omdrejningspunktet for samarbejdet, en restriktiv og forudsigelig standard for gruppen at arbejde ud fra. Altså en kommunikation som deles gruppen imellem.
- **Kulturel kommunikation:** er subjektiv betonet, den varetager de perspektiver og fortolkninger som kommunikeres af deltagerne, på baggrund af deres reference ramme; den formelle kommunikation.

Han eksemplificer tale og tekst, som værende to niveauer af kommunikation, hvor talen gør at hænderne er frie til at arbejde med tekst i dokumenter. Således er den formelle kommunikation bundet i produktet der arbejdes med og den kulturelle kommunikation er bundet i talen [Robinson, 1991].

Dette betyder at man kan være flere om at arbejde med en tekst, og det at udfærdige teksten er den formelle kommunikation. Hvorimod den diskussion der kan foregå omkring teksten – det kunne eksempelvis være tale eller en sideløbende chatkommunikation, klassificeres som kulturel kommunikation. Dette giver gruppen en

kommunikationsform til at reflektere over deres fælles produkt. Robinson fastslår at den formelle kommunikation er lige gyldig hvis ikke der foretages kulturelle fortolkninger af den og at den kulturelle kommunikation er vag og upræcis hvis ikke den knyttes til en fast og formel kommunikation; således er de to begreber afhængige af hinanden.

5.4 Awareness – Informationsstrømninger i systemet

I dette afsnit vil vi se på de informationer som skal gøre det muligt for en bruger at udnytte systemet optimalt. Enhver handling kræver information, ligesom hver handling samtidig også afgiver information. En bruger bør forholde sig til andre brugeres aktiviteter, når brugeren skal træffe en beslutning eller udføre en handling i et distribueret møde. Det skal derfor være klart hvilke informationer brugeren har til rådighed i den aktuelle situation. I dette afsnit vil vi afdække de awareness behov en gruppe kunne have til et system.

Awareness kan kort oprides som brugernes muligheder for at opfatte og bearbejde information vedrørende begivenheder eller sanseindtryk omkring hvad der sker i systemet og hvordan de andre brugere har det socialt.

Vi har sammenfattet en liste over de forskellige awareness typer vi ser relevante for vores program. Listen beskæftiger sig med brugerens aktiviteter og hvordan systemet behandler resultatet af disse aktiviteter [Steinfeld, et al. 1999]. Det er selvfølgelig afhængigt af systemet hvilke type awareness der er brug for, ligesom det kan afhænge af brugergruppen, hvor meget information de har behov for og lyst til at afgive/modtage.

5.4.1 Fire forskellige awareness typer

- **Activity:** Der skal skabes overblik over de projektrelateret aktiviteter der foretages af de enkelte gruppemedlemmer. Dette skal ske både synkront, dvs. hvad et gruppemedlem er i gang med at arbejde med på det pågældende tidspunkt, og asynkront. Brugere skal kunne danne sig et overblik over hvad der er sket siden man sidst har været logget på systemet.

- **Availability:** De enkelte gruppemedlemmers status, dvs. om de er tilstede eller ej, skal være synligt for de andre gruppemedlemmer. Det skal være muligt at identificere hvad de laver på det pågældende tidspunkt. Det kan dog ikke forventes at systemet altid selv kan afkode om en bruger er tilgængelig, og såfremt denne ikke er til stede, så bør brugeren selv udfylde sin status så den er til hjælp for de resterende gruppemedlemmer.
- **Process:** I situationer hvor der er tale om deciderede workflowlignende arbejdsopgaver, hvor det enkelte gruppemedlems opgaver skal passes ind i en større sammenhæng, bør der være awareness omkring dette workflow. Det er vigtigt at gruppemedlemmerne kan beholde overblikket og se hvordan deres opgaver passer ind i den samlede proces.
- **Perspective:** Med awareness om processen er det også vigtigt at kende til beslutningerne bag opgaverne, samt på hvilke grundlag de beslutninger er truffet. Ved at danne awareness omkring historikken, eller de informationer som knytter sig til beslutningsprocessen, får gruppemedlemmerne en bedre forståelse for hvor projektet er på vej hen og især hvorfor.

5.4.2 Passiv eller aktiv awareness

Det er vigtigt at der tages stilling til om (awareness-) informationerne leveres automatisk af systemet eller om det er noget personer selv skal opsøge.

- **Passiv:** En fordel er at brugeren kan blive informeret om noget han ikke vidste eller ikke var klar over han skulle holde sig ajour om. Faren er at brugeren nemt kan overvældes af de mange informationer, og dermed overse de vigtige af dem. Her skal man huske at awareness informationen helst skal være noget brugeren kan handle på baggrund af.
- **Aktiv:** En fordel er at brugeren informeres om det han er interesseret i at vide, men samtidig risikere han at 'glemme', eller miste vigtige informationer/opdateringer, hvis han selv skal opsøge dem.

I begge tilfælde handler det om fokus – forstyrres brugeren af awareness informationerne eller støtter de op om den aktivitet han har gang i? Der findes to tilgangsmetoder:

periferisk og central (focal) [Steinfeld, et al. 1999]. Disse skal implementeres i den udstrækning de støtter op om informationsformidlingen og den opgave brugeren sidder med. Man kunne forstille sig situationer hvor den periferiske awareness blev overset, og modsat hvor den centrerede awareness stjæler fokus fra den aktuelle opgave. Her vil det være en fornuftig mulighed, at brugeren måske ikke kunne vælge hvilke beskeder han vil have, men derimod har mulighed for at bestemme hvordan han vil gøres opmærksom på beskederne. Dette vil være mindre forstyrrende og gør det nemt for brugeren at tilpasse det til sin situation uden at han kommer til at mangle information [Grudin, 1994]. Designmæssigt skal der også tages stilling til om der skal differentieres mellem de informationer som sendes ud til brugerne; er der nogen som skal informeres om ting de andre ikke har brug for at vide.

5.4.3 Explicit vs. Embedded

I forhold til programmets indsamling af awareness data, kan det foregå enten eksplicit (ved en aktiv handling fra en bruger) eller implicit (ved en automatisk opdatering udenom brugeren). Det er meget information som systemet ville have svært ved at skaffe selv (for eksempel hvad brugeren laver udenfor systemet), men det at bede brugeren om aktivt at afgive den, vil ofte blive en ekstra arbejdsbelastning og derved en irritation, eller noget man helt glemmer. Her vil det være vigtigt at opnå en kritisk masse af brugere som rent faktisk anvender de eksplicite awareness funktioner, således at det vil blive en selvforstærkende effekt [Grudin, 1994].

5.4.4 Access anywhere

Nem tilgang til systemet er et vigtigt kriterium. Derfor bør hardware- og softwarekrav holdes på et minimum, så det ikke er nødvendigt at brugerne skal have det nyeste udstyr. Derudover er det centralt at produktet (helst) skal være tilgængeligt fra flere arbejdsstationer end brugerens egne, sådan at brugeren altid kan benytte systemet ligegyldigt hvor brugeren er. Det betyder, at det ikke bør være nødvendigt for brugeren at installere noget på sin maskine, og i forlængelse af det betyder det også at produktet skal være platformsuafhængigt - så brugeren både kan benytte f.eks. Linux eller Windows.

Et godt eksempel på produkter der kan bruges alle steder fra er webapplikationer. Disse giver mulighed for at "logge på" produktet ligegyldigt, hvor brugeren befinder sig. I forhold til platformsuafhængighed er Javaprogrammer yderligere et eksempel der sikrer denne uafhængighed.

6 Opsamling

I dette afsnit vil vi samle op på den gennemgåede teori set i forhold til vores erfaringer fra det udforskende eksperiment. Derefter vil vi se på hvad det er vores mødeadministrationsværktøj kan bidrage med i forhold til teorien.

6.1 Erfaringer i forhold til teorien

Det vi oplevede i eksperimentet i forhold til de teoretiske begreber, var at den konvergente tankegang blev begrænset. Den divergente tankegang gik i og for sig fint over Skype, men når vi skulle blive enige om hvilken retning vi skulle vælge (og hvad vi faktisk allerede havde besluttet), blev det sværere. Vi forsøgte at "være" konvergente ved at skrive et referat af vores samtale over chatten og på den måde fastholde vores beslutninger og snævre os ind. Men denne fremgangsmåde var langt fra optimal. Der skulle hele tiden rettes i referatet på chatten og det betød at vi skulle kopiere det gamle ind i tekstfeltet og skrive det nye på, hvilket lynhurtigt blev uoverskueligt. Da vi ikke kunne overskue referatet, blev det på den måde svært at fastholde "fælles forståelse" for vores mål og beslutninger - "*hvad var det lige vi havde aftalt?*". Dermed blev det også sværere at tænke reflektivt over vores mål og strategi for det videre arbejde. Derfor kan vi konkludere at vi godt kunne afholde et gruppemøde distribueret, men der var tydeligt nogle problemer med at opnå en "fælles forståelse" for vores beslutninger og videre mål.

Problemet var umiddelbart, at vi manglede et værktøj der kunne bruges til at kommunikere den formelle information. Vi havde den kulturelle kommunikation gennem Skype og vores diskussioner forløb godt. Men forskellige mennesker har forskellige opfattelser af hvad der bliver talt om, og derfor er det også nødvendigt at den formelle kommunikation fastholdes. Dette kan være med til at sikre at gruppedlemmerne har en nogenlunde fælles forståelse af de endelige beslutninger - og her manglede vi et effektivt værktøj. Et sådan værktøj kan forbedre muligheden for at afholde effektive synkrone gruppemøder i distribuerede grupper og skal yderligere understøtte availability og activity awareness. Disse typer af awareness anser vi for at være de mest relevante i

forhold til vores case, idet de giver brugeren en bedre opfattelse af de andre gruppemedlemmers tilstedeværelse i processen.

6.2 Fælles referencer i objekter

Hvorfor manglede vi specifikt denne formelle kommunikation i et distribueret samarbejde? Oftest kan de forskellige gruppemedlemmer i et face-to-face møde heller ikke se hvad der bliver skrevet i referatet. Begrundelsen for det kan måske findes i [Olson & Olson, 2000], hvor forfatterne gennem omfattende studier har forsøgt at finde karakteristika ved face-to-face møder. Her er der særligt to interessante karakteristika i forhold til lige denne problemstilling:

- co-reference, at det er nemt at etablere en fælles reference til objekter
- spatiality of reference, at både personer og arbejdsobjekter er til stede i samme rum

Disse to karakteristika betyder at personer ved et face-to-face møde kan gøre brug af objekter som fælles referencer - eksempelvis kan et gruppemedlem pege på et objekt som har en særlig værdi for gruppen, og så ved alle gruppemedlemmerne hvad der tales om. Objekterne giver en fælles referenceramme for gruppen, hvilket meget nemmere giver en fælles forståelse i gruppen. Objekter kan i princippet være hvad som helst, så længe at det giver gruppen en fælles reference. Oftest vil det dog være tegninger, dokumenter, diagrammer, modeller, skitser etc.

Denne mulighed mangler man oftest i distribuerede grupper. I vores udforskende eksperiment kunne vi alle se de samme dokumenter, og dette benyttede vi os af, hvis der for eksempel var en figur i et af dokumenterne - men vi havde ingen fælles objekter at "pege" på i forhold til vores diskussion og det vi forsøgte at nå frem til på mødet. Vi kunne ikke oprette og ændre objekter under selve mødet, som kunne give denne fælles reference. Derfor oplevede vi i eksperimentet at vi havde brug for den formelle kommunikation, som på den måde ville kunne udgøre den fælles reference.

I tabel 1 på side 20, som er taget fra [Olson & Olson, 2000], er der otte forskellige dimensioner, som forfatterne mener, bør være til stede for opnå fælles forståelse. Vi mener, at ”fælles referencer til objekter” kunne tilføjes som en niende dimension. At have objekter der kan give en ”fælles reference”, giver tydeligt en bedre fælles forståelse, og forbedrer en diskussion.

Det er altså vigtigt, at gruppemedlemmerne kan oprette og redigere objekter under selve mødet, og disse objekter kan fungere som fælles referencer under mødet. Her er det centrale for disse fælles objekter, at alle gruppemedlemmerne kan se objekterne, samtidig med at alle kan ændre i objekterne, og alle kan se når der bliver ændret i objekterne (contemporality og simultaneity).

Vi har i vores mødeadministrationsværktøj og vores samlede EMS, forsøgt at tilføje denne ekstra dimension. Referatet i mødeadministrationsværktøjet udgør den formelle kommunikation som [Robinson, 1991] taler om, og den fælles reference som [Olson & Olson, 2000] har i deres artikel. Referatet skal selvfølgelig have contemporality og simultaneity. På samme måde kan diagrammeringsværktøjet der kort er nævnt i det udforskende eksperiment også bruges til at skabe disse dokumenter.

6.3 Mødeadministrationsværktøjet som en ”ny kompetence”

I [Robinson, 1991] fremhæves der en række succeskriterier for CSCW-systemer. Et af disse har vi allerede nævnt, nemlig double-level language. Et anden er de såkaldte ”nye kompetencer”. Det Robinson mener, er at hvis den nye teknologi betyder at brugerne kan gøre nogle ting de ikke kunne tidligere, så er systemet rigtig godt på vej til at blive en succes. Selvom der måske er småfejl i systemet, så bærer brugerne i første omgang over med dette, fordi det at brugeren nu kan flere ting end tidligere overskygger småfejlene. På den måde er det nemmere at implementere et værktøj der tilføjer ny funktionalitet til brugeren, end at finpudse og videreudvikle på eksisterende værktøjer. Det at brugeren oplever en ny kompetence er selvsagt ikke en garanti for succesfuld implementering.

Vi mener netop at vores mødeadministrationsværktøj sammen med de andre programmer i vores EMS pakke, vil tilføje en ny kompetence til det at afholde distribuerede møder. Det at alle kan ændre, redigere, slette og tilføje objekter mens alle de andre gruppemedlemmerne kan se det, er en ny og bedre måde at arbejde distribueret på. Derfor mener vi at vores værktøj potentielt er en måde at forbedre mulighederne for at afholde distribuerende gruppemøder.

7 Samlet EMS design

Vi har i vores problemfelt kort skitseret hvad et elektronisk mødesystem er. Vi ser vores samlede programpakke som en form for et EMS. Det er ud fra det synspunkt, at hvis alle de programmer vi anbefaler bliver benyttet, så kan disse programmer i sammenhæng betegnes som et EMS. Det er altså ikke et samlet produkt som betegnes EMS, det er kombinationen mellem en række programmer der udgør det elektroniske mødesystem.

Ved udrulningen af et EMS vil en sådan moduleret opbygning med separate funktionaliteter i separate programmer være ønskelig, da det vil understøtte en gradvis implementering frem for en hurtig og måske forhastet overgang [Grudin, 1994]. En gradvis implementering kan eksemplificeres på følgende måde: En gruppe er allerede bekendt med diverse programmer til synkron og asynkron kommunikation via netværk. Der er behov for en yderligere funktion, hvorfor et enkelt nyt værktøj introduceres. I og med der kun introduceres et nyt modul vil der være en mindre belastning ved tilvæning end hvis et komplet nyt EMS skulle introduceres. Nu har brugerne den frihed til fortsat at vælge de programmer de helst benytter, i stedet for et stort samlet koncept, der tvinger brugerne til at benytte nye programmer, hvor de allerede har vænnet sig til andre programmer. Det kan f.eks. være et mailprogram brugerne er glade for, men det nye samlede koncept betyder, at man skal skrive mail fra det nye EMS. Modulopbygningen gør det i stedet nemmere for brugeren at vænne sig til den nye arbejdsmetode, da det forudsættes at vedkommende allerede er bekendt med de fleste programmer i forvejen.

Det betyder selvfølgelig, at de foreslåede konkrete produkter i figur 2 på side 31, såsom Skype, BSCW og Messenger, kan skiftes ud efter behov hvis brugerne i stedet ønsker at benytte andre programmer med lignende funktioner.

