# BRUKARTILPASSBARE PEDAGOGISKE AGENTAR FOR GRUPPEVARESYSTEM

av

Jan-Eirik Bakke Nævdal

Cand. scient

Universitetet i Oslo

Institutt for Informatikk / InterMedia

2004

# TAKKESIDE

Det er mange som skal takkast for det bidraget som dei har kome med til dette arbeidet. Først og fremst vil eg takke mine to rettleiarar, Anders Mørch for den fantastiske støtta og rettleiinga dei har gitt meg gjennom heile prosessen frå den spede begynnelse av oppgåva til siste innspurt, og Jens Kaasbøl for den gode rettleilinga og oppfølginga eg har fått av han i prosessen med oppgåva.

Eg vil og takke Jan Dolonen for all den hjelpa han har gitt meg både under utviklinga av programvaren og dei konstruktive diskusjonane om agentar i skriveprosessen. Sten Ludvigsen vil eg og takke for dei pedagogiske diskujsonane som var til stor nytte for eksempelbygginga av RuleEditor.

InterMedia vil eg takke spesielt då dei har lagt mykje til rette for meg gjennom eigen leseplass og dei maskinresursser som eg har fått til min disposisjon.

Denne oppgåva er ein del av DoCTA prosjektet, finanisert av ITU.

Institutt for Informatikk og InterMedia, Universitetet i Oslo

Februrar 2004

Jan Eirik Bakke Nævdal

# INNHALD

# OVERSIKT OVER FIGURER

Figur Side

# OVERSIKT OVER TABELLAR

Tabell side

# 1 Innleiing

I vårt digitale samfunn (Frønes 2002) blir ofte det fysiske møterommet bytta ut med virtuelle møterom. Kommunikasjonen mellom aktørane føregår ved hjelp av e-post og digitale diskusjonsforum der fysisk avstand ikkje spelar noko rolle. Menneskeleg kommunikasjon har utvikla seg under føresetnad av fysisk nærleik der yttergrensa vart sett av vilkåra for visuell og/eller auditiv persepsjon av naturlege signal. Den tekniske utviklinga har etter kvart sprengt denne grensa (telegraf, telefon, TV) og no mobiltelefonen og konstruksjonen av digitale nettverk og Internett.

Det fysiske møtet har likevel kvalitetar som i ulike samanhengar framleis er ein føresetnad for den fullkomne kommunikasjonen, anten føremålet med kommunikasjonen er sosial bodskapsformidling eller støtte for praktisk samarbeid.. For å motverke tapet av slike kvalitetar ved virtuelt samarbeid, prøver ein å personifisere deltakarane i det virtuelle møterommet/arbeidsplassen ved hjelp av representasjonar (eks. bilete eller animerte figurar). Slike representasjonar får form i samsvar med den bruksmetafor gruppevaresystema legg opp til.

I gruppevaresystema(Grudin 1994) spelar det ingen rolle, teknisk sett, om brukaren sitt i Australia eller Noreg. Problema er gjerne knytt til samarbeidshandlingar og samarbeidsstøtte gjennom gruppevaresystema for at samarbeidet i det virtuelle rommet kan bli like effektivt som i det fysiske. For å få dette til, implementerer ein gjerne såkalla medierande artifaktar i systema. Ein artifakt er her ein kunstig skapt representasjon som støttar brukarane alt etter kva fagområde, problemstilling eller oppgåve som ligg til grunn for samarbeidet.

Denne oppgåva har som formål å utvikle ein brukarstøttande artifakt, eller ein pedagogisk agent som tilfredsstiller definisjonen av agentar. Arbeidet inngår i eit større forskingsprosjekt, DoCTA NSS studie (Design and use of Collaborative Telelearning

Artefacts Natural Science Studies) (Wasson, Guribye & Mørch 2000). Dette er eit samarbeidsprosjekt mellom InterMedia UiB, InterMedia UiO og Telenor FoU. DoCTA-prosjektet sine føresetnadar og mål blir presenterte meir detaljert i kapittel 3. DoCTA prosjektet har, både gjennom DoCTA I og DoCTA NSS, vore ein god grobotn for hovudfagsoppgåver og kommande doktorgradsavhandlingar. Oppgåvene har hatt eit vidt spekter med fokus frå teknologiske til pedagogiske problemstillingar. Dei føregåande oppgåvene, som har mest relevans for mitt arbeid, er arbeida til Dolonen (2002), Jondahl (2001) og Omdahl (2002). Desse oppgåvene blir drøfta i frå delkapittel 3.3 til 3.5 ut frå den relevans dei har til dette arbeidet.

Spesifikt ligg problemstillinga innafor definisjonen av pedagogiske agentar slik dette er framstilt nedanfor.

Ein agent er av P. Maes ved Massachusetts Institute of Technology definert til å vere: *" Ein prosess som lever i ei virtuell verd av datamaskinar og nettverk, og som kan operere autonomt for å utføre eit sett arbeidsoppgåver"* (Maes 1994)

Nwana (Nwana 1996) set opp ein klassifiseringsstruktur for programvareagentar etter tre attributt: autonomi, læring og samarbeid. Denne klassifiseringsmetoden vil eg bruke vidare for å klassifisere den pedagogiske agenten i dette arbeidet. Både metode og attributt skildring kjem eg tilbake til i delkapittel 2.5.

Johnson (1999) definerer pedagogiske agentar som autonome agentar i samsvar med Nwana si klassifisering av software agentar sine potensielle attributt og funksjonalitetar. Funksjonsmessig er den pedagogisk agenten definert som følgjande:

*"Pedagogical agents have been incorporated in a wide range of educational systems from intelligent tutoring systems (ITS) to computer-supported collaborative learning (CSCL) environments. These agents act as tutors, coaches, critics and co-learners. Another role for agent is that of a facilitator. For example in a distributed collaborative learning environment where users are geographically distributed and collaborate*

*through a web-based learning environment, an agent can facilitate collaboration processes such as participation, coordination, teacher intervention, and group interaction."* (Chen, Mørch and Wasson 2003)

Arbeidet sitt mål er å utvikle ein pedagogisk agent med justerbare regelsett, basert på dei empiriske studia og dei tekniske løysningane som har komme fram i DoCTA I og DoCTA NSS prosjekta.

Følgjande arbeid baserer seg også på fagområda End User Development (EUD) og Usability. EUD kan beskrivast som tilpassing eller utvikling av programvare tilpassa brukaren sine behov og ønskjer utført av brukarane sjølve. Fagområdet kan sjåast på som dagens ledd i ei lang rekke med utviklingsmetodar og tilpassingar som kan lenkjast tilbake til Vannavar Bush (Bush 1945) og Dahl & Nygaard sine arbeid om objektorientert programmering (OOP) (1965) og deltakande design (Nygaard 1992) Objektorientert programmering og Ehn sine (1988) teoriar om deltakande design, Detaljert gjennomgang av desse ulike teoriane er lagt til avsnitt 2.2

## 1.1 Strukturen i oppgåva

Oppgåva er delt opp fire ulike delar. Første del tar for seg kva teoriar og bakgrunnsmateriale den utvikla agenten i oppgåva baserer seg på. Her tek eg med både dei direkte teoriane som er brukte spesifikt i oppgåva, og dei teoriane som dannar grunnlaget for rammeverket rundt pedagogiske agentar. Desse felta omhandlar CSCW (Computer Supported Cooprative Work), CSCL (Computer Supported Collaborative Learning) Menneske maskin interaksjon (MMI).

Andre del av oppgåva er problemstilling og metodekapittelet. Her set eg opp den definerte problemstillinga og avgrensing av arbeidet sitt hovudmål. Metodekapittelet er eit oppsett av kva verktøy som blei brukte i utviklingsprosessen av agenten.

Tredje del er utviklingsarbeidet. Dette er delt opp i to ulike kapittel der kvart av kapitla går mot ei problemstilling. Denne oppdelinga er gjort ut frå at det er stor forskjell mellom dei to utvikla delane.

Siste og fjerde del er diskusjonen kring dei to utviklingsmåla i arbeidet og konklusjonen av arbeidet mitt. Eg definerer også mogelege vidareføringar frå røynslene eg har gjort i denne oppgåva, og dei resultat og funn eg konkluderer med..

# 2 Teori

Den teoretiske bakgrunnen for arbeidet ligg hovudsakleg i to fagområde, informatikk og pedagogikk. Kapitla 2.1, 2.2, 2.3, og 2.5 har sitt grunnlag i informatikken, medan kapitelet 2.4 har bakgrunn i pedagogiske fag, og kapittel 2.6 er eit tverrfagleg felt som inkluderer tankar og idear frå både den tekniske og dei pedagogiske tradisjonar. Gjennom denne tverrfaglege tilnærminga prøver eg å integrere pedagogiske tankar i eit teknologisk rammeverk. Fagområda som blir omtala her er Menneske-maskin Interaksjon(MMI) Sluttbrukar utvikling(EUD) Computer Supported Collaborative Learning(CSCL) og Computer Supported Cooprative Work (CSCW).

## 2.1 Menneske-maskin interaksjon (MMI)

MMI er studiet av korleis menneske samarbeider / interagerer med datamaskiner, MMI-teoriar kan aldri gje svar på alle designaspekt, men kan leggje føringar for korleis ein kan unngå vanlege feil og/eller møte kjende problem som oppstår under design av programvare eller nettstadar. (Nielsen 1989,1992,1993; Norman 1988, Preece m. fl. 1994)

MMI er ein tverrfagleg vitskap med element frå informatikk, kognitiv psykologi, sosial- og organisasjonspsykologi og ergonomi. Ein finn også element frå kunstig intelligens, lingvistikk, filosofi, sosiologi antropologi, systemutvikling og design. (Preece m. fl.. 1994, s37).

Ein funksjonell definisjon av MMI er følgjande: *"Menneske-maskin interaksjon er eit fagfelt der vurdering av design, evaluering og implementering av interaktive datasystem for menneskeleg bruk, og studie av viktige fenomen som oppstår i og rundt desse systema"* (ACM SIGCHI 1992 p6, oversett av forfattar).

Resonnementa som ligg til grunn for fagfeltet, vart delvist framsette alt i 1945 i artikkelen "As we may think" av Vannevar Bush (1945), der forfattaren eksemplifiserer aktuelle prinsipielle problemstillingar knytt til utviklinga innanfor fototeknologien. Temaet er betre

brukartilgang, betre kvalitet (oppløysing) og raskare framkalling. Bush (ibid.) sin bodskap er at ny teknologi ikkje vil/ eller kan bli brukt av den allmenne delen i samfunnet så lenge ein av dei tre faktorane ikkje er oppfylt. Gjennom utvikling med omsyn til tilgang, forbetring og produksjonsforenkling vil den aktuelle teknologien bli akseptert som ein nødvendig og naturleg del av dagleglivet i samfunnet. Datahistorikken har fleire gode eksempel på denne utviklinga. Enkelt kan den oppsummerast slik:

Teknologi → Forbetring av teknologi. → Brukbarheit → Allmenn bruk.

Menneske-maskin fagfeltet kan oppsummerast slik ut frå Bush sine tankar. Referansane er til forfattarane eg har brukt i dette arbeidet og ikkje klassifisering av felta.

Bush → OOP(Dahl &Nygaard) → User Interface(Kay)→ Participation Design(Ehn) → EUD(Fisher/Mørch)

Eit av dei store framskritta innan nyare programmering var det Dahl & Nygaard (1965) som stod for gjennom utviklinga av programmeringsspråket SIMULA (Dahl & Nygaard ibid.). Dette språket skilde seg vesentleg frå andre språk i måten ein delte applikasjonane opp i objekt og klassar i staden for å nytte ein prosedural programmeringsstrategi som til då hadde vore den vanlege metoden. I dag er objektorientert programmering (OOP) ein strategi mest alle tilsvarande programvarer bruker. OO-tankegangen til Dahl & Nygaard (1965) har vorte grunnsteinen for mange nye metodar og teknikkar innan systemutvikling. Seinare har andre vidareutvikla ideane. I arbeidet med prosjektet STAR-workstation for Xerox Palo Alto utvikla Alan Kay eit grafisk brukargrensesnitt til STAR maskina. Kay har gitt uttrykk for at OO-tankegangen i SIMULA var ein av inspirasjonskjeldene til dette grafiske brukargrensesnittet. Apple vidareførte Dynabook-prosjektet til Xerox og utvikla først LISA og seinare Macintosh operativsystemet (MacOS). STAR-MacIntosh- designa kan seiast å vere grunnstammen til det moderne GUI vi har i dag(Preece 1994, s18). Brukartilgangen til GUI-systema er overlegne dei gamle kommandospråkbaserte systema

som MS-DOS. GUI-løysingane forbetra tilgang i samsvar med Bush sine idear (Bush, 1945) om føresetnader for allmenn bruk av teknologiske framsteg.

Medan fokuset på 60-70 talet var å forbetre brukbarheita på det ferdige produktet, kom Nygaard med ei alternativ tilnærming innan design av programvare med sitt deltakande design (Partision Design) eller skandinavisk design som det også blir kalla. Gjennom prosjekta DEMOS (Sverige 1975-79) og UTOPIA (Sverige og Danmark 1981-85) introduserte Nygaard sine kollegaer i Danmark og Sverige noko nytt ved at brukarane av det kommande systemet blei involverte i designa. Denne designmetoden starta som eit samarbeid mellom Nygaard og Norsk Jern og Metall arbeiderforbund (Ehn 1992).