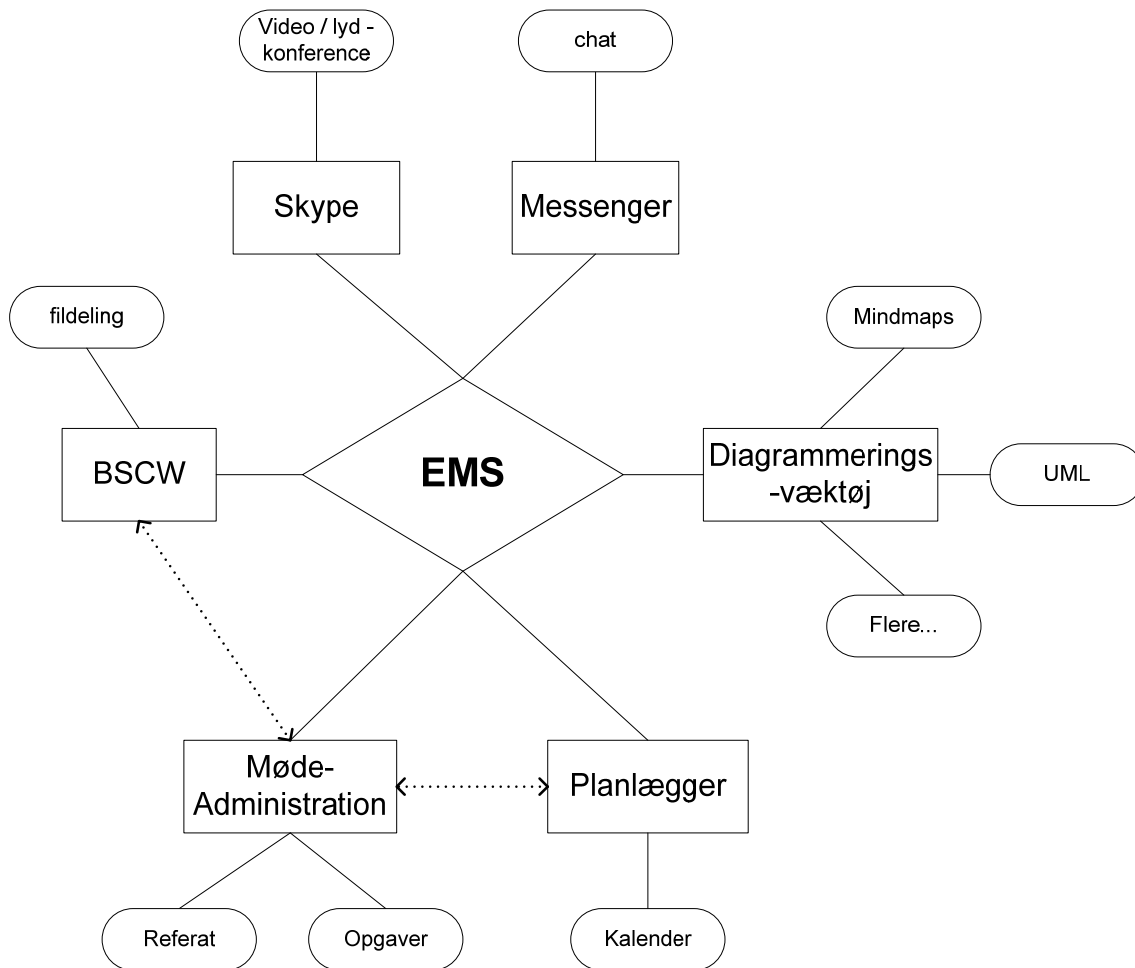
Derudover er store komplette systemer selvsagt dyrere og tager længere tid at udvikle, og derfor har vi i dette projekt valgt at holde os til ”modultanken”.

I dette kapitel vil vi begrunde hvorfor vi mener, at det netop er denne type værktøjer der skal bruges til sammensætte en ”EMS modul pakke”, som kan bruges til at afholde et distribueret gruppemøde.

På baggrund af vores teori og udforskende eksperiment, har vi udpeget seks forskellige typer værktøjer der vil være brugbare. Det drejer sig om:

- Mødeadministrationsværktøj
- Kalender
- Video/lyd
- Fildeling
- Instant messaging (chat)
- Diagrammeringsværktøj

I nedenstående figur, kan det ses hvilke typer af værktøjer vi mener, er nødvendige, eksempler på nogle af produkterne og sammenhængen mellem dem i forhold til det samlede EMS.



Figur 2. EMS systemet

Vi vil her gennemgå hvert af de seks værktøjer, for at give en forklaring på, hvorfor det givne værktøj er en del af den samlede EMS pakke.

7.1 Skype

Et vigtigt krav til et distribueret gruppemøde er audio kommunikation så gruppemedlemmerne kan diskutere og udveksle meninger. I Skype kan gruppemedlemmerne holde en distribueret videokonference og Skype opfylder dermed både behovet for lyd og eventuelt video. Alle kan tale, høre og se hinanden samtidig, og teknologien er så veludviklet, at selve lyd- og videofunktionaliteten er fuldt ud brugbar. I forhold til [Olson & Olson, 2000] er både ”visibility”, ”audibility”, ”contemporality”, ”simultaneity” og ”sequentiality” opfyldt ved brugen af Skype.

I den samlede EMS programpakke skal Skype bruges til at kommunikere med - samtidig med, at de andre produkter i "EMS pakken" anvendes. Dermed bruges Skype til at opfylde den kulturelle kommunikation med både lyd og video. Den kulturelle kommunikation er der behov for, således at gruppemedlemmerne kan opnå en "fælles forståelse", sådan at misforståelser eller tvivlsspørgsmål imellem gruppemedlemmerne hurtigt kan korrigeres. Særligt når det kreative arbejde med høj kobling skal udføres, vil det være nødvendigt med denne kulturelle kommunikation. Når der skrives et referat i mødeadministrationsværktøjet er det ligeledes nødvendigt, at de andre gruppemedlemmer kan kommentere løbende, så alle bliver enige om hvad der skal stå.

Skype er altså et uundværligt redskab i brugen sammen med resten af programpakken. Vi mener eksempelvis ikke at mødeadministrationsværktøjet kan benyttes uden at blive suppleret med minimum en "simultaneity", "contemporality" og "audibility" mellem gruppemedlemmerne.

Om der er behov for den visuelle del kommer an på gruppen. I vores første eksperiment oplevede vi ikke et særligt behov for en visuel repræsentation. Dette har sandsynligvis noget at gøre med, at vi i gruppen allerede havde mødt hinanden flere gange tidligere, kender hinandens stemmer og holdninger. Hvis det er en gruppe der ikke kender hinanden og aldrig har mødt hinanden vil de i højere grad have behov for det visuelle aspekt også. [Olson & Olson, 2000]

7.2 BSCW

BSCW er et fildelingsværktøj. Det er vigtigt for et velfungerende gruppearbejde, at der er gode muligheder for at dele filer i mellem gruppemedlemmerne, hvilket vi blandt andet også oplevede i vores udforskende eksperiment.

I mødeadministrationsværktøjet vil der være mulighed for at tilføje dybe links i en dagsorden eller et møde til filer i fildelingsværktøjet, sådan at gruppemedlemmerne hurtigt kan finde de dokumenter der diskuteres på det pågældende møde. Vi mener ikke det bør være muligt at gemme filer i mødeadministrationsværktøjet, da dette ville øge

risikoen for der opstår dubletter af dokumenter, og det dermed bliver svært for gruppen at holde styr på hvilket dokument der bliver redigeret i. Derfor skal alle dokumenter kun ligge i fildelingsværktøjet. Men det bør være muligt at indsætte dybe links i mødeadministrationsværktøjet til fildelingsværktøjet, sådan at brugerne dermed hurtigt kan tilgå det givne dokument.

7.3 Messenger

I vores programpakke har vi tilføjet Messenger. Det behøver ikke være Messenger, men kan være en hvilken som helst form for instant messaging. Til selve mødet mener vi umiddelbart ikke at der er behov for at bruge et instant messaging, i hvert fald ikke når vores møleadministrationsværktøj er færdigudviklet. Grunden til vi alligevel har Messenger med i programpakken er, at vi i begge vores eksperimenter har oplevet problemer med at få lyden til at fungerer i lyd/videokonferencerne. Inden lyden fungerer til gruppemødet er det derfor givtigt at kunne skrive til hinanden – så man sammen kan få løst ”lydproblemerne”. I den optik kan det siges, at Messenger er med i vores programpakke som en slags kommunikationsbackup. Indtil gruppen er 100 % sikre på de kan få lyd/videokonferencen til at fungere med det samme, mener vi det vil være nødvendigt med en eller anden form for instant messaging.

Det skal dog siges, at Skype også kan bruges som instant messaging. Vi har skrevet Messenger på som anbefaling, fordi flertallet af gruppens brugere helst benytter dette værktøj.

7.4 Diagrammeringsværktøjet

Vi har i vores samlede programpakke et diagrammeringsværktøj. Et diagrammeringsværktøj vil helt generelt øge den ”fælles forståelse”.

Diagrammeringsværktøjet kan indeholde mange forskellige typer af diagrammer. I vores figur har vi specifikt skrevet UML og Mindmap på, da dette er to typer af diagrammer vi i dette projekt gerne ville have haft. ’Flere’ i figuren betyder at her kan være en række af forskellige typer diagrammer, modeller tegninger osv., alt efter hvad gruppen har brug for. I diskussioner kan det oftest være en hjælp, hvis gruppemedlemmerne samtidig kan

se en model eller lave et fælles mindmap, da den visuelle repræsentation giver en ekstra mulighed for at forstå hvad hinanden mener.

Denne visuelle repræsentation kan yderligere øge muligheden for divergent tænkning. Særligt mindmaps er en almindelig teknik til at øge den divergente tænkning [Olson & Olson, 2000].

For at et diagrammeringsværktøj for alvor skal være brugbart i et distribueret gruppemøde, bør værktøjet som minimum understøtte "contemporality" og gerne også "simultaneity" på samme måde som vores mødeadministrationsværktøj. Dette er for at understøtte kreativitet. Hvis et gruppemedlem tegner en model i et diagrammeringsværktøj, bør det være muligt for de andre gruppemedlemmer at kommentere og ændre/tilføje til den pågældende model "on the fly" – ligesom man ville gøre hvis man var i samme rum, og lavede modeller på en tavle. Yderligere kunne et IT-diagrammeringsværktøj have den fordel også at have "sequentiality", "reviewability" samt "revisability". Det ville altså være muligt at gemme tegningerne efterhånden som de bliver redigeret, og det ville være muligt at kunne finde dem frem igen, for at kunne se hvordan modellen havde udviklet sig over tid, og det ville være muligt at et enkelt gruppemedlem justerede sin model, inden den blev "frigivet" så de andre gruppemedlemmer kunne se den. Dette ville være en ekstra "kompetence" i forhold til for eksempel en tavle eller whiteboard, hvor det ikke umiddelbart er muligt at gemme produktet efterhånden som det bliver udviklet.

Vi har på nuværende tidspunkt ikke kendskab til et nemt tilgængeligt værktøj der har den funktionalitet vi efterspørger. Derfor har vi ikke anbefalet et produkt til dette.

7.5 Planlægningsværktøjet

Den samlede EMS pakke bør også indeholde et planlægningsværktøj, såsom en kalender og/eller en tidslinie. Et planlægningsværktøj med en tidslinie vil eksempelvis være med til at fastholde den fælles forståelse for målene i projektet, hvilket giver et bedre og mere fokuseret samarbejde.

Fordele ved at planlægge og koordinere projekter er en diskussion det ikke vil være muligt, indenfor rammerne af dette projekt, at komme ind på her. Vi vil dog påstå at det vil være en fordel at benytte et planlægningsværktøj, og dette planlægnings-værktøj vil med fordel kunne integreres med mødeadministrationsværktøjet.

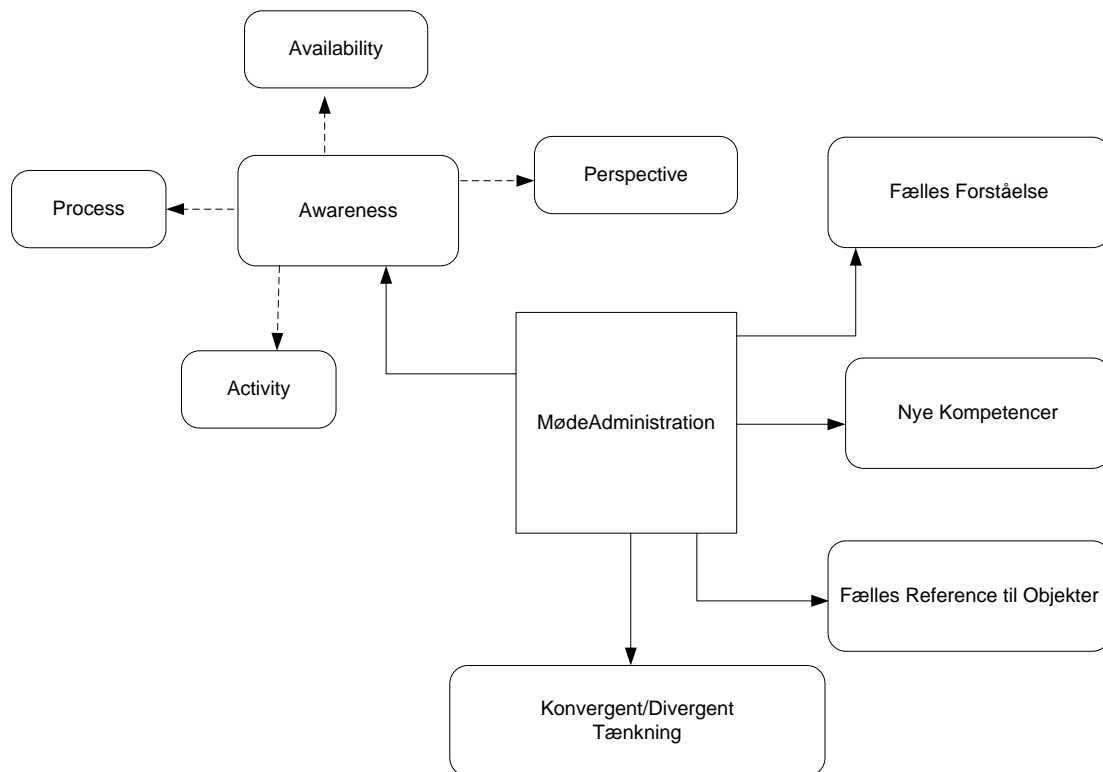
Det efterfølgende afsnit vil beskæftige sig dybere med mødeadministrationsværktøjet, ligesom vi også her vil tage fat på det vi ser som et fornuftigt samspil mellem planlægningsværktøjet og mødeadministrationsværktøjet.

8 Mødeadministrations design

Som i afsnit '5.1 Problematik ved distribuerede gruppemøder' kan det være svært at skabe overblik og koordinere arbejde i et distribueret samarbejde. Derfor er det yderst vigtigt at forbedre gruppemedlemmernes muligheder for at indgå i et virtuelt samarbejde [Grudin, 1994].

Vores fokus i projektet har derfor taget udgangspunkt i denne problematik, og vi har derfor som hovedpunktet i vores produkt valgt at beskrive og udvikle et mødeadministrationsværktøj. Værktøjet er en implementering af vores teoretiske erkendelser omkring hvordan man skaber et værktøj der forbedrer de teknologiske muligheder for synkront samarbejde over netværk. Mødestyringsværktøjet gør det muligt at lave referater og uddelegere opgaver imens mødet står på.

I den efterfølgende figur har vi lavet en oversigt over de teoretiske begreber vi umiddelbart knytter til vores design. Selve begreberne står beskrevet i vores teori, og hvorledes vi ser dem benyttes bliver forklaret i de efterfølgende afsnit.



Figur 3: En oversigt over de teoretiske begreber som mødeadministration anvender.

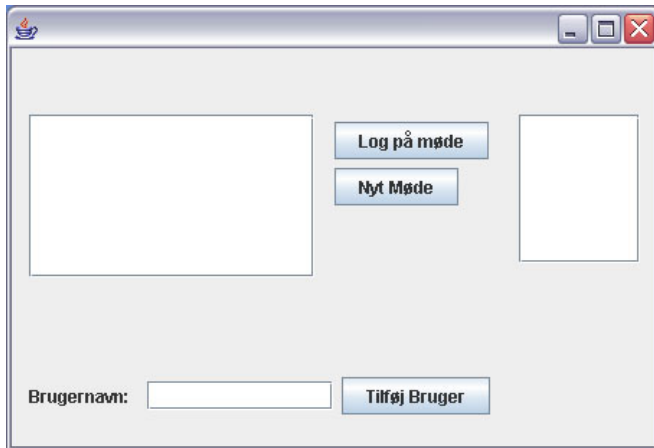
Hele vores design kredser om referatfunktionen og muligheden for at kunne udfærdige et skrift synkront, hvor alle gruppemedlemmers input kan komme i betragtning samtidig med at beslutningerne tages, mens produktet skabes. Det er denne mulighed vi betragter som at være den ”nye kompetence” vi tilføjer til de allerede eksisterende samarbejdsværktøjer. Det er en tilføjelse af ”Fælles reference til objekter”, hvor objekter værende mødereferatet. Dette gør det nemmere at skabe ”fælles forståelse”, enighed omkring de beslutninger der bliver taget og overblik over hvad der bliver lavet. Her har vi lagt vægt på, at det er de synkroner møder vi understøtter, hvor vi har fokus på at programmets data (referater og opgaver) opdateres synkront og kontinuerligt på alle computere således at det giver nemt overblik over hvad de forskellige brugere laver. Brugeren behøver ikke at gemme data for at de andre skal se det, programmet sørger selv for at opdatere alle klienter hele tiden. Dette gør at brugerne nemt og hurtigt kan rette fejlagtige opfattelser og opnå ”fælles forståelse”. Samtidig betyder det, at alle konstant kan se hvad der sker, at det bliver nemmere at arbejde reflektivt – det vil sige, revurdere og ændre gruppens oprindelige arbejde. Når referatet skrives vil de enkelte gruppemedlemmer løbende tage stilling til om referatet repræsenterer deres synspunkter og det gruppen bliver enige om. På den måde vil referatet hjælpe gruppen med at blive opmærksom på om der er erkendelsesmæssige konflikter i gruppen, så disse kan blive diskuteret igennem og der kan opnås en bedre og bredere ”fælles forståelse”.

Mødestyringsværktøjet vil primært understøtte den konvergente del af en kreativ arbejdsproces. Det er vigtigt for os at mødestyringsværktøjet kan integreres med planlægningsværktøjet således at gruppemedlemmerne kan opnå synergieffekterne ved værktøjer der aktivt arbejder sammen [Grudin, 1994].

8.1 Gennemgang af Mødeadministrationsværktøjet

Dette afsnit er en beskrivelse af programmet, med eksemplificeringer af hvordan vi mener awareness bør implementeres i designet. Værktøjet er en klient-server løsning hvor der startes en server og flere klienter kan så forbinde til denne over internettet.

Når mødeadministrationsværktøjet startes vil skærbillede 1 fremkomme. Eksemplet viser en tom mødeadministration, det vil sige lige når serveren er startet for første gang og der ikke er oprettet nogen møder i systemet.



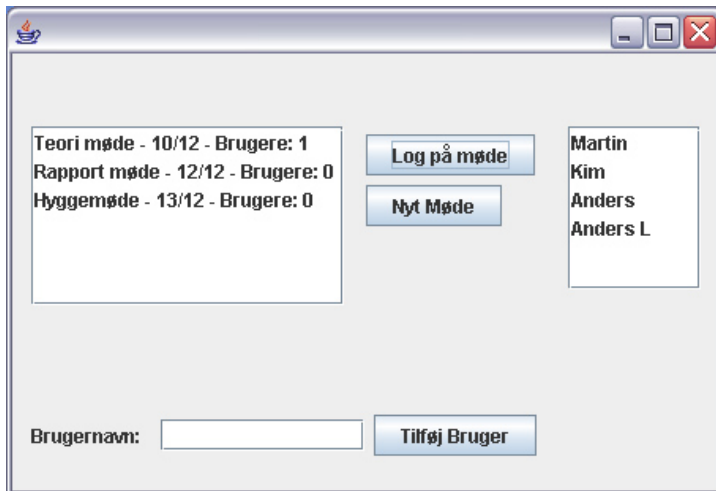
Skærbillede 1: Den tomme mødeadministration

Der kan i dette tilfælde kun oprettes et nyt møde eller en ny bruger, da der ikke er nogen tilgængelige møder at logge på. Var der tidligere oprettede møder i projektarbejdet, ville de være tilgængelige i listen. Et nyt møde oprettes ved at klikke på Nyt Møde knappen der åbner skærbillede 2.



Skærbillede 2: Oprettelse af møde med titel og dato.

Der er mulighed for at oprette et møde med titel og dato. Mødet oprettes i det øjeblik der trykkes på Opret Møde knappen. Alternativt kan der fortrydes ved at trykke på Annuller. Når et møde er oprettet vil det kunne ses i feltet til venstre, hvilket kan ses på skærbillede 3.



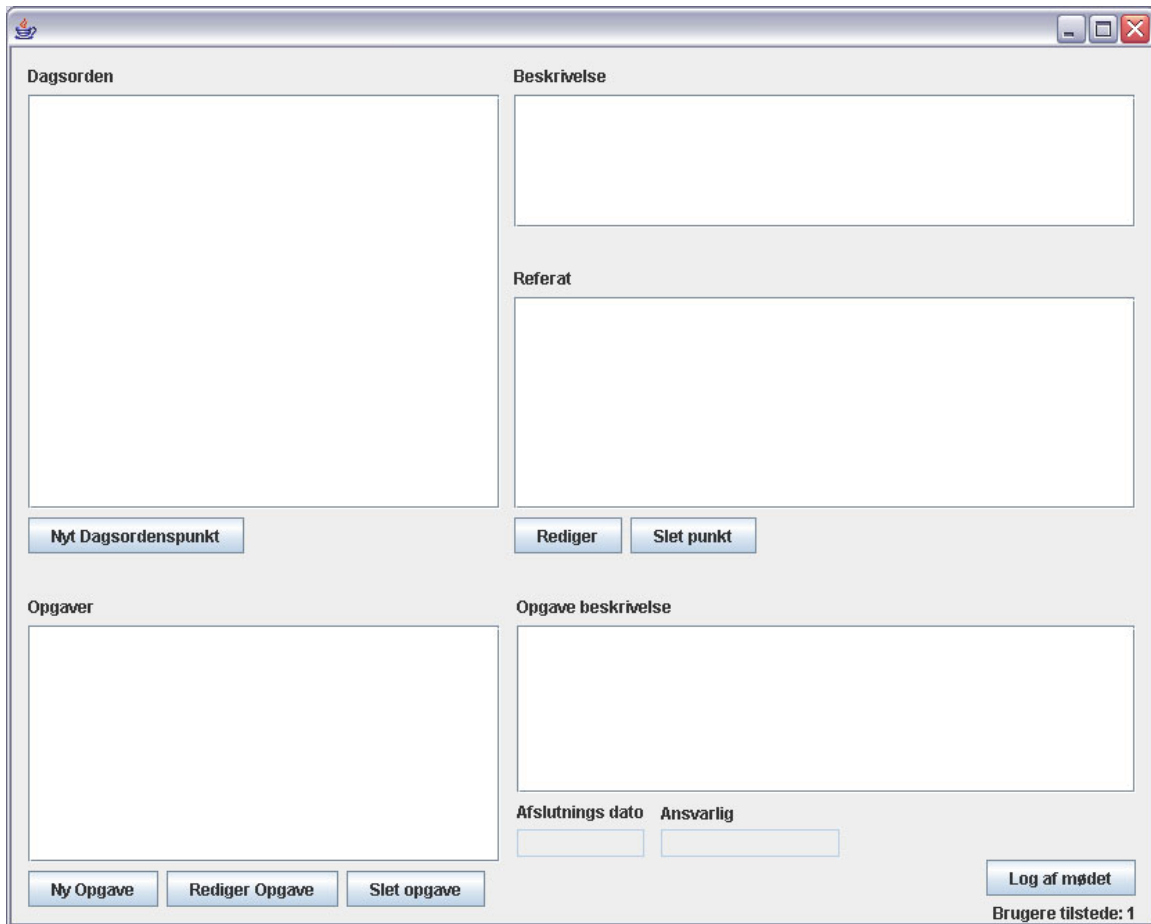
Skærbillede 3: Mødeadministrationen med tre oprettede møde og flere brugere er logget på.

Som vist på skærbilledet er der for et møde en titel, en dato og antallet brugere der er logget på det specifikke møde. Denne opstilling skaber activity og availability awareness om de andre medlemmer i projektet. Yderligere gives process awareness om fremtidige møder da der i oversigten bliver skabt overblik med hensyn til den fremtidige proces.

Det viste teorimøde har således en bruger logget på, men der er fire brugere på systemet i alt som alle er navngivet. Denne fremstilling er valgt for at give activity awareness til brugerne af systemet da det giver overblik over hvilke brugere der er på systemet og hvilke møder der er aktivitet i.

Availability awareness kommer til udtryk ved at man kan se hvilke andre brugere der er på systemet og klar til at deltage i møder.

I skærbilledet er der mulighed for at vælge et møde og derefter klikke på Log på Møde knappen, hvorved detaljerne for selve mødet bliver vist som på skærbillede 4.



Skærbillede 4: Tomt møde med kun en bruger logget på.

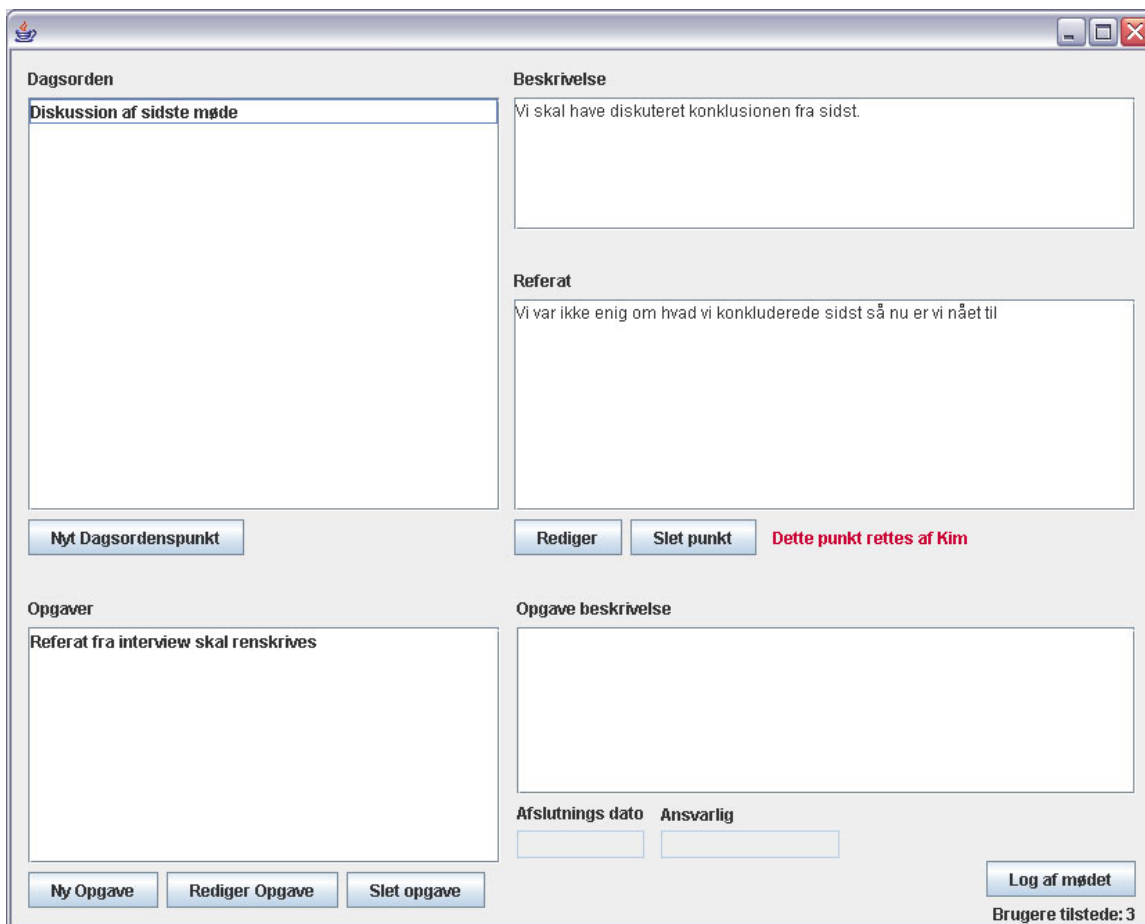
Når første bruger logger på et nyt møde vil det være tomt som overfor. Vinduet er delt op i to dele, den øverste del viser til venstre en liste over dagsordens punkter og til højre er der øverst en kort beskrivelse af punktet, dette udfyldes når punktet bliver oprettet. Under denne er der selve referatet. Et klik på "Nyt Dagsordenspunkt" vil bringe skærbillede 5 frem.

At der i bunden af skærbillede 4 er vist antallet af bruger på mødet er yderligere et eksempel på activity og availability awareness, da der gives overblik over at andre deltager i aktiviteten og der hermed indikeres hvad de foretager sig (De deltager i mødet).

The screenshot shows a standard Windows-style dialog box with a title bar containing a small icon on the left and minimize, maximize, and close buttons on the right. The dialog is titled "Overskrift" (Title) and contains a dropdown menu with the number "1" and a text input field containing "Diskussion af sidste mød". Below this is a section titled "Beskrivelse af opgaven" (Task description) with a large text area containing the text "Vi skal have diskuteret konklusionen fra sidst." At the bottom of the dialog are two buttons: "Opret" (Create) and "Annuller" (Cancel).

Skærbillede 5: Oprettelse af nyt punkt.

Her kan brugeren udfylde møde navnet og beskrivelsen, samt vælge placeringen i dagsordenen.



Skærbillede 6: En bruger retter i et dagsordenspunkt.

Nu er der så tilføjet et dagsordens punkt og en opgave til mødet. Man kan også i nederste højre hjørne på skærbillede 6 se at der er logget tre brugere på mødet. Der er en af brugerne der er i gang med at skrive på referatet, hvilket er vist med en rød tekst "Dette punkt rettes af Kim" ved siden af "Slet Punkt" knappen. Dette er et udtryk for passiv activity awareness, da det giver indsigt i hvad brugeren Kim foretager sig uden at denne selv skal gøre andet end at arbejde på den pågældende opgave. Process awareness kommer til udtryk ved at de andre brugere kan se hvad brugeren Kim skriver, mens han skriver det, samt at der er overblik over hvad de andre punkter i processen drejer sig om. Når der er en anden bruger der skriver vil det ikke være muligt for andre at skrive i referatet til dette punkt, men en anden bruger kan godt rette et andet punkt samtidigt. Punkter og opgaver kan også slettes. Hvis der oprettes en opgave på baggrund af et referat, så giver det den enkelte bruger mulighed for at gå tilbage og se grundlaget for opgaven. Dette giver perspective awareness, da gruppen nu kan danne sig et overblik

over de møder og opgaver der oprettes, samt finde det referat som forklarer hvorfor opgaven blev oprettet til at begynde med.

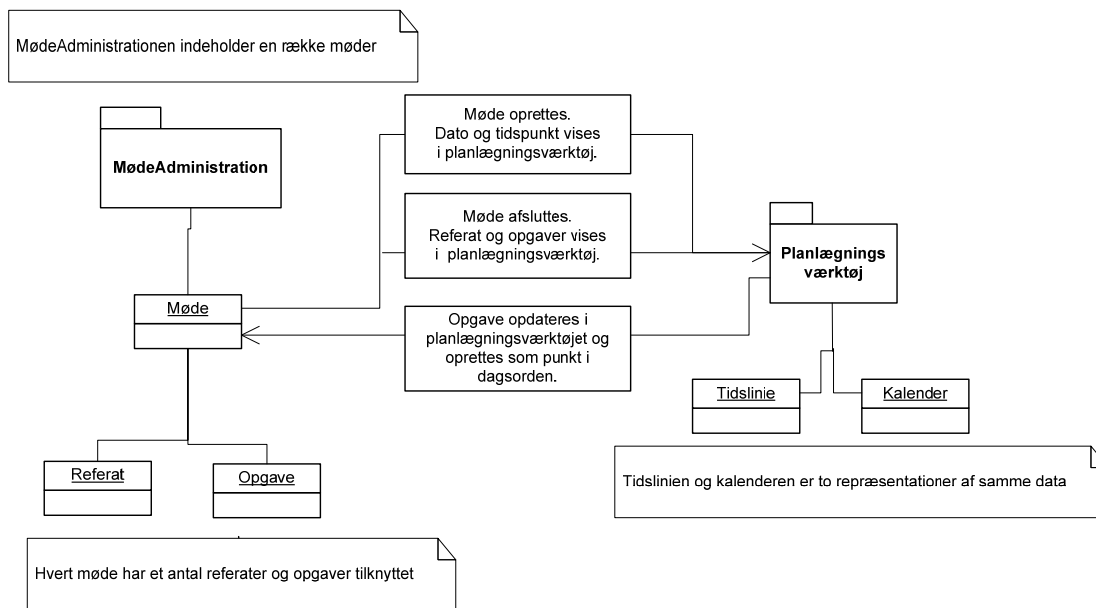
8.2 Integrationen med planlægningsværktøjet

I modsætning til mødeadministrationsværktøjet, så har vi ikke lavet nogen prototype af dette værktøj, ligesom vi ikke har designet de funktioner et sådan værktøj kunne indeholde. Vi har dog gjort os overvejelser omkring det sammenspil vi mener der skal være mellem mødeadministrationen og et planlægningsværktøj. Det skal være en fuldt ud integreret del sammen med mødeadministrationen, således at de oprettede møder automatisk overføres til planlægningsværktøjet. Møderne plottes automatisk ind i kalenderen/ tidslinien på det pågældende tidspunkt, og når et møde afsluttes vil alle de opgaver der er oprettet under mødet også kunne ses på tidsplanen. På den måde vil tidsplanen give et hurtigt overblik over hvad gruppen har aftalt at arbejde videre med, og hvornår en given opgave skal være færdig. Det vil samtidig være nemt for det enkelte gruppemedlem, at se hvem der er i gang med hvilken opgave, og hvornår den kan forventes at være færdig. Altså Process og Activity awareness omkring hvad de andre gruppemedlemmer arbejder med. Derudover er det også tiltænkt at når en bruger fuldfører en opgave eller forhindres i dette, grundet særlige omstændigheder, så kan brugeren vælge i planlægningsværktøjet, at opgaven skal med på et fremtidigt møde i mødeadministrationsværktøjet. Opgaven vil så automatisk blive overført til dagsordenen på det pågældende møde. Det er dette sammenspil som vi fokuserer på når vi nævner planlægningsværktøjet; det at mødeadministrationen modtager og afgiver information til et planlægningsværktøj giver vores mødeadministrationsværktøj mulighed for bedre at kunne støtte en projektgruppe i deres mødeplanlægning og i deres gruppemøder generelt.