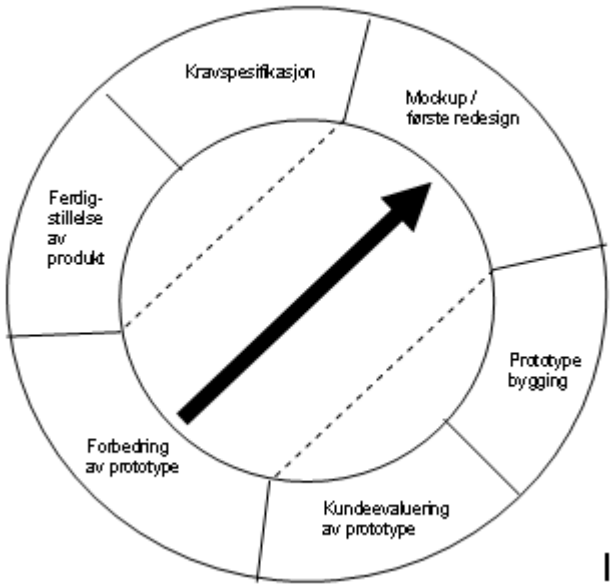
Overgangen frå deltakande design til at brukarane sjølve er utviklarane/tilpassarane representerer eit teknologisk framskritt. Framskrittet er at alle kan vere utviklarar til si eigen programvare og dermed få betre brukbarheit, funkjonalitet og kontroll.. Dette nye feltet er Sluttbrukar utvikling (End User Devlopment, EUD). Dette interessefeltet kan vidare delast opp i tre hovudretningar i) Sluttbrukar programmering (EUP), ii) Oppgåvespesifikt programmering (TSL) og iii) Sluttbrukar tilpassing (EUT). Då desse arbeidsfelta utgjer ein vesentleg del av dette arbeidet, vil dei bli omtalt nærare i del 2.3

### 2.1.1 Prototyping

Omgrepet "Prototyping" viser til ein stegvis prosess som gjer det mogleg for designutviklar å skape delmodellar av programmet som skal utviklast. (Pressmann 94). Gjennom prototyping framstiller ein delar av eit program/design/system for testing eller evaluering. Hovudpoenget er at ein ikkje utviklar heile systemet i ein prosess, slik "vassfallmodellen" legg opp til.

Prototypingsprosessen kan beskrivast som ein sirkulær prosess med repeterande syklus slik det er illustrert i Figur 1 nedanfor.

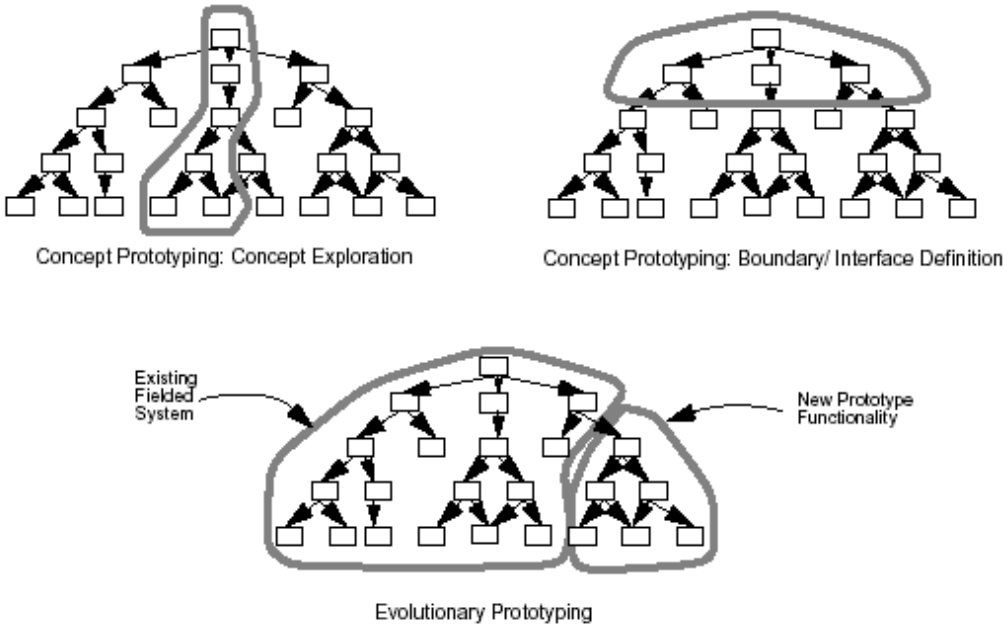
I denne prosessen blir kvart steg gjenteke til produktet eller delproduktet har tilfredsstillande kvalitet eller dekker det behovet prototypen blei tiltenkt. Boehm (1988) sette opp spiralmodellen der prototyping framstod som eit alternativ til vassfallsmodellen. Trongen for ein alternativ metode vaks fram som følgje av ei rekkje problem med styring og kontroll av utviklingsprosjekt der vassfallsmodellen vart nytta (Bohem 88)




Figur 1 (Pressman 1994 s27) Forenkla modell av Boehm`s (1988) spiral modell av prototyping som ein sirkulær repeterande prosess.

Innanfor programvareutkiling er det referert til to typar av utviklingsmetodar innanfor prototyping, i) den evolusjonære metoden og ii) bruk og kast metoden. Verktøyet som blir brukt i utviklinga, samt kvaliteten på kodinga, er grunnleggjande ulikt for dei to framgangsmåtane. Med omsyn til prototyping, skil ein også mellom to ulike formålsomsyn eller tilpassingsprinsipp , den vassrette modellen og den loddrette modellen (Figur 3). Kvart av desse prinsippa har ulike bruksområde.

Ved "bruk og kast metoden" er ikkje strukturen i koden viktig, men korleis "layouten" eller funksjonaliteten er konstruert. Evolusjonær prototyping vektlegg potensialet for å bli integrert i eit eksisterande system etter at prototypingsprosessen er ferdig. Begge metodane kan lett brukast både vassrett og loddrett og ein vil ofte bruke begge igjennom eit utviklingsprosjekt.



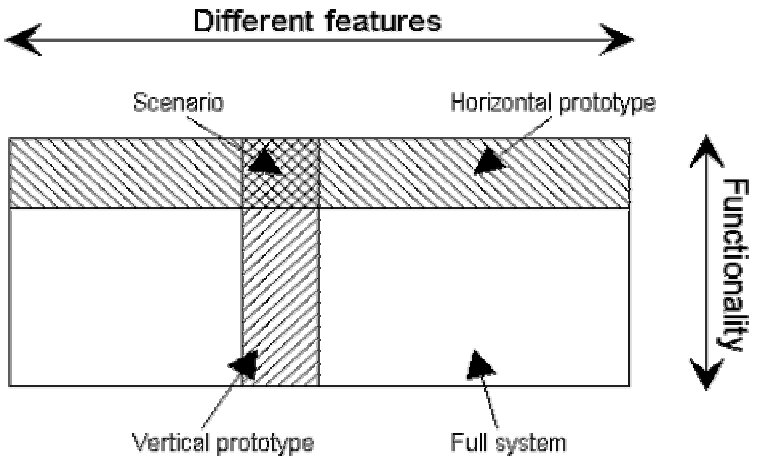

Figur 2 Modell forskjell mellom Evolusjonær og konsept prototyping (Wood&Kang 1992) Horisontal og vertikal prototyping er også vist i modellen

Den evolusjonære prototypinga er ein modell for utvikling saman med "bruk og kast" prototyping (rapid prototyping).Ved å bruke denne modellen for prototyping, er det eit vesentleg formål at programvarefragmentet som blir utvikla, skal inngå i - eller tilpassast eit fungerande program. Dermed blir val av verktøy og struktur ofte vesentleg annleis enn ved "bruk og kast". Ein avart av "bruk og kast" er konseptet prototyping. Modellen eg har brukt viser konsept prototyping men dei forskjellane figuren viser mellom konsept og evolusjonær prototyping gjeld også for forskjellen evolusjonær og bruk og kast prototyping.

C/C++, Java og Delphi er døme på språk ein gjerne nytta til evolusjonær prototyping. Typiske verktøy for programmeringa er visuelle kompilatorar slik som JBuilder eller VisualJ. VisiualC eller Borland Delphi®. Den underliggande strukturen på kodinga er det sentrale i evolusjonær prototyping.

Bruk og kast prototyping (rapid prototyping)

Bruk og kast prototyping seier mykje gjennom namnet. Teknikken er utvikla for å kunne lage "mock ups" av brukargrensesnitt anten som papirversjonar eller programvare modellar. Felles for alle bruk og kast modellane er at dei blir kasta etter at testen er gjennomført. Typiske verktøy for slik prototyping er MacroMedia Director®, MacroMedia Flash®, Borland JBuilder®, SmallTalk®, Microsoft Visual Basic® for å nemne nokre ulike. Konsept prototyping har svært mange likskapstrekk med bruk og kast metoden, men det er nokre skilnader.




Figur 3 Horisontal og vertikal prototyping.

Vassrett prototyping

Ved ei vassrett prototyping står det visuelle i fokus for utviklinga, der layout og utforming er det sentrale. Underliggjande funksjonalitet er underordna og ikkje tillagt noko særleg vekt i utviklingsfasen. Metoden blir mest brukt til å lage "mock ups" av brukargrensesnitt for å

teste og evaluere brukargrensesnitt og mindre delfunksjonar i ulike stadie av utviklingsprosessen. Metoden er særs nyttig i tidlege stadium av utviklingsprosessen. då det er enkelt å få endra dei delane som ikkje tilfredstiller brukarane sine ønskje eller behov.

Loddrett prototyping

Den loddrette prototypinga tar for seg utvikling av delfunksjonar i eit program der del-funksjonalitet til eit større program eller testprogram blir utvikla. Her er fokuset på utviklinga av djuptliggande funksjonalitet, og testing av desse funksjonane. Metoden passar godt saman med evolusjonær prototyping der ein utviklar modular for ein eksisterande applikasjon.

### 2.1.2 Usability

Usability eller brukbarheit har blitt eit viktig element i design av programvare og nettstader. Forutan Alan Kay, er Douglas Engelbart, Donald Norman og Jacob Nielsen sentrale namn. Usability er knytt til problemstillingar kring det å tilrettelegge systemet slik at brukarane føler at systemet er behageleg å bruke og intuitivt forståeleg. Jacob Nielsen (1993) har sett opp fleire punkt for design av menneske-maskin grensesnitt. Desse punkta har blitt aksepterte i fagfeltet som ein ISO- standard (ISO/DIS 9241-11 1996). Den tiltenkte effekten av å tilpasse eit system til desse punkta er betre brukbarheit sett frå brukarane sin synsvinkel, eller som Nielsen seier det sjølv: *"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."* (Nielsen ibid.).

Nielsens (ibid.) liste inneheldt 5 punkt, som skal vere ei rettesnor for design av programvare, websider og webapplikasjonar.

**Learnability**: Systemet skal vere lett å lære, slik at brukarane relativt raskt skal kunne setje seg inn i systemet og bruke det effektivt.

**Efficiency** : Når brukar først har lært seg systemet, skal det vere mogeleg å oppnå høg produktivitet/effektivitet.

**Memorability**: Systemet skal vere lett å setje seg inn att i om ein ikkje har brukt systemet over ein periode. Design som er lette å halde i minnet eller reetablere etter eit opphald, senkar terskelen for å oppta aktiviteten på nytt. Dette er viktig med omsyn til kontinuitet og reduksjon av brukarfråfall.

**Errors**: Systemet skal ha liten feilrate. Dette inneber at brukarane vert rettleia effektivt eller skjerma mot feilbruk. Høg feilrate tappar motivasjon og reduserer opplevinga av utbytte. Der feil ikkje kan eliminerast, er det viktig at ein kan retta feila utan altfor stor investering. Sjansen for å gjere fatale feil skal knapt vere til stades.

**Satisfaction**: Brukarane av systemet skal vere nøgd med systemet. Dette oppnår ein ved opplevinga av å meistre, vere effektiv og at systemet ikkje krev for stor investering eller fortrengjer hovudformålet med aktiviteten.

### 2.1.4. Evaluering

Heuristisk evaluering er ei form for testing der ein brukar eit førehandsoppsett kontrollskjema for å gå igjennom eit system for å evaluere det. Nielsen (Nielsen web1) har satt opp ein del punkt for bruk i heuristisk brukartesting basert på usability punkta som han sjølv har sett opp.

- **Synleg system status** (Visibility of system status). Det aktuelle systemet skal gje tilbakemelding om kva som føregår gjennom adekvat tilbakemelding
- **Brukar kontroll og fleksibilitet** (User control and freedom). Systemet må støtte kansellering av siste operasjon (undo) og gjenta siste operasjon (redo), slik at brukarane skal kunne angre det dei har gjort og komme seg ut av feilen som er blitt gjort.
- **Konsistens i utføring og innhald** (Consistency and standards). Systemet skal ha ein konsistens slik at det ikkje ser ut som om det er eit nytt system når ein delfunksjon blir aktivert.