De to værktøjer skal ses som to selvstændige programmer, men med den integration, at hver gang der oprettes opgaver til et møde, så overføres disse opgaver til planlægningsværktøjet. Når vi siger at planlægningsværktøjet er et selvstændigt program, så mener vi at det enten kan udvikles som et nyt modul, som er kompatibelt med vores mødeadministration, eller at det kan være et allerede udviklet produkt, som for eksempel

Microsoft Outlook/Exchange. Til at understøtte forbindelsen mellem de to programmer, forudser vi at vores mødeadministration skal kunne gemme/sende møder og opgaver i formater som er kompatible med de mest almindelige kalendere og planlægningsværktøjer.

Den efterfølgende figur viser forholdet mellem mødeadministrationen og planlægningsværktøjet i forhold til hvad det er for nogle objekter vores produkt sender til planlægningsværktøjet.



Figur 4: Viser sammenhængen mellem mødeadministration og planlægningsværktøjet.

8.3 Systembeskrivelse

Når vi nu har beskrevet design ideerne og de teorier og overvejelser som har ført til dem, finder vi det nu relevant at give en meget kort oversigt over hvordan vi har valgt at bygge systemet op. Vi vil indledningsvis beskrive hvordan vores prototype er bygget op, og derefter forklare hvad det er det giver os af fordele.

Prototypen er opbygget således at der er en central server der varetager datastyringen og klienten ikke selv skal varetage at gemme noget data lokalt. Således vil alle have mulighed for at tilgå de data serveren varetager uagtet hvor de måtte befinde sig - også selvom de ikke befinder sig på deres egen computer. Dette understøtter det distribuerede

samarbejde yderligere ved at frigøre deltagerne fra specifikke lokationer. Den programmerede del af systemet er implementeret i Java ved hjælp af RMI og er således platformuafhængigt, hvilket stemmer godt overens med vores tanker omkring at det skal være muligt for den enkelte bruger selv at vælge hvilket system og programmer han vil bruge. Platformuafhængigheden og brugen af en central server gør det muligt, såfremt man gør klientdelen tilgængelig på Internettet, at tilgå mødeadministration fra alle steder med Internet – bredbåndsforbindelser er dog at fortrække. Dette hænger sammen med 'Access anywhere', i form af en tilnærmet universal brugbarhed og det vil være muligt at tilgå Mødeadministrationen fra hvilken som helst computer med Internet.

9 Fokuseret eksperiment (eksperiment 2)

For at afprøve om vores program har været med til at give os svar på vores problemformulering har vi gentaget det første forsøg men denne gang hjulpet af vores eget udviklede værktøj – mødeadministrationen. Hvor fokus i det udforskende eksperiment var, at udforske hvilke problemer der var ved afholdelsen af et distribueret møde har vi i det fokuserede eksperiment særligt for øje hvad vores mødeadministrationsværktøj har af betydning, og hvad det kan bidrage med. Vi vil altså her prøve at afdække om værktøjet rent faktisk gør mødet nemmere at gennemføre og/eller giver os et bedre udbytte af mødet.

9.1 Forsøgsrammen

Opgave: At gennemføre et gruppemøde med følgende dagsorden, med samme programpakke som i det udforskende eksperiment, dog suppleret af vores mødeadministrationsværktøj.

Dagsorden

- Vi fremlægger hver især en status på hvad vi har nået fra i går.
- Diskussion af planen for resten af dagen.
- Uddelegering af opgaver til deltagerne.

Tidspunkt og varighed: 12/12/06 kl. 13 til 16

Deltagere: Anders Lorentz Hansen, Anders Nielsen, Kim Sonne og Martin Schultz

Placering: To var placeret i deres private hjem, to var på RUC.

Teknisk setup: En benyttede Linux, OpenSuse og de andre tre Windows. Skype blev benyttet til audio kommunikation, Kopete/Messenger som chat (hvis lyden skulle svigte), BSCW som fildelingsværktøj. Alle havde en bredbåndsforbindelse.

Vi havde lavet en jar-fil, som forbandt til vores server, og loggede på mødeadministrationsværktøj.

Efter eksperimentet - spørgsmål

For at kunne evaluere på eksperimentet efterfølgende, valgte vi inden eksperimentet at udforme syv spørgsmål, som hver af gruppedlemmerne skulle besvare når eksperimentet var blevet afholdt. Dette blev gjort for at få en så præcis beskrivelse som muligt, af hvordan mødet blev oplevet af gruppedlemmerne.

Spørgsmålene lød som følger;

- Hvad var din generelle opfattelse af værktøjet?
- Hvad var den største fordel ved at benytte værktøjet?
- Hvad var den største ulempe ved at benytte værktøjet?
- Hvilke behov/mangler/fejl oplevede du?
- Var mødet bedre end det første møde/eksperiment? Hvordan?
- Hvad synes du var den vigtigste konklusion fra dette møde?
- Hvad tror du de andre tre personer synes var den vigtigste konklusion?

De første fem spørgsmål går mest på selve oplevelsen af programmet, mens de sidste to er for at se i hvor høj grad gruppen deler den "fælles forståelse", som vi håber programpakken kan bidrage til at øge.

9.2 Mødeforløb

Ligesom til det udforskende eksperiment havde vi problemer med at få lyden til at gå klart igennem hos alle, og ligesom sidst var det her godt at kunne bruge instant messaging til at kommunikere med indtil lyden var oppe og køre. Vi havde på forhånd aftalt en dagsorden, men denne blev svagt justeret inden vi for alvor gik i gang med mødet. Da prototypen ikke kunne persistere referatet (Det er gemt i serverens hukommelse, men når server programmet lukkes forsvinder alt data), aftalte vi en person der var ansvarlig for at gemme på sin maskine lokalt, hvad vi skrev i referatet efter mødet var overstået. Vi aftalte yderligere ved hvert dagsordenpunkt hvem der skrev i referatet. Det er i vores

system muligt, at alle kan skrive og redigere i referatet – men vi oplevede det som nyttigt at det var en person der havde ansvaret for at der blev skrevet noget i referatet.

9.3 Evaluering af det fokuserede eksperiment

I dette afsnit vil vi gennemgå de erfaringer vi har gjort i vores fokuserede eksperiment. Vi vil forholde os til hvilke problematikker der var i vores udforskende eksperiment og sammenholde det, med den oplevelse vi havde af vores fokuserede eksperiment. Formålet er at give en vurdering af, hvorvidt vores mødeadministrationsværktøj har løst nogle af de problemstillinger, som vi oplevede i det udforskende eksperiment, samt de teoretiske overvejelser.

9.3.1 Oplevelse af det fokuserede eksperiment

Efter det fokuserede eksperiment besvarede alle gruppemedlemmer de forberedte spørgsmål omkring gruppemødet. Hovedkonklusionerne fra disse spørgsmål er at vores prototype af Mødeadministrationsværktøjet stadig mangler en del tilpasning. Det var en række ”irritationsmomenter” ved systemet, som gør at det endnu ikke er optimalt at bruge ved et distribueret møde. Dog var det også oplevelsen at på trods af disse fejl og mangler i programmet, så kunne prototypen stadig bruges, og alle gruppemedlemmer oplevede et potentiale i programmet.

Nedenfor følger en hurtig oversigt over de oplevede problemer samt fordele ved systemet.

9.3.2 Problemer

- Lang reaktionstid når serveren opdaterer, da der er et fast opdateringsinterval når der sker en ændring i objekterne, i stedet for en flydende opdatering.
- referatfeltet for lille, så man ikke kan se hele referatet.
- awareness mangler, for eksempel;

- kan man ikke se hvem der retter hvor, medmindre man klikker på hvert enkelt punkt.
- kan man ikke se hvem der er logget på mødet, kun hvor mange der er logget på.
- vi oplevede en enkelt gang at systemet frøs, da der var to inde og redigere et dagsordenspunkt – disse to måtte logge af mødet for at komme på igen
- pga. måden serveren opdaterer, bliver det dagsordenspunkt man kigger på ”ufokuseret” hver femte sekund og det betød at hvis referatet var større end vinduet så skulle man ”scrolle” ned til der hvor man var, hvert femte sekund.
- referatet kunne ikke gemmes permanent i et filsystem

9.3.3 Fordele

- programmet giver en bedre ”fælles forståelse” i gruppen, da alle netop kan se hvad der bliver skrevet i referatet
- programmet ”tvinger” gruppen til at lave dagsordner og faktisk følge dem
- hvis programmet benyttes, sikres det at gruppen har et referat og en opgaveliste efter hvert møde – nedskrevet. Dette giver en bedre og bredere forståelse for de fælles målsætninger

9.3.4 Konklusion på det fokuserede eksperiment

Løste mødeadministrationsværktøjet nogle af de problemer vi oplevede i det udforskende eksperiment? Både ja og nej. Der er ingen tvivl om, at mødeadministrationsværktøjet var en bedre måde at fastholde gruppemødets beslutninger på end chat-funktionen som vi benyttede i det udforskende eksperiment. Det at referatet kunne redigeres, mens alle kunne se selve redigeringen, var en funktion vi oplevede som brugbar. Vi oplevede at det var nemmere at nå frem til en beslutning, fordi beslutningen blev præsenteret i referatet. På den måde oplevede vi mødeadministrationsværktøjet som den formelle kommunikation, og Skype som den kulturelle, som vi har beskrevet i afsnittet ”Samlet

EMS design”. Det gav os altså en bedre kommunikation at vi havde denne fælles reference i referatet som vi kunne bruge i vores samtale/diskussion.

Dog var der en række problemer med vores prototype, som gjorde at systemet ikke var optimalt at bruge og hvis systemet skal kunne bruges, skulle disse mangler udbedres. Dette var ikke i så høj grad i forhold til funktionaliteten, men mere selve implementationen der haltede.

10 Konklusion

Vi har nu gennemgået en række teoretiske overvejelser, vi har lavet to eksperimenter og vi har designet og programmeret en prototype for at forsøge at besvare vores problemformulering.

Hvordan kan man forbedre de teknologiske muligheder for at afholde synkrone gruppemøder mellem geografisk adskilte studerende på universiteter?

I vores udforskende eksperiment blev det klart hvilken type af værktøj vi fandt brugbart og som ville kunne være med til at forbedre mulighederne for synkrone distribuerede gruppemøder. Dette værktøj har vi på baggrund af vores teoretiske erkendelser designet og programmeret. Værktøjet som hedder ”Mødeadministration” er en del af en samlet programpakke af allerede eksisterende programmer. På den måde vil det ikke give mening at benytte Mødeadministration uden de andre programmer fra programpakken. Det er nødvendigt med f.eks. verbal kommunikation samtidig med at Mødeadministration bruges. Vi har desuden foreslået en række konkrete produkter som repræsenterer de værktøjer vi mener, vil være nødvendige ved distribuerede gruppemøder.

I designet af vores Mødeadministration har vi lagt vægt på at vores værktøj skal understøtte en række elementer fra vores teoretiske begrebsapparat. Vores program skal være med til at give gruppemedlemmerne en større ”fælles forståelse” og give dem

muligheden for at arbejde reflektivt og understøtte konvergent tænkning. Hvis disse ting understøttes specifikt i programmet, så mener vi at programmet kan være med til at forbedre mulighederne for distribuerede gruppemøder.

Rent praktisk er Mødeadministration et værktøj, der for hvert af de møder en gruppe afholder, indeholder en række dagsordenspunkter og opgaver. Under selve mødet har gruppemedlemmerne mulighed for at skrive referat til hvert af dagsordenspunkterne. Mens et gruppemedlem skriver referatet kan samtlige andre gruppemedlemmer se hvad der bliver skrevet, og gruppen har dermed mulighed for at kommentere. Alle gruppemedlemmer har ligeledes mulighed for at redigere i alle punkter. Referaterne bliver på den måde et objekt som udgør en fælles reference mellem gruppemedlemmerne og disse objekter og referencer skabes under selve mødet. Dette har vi kaldt den niende dimension i forhold til at opnå en ”fælles forståelse”. Denne dimension bruges ofte i face-to-face møder, men er altså ikke en dimension der er normal i forhold til distribueret samarbejde. Hvis hele vores programpakke benyttes vil samtlige af de andre dimensioner også være opfyldt (selvfølgelig undtaget den der siger at personerne sidder fysisk i samme rum). Vi mener på den baggrund at vores programpakke og særligt vores Mødeadministration er med til at forbedre den ”fælles forståelse”.

Understøttelsen at alle i gruppen konstant kan se hvad der bliver skrevet i referatet er yderligere med til at understøtte den reflektive tankegang. Gruppemedlemmerne bliver ”presset” til at tage stilling til hvad der står i referatet og på den måde revurdere og genoverveje om det er den retning gruppen bør gå. Et referat er derudover en almindelig måde at forsøge at tænke konvergent. Når den divergente tankegang har fået frit løb, er et referat netop sammenkøningen af gruppens ideer og det er i referatet samt de kommende opgaver, at det kan ses hvad gruppen beslutter og hvad der skal arbejdes videre med. Det styrker ligeledes gruppens fælles målsætninger.

I forhold til den grafiske brugergrænseflade, er awareness et begreb vi har haft godt fat i, da dette er særlig vigtigt i forhold til distribuerede samarbejdsværktøjer. Vores prototype implementerer ikke alle de ”awareness tiltag” som programmet burde, for at for alvor at

blive brugbart. Men vi mener at vi i rapporten vist, hvordan og hvad der skal til, hvis prototypen skulle færdigudvikles. Hvis gruppe-medlemmerne aldrig har mødt hinanden før, så vil dette aspekt være særligt vigtigt.

Eksperiment 2 er en evaluering af om vores mødeadministrationsværktøj bidrager med en forbedring af den distribuerede arbejdsform, der blev eksemplificeret i eksperiment 1. Den udviklede prototype var et forsøg på at teste den funktionalitet vi ønskede at tilføje til distribuerede gruppemøder. De implementeringsfejl og mangler prototypen besidder er på nuværende tidspunkt en begrænsning for en direkte anvendelse. Vi mener at en færdigudvikling af vores design med alle funktioner og awareness faciliteter implementeret, vil kunne forbedre distribuerede gruppemøder. Den fastholdelse af fælles forståelse og fælles målsætning som vi mener referatfunktionen medfører, kan benyttes ikke kun af studerende, men også af distribuerede grupper generelt. Derfor mener vi at rapporten skal ses som et svar på problemformuleringen; den giver et bud på et design som vi mener, er interessant for målgruppen og som vi selv kunne forestille os at anvende under senere projekter.