- **Feilretting og feilhindring** (Error prevention). Hindring av at brukarar gjer feil. Dette vert gjort ved å utvikle gode designløysningar, og å nytte lett gjenkjennelege metaforar.
- **Gjenkjenning framfor minne** (Recognition rather than recall). Brukarane skal sleppe å måtte hugse kvar ulike funksjonar ligg, men gjenkjenne ikon eller liknande for ulike funksjonar.
- **Effektivt og funksjonelt å bruke** (Flexibility and efficiency of use). Tilpassing for ulike brukargrupper, lett å lære for nybyrjaren og effektivt for eksperten.
- **Stilreint design** (Aesthetic and minimalist design). Design på meldingar, utforming av layout bør vere reine, slik at fokuset til brukar ikkje blir distrahert av andre ting enn den funksjonen brukaren skal utføre.
- **Muligheit for brukarane å gjenkjenne og rette feil.** (Help users recognize, diagnose, and recover from errors). Feilmeldingar skal ikkje vere kryptiske kodelinjer eller reine tal, men informative meldingar som gjer brukaren i stand til å identifisere og rette feilen.
- **Hjelpefunksjonar og dokumentasjon for systemet** (Help and documentation) Muligheit for hjelpefunksjonar som er lett tilgjengeleg i sjølve systemet

Wizard of Oz er ein annan måte for prototyping der systemet ikkje blir utvikla men simulert av menneske. I datasamanheng blir denne teknikken brukt blant anna for å teste ut brukargrensesnitt- agentar utan å måtte programmere agenten på førehand.

Sjølve namnet kjem frå eventyret The Wizard og Oz av Frank L. Baum i 1900, der ein trollmann gjer handlingar som ser ut som magi, men det er hjelparane hans som utfører skjulte handlingar og slik skapar illusjonen.

Definisjon:[frase] *Wizard of Oz testing er testing der den automatiske maskinkomponenten er erstatta av ein eller annen form for menneskelig innblanding, slik at brukaren i testen ikkje er oppmerksam på erstatninga. Denne metoden er ofte brukt i evaluerings- prosessar , spesielt for å verifisere effektiviteten til dialogbaserte system.* (http://portalen.bibliotekivest.no/terminologi.htm )

Fordelen med denne teknikken er at ein kan teste og evaluere nye konsept og funksjonar utan å måtte implementere funksjonane fullstendig i systemet. (Ericsson 1999)

Dahlback Jønsson, Ahrenberg (1993) sette opp tre evalueringspunkt som er særleg viktige i vurderinga av eit Wizard of Oz eksperiment

- Bakgrunnssystemet, Viktig å ha eit stabilt system som bakgrunn for eksperimentet
- Oppgåvene forsøkspersonane skal løyse.
- Retningslinjene til dei som skal agere som agentar må vera utarbeidde etter gjennomført test.

Problem knytt til denne teknikken er at maskinar er raske, men rigide i reaksjonsmønster. Skrivefeil o.l. blir ikkje godtatt. Menneske er i motsetnad til maskinen fleksibel, intelligente men "trege" med omsyn til reaksjonstid og motorisk aktivering. Menneskeleg simulering av ei programvare står alltid i fare for å bli avslørt, sidan maskinen reagerer spontant medan mennesket vil bruke lengre tid på å prosessere førespurnaden og kome med eit svar.

Aktiviteten i samhandlinga mellom Wizarden og brukaren er eit kriterium for kor vellykka simuleringa er. (Ericsson 1999)

## 2.2 End User Development (EUD)

Innafor fagfeltet programvareutvikling og MMI er det eit klårt definert mål å vinna innsikt i brukarane sine behov og ønskje. I forskinga innafor paradigmet EUD, (http://www.eudnet.net) der problemstillingane omhandlar tilpassingsprosessen mellom brukar og teknologi, set ein brukarbehovet i sentrum for tilpassing i komplekse system (Klann 2002). Studium av ulike aspekt ved EUD-paradigmet er omtala blant anna av Summer & Klepper (1986) og Lieberman m. fl. (2003), men også fleire forfattarar har omtalt det.

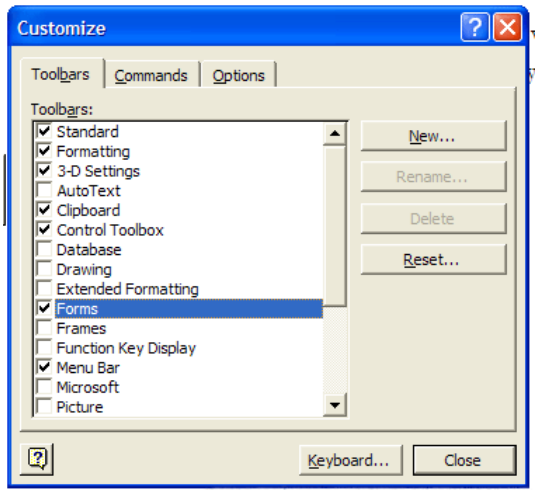
Klann (2002) sette fram eit framlegg til definisjon av kva End User Development er:*" End User Development is a set of activities or techniques that allows people, who are non-professional developers, at some point to create or modify a software artifact"*

Fisher & Girgensohn (1990) hevdar at eit moderne datasystem består av ein av to generelle kategoriar:

1) General purpose programming languages„ som er det språket systemutviklarar kjenner som Java, C/C++, Pascal, C#, Delphi, Fortran, Cobol o.s.v. Alt er mogeleg å gjere, men ein må ha kunnskapen om programmering for å utnytte programspråka sin funksjonalitet.

2) Turn-key – systems er derimot enkle og brukarvenlege system, men med liten muligheit for endring eller tilpassing til problemstillingar og oppgåver utanfor det systemet opphavleg vart designa for. Standard programvare slik som gamle WordPerfect 5.1 er eit eksempel på slike system. Her finn ein eit standard (felles) grensesnitt der brukarane ikkje har høve til tilpassing av menyar eller hurtigkommandoar i noko særleg grad.

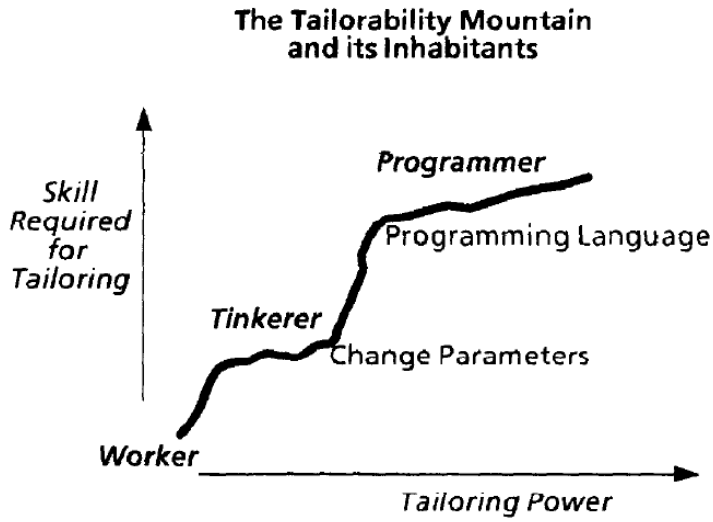
EUD verktyet finn ein balanse mellom desse to typane ved at ein gjer kraftige verktøy tilgjengeleg for modifisering og endring av det aktuelle systemet med enkle metodar slik som i Turn-key systema.




Figur 4 Eksempel på customization i MS Office. Der er det ein eigen funksjon for å tilpasse forskjellige menygrupper ut frå kva ein sjølv treng og vil bruke i dokumenta

Arbeidet med å få dette til, har splitta EUD opp i tre ulike retningar. End User Programming (EUP), Task Spesific Language (TSL) og End user Tailorablilty (EUT) Dei ulike metodane blir tilpassa brukarane sine.

I MacLean m. fl. (1990) blir tre typar brukarar tatt opp Worker, Tinkerer og Programmer. Ut frå dei tre typane EUD kan ei sei at EUT passer best hjå Worker og Tinkerer, TSL for Tinkerer og Programmer, mens EUP passar best for Programmer åleine. Desse EUD felta vil eg beskrive i detalj i dei neste avsnitta.




Figur 5 Tailoring slope (MacLean m. fl. 1990)

### 2.2.1 End User Programming (EUP)

Sluttbrukar Programmering (EUP) er det kraftigaste verktøyet for sluttbrukartilpassing då det tillet at brukarane sjølv lager programvara og ikkje berre tilpasser den til sitt bruk. Eit fellestrekk for EuP språk er at utviklarane startar på scratch eller berre med eit rammeverk definert. Dermed står sluttbrukarutviklarane fritt til å definere sin eiga røyndom for sine applikasjonar. Typiske utviklingsverktøy for sluttbrukar programmering er Applescript, Visiual Basic og Hypertalk. Visiual Basic(VBA) kan både vere eit scriptspråk og eit makrospråk, gjennom den implementeringa Microsoft har gjort med Visiual Basic kan ein programmere applikasjonar i VBA som er sjølvstendige program. Men VBA blir også brukt til makro programmering i Office pakken. Her blir VBA eit TSL.

### 2.2.2 Task Specific Language (TSL)

TSL kan definerast som eit modulbasert høgnivå programmeringsspråk, eit døme på TSL system er  reknearksmakroer i MS Excel der ein kan lage formlar og funksjonalitet utan å måtte kunne programmere. Verktøyet ligg som visuelle objekt, slik at brukaren kan

kombinere og strukturere slik han/ho vil at funksjonaliteten til reknearket skal vere. I desse makroane er det VBA som ligg til grunn for elementprogrammeringa.

Eit anna eksempel er kjøkkendesignprogrammet JANUS (Fisher, Girgensohn 1990, Fisher, McCall, Mørch, 1989) blir det vist eit eksempel på eit kjøkkendesign program med ein svært utvida funksjonalitet. I tillegg til å ha dei vanlege funksjonane som val av ulike objekt, storleik på objekta, storleiken på rommet og fasongen på rommet, er det i Janus mogeleg å lage nye objekt og importere dei nye objekta inn i designet av kjøkkenet. Her blei det laga eit språk for kjøkkendesign. Dette bestod av domenespesifikke symbol og relasjonar mellom desse.

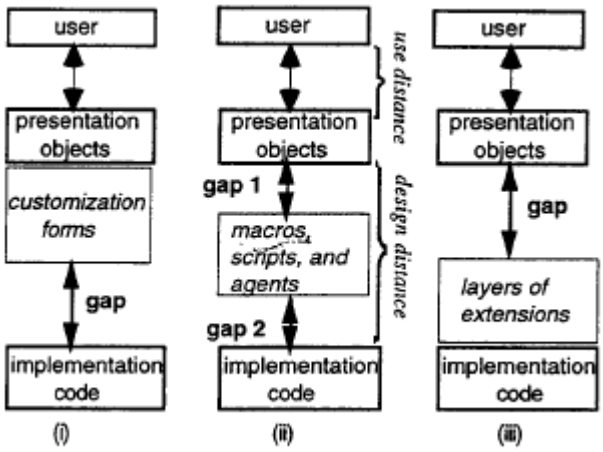
### 2.2.3 End User Tailoring (EUT)

Tailoring er å tilpasse generell programvare for å dekke ulike behov brukarane har. Ulike skriveprogram, rekneark- applikasjonar, teikneprogram og e-postapplikasjonar er døme på dette. (Mørch, 1995). I denne artikkelen set Mørch opp to ulike typar av Tailoring:

**Tailoring as design** er tilpassing som ei kontinuerleg utvikling av programvarepakken, gjennom å lage stabile modifikasjonar til programvarepakken, i motsetnad til tilpassingsmåtar der ein endrar dei spesifikke programvarene i pakken.(Cyber 1991).

**Tailoring as use** står for ein tilpassingsmåte der brukarar tilpassar programmet fordi det er ineffektivt eller for vanskeleg innafor det bruksområdet brukaren har behov for eller ønskjer å arbeide på. (Mørch 1995)

Vidare set han opp tre nivå av dei ulike tailoringtypane. Tilrettelegging (Customaztion), integrering (Integration) og utviding (Extention). HyperCard blir brukt som eksempel for tilrettelegging og integrasjon. HyperCard gir moglegheit for modifisering av knappane sine attributt eller dynamisk linking mellom fleire HyperCard sider. Slike endringar er tilrettelegging og integrering i Mørch sin modell (Figur 6) Utviding treng generelle programmeringsspråk slik som LISP for å kunne utnytte.




Figur 6. Tailoring by Customization Integration and Extension. Vist med bruene mellom User distance og Design distance (Mørch 1995)

Mørch definerer også to variablar for tailoring som Use distance og Design distance. Use distance er mellomrommet mellom brukar og presentasjonsobjekta mens Design distance er avstanden mellom tailoringsreiskapane og den implementerte programkoden.