11 Litteraturliste

Primær litteratur

- [Dennis, et al. 1988], A. R. Dennis, J. F. George, L. M. Jessup, J. F. Nunamaker Jr., D. R. Vogel, Information Technology to Support Electronic Meetings, Management Information Systems Quarterly, Volume 12 , Issue 4 (December 1988)
- [Farooq, et al 2005], U. Farooq, J. M Carroll, C. H. Ganoe, Supporting Creativity in Distributed Scientific Communities, The Pennsylvania State University, Conference on Supporting Group Work - Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work, ACM Press (2005)
- [Grudin 1994], J. Grudin, Groupware and social dynamics: Eight challenges for developers, Communications of the ACM, (January 1994)
- [Olson & Olson 2000], G. Olson & J. Olson, Distance Matters, Human Computer Interaction Press (2000)
- [Robinson 1991], M. Robinson, Double-level language and Co-operative working, AI & Society, Volume 5 , Issue 1 (January-March 1991)
- [Steinfeld, et al. 1999], C. Steinfeld, C. Jang & B. Pfaff, Supporting virtual team collaboration: the TeamSCOPE system, Conference on Supporting Group Work archive Proceedings of the international ACM SIGGROUP conference on Supporting group work, ACM Press (1999)

Sekundær litteratur

- A. Malhotra & A. Majchrzak , Enabling Knowledge Creation in Far-Flung Teams: Best Practices for IT Support and Knowledge Sharing, Journal of knowledge Management, Vol 8 NO. 4 (2004)
- B. Shneiderman, Creating Creativity for Everyone: User Interfaces for Supporting Innovation, ACM Transactions on Computer-Human Interaction (TOCHI) archive Volume 7 , Issue 1, ACM Press (March 2000)
- M. Sumner & D. Hostetler, A Comparative Study of Computer Conferencing and Face-to-face Communications in Systems Design, Special Interest Group on Computer Personnel Research Annual Conference , Proceedings of the 2000 ACM SIGCPR conference on Computer personnel research, ACM Press (2000)
- M. Warkentin & P. M. Beranek, Training to Improve Virtual Team Communication, Info Systems J, Blackwell Science Ltd (1999)
- P. J. Hinds & M. Mortensen, Understanding Conflict in Geographically Distributed Teams: The Moderating Effects of Shared Identity, Shared Context, and Spontaneous Communication, Organization Science, Vol 16, NO. 3 (may-june 2005)

12 Bilag

MOEDEADMINISTRATION.JAVA 59

SERVERIFACE.JAVA 62

MOEDE.JAVA 63

MOEDEPUNKT.JAVA 66

DPUNKT.JAVA 68

OPGAVE.JAVA 68

BRUGER.JAVA 69

MOEDEADMINISTRATIONSJUL.JAVA 70

NETCONNECT.JAVA 74

MOEDEGUL.JAVA 75

OPRETNYOPGAVEGUL.JAVA 92

OPRETNYTDPUNKTGUL.JAVA 96

MoedeAdministration.java

```
1 package moedevaerktoej;  
2  
3 import java.io.Serializable;  
4 import java.rmi.RMI SecurityManager;  
5 import java.rmi.server.UnicastRemoteObject;  
6 import java.util.ArrayList;  
7 import java.rmi.*;  
8  
9 /**  
10  * MoedeAdministration er serverens hovedklasse og der køres først  
når serveren  
11  * startes. Implementere interfacet ServerIface  
12  */  
13 public class MoedeAdministration extends UnicastRemoteObject
```

```

14     implements ServerIface {
15     public Moede[] moeder = new Moede[50];
16     public int moedeTaeller = 0;
17     public int brugerTaeller = 0;
18     public Bruger[] brugerListe = new Bruger[100];
19     private String name;
20
21     /**
22     *Construtoren for klassen. Er en af de RMI eksporterede
funktioner så den
23     *kaster en undtagelse
24     */
25     public MoedeAdministration(String s) throws java.rmi.RemoteException{
26         name = s;
27     }
28
29     /**
30     *Opretter et nyt møde og ligger det på listen over møder
31     */
32     public void nytMoede(Bruger bruger, String titel, String
moedeDato){
33         Moede nytMoede = new Moede(bruger, titel, moedeDato);
34         moeder[moedeTaeller] = nytMoede;
35         moedeTaeller++;
36     }
37     /**
38     *Henter et møde ud
39     */
40     public Moede hentMoede(int i, Bruger b){
41         return moeder[i];
42     }
43     /**
44     *Opretter en ny opgave
45     */
46     public int nyOpgave(String titel, int moedeNR){
47         moeder[moedeNR].opretOpgave(titel);
48         return moeder[moedeNR].opgaveTaeller;
49     }
50     /**
51     *Henter en opgave ud til klienten
52     */
53     public Opgave hentOpgave(int moedeNR, int opgaveNummer){
54         return moeder[moedeNR].opgaver[opgaveNummer];
55     }
56
57     /**
58     *En opgave er redigeret og skal skrives tilbage til serveren
59     */
60     public void skrivOpgave(int moedeNR, int opgaveNummer, Opgave o){
61         moeder[moedeNR].opgaver[opgaveNummer] = o;
62     }
63     /**
64     *Et Dpunkt er redigeret og skal skrives tilbage til serveren
65     */
66     public void skrivDPunkt(int moedeNR, int dpunktNummer, dPunkt d){

```

```

67         moeder[moedeNR].dPunkter[dpunktNummer] = d;
68     }
69     /**
70      *Opretter et nyt dpunkt
71      */
72     public void nytDPunkt(String titel, String beskrivelse, int
moedeNR){
73
74         moeder[moedeNR].opretDPunkt(titel, beskrivelse);
75     }
76     /**
77      *Henter et dpunkt fra serveren
78      */
79     public dPunkt hentDPunkt(int moedeNR, int dpunktNummer){
80         return moeder[moedeNR].dPunkter[dpunktNummer];
81     }
82
83     /**
84      *Skriver et møde til serveren
85      */
86     public void afleverMoede(Moede m, int i){
87         Moede temp = m;
88         moeder[i] = temp;
89     }
90
91     /**
92      *Returnere over møderne med titel, dato og antal brugere der
er logget på
93      */
94     public ArrayList<String> moedeListe(){
95         ArrayList<String> temp = new ArrayList();
96         for(int i = 0; i < moedeTaeller; i++ )
97             temp.add(moeder[i].getTitel() + " - " + moeder[i].getDato()
+ " - Brugere: " + moeder[i].getDeltagerAntal());
98         return temp;
99     }
100    /**
101     *Skriver en liste af brugere på systemet ud
102     */
103    public ArrayList<String> brugerListe(){
104        ArrayList<String> temp = new ArrayList();
105        for(int i = 0; i < brugerTaeller; i++ )
106            temp.add(brugerListe[i].getNavn());
107        return temp;
108    }
109
110    /**
111     *Opretter en ny bruger
112     */
113    public void nyBruger(String n){
114        brugerListe[brugerTaeller] = new Bruger(n);
115        brugerTaeller++;
116    }
117
118    /**

```

```

119     *Logger en bruger på et specifikt møde
120     */
121     public void brugerTilMoede(int b, int m){
122         moeder[m]. tilfoejDeltager(brugerListe[b]);
123     }
124
125     /**
126     *Sletter en bruger
127     */
128     public void fjernBruger(int b, int m){
129         moeder[m]. fjernBruger(brugerListe[b]);
130     }
131
132     /**
133     *Henter en bruger ud
134     */
135     public Bruger getBruger(int i){
136         return brugerListe[i];
137     }
138
139     /**
140     *Returnere antallet af møder
141     */
142     public int getAntal(int m){
143         return moeder[m]. getDeltagerAntal();
144     }
145
146     /**
147     *Starter programmet og registrere det med rmiregistry
148     */
149     public static void main(String args[]) {
150         System.setSecurityManager(new RMISecurityManager());
151         try {
152             MoedeAdministration obj = new
153             MoedeAdministration("StartServer");
154             Naming.rebind("StartServer", obj);
155             System.out.println("Serveren er startet....");
156         } catch (Exception e) {
157             e.printStackTrace();
158         }
159     }
160 }
161

```

ServerIface.java

```

1  /*
2  *Serveriface interfacet "eksportere" de funktioner som klienten kan
3  kalde via
4  *RMI. MoedeAdministration implementere dem.
5  */
6  package moedevaerktoej;

```

```

7 import java.util.ArrayList;
8
9 public interface ServerIface extends java.rmi.Remote {
10
11     void nytMoede(Bruger bruger, String titel, String moedeDato) throws
java.rmi.RemoteException;
12     void skrivDPunkt(int moedeNR, int dpunktNummer, dPunkt d) throws
java.rmi.RemoteException;
13     dPunkt hentDPunkt(int moedeNR, int dpunktNummer) throws
java.rmi.RemoteException;
14     Moede hentMoede(int i, Bruger b) throws java.rmi.RemoteException;
15     void afleverMoede(Moede m, int i) throws java.rmi.RemoteException;
16     ArrayList<String> moedeListe() throws java.rmi.RemoteException;
17     ArrayList<String> brugerListe() throws java.rmi.RemoteException;
18     void nyBruger(String navn) throws java.rmi.RemoteException;
19     Bruger getBruger(int i) throws java.rmi.RemoteException;
20     void nytDPunkt(String titel, String beskrivelse, int moedeNR) throws
java.rmi.RemoteException;
21     Opgave hentOpgave(int moedeNR, int opgaveNummer) throws
java.rmi.RemoteException;
22     void skrivOpgave(int moedeNR, int opgaveNummer, Opgave o) throws
java.rmi.RemoteException;
23     int nyOpgave(String titel, int moedeNR) throws
java.rmi.RemoteException;
24     int getAntal(int m) throws java.rmi.RemoteException;
25     void brugerTilMoede(int b, int m) throws java.rmi.RemoteException;
26     void fjernBruger(int b, int m) throws java.rmi.RemoteException;
27
28
29 }
30

```

Moede.java

```

1 package moedevaerktoej;
2
3 import java.io.Serializable;
4 import java.util.ArrayList;
5 /**
6  *Moede er den klasse der styre alt omkring et specifikt moede.
7  *Brugere, referater og opgaver
8  */
9 public class Moede implements Serializable {
10     ArrayList<Bruger> moedeDeltagere = new ArrayList();
11     public Opgave[] opgaver = new Opgave[50];
12     public dPunkt[] dPunkter = new dPunkt[50];
13     public int dPunktTaeller = -1;
14     public int opgaveTaeller = -1;
15     private String titel = "";
16     private String moede_dato;
17     private Bruger rettet_af;

```

```

18     private String rettet_den;
19
20     /**
21      *Opretter en ny opgave og tilføjer den til moedets liste over
opgaver
22      */
23     public void opretOpgave(String titel){
24         Opgave temp = new Opgave(titel, opgaveTaeller+1);
25         opgaver[opgaveTaeller+1] = temp;
26         opgaveTaeller++;
27     }
28
29     /**
30      *Opretter et nyt DPunkt (dagsordens punkt) og tilføjer det til
moedets
31      * liste over DPunkter
32      */
33     public void opretDPunkt(String titel, String beskrivelse){
34         dPunkt temp = new dPunkt(titel, dPunkter.length,
beskrivelse);
35         dPunkter[dPunktTaeller+1] = temp;
36         dPunktTaeller++;
37     }
38
39     /**
40      *Henter en opgave ud til redigering af en klient. Sørger for
den kun kan
41      *redigeres hvis der ikke er en anden der redigere den.
42      *Returnerer null hvis brugeren ikke må rette ellers returnere
den opgaven.
43      */
44     public Opgave hentOpgave(int n, Bruger b){
45         if(opgaver[n].redigerbar(b)){
46             System.out.println("Brugernavn: " + b.getNavn());
47             return opgaver[n];
48         }
49         return null;
50     }
51
52     /**
53      *Henter et dpunkt ud til redigering af en klient. Sørger for
den kun kan
54      *redigeres hvis der ikke er en anden der redigere den.
55      *Returnerer null hvis brugeren ikke må rette ellers returnere
den dpunktet.
56      */
57     public dPunkt hentDPunkt(int n, Bruger b){
58         if(dPunkter[n].redigerbar(b)){
59             return dPunkter[n];
60         }
61         System.out.println("hentopgaver er FALSE - sur r&v");
62         return null;
63     }
64
65

```

```

66     /**
67     *Returnere en opgave når en klient er færdig med at rette i
den
68     *og gør den redigerbar ved at sætte retter til null.
69     */
70     public void afleverOpgave(Opgave o){
71         o.retter = null;
72         opgaver[o.getNummer()-1] = o;
73     }
74 }
75
76 /**
77 *Returnere et dpunkt når en klient er færdig med at rette i
den
78 *og gør den redigerbar ved at sætte retter til null.
79 */
80 public void afleverDPunkt(dPunkt d){
81     d.retter = null;
82     dPunkter[d.getNummer()] = d;
83 }
84
85 /**
86 *Henter en opgave ud til læsning, dvs der kan ikke redigeres i
det der
87 * returneres.
88 */
89 public Opgave laesOpgave(int n){
90     return opgaver[n];
91 }
92 /**
93 *Sletter en opgave fra listen over opgaver
94 */
95 public void fjernOpgave(int n){
96     opgaver[n] = null;
97 }
98
99
100 /** Skaber en ny instans af Moede */
101 public Moede(Bruger b, String titel, String moedeDato) {
102     this.titel = titel;
103     rettet_af = b;
104     rettet_den = "dato";
105     moede_dato = moedeDato;
106 }
107
108 /**
109 *Tilføjer en deltager til mødet
110 */
111 public void tilfoejDeltager(Bruger b){
112     moedeDeltagere.add(b);
113 }
114
115 /**
116 *Returnere en arrayliste med titlen på de opgaver der i mødet
117 */

```

```

118     public ArrayList<String> opgaveArray(){
119         ArrayList<String> temp = new ArrayList();
120         for(int i = 0; i < opgaveTaeller+1; i++ ){
121             temp.add(opgaver[i].getTitel());
122         }
123         return temp;
124     }
125
126     /**
127      *Returnere en arrayliste med titlen på de dPunkter der i
mødet
128      */
129     public ArrayList<String> dagsordenArray(){
130         ArrayList<String> temp = new ArrayList();
131         for(int i = 0; i < dPunktTaeller+1; i++ ){
132             temp.add(dPunkter[i].getTitel());
133         }
134         return temp;
135     }
136
137     /**
138      *Returnere titlen på mødet
139      */
140     public String getTitel(){
141         return titel;
142     }
143     /**
144      *Returnere datoen for mødet
145      */
146     public String getDate(){
147         return moede_dato;
148     }
149
150     /**
151      *Returnere antallet af deltagere i mødet
152      */
153     public int getDeltagerAntal(){
154         return moedeDeltagere.size();
155     }
156
157     /**
158      *Fjerner en bruger fra mødet, brugeren er logget af
159      */
160     public void fjernBruger(Bruger b){
161         moedeDeltagere.remove(b);
162     }
163
164 }
165
166

```

MoedePunkt.java

```

1 package moedevaerktoej;

```

```

2
3 import java.io.Serializable;
4
5 /**
6  *Er en overklasse som dpunkterne og opgaver arver fra
7  *Indeholder de data som de har tilfældes
8  */
9 public class MoedePunkt implements Serializable {
10     public String titel;
11     protected String beskrivelse = "";
12     protected Bruger retter;
13     protected int nummer;
14
15     /** Tom konstruktor */
16     public MoedePunkt() {
17     }
18
19     /**
20     *Funktionen afgøre om punktet må rettes
21     *Hvis retter er null er der ikke en bruger igang med at
redigere
22     *ellers er retter = brugernavnet på den bruger der retter
23     */
24     public boolean redigerbar(Bruger b){
25         if (retter == null || retter == b){
26             retter = b;
27             return true;
28         }
29
30         return false;
31     }
32
33     /**
34     *Returnere punktets nummer
35     */
36     public int getNummer(){
37         return nummer;
38     }
39
40     // Her følger en masse set og get metoder til at sætte punktets
41     // data
42     public void setTitel(String s){
43         titel = s;
44     }
45
46     public String getTitel(){
47         return titel;
48     }
49
50     public void setBeskrivelse(String s){
51         beskrivelse = s;
52     }
53
54     public String getBeskrivelse(){
55         return beskrivelse;

```

```

56     }
57
58     public void setRetter(Bruger s){
59         retter = s;
60     }
61
62
63     public Bruger retter() {
64         return retter;
65     }
66 }
67

```

dPunkt.java

```

1 package moedevaerktoej;
2
3 import java.io.Serializable;
4
5 /**
6  *dPunkt arver fra MoedePunkt
7  *dPunkt står for dagsordenspunkt og gemmer referatet til
dagsordenspunktet
8  */
9 public class dPunkt extends MoedePunkt implements Serializable {
10     private String referat;
11
12     public dPunkt(String t, int n, String beskrivelse) {
13
14         titel = t;
15         nummer = n;
16         referat = "";
17         this.beskrivelse = beskrivelse;
18     }
19
20
21     public void setReferat(String r){
22         referat = r;
23     }
24
25     public String getReferat() {
26         return referat;
27     }
28 }
29

```

Opgave.java

```

1 package moedevaerktoej;
2
3 import java.io.Serializable;
4 /**

```

```

5  *Opgave arver fra moedepunkt
6  */
7  public class Opgave extends MoedePunkt implements Serializable {
8
9      private String opgaveDato = "";
10     private String ansvarlig;
11
12     /** Opretter en opgave med titel og nummer */
13     public Opgave(String t, int n) {
14         titel = t;
15         nummer = n;
16     }
17
18     public void setAnsvarlig(String s){
19         ansvarlig = s;
20     }
21
22     public String getAnsvarlig(){
23         return ansvarlig;
24     }
25
26     public void setOpgaveDato(String d){
27         opgaveDato = d;
28     }
29
30     public String getOpgaveDato(){
31         return opgaveDato;
32     }
33
34 }