## 2.3 Computer Supported Cooprative Work (CSCW)

CSCW oppstod som eige fagfelt i midten av 1980-talet, og har auka sin oppslutnad frå 90-talet. Paul Cashman og Irene Greif starta i 1984 med ein serie konferansar sponsa av ACM med forskarar og utviklarar der temaet var studie av korleis folk arbeider i grupper og korleis teknologi kan støtte dette samarbeidet (Grudin 1991)

Ein felles definisjon av kva CSCW er, er ikkje allment akseptert. For å vise kva retning dette arbeidet ligg i, set eg opp nokre definisjonar av fagfeltet som er relevante i denne samanhengen.

Ellis m. fl. (1991) definerer CSCW som "*Computer based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment*".

Suchman (1989) definerer CSCW ".. *the design of computer-based technologies with explicit concern for the socially organized practise of their intended users".*

Bannon m. fl.. (1988) meiner at CSCW kan sjåast på som ein paraplyterm for ulike forskingsområde der dei har overlappande faginteresser. I det følgjande er forståinga av CSCW influert av Bannon sitt syn på omgrepet (Bannon m. fl. 88, Bannon 1992).

CSCW er basert på gruppevare som tilbyr høgare nivå av koordinering og samarbeid mellom personar som arbeider i ein organisasjon (Koschmann 1996). Hovudfokuset i CSCW er arbeidseffekten og i kva grad organisasjonen får utbytte av samarbeidet.

### 2.3.1 Gruppevaresystem

Termen "Groupware" blei først brukt av Johnson-Lenz & Johnson-Lenz i 1978 på ein konferanse (Grudin 1991). Etter kvart har termen gruppervaresystem (groupware) blitt definert for å beskrive system eller produkt som støttar samarbeid i grupper.

Gruppevaresystem har blitt definert i ulike typar system etter to attributt: lokalisasjon og tid. Johansen (1988) laga ei lokalitets- / tidsmatrise for å kunne klassifisere ein grunnstruktur innan CSCW relaterte gruppevaresystem. Shneiderman (1992) har sett opp ei forklaring på kva dei to attributta til Johnsen (1988) inneber:

*Groupware technologies are typically categorized along two primary dimensions: whether users of the groupware are working together at the same time ("realtime" or "synchronous" groupware) or different times ("asynchronous" groupware), and whether users are working together in the same place ("colocated" or "face-to-face") or in different places ("non-colocated" or "distance").* (Shneiderman 1992)

Johansen (1988) si klassifisering av gruppevaresystem er den oppdelingsstrukturen innan CSCW som har fått størst tilslutnad (Baggetun 2001)

| | **Samlokalisert** | **Distribuert** |
|---|---|---|
| **Samtidig** | Ansikt til ansikt samhandling/samlokalisert(1) | Ikkje samlokalisert Samtidssamhandling (2) |
| **Ulik tid** | Asynkron og samlokalisert samhandling (3) | Asynkron og ikkje samlokalisert samhandling (4) |

Tabell 1 Tid og stad matrisen frå Johansen (1988)

Dei to dimensjonane, tid og stad, definerer fire ulike typar av gruppevarer ut frå kva funksjonalitet dei skal støtte opp om i arbeidsprosessen. Tabellen er meint som ei rettleiing med omsyn til funksjonalitet, meir enn ei streng og gjensidig utelukkande kategorisering av gruppevaresystemet. Mange system støttar både synkron og asynkron interaksjon. Andre system støttar samlokaliserte og distribuerte arbeidsformer/metodar. Eg har sett dei originale titlane på felta i parentes i kvar beskriving av feltet.

1) Ansikt til ansikt samhandling/samlokalisert (Face to face) tilseier at ein i møterommet har fysisk nærleik og direkte augnekontant mellom deltakarene: I ei slik setting utgjer gjerne artifaktene tavler og andre visuelle presentasjonsverktøy for felles fokus.

2) Ikkje samlokalisert samtidssamhandling (Synchron and Distributed Interaction) klassifiserer gruppeverktøy der ein har samtidig handling eller diskusjon eller chat-verktøy slik som MSN messenger eller NetMeeting.

3) Asynkron og samlokalisert samhandling (Asynchron. Co-located Interaction ) inneber at ein kommuniserer, men handlar til ulike tider. Asynkron samhandling kan vere at ein deler dataressursar, og har felles bruk av datamaskiner eller tilgang til andre felles ressursar i same lokalitet.

4) Asynkron og ikkje samlokalisert samhandling (Asynchron. Distr. Interaction) Denne kategorien beskriv ulike diskusjonsforum og gruppesystem som baserer seg på postar, deling av dokument der ein ikkje arbeider på same tid eller i same fysiske lokalitet.

### 2.3.2 Awareness

Dourish og Bellotti definerer awareness som: *"Awareness is an understading of the activities of others, which provides a context for your own activity"* (1992 s107)". Med dette kan ein seie at awareness er forståinga av kva andre gjer og korleis dette påverkar ditt eige handlingsmønster. I gruppevaresystem, er awareness eit svært viktig aspekt i samarbeidet, då det legg premissane for kor godt samarbeidet kan fungere. Det er definert ulike kategoriar av awareness. Dourish og Bellotti (ibid.) har sett opp to kategoriar: i) social awareness, ii) og Task awareness Andre forfattarar har også definert fleire typar awareness. Eg tek med to slike tilleggskategoriar i mitt arbeid, då dei er relevante for pedagogiske agentar: iii) Workspace awareness (Gutwin, Stark & Greenberg 1995) og iv) Conceptual awareness (Mørch, Jondahl & Dolonen 2004).

Social awareness (sosial viten) er den viten studentane i same gruppe har om dei sosiale relasjonane seg i mellom.

Task awareness er saksrelatert og står for den viten den enkelte gruppemedlem har av korleis ein kan løyse den tillagte oppgåva best mogeleg .

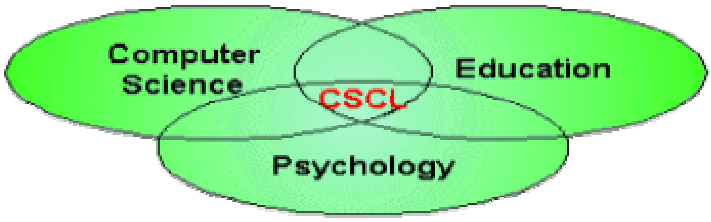
Workspace awareness (Gutwin, Stark & Greenberg 1995, Gutwin & Greenberg 1996,1998) blir forklart som forståinga av korleis personar samarbeider innan eit delt felles virtuelt arbeidsbord/tavle (eng workspace). Workspace awareness rettar seg mest mot synkrone gruppevaresystem, men er også relevant for asynkrone verktøy. Behovet for oversikt og kontroll over kva andre gruppemedlemmar har gjort medan ein har vore avlogga er også innanfor denne typen awareness.

Conceptual awareness definerer Mørch, Jondahl og Dolonen (2004) på følgjande måte: *Making learners aware of their own and other learners` action and activities; and identifying the common attitude (Mead 1934) and shared principles of a group of learners.*

## 2.4 Computer Supported Collaborative Learning (CSCL)

CSCL-paradigmet oppstod i 1989. Kochmann (1996) hevdar at ein NATO-workshop 1989 i Italia er utgangpunktet for CSCL paradigmet. Etterkvart erstatta CSCL det føregåande paradigmet, Logo as Latin[1], som til då hadde vore det dominerande tilnærminga til læring og IKT. CSCL sin metodikk er samarbeidslæring. I staden for at kunnskapsformidlaren er autoritær informasjonsoverførar, er den meir ei støtte og ressurs for det lærande fellesskapet (Kochmann ibid. s 16)

CSCL består av eit svært samansatt fagfelt det integrerer informasjonsvitenskap/ informatikk, psykologi, pedagogikk og undervisning slik som vist i Figur 7.




Figur 7 Fagfeltet CSCL samansemltingabestår av dei forskjellige fagfelta, modellen viser kva område som er skjæringspunkta.

CSCL er berre ein av dei læringsteoriane som blir brukt i dagens undervisning, i Tabell 2 gjer eg ein kort oppsummering av desse læringsstrategiane ut frå kva type undervisning dei er utvikla for og kva verden deira prosesser finner stad i forhold til Popper (1972) sine

[1] Logo as Latin kjem frå Koschmann (1996), som ein kritikk av det dominerande Logo programmeringspråket

begrep om World 1, World 2 og World 3. I tabellen har eg sett det undervisningstypane å vise kva andre læringmetoder som finst i tillegg til CSCL

Tabell 2 Kva felt dei ulike læringteorier passar i forhold til svarmuligheiter og referanser til Poppers 3 ulike verder

| **Paradigme** | **Riktig svar** | **Mange svar** | **Ingen svar /mange vegar** | **Poppers verden** |
|---|---|---|---|---|
| **ITS**[2] | V | (v) | | W2 |
| **LasL** | (V) | V | | W1 |
| **CSCL** | | (V) | V | W3 |

opp som riktig svar, mange svar, og ingen svar/mange veier. Dei matematiske fagretningane har som oftast eit svar med to strek under, og ITS-paradigme (Intelligent Tutoring System) er designet lagt opp til ei rettleiing fram til det rette svaret. Logo as Latin (LasL) har ein beslutningsprosess der fleire mål kan nåes men kan også brukast til matematiske oppsett.

CSCL består av tre ulike hovudteoriar. Eg vil beskrive to aktuelle tilnærmingsmåtar (teoriar) som har særleg relevans for arbeidet mitt: sosialkonstruktivismen og den sosio-kulturelle teorien. Situert kognisjon er den siste av dei teoretisk hovudsøylene, men vert mindre vektlagt i dette arbeidet.

### 2.4.1 Sosialkonstruktivismen

Sosialkonstruktivismen har tre kriterium som må oppfyllast for at det skal oppstå ny brukskunnskap hjå elevane/studentane, desse punkta er:

[2] ITS = Intelligent Tutoring System

1) Kunnskapsbygging og ikkje kunnskapsreproduksjon.

2) I interaksjon med andre utviklar ein ny kunnskap.

3) For å oppnå denne utviklinga av ny kunnskap bør interaksjonen involvere meir kunnskapsrike personar f.eks. ein lærar eller medelevar.

Denne prosessen kan gjentakast og dermed vil den ny kunnskapen igjen utvikle kunnskap. Denne teorien skil seg vesentleg frå teorien som var gjeldande før, der læraren gav kunnskapen til elevane, og elevane var passive mottakarar av informasjonen. Dermed var dei ikkje med å "utvikle" den, men fekk den i "ferdig" tilstand frå læraren. Tilpassinga av kunnskapen til elevane stod læraren for.

### 2.4.2 Den sosio-kulturelle teorien

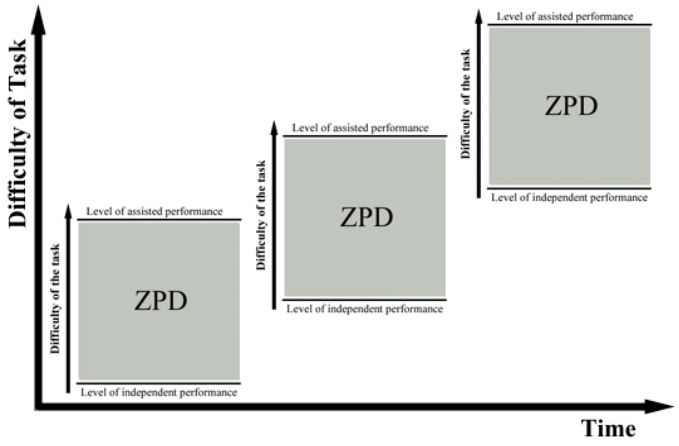
Stammer frå sovjetisk psykologi og forsking og er basert på ideane til Vygotsky, Luria, og Leontiev. Andre har vidareført og integrert desse pedagogiske synsmåtane i meir omfattande pedagogiske teorisystem (Wertsch m. fl. 1995). Den sosiokulturelle tilnærminga har som mål å forklare forholdet mellom mentale funksjonar og kulturelle, institusjonelle og historiske situasjonar (Wertsch m.fl ibid.). Kaptelinin (1996 s109) hevdar at indre mentale prosesser ikkje kan forståast utan referanse til dei ytre prosessane slik dette kjem til uttrykk i kulturelle kontekstar og rasjonelle strukturar.

Lev Vygotsky (1978) introduserte omgrepet "Zone of Proximal Development" (ZPD) eller sona for den næraste utviklinga. *"The dictance between the actual development level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidence or in a collaboration with more capable peers"* (Vygotsky, 1979:86)

Dette er eit er eit omgrep som kan brukast til å forstå læring og utvikling, og som inkluderer forholdet mellom lærarar og elevar. Omgrepet omfattar sona mellom det ein er klar for å meistre og det ein alt meistrar. Ved støtte frå andre kan ein utvikle latent kunnskap til

eksplisitt og sjølvforvalta kunnskap og kompetanse. Dette prosesskonseptet er grunnlaget for samarbeidslæring (collaborative learning).

Figur 8 viser korleis ZPD sona utviklar seg saman med personen over tid. Dess meir kunnskap personen har, dess høgare vil utviklingssona til personen gå. Den nedre grensa er det nivået eit individ alt meistrar, mens den øvre grensa i modellen er det maksimale nivået som eit individ kan greie ved hjelp av støtte eller samarbeid med meir kunnskapsrike personar.