```

Bruger.java

```

1  package moedevaerktoej;
2
3  import java.io.Serializable;
4
5  /**
6   *Bruger er den klasse brugerne bliver styret ved hjælp af. Meget
7   simpel klasse
8   */
9  public class Bruger implements Serializable {
10
11     private String navn;
12
13     public Bruger(String n) {
14         navn = n;
15     }
16
17     public String getNavn(){
18         return navn;
19     }

```

```
18 }
19
```

MoedeAdministrationsGUI.java

```
1 package moedevaerktoej;
2
3 import java.awt.event.ActionEvent;
4 import java.rmi.RemoteException;
5 import javax.swing.JOptionPane;
6
7 import javax.swing.AbstractAction;
8 import javax.swing.Action;
9 import javax.swing.Timer;
10
11 public class MoedeAdministrationGUI extends javax.swing.JFrame {
12
13     public MoedeAdministrationGUI() {
14         initComponents();
15         new Timer(5000, opdater).start();
16     }
17
18     // <editor-fold defaultstate="collapsed" desc=" Generated Code
19     ">
20     private void initComponents() {
21         nytMoedeKnap = new javax.swing.JButton();
22         jButton2 = new javax.swing.JButton();
23         jScrollPane1 = new javax.swing.JScrollPane();
24         moedeListe = new javax.swing.JList();
25         brugernavn = new javax.swing.JTextField();
26         brugerLBL = new javax.swing.JLabel();
27         jScrollPane2 = new javax.swing.JScrollPane();
28         brugerListe = new javax.swing.JList();
29         tilfoejBrugerBTN = new javax.swing.JButton();
30
31         setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
32         addWindowListener(new java.awt.event.WindowAdapter() {
33             public void windowActivated(java.awt.event.WindowEvent evt) {
34                 vindueAktiveret(evt);
35             }
36         });
37
38         nytMoedeKnap.setText("Nyt M\u00f8de");
39         nytMoedeKnap.addMouseListener(new java.awt.event.MouseAdapter()
40         {
41             public void mouseClicked(java.awt.event.MouseEvent evt) {
42                 nytMoedeKlik(evt);
43             }
44         });
45         jButton2.setText("Log p\u00e5 m\u00f8de");
```

```

46     jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
47         public void mouseClicked(java.awt.event.MouseEvent evt) {
48             logPaaMoede(evt);
49         }
50     });
51
52     moedeListe.setModel(new javax.swing.AbstractListModel() {
53         String[] strings = { "Item 1", "Item 2", "Item 3",
"Item 4", "Item 5" };
54         public int getSize() { return strings.length; }
55         public Object getElementAt(int i) { return strings[i]; }
56     });
57     jScrollPane1.setViewportViewView(moedeListe);
58
59     brugerLBL.setText("Brugernavn:");
60
61     brugerListe.setModel(new javax.swing.AbstractListModel() {
62         String[] strings = { "Item 1", "Item 2", "Item 3",
"Item 4", "Item 5" };
63         public int getSize() { return strings.length; }
64         public Object getElementAt(int i) { return strings[i]; }
65     });
66     jScrollPane2.setViewportViewView(brugerListe);
67
68     tilfoejBrugerBTN.setText("Tilf\u00f8j Bruger");
69     tilfoejBrugerBTN.addMouseListener(new
java.awt.event.MouseAdapter() {
70         public void mouseClicked(java.awt.event.MouseEvent evt) {
71             tilfoejBrugerBTNMouseClicked(evt);
72         }
73     });
74
75     org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
76     getContentPane().setLayout(layout);
77     layout.setHorizontalGroup(
78
layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
79         .add(layout.createSequentialGroup()
80             .addContainerGap()
81
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
82             .add(layout.createSequentialGroup()
83                 .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 198,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
84                 .add(14, 14, 14)
85
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
86             .add(nytMoedeKnap)
87             .add(jButton2))
88         .add(layout.createSequentialGroup()
89             .add(brugerLBL)
90             .add(13, 13, 13)

```

```

91             .add(brugernavn,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 129, Short.MAX_VALUE)
92
93     .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
94             .add(tilfoejBrugerBTN)
95             .add(20, 20, 20))
96
97     .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
98             .add(jScrollPane2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 84,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
99             .addContainerGap()
100
101     );
102     layout.setVerticalGroup(
103
104     layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
105         .add(layout.createSequentialGroup()
106             .addContainerGap()
107
108             .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
109                 .add(layout.createSequentialGroup()
110                     .add(51, 51, 51)
111                     .add(jButton2)
112
113                     .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
114                         .add(nytMoedeKnap) )
115                         .add(layout.createSequentialGroup()
116                             .add(46, 46, 46)
117
118                             .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
119                                 .add(jScrollPane2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 103,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
120                                 .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 113,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
121                                 .add(69, 69, 69)
122
123                                 .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
124                                     .add(brugerLBL)
125                                     .add(brugernavn,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
126                                     .add(tilfoejBrugerBTN)
127                                     .add(22, 22, 22))
128
129                                     );
130
131                                 pack();
132
133                             }// </editor-fold>
134                             /**
135                             *Knappen tilføjer en ny bruger til serveren
136                             */
137                             private void tilfoejBrugerBTNMouseClicked(java.awt.event.MouseEvent
138                             evt) {

```

```

127         try {
128             if (brugernavn.getText().equals(""))
129                 return;
130             NetConnect.getInstance().obj.nyBruger(brugernavn.getText());
131
brugereListe.setListData(NetConnect.getInstance().obj.brugereListe().toArray());
132             brugernavn.setText("");
133         } catch (RemoteException ex) {
134             ex.printStackTrace();
135         }
136     }
137 }
138 /**
139  *Bliver kørt når brugeren vil logge på et møde
140  */
141 private void logPaaMoede(java.awt.event.MouseEvent evt) {
142     if (moedeListe.getSelectedIndex() != -1 &&
brugereListe.getSelectedIndex() != -1 ){
143
144         String [] param = new String[2];
145         param[0] = String.valueOf(moedeListe.getSelectedIndex());
146         param[1] = String.valueOf(brugereListe.getSelectedIndex());
147         try {
148
149             NetConnect.getInstance().obj.brugerTilMoede(brugereListe.getSelectedIndex(),
brugereListe.getSelectedIndex());
150         } catch (RemoteException ex) {
151             ex.printStackTrace();
152         }
153         MoedeGUI.main(param);
154     } else JOptionPane.showMessageDialog(null, "Du mangler at vælge
møde eller brugernavn.", "Advarsel", JOptionPane.PLAIN_MESSAGE);
155
156     }
157
158 /**
159  *Hvergang billedet bliver aktiveret opdater listen over møder
160  *og brugere
161  */
162 private void vindueAktiveret(java.awt.event.WindowEvent evt) {
163     try {
164
brugereListe.setListData(NetConnect.getInstance().obj.brugereListe().toArray());
165
moedeListe.setListData(NetConnect.getInstance().obj.moedeListe().toArray());
//.moedeListe().toArray());
166     } catch (RemoteException ex) {
167         ex.printStackTrace();
168     }
169 }
170
171 /**
172  *Vis vinduet der laver et nyt møde

```

```

173     */
174     private void nytMoedeKlik(java.awt.event.MouseEvent evt) {
175         new NyMoedeGUI().setVisible(true);
176     }
177
178     /**
179      * Denne bliver kørt af timeren til at opdatere data
180      */
181     Action opdater = new AbstractAction() {
182
183         public void actionPerformed(ActionEvent e) {
184             try {
185
186                 moedeListe.setListData(NetConnect.getInstance().obj.moedeListe().toArray());
187                 brugerListe.setListData(NetConnect.getInstance().obj.brugerListe().toArray());
188             } catch (RemoteException ex) {
189                 ex.printStackTrace();
190             }
191         };
192     /**
193      * Starter programmet
194      */
195     public static void main(String args[]) {
196         java.awt.EventQueue.invokeLater(new Runnable() {
197             public void run() {
198                 new MoedeAdministrationGUI().setVisible(true);
199             }
200         });
201     }
202
203     // Variables declaration - do not modify
204     private javax.swing.JLabel brugerLBL;
205     private javax.swing.JList brugerListe;
206     private javax.swing.JTextField brugernavn;
207     private javax.swing.JButton jButton2;
208     private javax.swing.JScrollPane jScrollPane1;
209     private javax.swing.JScrollPane jScrollPane2;
210     private javax.swing.JList moedeListe;
211     private javax.swing.JButton nytMoedeKnap;
212     private javax.swing.JButton tilfoejBrugerBTN;
213     // End of variables declaration
214
215 }
216

```

NetConnect.java

```

1 package moedevaerktoej;
2 import java.rmi.*;
3
4 public class NetConnect {

```

```

5
6  /**
7   *Referencen til serveren
8   */
9  public ServerIface obj = null;
10
11  /**
12   *opretter netconnect som singleton
13   * Naming.lookup skal ændres til den server man vil forbinde til
14   *
15   *NetConnect er bare en "wrapper" til server objektet
16   */
17  public NetConnect() {
18      try {
19          obj = (ServerIface)
Naming.lookup("rmi://voesserver.??/StartServer");
20          //Voesserver.?? skal rettes til IP adressen eller
domænet på den
21          //maskine serveren køre på.
22          } catch (Exception e) {
23              e.printStackTrace();
24          }
25
26  public static NetConnect getInstance(){
27      return theInstance;
28  }
29
30  private static NetConnect theInstance = new NetConnect();
31  public static void main(String[] args){
32      NetConnect n = new NetConnect();
33  }
34 }
35

```

MoedeGUI.java

```

1  package moedevaerktoej;
2
3  import java.awt.Color;
4  import java.awt.event.ActionEvent;
5  import java.rmi.RemoteException;
6  import java.util.ArrayList;
7  import javax.swing.AbstractListModel;
8  import javax.swing.JDialog;
9  import javax.swing.AbstractAction;
10 import javax.swing.Action;
11 import javax.swing.Timer;
12
13 public class MoedeGUI extends javax.swing.JFrame {
14
15     public MoedeGUI() {

```

```

16     initComponents() ;
17 }
18
19 // <editor-fold defaultstate="collapsed" desc=" Generated Code
">
20 private void initComponents() {
21     jScrollPane1 = new javax.swing.JScrollPane();
22     jreferat = new javax.swing.JTextArea();
23     jScrollPane2 = new javax.swing.JScrollPane();
24     dagsordensListe = new javax.swing.JList();
25     jScrollPane3 = new javax.swing.JScrollPane();
26     opgaveListe = new javax.swing.JList();
27     jLabel1 = new javax.swing.JLabel();
28     jLabel2 = new javax.swing.JLabel();
29     jLabel3 = new javax.swing.JLabel();
30     jLabel4 = new javax.swing.JLabel();
31     jButton1 = new javax.swing.JButton();
32     nyOpgaveBTN = new javax.swing.JButton();
33     redigerKnap = new javax.swing.JButton();
34     jLabel5 = new javax.swing.JLabel();
35     jScrollPane5 = new javax.swing.JScrollPane();
36     DPunktBeskrivelse = new javax.swing.JTextArea();
37     jScrollPane4 = new javax.swing.JScrollPane();
38     jopgave = new javax.swing.JTextArea();
39     opgaveDato = new javax.swing.JTextField();
40     opgaveAnsvarlig = new javax.swing.JTextField();
41     opgaveDatoLBL = new javax.swing.JLabel();
42     opgaveAnsvarligLBL = new javax.swing.JLabel();
43     opgaveRedigerBTN = new javax.swing.JButton();
44     sletPunktBTN = new javax.swing.JButton();
45     sletOpgaveBTN = new javax.swing.JButton();
46     jLabel6 = new javax.swing.JLabel();
47     antalBrugere = new javax.swing.JLabel();
48     retterLBL = new javax.swing.JLabel();
49     logAfBTN = new javax.swing.JButton();
50
51
52 setDefaultCloseOperation( javax.swing.WindowConstants.DISPOSE_ON_CLOSE );
53 addWindowFocusListener(new java.awt.event.WindowFocusListener() {
54     public void windowGainedFocus( java.awt.event.WindowEvent
55     evt) {
56         fokus(evt);
57     }
58     public void windowLostFocus( java.awt.event.WindowEvent evt)
59     {
60     }
61     });
62 addWindowListener(new java.awt.event.WindowAdapter() {
63     public void windowActivated( java.awt.event.WindowEvent evt) {
64         formWindowActivated( evt );
65     }
66     public void windowOpened( java.awt.event.WindowEvent evt) {
67         startProgram( evt );
68     }
69     });

```

```

65     }
66   });
67
68   jreferat.setColumns(20);
69   jreferat.setEditable(false);
70   jreferat.setRows(5);
71   jScrollPane1.setViewportView(jreferat);
72
73   dagsordensListe.setModel(new javax.swing.AbstractListModel() {
74     String[] strings = { "Diverse punkter" };
75     public int getSize() { return strings.length; }
76     public Object getElementAt(int i) { return strings[i]; }
77   });
78
dagsordensListe.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
79   dagsordensListe.addMouseListener(new
java.awt.event.MouseAdapter() {
80     public void mouseClicked(java.awt.event.MouseEvent evt) {
81       dagsordensListeMouseClicked(evt);
82     }
83   });
84
85   jScrollPane2.setViewportView(dagsordensListe);
86
87   opgaveListe.setModel(new javax.swing.AbstractListModel() {
88     String[] strings = { "Diverse opgaver" };
89     public int getSize() { return strings.length; }
90     public Object getElementAt(int i) { return strings[i]; }
91   });
92   opgaveListe.addMouseListener(new java.awt.event.MouseAdapter() {
93     public void mouseClicked(java.awt.event.MouseEvent evt) {
94       opgaveListeMouseClicked(evt);
95     }
96   });
97
98   jScrollPane3.setViewportView(opgaveListe);
99
100  jLabel1.setText("Dagsorden");
101
102  jLabel2.setText("Referat");
103
104  jLabel3.setText("Opgaver");
105
106  jLabel4.setText("Opgave beskrivelse");
107
108  jButton1.setText("Nyt Dagsordenspunkt");
109  jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
110    public void mouseClicked(java.awt.event.MouseEvent evt) {
111      opretDagsordensPunkt(evt);
112    }
113  });
114
115  nyOpgaveBTN.setText("Ny Opgave");

```

```

116     nyOpgaveBTN.addMouseListener(new java.awt.event.MouseAdapter() {
117         public void mouseClicked(java.awt.event.MouseEvent evt) {
118             nyOpgaveBTNMouseClicked(evt);
119         }
120     });
121
122     redigerKnap.setText("Rediger");
123     redigerKnap.addMouseListener(new java.awt.event.MouseAdapter() {
124         public void mouseClicked(java.awt.event.MouseEvent evt) {
125             redigerKnap(evt);
126         }
127     });
128
129     jLabel5.setText("Beskrivelse");
130
131     DPunktBeskrivelse.setColumns(20);
132     DPunktBeskrivelse.setEditable(false);
133     DPunktBeskrivelse.setRows(5);
134     jScrollPane5.setViewportViewView(DPunktBeskrivelse);
135
136     jScrollPane4.setMinimumSize(new java.awt.Dimension(32, 32));
137     jScrollPane4.setPreferredSize(new java.awt.Dimension(162, 77));
138     jopgave.setColumns(20);
139     jopgave.setEditable(false);
140     jopgave.setRows(2);
141     jScrollPane4.setViewportViewView(jopgave);
142
143     opgaveDato.setEditable(false);
144
145     opgaveAnsvarlig.setEditable(false);
146
147     opgaveDatoLBL.setText("Afslutnings dato");
148
149     opgaveAnsvarligLBL.setText("Ansvarlig");
150
151     opgaveRedigerBTN.setText("Rediger Opgave");
152     opgaveRedigerBTN.addMouseListener(new
java.awt.event.MouseAdapter() {
153         public void mouseClicked(java.awt.event.MouseEvent evt) {
154             opgaveRedigerBTN(evt);
155         }
156     });
157
158     sletPunktBTN.setText("Slet punkt");
159     sletPunktBTN.addMouseListener(new java.awt.event.MouseAdapter()
{
160         public void mouseClicked(java.awt.event.MouseEvent evt) {
161             sletPunkt(evt);
162         }
163     });
164
165     sletOpgaveBTN.setText("Slet opgave");
166     sletOpgaveBTN.addMouseListener(new java.awt.event.MouseAdapter()
{

```

```

167         public void mouseClicked(java.awt.event.MouseEvent evt) {
168             sletOpgave(evt);
169         }
170     });
171
172     jLabel6.setText("Brugere tilstede:");
173
174     retterLBL.setForeground(new java.awt.Color(204, 0, 51));
175
176     logAfBTN.setText("Log af m\u00f8det");
177     logAfBTN.addMouseListener(new java.awt.event.MouseAdapter() {
178         public void mouseClicked(java.awt.event.MouseEvent evt) {
179             logAf(evt);
180         }
181     });
182
183     org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
184     getContentPane().setLayout(layout);
185     layout.setHorizontalGroup(
186
layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
187         .add(layout.createSequentialGroup()
188             .addContainerGap()
189
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
190             .add(layout.createSequentialGroup()
191                 .add(jLabel1)
192
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 258,
Short.MAX_VALUE))
193                 .add(jLabel3)
194                 .add(layout.createSequentialGroup()
195                     .add(nyOpgaveBTN)
196
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
197                     .add(opgaveRedigerBTN)
198
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
199                     .add(sletOpgaveBTN)
200                     .add(11, 11, 11))
201                 .add(jButton1)
202                 .add(jScrollPane3,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 310, Short.MAX_VALUE)
203                 .add(jScrollPane2,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 310, Short.MAX_VALUE))
204
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
205         .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
206             .add(layout.createSequentialGroup()
207
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

```

```

208
209 .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
210     .add(jScrollPane4,
211     org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 429, Short.MAX_VALUE)
212     .add(layout.createSequentialGroup()
213     .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING,
214     false)
215     .add(opgaveDatoLBL,
216     org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
217     org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
218     .add(opgaveDato,
219     org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 80, Short.MAX_VALUE))
220     .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
221     .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
222     .add(opgaveAnsvarligLBL)
223     .add(opgaveAnsvarlig,
224     org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 129,
225     org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
226     .add(layout.createSequentialGroup()
227     .add(jLabel4)
228     .add(335, 335, 335)))
229     .add(layout.createSequentialGroup()
230     .add(10, 10, 10)
231     .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
232     .add(jLabel2)
233     .add(jScrollPane1,
234     org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 425, Short.MAX_VALUE)
235     .add(jLabel5)
236     .add(jScrollPane5,
237     org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 425, Short.MAX_VALUE)
238     .add(layout.createSequentialGroup()
239     .add(redigerKnap)
240     .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
241     .add(sletPunktBTN)
242     .add(12, 12, 12)
243     .add(retteLBL)))
244     .add(org.jdesktop.layout.GroupLayout.TRAILING,
245     layout.createSequentialGroup()
246     .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
247     .add(jLabel6)
248     .add(2, 2, 2)
249     .add(antalBrugere))
250     .add(layout.createSequentialGroup()
251     .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
252     .add(logAfBTN))

```

```

242         .addContainerGap()
243     );
244     layout.setVerticalGroup(
245
layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
246         .add(layout.createSequentialGroup()
247             .addContainerGap()
248
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
249             .add(layout.createSequentialGroup()
250
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
251             .add(jLabel1)
252             .add(jLabel5))
253
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
254
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING,
false)
255
.add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup()
256             .add(jScrollPane5,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 96,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
257
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
258             .add(jLabel2)
259
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
260             .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 153,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
261             .add(jScrollPane2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 299,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
262
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
263
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
264             .add(jButton1)
265             .add(redigerKnap)
266             .add(sletPunktBTN)
267             .add(retterLBL))
268             .add(30, 30, 30)
269
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
270             .add(jLabel3)
271             .add(jLabel4))
272
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
273
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)

```

```

274
    .add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup()
275
        .add(jScrollPane4,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 121, Short.MAX_VALUE)
276
    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
277
    .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
278
        .add(layout.createSequentialGroup()
279
            .add(opgaveDatoLBL,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 14,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
280
    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
281
        .add(opgaveDato,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
282
        .add(layout.createSequentialGroup()
283
            .add(20, 20, 20)
284
            .add(opgaveAnsvarlig,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
285
        .add(opgaveAnsvarligLBL))
286
        .add(4, 4, 4))
287
    .add(org.jdesktop.layout.GroupLayout.TRAILING, jScrollPane3,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 170, Short.MAX_VALUE))
288
    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
289
    .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
290
        .add(layout.createSequentialGroup()
291
    .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
292
        .add(nyOpgaveBTN)
293
        .add(opgaveRedigerBTN)
294
        .add(sletOpgaveBTN))
295
        .addContainerGap())
296
    .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
297
        .add(jLabel6)
298
        .add(antalBrugere)))
299
        .add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup()
300
            .add(logAfBTN)
301
            .add(20, 20, 20)))
302
        );
303
    pack();
304
} // </editor-fold>
305
306
/**

```

```

307     *Fjerner en bruger fra listen over de aktive brugere i mødet
308     */
309     private void logAf(java.awt.event.MouseEvent evt) {
310         try {
311             NetConnect.getInstance().obj.fjernBruger(brugerNummer,
moedeNummer);
312         } catch (RemoteException ex) {
313             ex.printStackTrace();
314         }
315         dispose();
316     }
317     /**
318     *Sletter en opgave fra listen
319     */
320     private void sletOpgave(java.awt.event.MouseEvent evt) {
321         if(opgaveListe.getSelectedIndex() != -1)
322             opgaveNummer = dagsordensListe.getSelectedIndex();
323         if(opgaveNummer > -1){ //Tjekker om der er valgt et
listepunkt
324             Opgave sletOpgave = null;
325             try {
326                 sletOpgave =
NetConnect.getInstance().obj.hentMoede(moedeNummer,
NetConnect.getInstance().obj.getBruger(brugerNummer)).hentOpgave(opgaveNummer,
NetConnect.getInstance().obj.getBruger(brugerNummer));
327             } catch (RemoteException ex) {
328                 ex.printStackTrace();
329             }
330             if(sletOpgave == null){
331                 //Gør ingen ting, brugeren må ikke redigere
332                 System.out.println("Du må ikke redigere");
333             }
334             else{
335                 sletOpgave.setAnsvarlig(" ");
336                 sletOpgave.setBeskrivelse(" ");
337                 sletOpgave.setOpgaveDato(" ");
338                 sletOpgave.setTitel(" ");
339                 try {
340
341
NetConnect.getInstance().obj.skrivOpgave(moedeNummer, opgaveNummer,
sletOpgave);
342             } catch (RemoteException ex) {
343                 ex.printStackTrace();
344             }
345             System.out.println("opgave slettet");
346             opgaveNummer = -1;
347             start();
348         }
349     }
350 }
351
352 /**
353     *Sletter et referart punkt fra listen

```

```

354     */
355     private void sletPunkt(java.awt.event.MouseEvent evt) {
356         if(dagsordensListe.getSelectedIndex() != -1)
357             dpunktNummer = dagsordensListe.getSelectedIndex();
358         if(dpunktNummer > -1){ //Tjekker om der er valgt et
listepunkt
359             dPunkt sletDPunkt = null;
360             try {
361                 sletDPunkt =
NetConnect.getInstance().obj.hentMoede(moedeNummer,
NetConnect.getInstance().obj.getBruger(brugerNummer)).hentDPunkt(dpunktNummer
, NetConnect.getInstance().obj.getBruger(brugerNummer));
362             } catch (RemoteException ex) {
363                 ex.printStackTrace();
364             }
365             if(sletDPunkt == null){
366                 //Gør ingen ting, brugeren må ikke redigere
367                 System.out.println("Du må ikke redigere");
368             } else{
369                 sletDPunkt.setReferat("");
370                 sletDPunkt.setBeskrivelse("");
371                 sletDPunkt.setTitel("");
372                 try {
373
374
NetConnect.getInstance().obj.skrivDPunkt(moedeNummer, dpunktNummer,
sletDPunkt);
375             } catch (RemoteException ex) {
376                 ex.printStackTrace();
377             }
378             System.out.println("dPunkt slettet");
379             dpunktNummer = -1;
380             start();
381         }
382     }
383 }
384
385 private void formWindowActivated(java.awt.event.WindowEvent evt) {
386 // TODO add your handling code here:
387 }
388 private Opgave redigerOpgave = null; //Global variabel til at huske
hvilken opgave der redigeres pt
389
390 /**
391  *Funktion til at teste om en opgave kan redigeres eller ej.
392  *returnere en boolean
393  */
394 private void testEditable(){
395     if (jopgave.isEditable() != true){
396         opgaveDato.setEditable(false);
397         opgaveAnsvarlig.setEditable(false);
398         System.out.println("Dato = false");
399     } else {
400         opgaveDato.setEditable(true);

```

```

401         opgaveAnsvarlig.setEditable(true);
402         System.out.println("Dato = true");
403     }
404 }
405
406 /**
407  *Funktion der kører når brugeren trykker på knappen til at
redigere et
408  *opgave punkt. Tjekker først om der er valgt et punkt og
derefter om brugeren
409  *har ret til at redigere det (Der kunne være en anden bruger
der redigerede på
410  *samme tidspunkt). Hvis det er ok skifter den knappens tekst
til
411  * "Afslut redigering". Når brugeren trykker på "Afslut
redigering" skriver den
412  *resultatet til serveren og frigiver punktet.
413  */
414 private void opgaveRedigerBTN(java.awt.event.MouseEvent evt) {
415     if(opgaveListe.getSelectedIndex() != -1)
416         opgaveNummer = opgaveListe.getSelectedIndex();
417     if(opgaveNummer > -1){ //Tjekker om der er valgt et
listepunkt
418         if(jopgave.isEditable()!=true) {
419             try {
420                 redigerOpgave =
NetConnect.getInstance().obj.hentMoede(moedeNummer,
NetConnect.getInstance().obj.getBruger(brugerNummer)).hentOpgave(opgaveListe.g
etSelectedIndex(), NetConnect.getInstance().obj.getBruger(brugerNummer));
421
422                 } catch (RemoteException ex) {
423                     ex.printStackTrace();
424                 }
425                 if(redigerOpgave == null){
426                     //Gør ingen ting, brugeren må ikke redigere
427                     System.out.println("Du må ikke redigere");
428                     opgaveRedigerBTN.setForeground(Color.RED);
429
430                 } else{
431                     System.out.println("Editable true");
432                     opgaveListe.setEnabled(false);
433                     try {
434                         redigerOpgave.retter =
NetConnect.getInstance().obj.getBruger(brugerNummer);
435
NetConnect.getInstance().obj.skrivOpgave(moedeNummer, opgaveNummer,
redigerOpgave);
436                     } catch (RemoteException ex) {
437                         ex.printStackTrace();
438                     }
439
440                     jopgave.setEditable(true);
441                     opgaveDato.setEditable(true);
442                     opgaveAnsvarlig.setEditable(true);

```

```

443         opgaveRedigerBTN.setText("Afslut redigering");
444     }
445     } else {
446         System.out.println("Editable false");
447         opgaveListe.setEnabled(true);
448         redigerOpgave.setBeskrivelse(jopgave.getText());
449         redigerOpgave.setAnsvarlig(opgaveAnsvarlig.getText());
450         redigerOpgave.setOpgaveDato(opgaveDato.getText());
451         redigerOpgave.retter = null;
452         jopgave.setEditable(false);
453         opgaveDato.setEditable(false);
454         opgaveAnsvarlig.setEditable(false);
455         opgaveRedigerBTN.setText("rediger");
456         try {
457
NetConnect.getInstance().obj.skrivOpgave(moedeNummer, opgaveNummer,
redigerOpgave);
458         } catch (RemoteException ex) {
459             ex.printStackTrace();
460         }
461     }
462 }
463 }
464 /**
465  *Vis vinduet til oprettelse af ny opgave.
466  */
467 private void nyOpgaveBTNMouseClicked(java.awt.event.MouseEvent evt) {
468 // new OpretNytDPunktGUI().setVisible(true);
469     String [] param = new String[1];
470     param[0] = String.valueOf(moedeNummer);
471     //new MoedeGUI().setVisible(true);
472     OpretNyOpgaveGUI.main(param);
473 }
474 private dPunkt redigerDPunkt=null;
475 private int dpunktNummer = -1; //Det pt valgte dPunkt
476
477
478 /**
479  *Funktion der kører når brugeren trykker på knappen til at
redigere et
480  *møde punkt. Tjekker først om der er valgt et punkt og
derefter om brugeren
481  *har ret til at redigere det (Der kunne være en anden bruger
der redigerede på
482  *samme tidspunkt). Hvis det er ok skifter den knappens tekst
til
483  * "Afslut redigering". Når brugeren trykker på "Afslut
redigering" skriver den
484  *resultatet til serveren og frigiver punktet.
485  */
486 private void redigerKnap(java.awt.event.MouseEvent evt) {
487     if(dagsordensListe.getSelectedIndex() != -1)
488         dpunktNummer = dagsordensListe.getSelectedIndex();

```

```

489         if(dpunktNummer > -1){ //Tjekker om der er valgt et
listepunkt
490             if(jreferat.isEditable()!=true) { //Er brugeren igang med at
redigere eller vil han afslutte redigeringen?
491                 try {
492                     redigerDPunkt =
NetConnect.getInstance().obj.hentMoede(moedeNummer,
NetConnect.getInstance().obj.getBruger(brugerNummer)).hentDPunkt(dpunktNummer
, NetConnect.getInstance().obj.getBruger(brugerNummer));
493
494                 } catch (RemoteException ex) {
495                     ex.printStackTrace();
496                 }
497                 if(redigerDPunkt == null){
498                     //Gør ingen ting, brugeren må ikke redigere
499                     System.out.println("Du må ikke redigere");
500
501                 } else{
502                     System.out.println("Editable true");
503
504                 try {
505                     redigerDPunkt.retter =
NetConnect.getInstance().obj.getBruger(brugerNummer);
506
NetConnect.getInstance().obj.skrivDPunkt(moedeNummer, dpunktNummer,
redigerDPunkt);
507                     dagsordensListe.setEnabled(true);
508                 } catch (RemoteException ex) {
509                     ex.printStackTrace();
510                 }
511                 dagsordensListe.setEnabled(false);
512                 jreferat.setEditable(true);
513                 redigerKnap.setText("Afslut redigering");
514             }
515         } else { //Han er igang med at redigere
516             System.out.println("Editable false");
517             dagsordensListe.setEnabled(true);
518             redigerDPunkt.setReferat(jreferat.getText());
519             redigerDPunkt.retter = null;
520             jreferat.setEditable(false);
521             redigerKnap.setText("rediger");
522             try {
523
NetConnect.getInstance().obj.skrivDPunkt(moedeNummer, dpunktNummer,
redigerDPunkt);
524                 redigerDPunkt = null;
525             } catch (RemoteException ex) {
526                 ex.printStackTrace();
527             }
528         }
529     }
530     return;
531
532 }

```

```

533     /**
534     *Opdatere referat felt fra serveren. Bliver kørt fra timeren
535     */
536     private void sendOpdatering(){
537         try {
538             if (redigerDPunkt != null){
539                 redigerDPunkt.setReferat(jreferat.getText());
540                 NetConnect.getInstance().obj.skrivDPunkt(moedeNummer,
541                 dpunktNummer, redigerDPunkt);
542             }
543         } catch (RemoteException ex) {
544             ex.printStackTrace();
545         }
546     }
547
548     /**
549     *Opdater hvergang programmet får fokus
550     */
551     private void fokus(java.awt.event.WindowEvent evt) {
552         start();
553     }
554
555     /**
556     *Vis vinduet til oprettes af nyt referat punkt
557     */
558     private void opretDagsordensPunkt(java.awt.event.MouseEvent evt) {
559         String [] param = new String[1];
560         param[0] = String.valueOf(moedeNummer);
561         OpretNytDPunktGUI.main(param);
562     }
563     int opgaveNummer = -1; //Den sidst valgte opgave
564
565     /**
566     *Vis opgave detaljer når der bliver klikket på en opgave i
567     listen
568     */
569     private void opgaveListeMouseClicked(java.awt.event.MouseEvent evt) {
570         // DEBUG
571         System.out.println(opgaveListe.getSelectedIndex());
572         opgaveNummer = opgaveListe.getSelectedIndex();
573         try {
574             if(opgaveListe.getSelectedIndex() > -1){ //Tjekker om der er
575                 valgt et listepunkt
576                 Opgave temp =
577                 NetConnect.getInstance().obj.hentOpgave(moedeNummer, opgaveNummer);
578                 jopgave.setText(temp.getBeskrivelse());
579                 opgaveDato.setText(temp.getOpgaveDato());
580                 opgaveAnsvarlig.setText(temp.getAnsvarlig());
581             }
582         } catch (RemoteException ex) {
583             ex.printStackTrace();
584         }
585     }