Figur 8 Zone of Proximal Development

### 2.4.3 Knowledge Building

Knowledge building (KB, norsk Kunnskapsbygging) baserer seg på teoriar om kollektiv arbeid for utvikling og fordjuping av konseptuelle artifaktar, ut frå Popper (1972) sine konsept om World 1, World 2 og World 3. World 1 er den fysiske verda, World 2 er den mentale/subjektive verda og World 3 er den konseptuelle verda (Paavola m. fl. 2002). Eit viktig aspekt ved Bereiter sin teori er å skilje mellom læring og kunnskapsbygging, Der Bereiter meiner at læring skjer i World 2, altså i hovudet, medan kunnskapsbygging skjer i World 3. Truleg skjer kunnskapsbygginga som ein interaksjon mellom dei tre ulike verdene.

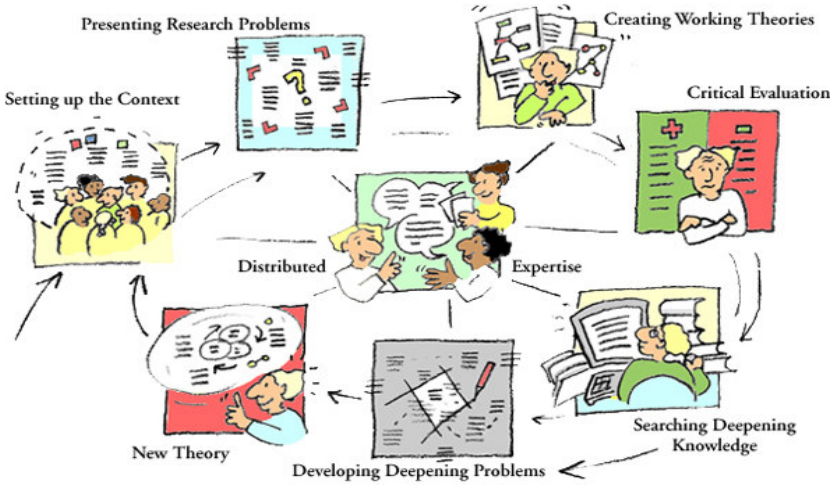
### 2.4.4 Progressive Inquiry Learning

Hakkarainen var PhD student av Carl Bereiter ved U. of Toronto og vidareutvikla kunnskapsbyggings- teoriane til Bereiter til teorien Progressive Inquiry Learing då han kom tilbake og stifta Centre for Research on Networked Learning and Knowledge Building, Department of Psychology, Universtiy of Helsinki, Finland .

Teoretisk er det vanskeleg å skilje Knowledge building og Progressive Inquiry Learning, men om ein ser på implementasjonen av dei to teoriane, er det ein del forskjell ein kan påpeike. Programvareimplementasjonen til Bereiter er Knowledge Forum (Bereiter & Scardamalia,1993), mens Hakkarainen var med å utvikla FLE (2 &3) (Hakkarainen,1998).

Progressive Inquiry Learning (PIL) baserer seg på utvikling av seg sjølv med omsyn til regulative eigenskapar (Zimmerman 1989), reflekterande tankegang, kritisk tankegang (Beyer 1985) og akademiske ferdigheitar i skriving og lesing (Geisler 1994). Modellen har som mål å danne aktive studentar (active learners). Desse kan lettare og effektivt styre sin eigen læringsprosess. Karakteristisk for PIL er at studentane skal behandle ny informasjon som problem, som då treng ei forklaring (Bereiter & Scardamalia 1993) Strukturen for PIL imiterer akademiske forskingsstrategiar. Studentar kan bli guida i ein større prosess der eit spørsmål dannar starten. Dette blir svara på, deretter set svaret opp nye spørsmål.

Måten PIL er implementert i FLE skal prøve å få brukarane til å inngå i ein sjølvdriven spørsmålsprosess der ny kunnskap blir utvikla undervegs. Prosessen baserer seg på seks steg: i) Starte med eit ofte uklart spørsmål ii) Danne seg eigen teori om korleis skal svare på og forklare spørsmålet iii) Gjennom å samarbeide med resten av gruppa, få evaluert og revurdert spørsmålet/problemet(Inquiry). iv) Søk etter grundigare/utfyllande kunnskap om problemet. v) Danne nye problemspørsmål på grunnlag av det avklarte problemspørsmålet. vi) Produsere utfyllande forklaringar på problemet for heile læremiljøet slik det er vist i Figur 8.




Figur 9 Progressive Inquiry Learning, FLE sin underliggande pedagogiske arbeidsmodell. Modellen viser korleis

To primærkjelder ligg til grunn for framstilling av "Progressive Inquiry Learning modellen": i) filosofisk tilnærming og ii) kognitiv tilnærming (Hakkarainen, Lipponen & Jävelä 2001, Ludvigsen & Mørch 2003) Det er store forskjellar mellom desse to: filosofisk vitskap og kognitiv vitenskap, men begge har felles fokus på problemløysning. I forståinga av at eit problem og eit tilordna sett med spørsmål, startar ein kunnskapsbyggingsprosess som har som føremål å løyse problemet (Ludvigsen & Mørch 2003)

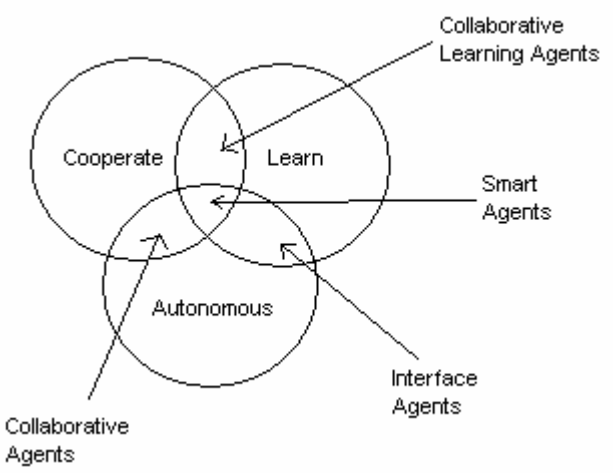
## 2.5 Programvare agentar (software agents)

Omgrepet "agent" kjem frå miljøa som forskar på kunstig intelligens. Etter kvart har andre fagområde også tatt i bruk termen. Den blir no også brukt innan distribuerte system og menneske maskin interaksjon (MMI). Ved at programvareagentar har fått eit stort nedslagsfelt innafor ulike miljø, og bruksmuligheitene er nesten uendelege, vil nok feltet auke framover (Jennings & Wooldridge 1996).

Termen vert nytta noko ulikt innan dei ulike fagområda. I denne oppgåva vil klassifiseringa av ein agent følgje Nwana (1996) si generelle attributtbaserte klassifisering, medan

definisjonen av den spesifikke agenten vil følgje definisjonen av ein pedagogisk agent som Chen, Mørch & Wasson (2003) har sett fram.

Nwana definerer ei generell gruppering av agentar i sin artikkel. Eg vil fokusere på den klassifiseringsbestemminga som er mest relevant for agenten.




Figur 10 Software agentar klasse typar (Nwana 1996) Oppdelt etter dei tre attributta Nwana definerte.

Både i Jondahl (2001), Omdahl (2002) og Johnson (1998) blir pedagogiske agentar lagt inn under Interface agentar i samsvar med Nwana (1996) si klassifisering. Dei tre attributta ein agent kan definerast etter her er: Samarbeid, Læring og Autonomi.

### Samarbeid

Støtter opp om samarbeid mellom andre agentar eller brukarar i same system eller andre system dei kan kommunisere med, gjennom ei eller annan form for språk (Wooldridge & Jennings 1995). KQML kan vere eit slikt språk.

### Læring

Evna til å lære frå korleis agenten og dei eksterne omgivingane intervener med kvarandre (Nwana 1996).

### Autonomi

Autonomi viser til grad av sjølvstende. Det viktigaste elementet for ein autonom agent er muligheita for å "ta initiativet" i staden for å vente på respons frå systemet eller miljøet agenten er implementert i (Wooldridge & Jennigs 1995).

Klassifiseringa til Nwana (Nwana 1996) gjev fire samansette typar av agentar: Samarbeidsagentar (SaA), Samarbeids-Læringsagentar (SLA), Smarte Agentar (SmA) og Brukargrensesnittagentar (BgA). Kategoriane er i følgje Nwana (ibid.) ikkje absolutte, men ein grupperingsmetode der hovudfunksjonane til agenten er kategorisert. Ein samarbeidsagent vil vektleggje samarbeid og autonomi, men det kan godt vere at ein slik agent også kan lære, men ikkje nok til at den kan kallast ein Smart Agent.

I Baggetun, Dolonen og Dragsnes (2001) blei det identifisert eit sett med attributt ei programvare agent bør ha for å kallast ein agent:

- Muligheit til å operere utan menneskeleg interferens (autonomi)
- Evne til å kommunisere med andre agentar eller brukarar (kommunikativ)
- Muligheit for overvakning og handling i forhold til sitt eige miljø (reaktiv)
- Muligheit til å endre /lære av det miljøet den eksisterer i (adaptiv)

## 2.6 Pedagogiske agentar

Ein einskapleg definisjon av kva ein pedagogisk agent er, er ikkje klarlagt, men mine tolkingar samsvarer med Chen, Mørch & Wasson (2003). Dette vil og vere naturleg då

arbeidet fell innanfor sentrale arbeidsområde i DoCTA-prosjektet. Ei anna tilnærming eg og vil vektleggje er den Johnson & Haye-Rothes (1998A) står for: *"Because pedagogical agents are autonomus agents, they inherit many of the same concerns that autonomus agents must address ".*

Denne definisjonen legg pedagogiske agentar innanfor klassedefineringa til Nwana (1996). Johnson (1999) seier også at pedagogiske agentar er autonome agentar der agenten si hovudoppgåve er å støtte læring mellom personar gjennom interaksjon med deltakarene innafor konteksten i det interaktive læremiljøet.

Ein kan identifisere fleire typar av pedagogiske agentar etter kva primære funksjonar dei har. Ein overvakingsagent (eng monitoring) vil vere ein typisk autonom agent som hentar og analyserer data, men den tek ikkje direkte kontakt med brukarane. Informasjonen må hentast ut av agenten anten som ein presentasjon eller reine rå data.

Ein annan type er brukargrensesnittagentane, Interface agent etter Nwana (1996). Desse har ein analysedel slik den autonome agenten har. Den store skilnaden ligg i respons agenten. Her blir reaksjonar på det analyserte materialet vurdert ut frå definerte reglar, og agenten tar kontakt med brukarane gjennom auditive eller visuelle effektar. Animerte agentar er ein slik type. Johnson beskriv aminerte agentar som "*animated pedagogical agents make it possible to more accuratly model the kinds of dialogs that occurs during apprenticeship learning and one to one tutoring*" (Johnson & Haye-Rothes 1998).

Eksempel på slike animerte agentar er Adele (Johnson 1999), COLER (Constantino-Gonzàles & Suthers 2001) og Herman the bug (Lester & Stone 1997). Desse tre animerte agentsystema vil eg beskrive i meir detalj.

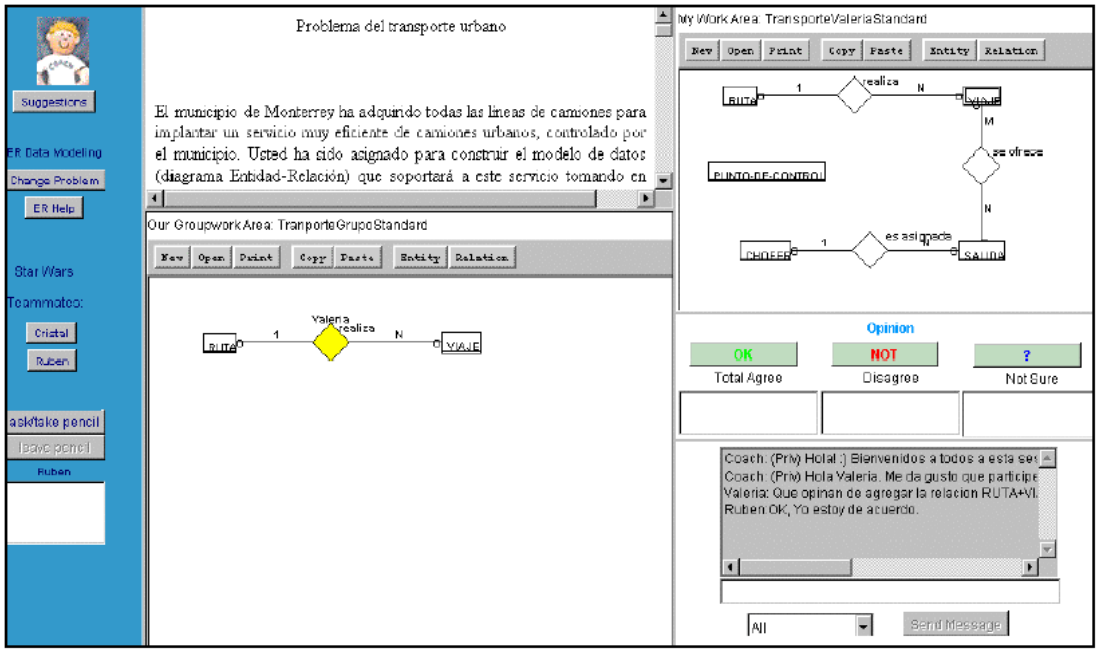
Johnson (1999) beskriv Adele (Agent for Distant Learning: Light Edition) utvikla ved CARTE (Center for Advanced Research in Techology for Education). Denne agenten er designa for å støtte opp om eit virtuelt læringsmiljø for legestudentar.


Figur 11 ADELE brukargrensesnitt i instruksjon tilbakemelding (Johnson 1999)

Agenten fungerer gjennom eit førehandsdefinert scenario som studentane skal gå igjennom. Agenten har oppsette reglar for korleis denne gjennomgangen skal bli gjort. Ved brot på desse reglane vil agenten ta kontakt med brukaren for å rettleie gjennom å stille spørsmål eller komme med kommentarar om kvifor studenten tok det valet den gjorde. Den kan også komme med råd om kva studenten skal gjere vidare ut frå råda frå Adele. Denne funksjonen ligg WHY knappen.

COLER (Constantino-Gonzàles & Suthers 2001) er eit eksempel på implementerte pedagogiske agentar i webbaserte gruppevaresystem. I dette tilfellet er både gruppevara og agent utvikla i fellesskap. Gruppevara er bygd opp rundt eit felles vindauga (Our GroupworkArea) der alle gruppemedlemane kan sjå, og eit eige vindauga (My workArea) som er privat. I tillegg er det ein synkron chat funksjon implementert. Agenten blir kalla

"personal coach", gjennom at ein agent skal rettleie (coaching) kvar enkelt student i arbeidet.




Figur 12 Webgrensesnittet i COLER hovudbilete (Constantino-Gonzàles & Suthers 2001)

Agentfunksjonen i COLER har fleire funksjonar. Desse blir delt opp i 7 kategoriar, diskusjon (Discussion), deltaking (Participation), tilbakemelding (Feedback), Sjølvrefleksjon (Self-Reflection) ER-modellering (ER Modeling), Velkommen (Welcome) og farvel (Goodbye). Kvar av kategoriane har eitt eller fleire typar av råd som blir viste til studentane når brot på regelsetta er oppdaga. Ein rådgenerator prioriterer ut kva råd som skal komme først basert eit AND/OR situasjons tre.

Lester & Stone (1997) kom med ein litt anna agentutforming enn kva eg tidlegare har vist, Herman the bug har eit anna design på den animerte agenten enn kva agentane til Johnson og Constantino-Gonzàles og Suthers har.


Figur 13 Herman the bug. Lester & Stone 1997. Startvindauga til Herman programmet

Desse tre agentane er brukargrensesnitt agentar (BG-A), men det finst andre typar av slike agentar. Ein kan vidare dela BG-A opp i ulike brukarfunksjonar der funksjonen til agenten er tilpassa eit spesifikt mål. Dei funksjonane eg interesserer meg for i dette arbeidet, er samarbeidsbygging, kunnskapsbygging og verktystøtte. I Jondahl (2001) sitt forsøk (CoPAS[3]) blei det sett opp ein agent for kvar funksjon, men det er ingen hindring for at ein og same agent kan ha alle funksjonane.

### Samarbeidsagent

Samarbeidsagentar støttar opp om samarbeid og gruppearbeid i den virtuelle gruppa. Optimalt vil dette seie at fleire av fordelane ein har i det samlokaliserte gruppemøtet blir overført til nettet der ein og får nytta seg av dei fordelane som nettbaserte system har med omsyn til fleksibilitet og lokaliseringsfridom. Agenten vil leggje ein overordna struktur som brukarane kan følgje, viss ønskeleg.

[3] CoPAS Collaboration Patterns Agent Simulation, Eit studium gjort ved IFI, UIB.

### Kunnskapsbyggingsagent (domain agent I)

Kunnskapsbyggingsagentar støttar kunnskapsbygginga i gruppa då det er agenten som har kunnskap om fagfeltet og kan vere ein mentor for elevane medan dei jobbar i gruppa. Agenten sin funksjon er å svare brukarane på spørsmål.

### Statistikkagent

Statistikkagentar er loggførande og gjev på den måten oversikt over tekniske registreringar i arbeidsprosessen. Desse agentane er difor kopla opp mot det administrative systemet og er autonome.(Chen & Wasson 2002)

### Verktøyagentar (domain agentar II)

Verktøyagentar (domain agentar II) er ein type agentar som rettar støttefunksjonane sin mot det verktøyet studentane bruker for å oppnå kunnskapbygginga. I Jondahl (2001) var ein verktøysagent "implementert" i systemet med oppgåva å svare på spesifikke spørsmål knytt til gruppevaresystemet TeamWave.

## 2.7 Oppsummering av teori

Gjennom dette kapittelet har eg presentert det teoretiske grunnlaget som arbeidet mitt byggjer på. Noko av teorien har berre indirekte relevans, medan noko av den er direkte relatert til utviklingsarbeidet og problemstillinga. Teoriane som dannar rammeverket rundt pedagogiske agentar og kunnskapsbygging/gruppesamarbeid både i digitale og virtuelle omgjevnader og i vanlege klasserom, ser eg som indirekte, men likevel naudsynte og informative om ein skal forstå dette arbeidet sin plass i heilskapen. I og med at dette arbeidet er ei utviklingsoppgåva, er mykje av teorien lagt opp mot det praktiske arbeidet med å utvikle ein agent. Dei pedagogiske teoriane kan virke litt gløymt i oppgåva, men mykje av designa og løysningane som har blitt vurdert opp mot spesielt CSCL teoriar for implemetasjon.

Materialet som blir presentert i kapitel 3 bygger mykje på  rammeverket eg har presentert i dette kapittelet.

# 3 Tidlegare arbeid

Denne oppgåva er ein av fleire oppgåver som har komme ut av DoCTA I og DoCTA NSS prosjekta, og byggjer vidare på materialet som har komme ut til no. Sentralt står Jan Dolonens oppgåve: "The Develompment of a Pedagogical Agent System for Computer Supported Collaborative Leraning" (Dolonen 2002), der mi oppgåve kan sjåast på som ei direkte vidareutvikling av denne pedagogiske agenten.

Ein del av koden i Dolonens agent er gjenbrukt i min agent og agentane har fleire fellestrekk, både med omsyn til struktur for kommunikasjon og generell oppbygnad. Store deler av den teoretiske plattforma for arbeidet har Silje Jondahl lagt i si hovudfagsoppgåve: "Simulering av pedagogiske agentar i eit virtuelt læremiljø ved bruk av Wizard of Oz teknikken" (Jondahl 2001). Stort bidrag til teoretisk plattform har også Karianne Omdahl tilført med si hovudfagsoppgåve "Designing Pedagogical Agents for Collaborative Learning: An Empirical Study" (Omdahl 2002). Desse to oppgåvene er difor ein vesentleg del av referanserama for det vidare arbeidet med agentutviklinga.

## 3.1 DoCTA prosjektet

Første del av DoCTA prosjektet DoCTA I (Design and use of Collaborative Telelearning Artifacts) (Wasson, Guribye & Mørch 2000), hadde fokus på IKT-meditert samarbeidslæring med fokus retta mot opplæring av lærarar. Studiet tok for seg dei sosiale, kulturelle, pedagogiske og psykologiske aspekta i prosessen der IKT-artifaktar er tekne i bruk. Hovudfokuset for forskninga var 3-delt:

1) Å definere eit sosiokulturelt perspektiv på læringsaktivitetar der settinga var samarbeidslæring og temaet var studie av dei mellommenneskelege interaksjonane.

2) Å bidra med kunnskap om samarbeidslæring i digitale omgjevnader i høve til pedagogiske design av læringsscenarium.

3) Å studere og evaluere dei sosiale og kulturelle aspekta ved distribuert samarbeidslæring i digitale omgjevnader (Brendshøi 2003) .

Den andre fasen av DoCTA, DoCTA NSS (Natural Science Studios) (Wasson & Ludvigsen 2003) gjekk vidare med røynsler frå DoCTA I . Kunnskapen skulle no overførast til eit nytt område der ein la til design og utvikling av teknologi til det eksisterande teoretiske grunnlaget for prosjektet.

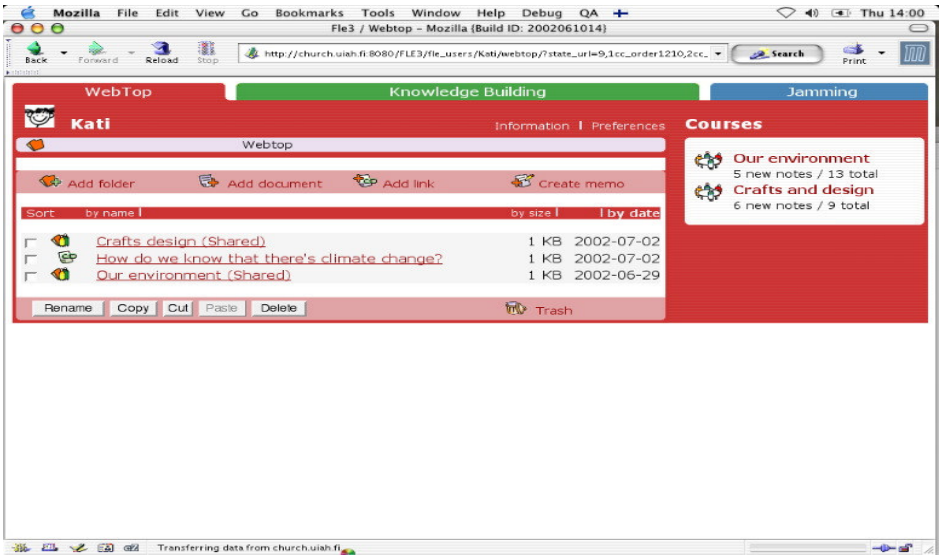
Det overordna målet til prosjektet er å studere korleis eit pedagogisk design av gruppevaresystem for samarbeid og læring mogeleggjer diskusjon, og korleis denne prosessen støtter kunnskapsbygging. Eit sentralt føremål er å forstå korleis elevane brukar eit spesifikt verktøy (FLE2 / FLE3) til kunnskapsbyggingsprosessen, og korleis dette støttar elevane i å "resonnere vitskapleg" (Brendshøi 2003). Den teknologiske utviklinga av programvare /agentar er ein konsekvens av innsikta i desse grunnprosessane.

DoCTA prosjektet sitt teoretiske rammeverk inkluderer to paradigme. Computer Supported Collaborative Learning (CSCL) og Koordineringsteori (Malone & Crowston 1994). I prosjektet vert "telelearning" miljø kombinert med samarbeidslæring for å meistre den tette samanhengen mellom ulike aktivitetar (Malone & Crowston ibid.) og støtte den gjensidige avhengigheiteten mellom dei forskjellige aktørane (Wasson 1997). DoCTA NSS bygger vidare på DoCTA I prosjektet der ein basismodell av gjensidig avhengigheit mellom brukarar i elektroniske samarbeidsmedier har vart utvikla (Wasson, Guribye & Mørch 2000)

Dei IKT verktøya som blei brukte i DoCTA I og DoCTA NSS, var blant anna FLE2 og FLE3 (Future Learing Enviroment 2 og 3). FLE2 blei brukt i pilotstudium (Genetikkscenariet) og involverte to skular: Holmlia (Oslo) og Ytrebygda (Bergen). I feltstudiet vart FLE3 nytta. Hovseter (Oslo) og Sandgotna (Bergen) vart no involverte.

## 3.2 Future Learning Environment 3 (FLE3)

Future Learning Enviroment 3[4] er utvikla ved Media Lab, University of Art and Design Helsinki (UIAH) Finland, og er eit verktøy for gruppesamarbeid via nettet. Det er bygd opp med eit felles grensesnitt mot brukarane, etter metaforen om filmapper i arkivskap. Brukarane startar med hovudsida som er kalla Webtop. Denne gjev tilgang til ein del personlege muligheiter som å laste opp filer, lage memo og leggje til linkar. Webtopen har då funksjon som eit skrivebord for den enkelte brukaren. Det er også mogleg å dele eigne filer med andre ved hjelp av ein eigen kursfoldar for kvart fag eller kurs ein er medlem av. GUI designet til FLE3 er utvikla av Kligyte (2001). Nokre av kategoriane for katalogiseringa av innlegg, fekk endra namn. Nokre av dei gamle katagoriane blei fjerna. Dette fordi antalet og typane av katagoriar forvirra meir enn dei støtta opp om kunnskapsbyggingsprosessen (Kligyte ibid.)




Figur 14 Startvindauga til FLE3 (Webtop)

[4] http://fle3.uiah.fi

FLE byggjer på det pedagogiske konseptet i Progressive Inquiry Learning (Hakkarainen 1998), som er ei vidareføring av Knowledge building (Bereiter & Scardamalia 1993) Begge læringsmodellane er beskrivne meir detaljert i kapittel 2.4.3 og 2.4.4

Hovudstrategien i FLE3 fell saman med ideen om Problembasert læring. Det sentrale her er prosessfasane:

1) Definering av problemet

2) Mi forklaring/ mi forståing av problemet

3) Vitskapleg forklaring

4) Konklusjon/oppsummering.