```

```

583
584     /**
585     *Vis opgave detaljer når der bliver klikket på et dPunkt i
listen
586     */
587     private void dagsordensListeMouseClicked(java.awt.event.MouseEvent evt)
{
588         // DEBUG
System.out.println(dagsordensListe.getSelectedIndex());
589         dpunktNummer = dagsordensListe.getSelectedIndex();
590         if(dagsordensListe.getSelectedIndex() > -1){ //Tjekker om der er
valgt et listepunkt
591             try {
592                 // dPunkt temp =
moedeObjekt().laesDPunkt(dagsordensListe.getSelectedIndex());
593
jreferat.setText(NetConnect.getInstance().obj.hentDPunkt(moedeNummer,
dpunktNummer).getReferat());
594
DPunktBeskrivelse.setText(NetConnect.getInstance().obj.hentDPunkt(moedeNummer
, dpunktNummer).getBeskrivelse());
595         if(NetConnect.getInstance().obj.hentDPunkt(moedeNummer,
dpunktNummer).retter() != null){
596             retterLBL.setText("Dette punkt rettes af " +
NetConnect.getInstance().obj.hentDPunkt(moedeNummer,
dpunktNummer).retter().getNavn());
597         }
598     } catch (RemoteException ex) {
599         ex.printStackTrace();
600     }
601 }
602     return;
603 }
604
605     private boolean start = false;
606     private Moede moedet = null;
607     private NetConnect forbindelse = NetConnect.getInstance();
608
609     /**
610     *Funktion der opdatere listen over dPunkter og opgaver fra
serveren
611     */
612     private void start(){
613         try {
614
antalBrugere.setText(String.valueOf(NetConnect.getInstance().obj.getAntal(moede
Nummer)));
615         if(NetConnect.getInstance().obj.hentMoede(moedeNummer,
NetConnect.getInstance().obj.getBruger(brugerNummer)).dagsordenArray() != null){
616             try {
617
dagsordensListe.setListData(NetConnect.getInstance().obj.hentMoede(moedeNummer
,

```

```

NetConnect.getInstance().obj.getBruger(brugerNummer)).dagsordenArray().toArray()
);
618             if(redigerDPunkt == null && dpunktNummer != -1){
619
jreferat.setText(NetConnect.getInstance().obj.hentDPunkt(moedeNummer,
dpunktNummer).getReferat());
620
if(NetConnect.getInstance().obj.hentDPunkt(moedeNummer, dpunktNummer).retter()
!= null){
621             retterLBL.setText("Dette punkt rettes af
" + NetConnect.getInstance().obj.hentDPunkt(moedeNummer,
dpunktNummer).retter().getNavn());
622             }
623         }
624     } catch (RemoteException ex) {
625         ex.printStackTrace();
626     }
627
628     }
629 } catch (RemoteException ex) {
630     ex.printStackTrace();
631 }
632 try {
633
634     if(NetConnect.getInstance().obj.hentMoede(moedeNummer,
NetConnect.getInstance().obj.getBruger(brugerNummer)).opgaveArray() != null)
635
opgaveListe.setListData(NetConnect.getInstance().obj.hentMoede(moedeNummer,
NetConnect.getInstance().obj.getBruger(brugerNummer)).opgaveArray().toArray());
636     } catch (RemoteException ex) {
637         ex.printStackTrace();
638     }
639
640 }
641 /**
642  *Funktion der gør 1 gang, når vinduet bliver skabt. Det laver
en timer
643  *der sørger for at synkronisere programmets data imod serveren
644  */
645 private void startProgram(java.awt.event.WindowEvent evt) {
646     new Timer(5000, opdater).start();
647
648 }
649
650
651 private static int moedeNummer; //Hvilket møde er vi i
652 private static int brugerNummer; //Hvilken bruger har vi
653
654 /**
655  *Funktion der køres af vores timer
656  */
657 Action opdater = new AbstractAction() {
658
659     public void actionPerformed(ActionEvent e) {

```

```

660
661         start();
662         sendOpdatering();
663     }
664 };
665
666 /**
667  *Funktionen der starter klassen. Tager 2 argumenter
668  * @param moedeNummer, mødet den skal vise
669  * @param hvilken bruger kører programmet
670  */
671 public static void main(final String args[]) {
672     java.awt.EventQueue.invokeLater(new Runnable() {
673         public void run() {
674             new MoedeGUI().setVisible(true);
675             moedeNummer = Integer.parseInt(args[0]);
676             brugerNummer = Integer.parseInt(args[1]);
677             // DEBUG moedeNummer = 0;
678         }
679     });
680 }
681
682 // Variables declaration - do not modify
683 private javax.swing.JTextArea DPunktBeskrivelse;
684 private javax.swing.JLabel antalBrugere;
685 private javax.swing.JList dagsordensListe;
686 private javax.swing.JButton jButton1;
687 private javax.swing.JLabel jLabel1;
688 private javax.swing.JLabel jLabel2;
689 private javax.swing.JLabel jLabel3;
690 private javax.swing.JLabel jLabel4;
691 private javax.swing.JLabel jLabel5;
692 private javax.swing.JLabel jLabel6;
693 private javax.swing.JScrollPane jScrollPane1;
694 private javax.swing.JScrollPane jScrollPane2;
695 private javax.swing.JScrollPane jScrollPane3;
696 private javax.swing.JScrollPane jScrollPane4;
697 private javax.swing.JScrollPane jScrollPane5;
698 private javax.swing.JTextArea jopgave;
699 private javax.swing.JTextArea jreferat;
700 private javax.swing.JButton logAfBTN;
701 private javax.swing.JButton nyOpgaveBTN;
702 private javax.swing.JTextField opgaveAnsvarlig;
703 private javax.swing.JLabel opgaveAnsvarligLBL;
704 private javax.swing.JTextField opgaveDato;
705 private javax.swing.JLabel opgaveDatoLBL;
706 private javax.swing.JList opgaveListe;
707 private javax.swing.JButton opgaveRedigerBTN;
708 private javax.swing.JButton redigerKnap;
709 private javax.swing.JLabel retterLBL;
710 private javax.swing.JButton sletOpgaveBTN;
711 private javax.swing.JButton sletPunktBTN;
712 // End of variables declaration

```

```
713
714 }
715
```

OpretNyOpgaveGUI.java

```
1 package moedevaerktoej;
2
3 import java.rmi.RemoteException;
4
5 public class OpretNyOpgaveGUI extends javax.swing.JFrame {
6
7
8     public OpretNyOpgaveGUI() {
9         initComponents();
10    }
11
12
13    // <editor-fold defaultstate="collapsed" desc=" Generated Code
">
14    private void initComponents() {
15        annullerKnap = new javax.swing.JButton();
16        opgaveOverskrift = new javax.swing.JTextField();
17        opgaveOverskriftLBL = new javax.swing.JLabel();
18        opgaveBeskrivelseLBL = new javax.swing.JLabel();
19        jScrollPane1 = new javax.swing.JScrollPane();
20        opgaveBeskrivelse = new javax.swing.JTextArea();
21        opgaveAnsvarligLBL = new javax.swing.JLabel();
22        opgaveDatoLBL = new javax.swing.JLabel();
23        opgaveAnsvarlig = new javax.swing.JTextField();
24        opgaveDato = new javax.swing.JTextField();
25        opretNyOpgaveBTN = new javax.swing.JButton();
26
27
28        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
29        annullerKnap.setText("Annuller");
30        annullerKnap.addMouseListener(new java.awt.event.MouseAdapter()
31    {
32        public void mouseClicked(java.awt.event.MouseEvent evt) {
33            annullerKnapMouseClicked(evt);
34        }
35    });
36
37        opgaveOverskriftLBL.setText("Overskrift");
38
39        opgaveBeskrivelseLBL.setText("Beskrivelse af opgaven");
40
41        opgaveBeskrivelse.setColumns(20);
42        opgaveBeskrivelse.setRows(5);
43        jScrollPane1.setViewportView(opgaveBeskrivelse);
44
45        opgaveAnsvarligLBL.setText("Ansvarlig");
```

```

45         opgaveDatoLBL.setText("Dato for f\u00e6rdigg\u00f8relse");
46
47         opretNyOpgaveBTN.setText("Opret");
48         opretNyOpgaveBTN.addMouseListener(new
java.awt.event.MouseAdapter() {
49             public void mouseClicked(java.awt.event.MouseEvent evt) {
50                 opretNyOpgaveBTNMouseClicked(evt);
51             }
52         });
53
54         org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
55         getContentPane().setLayout(layout);
56         layout.setHorizontalGroup(
57
layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
58             .add(layout.createSequentialGroup()
59                 .addContainerGap()
60
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
61                 .add(layout.createSequentialGroup()
62
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
63                     .add(opgaveOverskriftLBL)
64                     .add(opgaveOverskrift,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 146,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
65                 .add(14, 14, 14)
66
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
67                     .add(opgaveAnsvarligLBL)
68                     .add(opgaveAnsvarlig,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 115,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
69                 .add(14, 14, 14)
70
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING,
false)
71                     .add(opgaveDato)
72                     .add(opgaveDatoLBL,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
73                 .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 453,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
74                 .add(opgaveBeskrivelseLBL)
75                 .add(layout.createSequentialGroup()
76                     .add(opretNyOpgaveBTN)
77
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
78                     .add(annullerKnap))
79
.addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

```

```

80         );
81         layout.setVerticalGroup(
82
layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
83             .add(layout.createSequentialGroup()
84                 .add(14, 14, 14)
85
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
86                 .add(opgaveOverskriftLBL)
87                 .add(opgaveAnsvarligLBL)
88                 .add(opgaveDatoLBL))
89
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
90
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
91     .add(opgaveOverskrift,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
92     .add(opgaveAnsvarlig,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
93     .add(opgaveDato,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
94     .add(13, 13, 13)
95     .add(opgaveBeskrivelseLBL)
96
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
97     .add(jScrollPane,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 139,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
98     .add(15, 15, 15)
99
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
100         .add(annullerKnap)
101         .add(opretNyOpgaveBTN))
102
.addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
103     );
104     pack();
105 }// </editor-fold>
106 /**
107  *Opretter et nyt møde på serveren
108  */
109
110 private void opretNyOpgaveBTNMouseClicked( java.awt.event.MouseEvent
evt) {
111     try {
112

```

```

113         int opg =
NetConnect.getInstance().obj.nyOpgave(opgaveOverskrift.getText(),
moedeNummer);
114
115         Opgave opgaven =
NetConnect.getInstance().obj.hentOpgave(moedeNummer, opg);
116         opgaven.setAnsvarlig(opgaveAnsvarlig.getText());
117         opgaven.setOpgaveDato(opgaveDato.getText());
118         opgaven.setBeskrivelse(opgaveBeskrivelse.getText());
119         NetConnect.getInstance().obj.skrivOpgave(moedeNummer, opg,
opgaven);
120     } catch (RemoteException ex) {
121         ex.printStackTrace();
122     }
123
124     dispose();
125 }
126 /**
127  *Luk vinduet hvis brugeren alligevel ikke vil oprette et nyt
møde
128  */
129 private void annullerKnapMouseClicked(java.awt.event.MouseEvent evt) {
130 // TODO add your handling code here:
131     this.dispose();
132 }
133 private static int moedeNummer; //Til at huske hvilket møde opgaven
skal tilføjes.
134
135 /**
136  * @param args the command line arguments
137  */
138 public static void main(final String args[]) {
139     java.awt.EventQueue.invokeLater(new Runnable() {
140         public void run() {
141             new OpretNyOpgaveGUI().setVisible(true);
142             moedeNummer = Integer.parseInt(args[0]);
143         }
144     });
145 }
146 }
147
148 // Variables declaration - do not modify
149 private javax.swing.JButton annullerKnap;
150 private javax.swing.JScrollPane jScrollPane1;
151 private javax.swing.JTextField opgaveAnsvarlig;
152 private javax.swing.JLabel opgaveAnsvarligLBL;
153 private javax.swing.JTextArea opgaveBeskrivelse;
154 private javax.swing.JLabel opgaveBeskrivelseLBL;
155 private javax.swing.JTextField opgaveDato;
156 private javax.swing.JLabel opgaveDatoLBL;
157 private javax.swing.JTextField opgaveOverskrift;
158 private javax.swing.JLabel opgaveOverskriftLBL;
159 private javax.swing.JButton opretNyOpgaveBTN;
160 // End of variables declaration

```

```
161
162 }
163
```

OpretNytDPunktGUI.java

```
1 package moedevaerktoej;
2
3 import java.rmi.RemoteException;
4
5 public class OpretNytDPunktGUI extends javax.swing.JFrame {
6
7     public OpretNytDPunktGUI() {
8         initComponents();
9     }
10
11     // <editor-fold defaultstate="collapsed" desc=" Generated Code
">
12     private void initComponents() {
13         Overskrift = new javax.swing.JTextField();
14         opretDPunktKnap = new javax.swing.JButton();
15         jLabel1 = new javax.swing.JLabel();
16         annullerKnap = new javax.swing.JButton();
17         jComboBox1 = new javax.swing.JComboBox();
18         jScrollPane1 = new javax.swing.JScrollPane();
19         opgaveBeskrivelse = new javax.swing.JTextArea();
20         jLabel3 = new javax.swing.JLabel();
21
22
23         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
24         opretDPunktKnap.setText("Opret");
25         opretDPunktKnap.addMouseListener(new
26 java.awt.event.MouseAdapter() {
27             public void mouseClicked(java.awt.event.MouseEvent evt) {
28                 opretDPunkt(evt);
29             }
30         });
31         jLabel1.setText("Overskrift");
32
33         annullerKnap.setText("Annuller");
34         annullerKnap.addMouseListener(new java.awt.event.MouseAdapter()
35 {
36             public void mouseClicked(java.awt.event.MouseEvent evt) {
37                 annullerKnapMouseClicked(evt);
38             }
39         });
40         jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new
41 String[] { "1", "2", "3" }));
42
```

```

42     opgaveBeskrivelse.setColumns(20);
43     opgaveBeskrivelse.setRows(5);
44     jScrollPane1.setViewportViewView(opgaveBeskrivelse);
45
46     jLabel3.setText("Beskrivelse af opgaven");
47
48     org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
49     getContentPane().setLayout(layout);
50     layout.setHorizontalGroup(
51
layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
52         .add(layout.createSequentialGroup()
53             .addContainerGap()
54
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
55             .add(org.jdesktop.layout.GroupLayout.LEADING,
layout.createSequentialGroup()
56                 .add(opretDPunktKnap)
57
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
58                 .add(annullerKnap))
59             .add(org.jdesktop.layout.GroupLayout.LEADING,
layout.createSequentialGroup()
60                 .add(jComboBox1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
61
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
62                 .add(Overskrift,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 146,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
63             .add(org.jdesktop.layout.GroupLayout.LEADING,
layout.createSequentialGroup()
64                 .add(7, 7, 7)
65                 .add(jLabel1))
66             .add(org.jdesktop.layout.GroupLayout.LEADING,
jLabel3)
67             .add(org.jdesktop.layout.GroupLayout.LEADING,
jScrollPane1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 453,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
68         .addContainerGap(38, Short.MAX_VALUE))
69     );
70     layout.setVerticalGroup(
71
layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
72         .add(layout.createSequentialGroup()
73             .addContainerGap()
74             .add(jLabel1)
75
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

```

```

76 .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
77     .add(Overskrift,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
78     .add(jComboBox1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
79     .add(12, 12, 12)
80     .add(jLabel3)
81
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
82     .add(jScrollPane1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 139,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
83
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 23,
Short.MAX_VALUE)
84
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
85     .add(opretDPunktKnap)
86     .add annullerKnap))
87     .addContainerGap()
88 );
89 pack();
90 }// </editor-fold>
91 /**
92  *opretter et nyt dPunkt på serveren
93  */
94 private void opretDPunkt(java.awt.event.MouseEvent evt) {
95     NetConnect forbindelse = NetConnect.getInstance();
96     try {
97         NetConnect.getInstance().obj.nytDPunkt(Overskrift.getText(),
opgaveBeskrivelse.getText(), moedeNummer);
98     } catch (RemoteException ex) {
99         ex.printStackTrace();
100     }
101     dispose();
102 }
103
private void annullerKnapMouseClicked(java.awt.event.MouseEvent evt) {
104     this.dispose();
105 }
106
107
private static int moedeNummer;
108
109
public static void main(final String args[]) {
110     java.awt.EventQueue.invokeLater(new Runnable() {
111         public void run() {
112             new OpretNytDPunktGUI().setVisible(true);
113             moedeNummer = Integer.parseInt(args[0]);
114         }
115     }

```

```
116     });
117 }
118
119 // Variables declaration - do not modify
120 private javax.swing.JTextField Overskrift;
121 private javax.swing.JButton annullerKnap;
122 private javax.swing.JComboBox jComboBox1;
123 private javax.swing.JLabel jLabel1;
124 private javax.swing.JLabel jLabel3;
125 private javax.swing.JScrollPane jScrollPane1;
126 private javax.swing.JTextArea opgaveBeskrivelse;
127 private javax.swing.JButton opretDPunktKnap;
128 // End of variables declaration
129
130 }
131
```