Denne strategien skal gje ein betre læreeffekt i høve til gruppevaresystem som er ukategoriserte, i den forstand at det ikkje har katagoriar som kvart innlegg skal sorterast under. Dermed vil ein gå igjennom ein prosess der kunnskap blir danna ut frå kva ein har definert problemet som. Forklaringar kan skape nye problem som igjen fører til nye prosessar som aukar kunnskap og innsikt i spiralformasjon. Ved denne fasetenkinga prøver ein å operasjonalisere og realisere læringsprosessen slik Vygotsky framstiller det. På denne måten knyter ein aktivt ny informasjon til eigne førestillingar i ein kontinuerlige endrings- og justeringsprosess i staden for å bygge opp ein fragmentert leksikalsk kunnskap utan samanbindande strukturar. Problemdefineringa føreset at ein alt har tankar om tema, noko som indikerer grunnlaget i den proksimale læringssona.

## 3.3 Simulering av Agentar i eit virtuelt læringsmiljø

I Jondahl (2001) blei det utført ei simulering av pedagogiske agentar i eit scenario. Simuleringa blei gjennomført i prosjektet CoPAS (Collaboration Pattern Agent Simulation)

Gruppevaresystemet TeamWave vart valt som verktøy for simuleringa. Dette verktøyet er eit synkront (Gutwin & Greenberg 1995) og var høveleg som simuleringsverktøy for agentfunksjonaliteten. Simuleringsteknikken i forsøket var Wizard of Oz teknikken (Jondahl 2001).

Forsøket til Jondahl (ibid.) involverte 4 grupper. Av desse fire var det ei kontrollgruppe. Medan forsøket blei gjennomført, registerte "agentane" kva kommentarar som blei brukte. Desse kommentarane vart så analyserte med tanke på kva nytte deltakarane hadde hatt av dei pedagogiske agentane og kva påverknad desse hadde hatt på deltakarane si forståing av problemstillinga dei var sett til å løyse. I testen hadde ein lagt inn 3 ulike simulerte agentar: Verktøysagenten, Samarbeidsagenten og Fagagenten. Kvar av desse hadde førehands definerte fraser for svaralternativ til brukarane av systemet. Tilbakemeldingane var katalogiserte etter type og agenttilhørigheit. Tre hovudfagsstudentar ved Intermedia UIB ikledde seg rollane som agentar.

Agentane kommuniserte ved hjelp av popup-boksar med tekst. Meldingane frå agentane til studentane var førehandsdefinerte i to kategoriar: generell info og spesifikk info. Fordelinga mellom agentane av kva type informasjon dei skulle gje var i samsvar med den rollen/funksjonen dei hadde. Samarbeidsagenten hadde flest generelle svar av typen som vist i meldinga i Figur 15.


Figur 15 Ei av melding frå samarbeidsagenten i CoPAS forsøket.
(Jondahl 2001)

Gjennom testen av agentane kom det opp fleire problem. Desse måtte løysast medan testen pågjekk. Det viktigaste problemet var at dei førehandsdefinerte frasene ikkje var nok til å dekke alle spørsmål som kom inn til agentane, samt at nokre fraser ikkje blei brukte i det heile (Jondahl, 2001 Vedlegg G).

Etter testkjøringa vart agentypar og aktivitetstype diskuterte ut frå intervju med testbrukarane og evalueringa av datamaterialet. Følgjande funn blei rapporterte:

- Breakdown (agentmuligheiter / problem )
- Aktive og passive agentar
- Funkjsonalitet og bruksområde. Ein agent kan vere begge delar.
- Reaktive agentar mot animerte agentar

Jondahl (ibid.) konkluderer med at dialogforma mellom agent og brukarar bør delast opp i to kategoriar: generell informasjon og spesifikk informasjon. Den generelle informasjonen, føreslår ho, bør bli presentert i eit permanent vindauga integrert i websida/applikasjonen. Når det er viktig informasjon eller kritisk informasjon agenten skal formidle, bør popup boksar nyttast. Overbruk av popup boksar vil distrahere eller irritere brukarane slik at dei lett kan kome til å ignorere meldingane frå agenten.

## 3.4 Design av pedagogiske agentar

Omdahl (2002) set opp tre sannsynlege område agentar kan støtte opp om i eit samarbeidslæringsscenario slik DoCTA NSS legg opp til.

- Støtte til asynkront arbeid.
- Støtte kategoriseringa i FLE

- Støtte kunnskapsbyggingsprosessen som skal føregår i FLE.

Ut frå Omdahl sine tre punkt kan ein sette opp 2 ulike typar av pedagogiske agentar som vil tilfredstille alle tre områda: samarbeidsagent og kunnskapsbygginings agent.

### 3.4.1 Design av samarbeidsagent

Ein samarbeidsagent vil kunne dekke to av Omdahl sine tre punkt for agentstøtta prosessar i asynkrone verktøy. Den asynkrone arbeidsforma gjer det vanskeleg for instruktørar/lærar å halde oversikt over læringsprosessane til deltakarane (Omdahl 2002). Denne oversikta er lettare å overlate til ein agent, sidan denne kan ha mogelegheiten til konstant å overvake gruppevaresystemet og dei endringane brukarane gjer der. Eit av forslaga, som blir sett på som ein agentfunksjon, er moglegheita for ein nyhendebulletin om endringar/nye postar o.l i det ein brukar loggar seg inn på systemet etter å ha vore avlogga for ein periode.

Kategoriseringa i FLE viste seg å ikkje vere så enkel som ein på førehand hadde tenkt. Pilotstudiet i DoCTA viste tydeleg dette. Då postane blei gjennomgått manuelt, var det endringar på ein del av kategoriane. Den største endringa var "Prosess kommentar" som gjekk frå 4% originalt til 17% etter den manuelle gjennomgangen (Omdahl 2002, s 61, figur 6-4 a og b) Omdahl foreslår at ved postingar skal agenten komme opp ganske raskt slik at den kan hjelpe usikre brukarar til å velje rett kategori for notatet dei er på veg til å poste (2002). Ein av dei viktige måla i CSCL er å oppnå ei felles kunnskapsforståing mellom brukarane innad i gruppevaresamfunnet (Community of learners). Omdahl tenkjer seg at ein pedagogisk agent kan fremje denne felles kunnskapsaktiviteten. Data Omdahl brukte stammar frå DoCTA I og FLE2. Då dette arbeidet fokuserer på ein ny versjon av FLE, er det nokre av Omdahl sine aspekt som ikkje direkte er naudsynte då FLE3 har implementert løysningar på desse problema.

### 3.4.2 Design av kunnskapsbyggingsagent

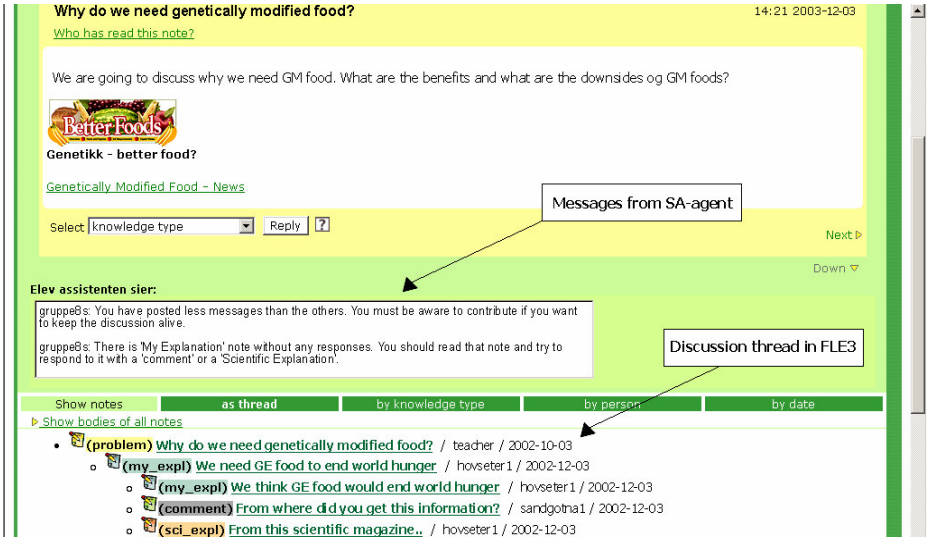
I og med at kategoriseringa var eit problem, og dermed hindra kunnskapsbygginga hjå studentane, vil Omdahl ha agenthjelpefunksjonar inne i diskusjonsforumet. Eit døme på dette er når agenten rettleier brukarane i korleis Progressive Inquiry Learning fungerer, og korleis dei ulike notattypane skal følgje etter kvarandre. Agenten skal også overvake postane slik at feilposting ikkje førekjem. Eit eksempel på dette er følgjande: Dersom eit problem blir respondert på med oppsummering, vil agenten agere mot brukaren som posta feil, med råd om at det er betre å poste ei foreløping oppfatting, eller ei spesifisering av problemet som ein kommentar. Ein av dei funksjonane Omdahl tenkjer seg vil kunne ha ein god effekt, er ein "skjul /vis"- funksjon for kvar av kategoriane. Der studentane startar med berre to kategoriar og når eit notat er posta med desse kategoriane, gjev agenten brukarane høve til å nytte fleire kategoriar. Agentideen til Omdahl har ho henta frå eit anna samarbeidslæring-system, Kolumbus (Misch, 2001).

## 3.5 Tidlegare implementerte agentar

I DoCTA prosjektet er det utvikla to forskjellige agentar. Dolonen (2002) utvikla ein pedagogisk agent for gruppevaresystem der agenten hadde fokus på å assistere studentane. Første versjon av agenten var laga for FLE2, men då FLE3 kom, blei ein ny versjon av agenten utvikla spesielt for denne versjonen. Chen (Chen & Wasson 2002) utvikla ein agent som var retta mot instruktørane/lærarane i gruppevaresystemet.

### 3.5.1 Student Assistant agent (SA)

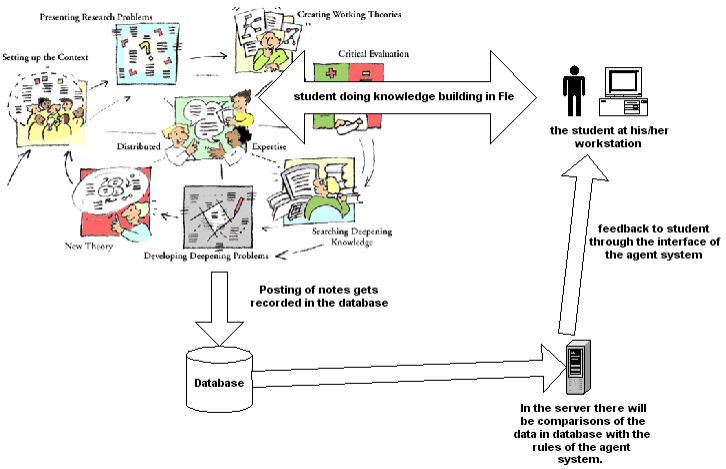
Dolonen 2002 tok føre seg utvikling av ein pedagogisk agent i FLE 2, migrert til FLE 3. Agenten var programmert i Java med ein progress-sql database løysning som lager for agenten. Agenten er basert på funn i Jondahl (2001) sjølv om gruppevaresystema Teamwave og FLE3 er ulike både i design og bruksområde. Figur 16 viser agenten sitt permanente vindauga integrert i FLE3. Etterkvart som agenten sine reglar blir brotne, vil meldingane kome opp i vindauga.




Figur 16 SA-agent integrert i FLE3 (Dolonen 2002)

Dolonen sin agent er bygd opp for å fungere i kunnskapsbyggingsmodulen til FLE. Hovudmålet til agenten er å få kvar student til å bli merksam på kva ansvar dei har i kunnskapsbyggingsprosessen som FLE legg opp til. Dette ansvaret skal dei bli medvitne om gjennom ulike meldingar frå agenten. For viktige meldingar kjem det opp ein popup boks med den aktuelle meldinga i tillegg til at den kjem opp i det permanente vindauga.

Figuren nedanfor viser korleis SA-agenten er sett opp til å fungere i FLE sin kunnskapsbyggingsmodul. Prosessen mellom FLE og agenten går parallelt med kunnskapsbyggingsprosessen mellom studenten og FLE.




Figur 17 Agent support in FLE (Dolonen 2002) Agentens strukturintegrering i FLE sin kunnskapsbygging modul

Agenten henter datanene i FLE ut gjennom databasekall.. Tabell 3 viser dei 3 variablane som blir henta ut og sett opp i tilhøyrande databasetabell, Denne tabellstrukturen er også gjenbrukt i SA[2]-agenten.

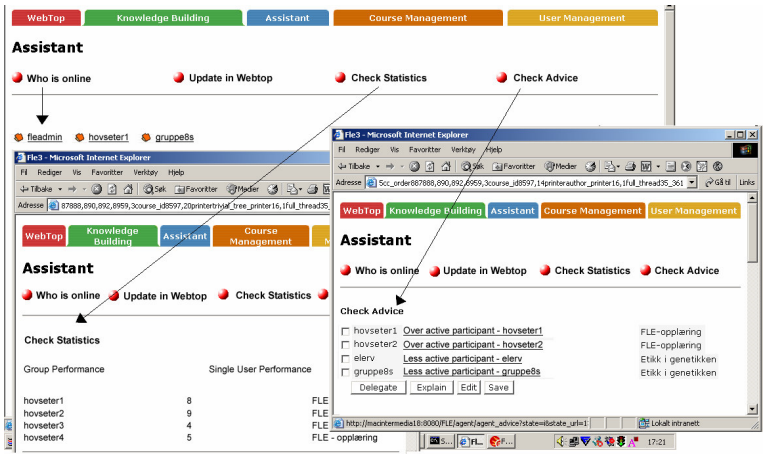
| Table names: | Field in the tables: |
|---|---|
| User | Name, ip_address, login time, is_active |
| Note | Name, note_id, note_title, in_reply_to, path, thinking_type, timestamp |

Tabell 3 Database tabellene for info henta ut frå FLE (Dolonen 2002)

Databasetabellen "User" held styr på brukarane, kvar dei sit (ip_adress) og om dei er aktive. Dolonen peikar på at SA-agenten er lite fleksibel for regelendring. Den støttar endringar i tekstfrasene. Jondahl (2001) oppdaga dette behovet i CoPAS forsøket, og dermed blei det implementert i SA agenten. Grunnreglane er fortsatt ikkje editerbare "On the fly". For å få redigert, sletta eller endra reglane, må agenten omprogrammerast og kompilerast på nytt. Dette er ein tidkrevjande prosess og føreset gode programmeringskunnskapar.

### 3.5.2 Instructor Assistant agent (IA)

Chen og Wasson (2002) utvikla ein annan type agent enn det som tidlegare var forsøkt i DoCTA NSS prosjektet. Ein statistikkagent har som oppgåve å observere bruken av systemet og lage ein statistisk presentasjon som så vert sendt til læraren/instruktøren. Det som skil denne agenten frå andre agentar, er at den er passiv i høve til brukarane. Den agerer ikkje ut mot brukarane med informasjonen den register (Dolonen 2002). Eit anna aspekt med Chen sin agent er at den er direkte integrert i FLE3 som eit eige element på lik linje med dei andre funksjonane til FLE3. Dette medfører at den ikkje er tilgjengleg for andre enn dei som er lærarar, instruktørar eller administratorar i FLE3. Agenten adopterer designet av brukargrensesnittet til FLE 3 og ser ut som ein modul av FLE3. Chen & Wasson (2002) adresserer litt dei same problema som Omdahl når det gjeld å halde oversikta i asynkrone gruppevaresystem. IA-agenten er designa for å lette på problema lærarrolla får i distribuert samarbeidslæring samanlikna med tradisjonell undervisning (Chen & Wasson 2002).




Figur 18 Instructional Assistant Agent (Chen, Dolonen & Wasson 2003)

## 3.6 Plattforma for mitt arbeid

I det føregåande har eg berkrive nokså inngåande en den av det tidlegare arbeidet som er gjort ved InterMedia UIB og i DoCTA-prosjektet og utvikling av FLE3. Dolonen (2002) sitt arbeide er det viktigaste fundamentet (Figur 19) for den praktiske delen av arbeidet mitt, medan teortiske har røter i mange fagretningar og trekkje vekslar på forskingsarbeid og publikasjonar som kjem frå miljøet rundt DoCTA prosjekta, men også frå ekstrene forskningsgrupper, særleg frå forskningsmiljøa rundt CSCL i Finland og Canada.

Hovudfagområda i prioritert rekke følgje:

Direkte relevans for oppgåva

- Agentteknologi
- Menneske Maskin Interaksjon
- Sluttbrukar utvliking (EUD)

Indirekte relavans eller rammeverk for arbeidet.

- Computer Supported Collaborative Learning
- Computer Supported Coopartive Worck

# 4 Problemskildring og avgrensing

Først vil eg setje opp aktuelle problemsituasjonar der eg meinar ein kan ha nytte av programvareagentar integrert i systema.

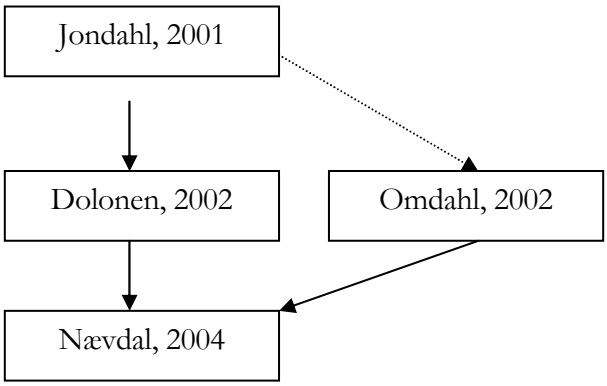
I eit gruppevaresystem kan det oppstå problem med samarbeidet mellom deltakarane når dei ikkje er samlokaliserte. Eit gruppevaresystem er basert på gruppesamspelet, og det er difor vesentleg for læringsprosessen at dette fungerar tilfredsstillande.

Gjennom gruppevaresystema skal ein frigjere seg frå det fysiske møterommet og ta steget inn i det virtuelle grupperommet. Det ønskjelege er at ein tek med dei positive elementa frå den fysiske møtetradisjonen på same tid som ein tek i bruk dei fordelane ein kan få i eit virtuelt miljø. Erfaringane viser likevel at dette ikkje alltid er like lett for brukarane (Grudin 1991). Hyperstrukturen i det virtuelle miljøet gjer det og vanskeleg for brukarane å følgje med kva som blir gjort av dei andre i systemet. Dette problemet kan truleg reduserast vesentleg ved hjelp av samarbeidsagentar som overvakar og rettleiar brukarane fram mot eit funksjonelt samarbeid. Agenten kan operere både ved "tips" i starten og ved intervenering når brukaren utfører "unaturlege" handlingar undervegs.

Gruppevaresystema i denne oppgåva ligg innanfor CSCL feltet (Computer Supported Collaborative Learning) der virtuelle læringsrom erstattar klasseromma når gruppeoppgåver skal gjennomførast, og der felles fysisk lokalisasjon ikkje er mogleg eller er ugunstig (langt å reise o.l.). Gruppevaresystem kan også vere eit supplement i den ordinære klasseroms-undervisninga der ein brukar datamaskinane og gruppevaresystemet som supplerande aktivitet.

Jondahl (2001) gjennomførte eit empirisk studie av agentar gjennom simuleringsteknikken "Wizard of Oz". Omdahl (2002) beskriv Interface agentar ut frå eit teoretisk grunnlag og sorterer agentane i grupper, basert på kognitive karakteristika (gjenkjenning, tillit). Forskingsrapportar (Jondahl 2002) har vist at det å oppnå brukarane sin tillit, slik at dei

lyttar til meldingane agenten kjem med, synest å vere ei av dei største utfordringane i konstruksjonen av funksjonelle gruppevareagentar. Omdahl (2002) set fram hypotesar basert på det empiriske materialet som tilgjengeleg frå agenttesting og frå studium av samarbeidslæring (Wasson, Guribye & Mørch 2000). Desse hypotesane dannar det praktiske rammeverket for dette arbeidet knytt til den pedagogiske sida av agentutviklinga. Figur 19 viser samanhengen mellom dei tidlegare hovudfagsoppgåvene ved Institutt for Informasjonsvitenskap, Universitetet i Bergen og mitt arbeid.




Figur 19 Samanhengsoversikt mellom mitt arbeid og tidlegare arbeid ved IFI/UIB

Dolonen (2002) har utvikla agentar som opererer etter fastlagde reglar. Slike regelrigide agentar har nokre fordelar, som t.d. lettare implementering, men også ei rekkje ulemper, særleg når det er trong for endring av reglane. Kvart forsøk / brukssituasjon krev ofte ulike reglar eller regelsett. Ved å implementere eit sluttbrukargrensesnitt (EUD) skal læraren/ forsøksleiaren kunne lage, laste inn eller endre regler som legg ulike grunnlag for oppførselen til agenten. Reglane som agenten skal bruke i samarbeidssituasjonen, blir programmert i ein brukar-orientert editor. Ut frå studie av pedagogiske agentar som er blitt gjort i andre arbeid (Johnson 1998, Omdahl 2002, Baggetun, Dragsnes & Dolonen 2001), kan ein konkludere med at pedagogiske agentar har ein autonom funksjon og er dermed ein autonom agent.

Ein pedagogisk agent kan ha fleire attributt. Dette er omtalt hjå blant anna Baggetun, Dragsnes og Dolonen (2001). Ved å setje opp eit kriterium for dei attributta ein pedagogisk agent må ha for å kunne kallast pedagogisk agent, har eg utvikla ein tabell med tre hovudattributt for autonome / Interface agentar (Nwana 1996) Etter å ha definert agenten som autonom eller som Interface agent etter Nwana (ibid.), vil eg hevde at ein pedagogisk agent bør ha desse tre overordna attributta : i) oppførsel, ii) kommunikasjon og iii) funksjonstype, og iv) Peadagogikk.

- **Oppførsel /Intervenering**
    - Aktiv
    - Reaktiv
    - Passiv
- **Kommunikasjon**
    - Animert,
    - Pop-up
    - Permanent
- **Funksjonstype**
    - Domain
    - Collaborate
    - Instuctor

- **Pedagogikk**
    - Progresive Inqiry Learning
    - Knowlegde Building,
    - Intelligent Tutoring System
    - Lærar resurssar.

Ein pedagogisk agent treng å ha alle fire attributta for å kunne kallast ein pedagogisk agent. Kvart av desse attributta har underkatagoriar som vil kunne skilje mellom forskjellige typar av pedagogiske agentar.

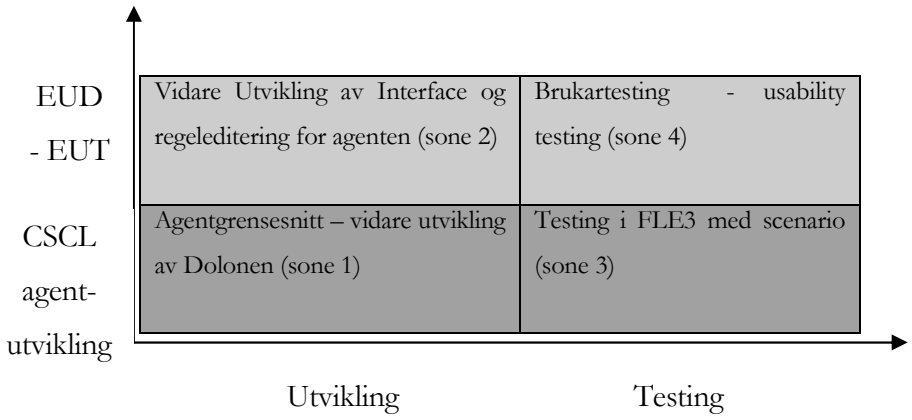
## 4.1 Oppgåveformulering

Eg setje opp følgjande to utviklingsmål som mål for arbeidet:

- Konstruere ein editerbar agent som aukar fleksibiliteten i læringsprosessen.
- Utvikle ei nøytral animert kommunikasjonsform som reduserer agentpersonifisering i høve til ordinære animerte agentar på same tid som ein utnyttar sentrale fordelar ved animerte agentar.

### 4.1.1 Avgrensing

Utviklingssyklusen til agenten kan delast inn i fire soner (Figur 20) Kvar del har fokus på ulike aspekt av informatikk- og pedagogiske fagretningar. Følgjande utviklingsarbeid tar for seg sone 1 og sone 2 som utviklingsoppgåve. Sone 3 og 4 er relatert mot evaluering og testing av agenten i pedagogiske scenarier. Dei to siste sonene her eg har sett opne til framtidig arbeid av to årsaker : 1: Arbeidet mitt er eit utviklingsarbeid, der ei vurdering på pedagogisk grunnlag fell utanfor rammene til oppgåva. 2: Ein med bakgrunn i pedagogikk vil vere mykje betre eigna til å gjere dette enn ein informatikar er.



Figur 20: Avgrensing av oppgåva sonedelt

Figuren over viser det innhaldet eg har lagt i dei 4 sonene. Nedanfor vert dette innhaldet kommentert meir detaljert.

### Sone 1: CSCL/Utvikling :

Denne sona omhandlar sjølve agentutviklinga. Dette er ei vidareutviklinga av Dolonen (2002) sin agent, der aspekta er i) Bruker-agent dialog, ii) Gjenbruk, iii) Integrering i gruppevaresystemet. Namnet er har gitt på denne modulen er Student Assistent 2 agent(SA2) Kapittel 6 omhandlar denne delen av oppgåva.

### Sone 2: EUD/Utvikling

Utvikle regelediterings-funksjonen slik at ein oppnår ein stabil applikasjon til å nykonstruere, endre, eller slette agentreglar i samsvar med dei behov brukarane har i ulike lærinsgsituasjonar. RuleEditor er det navnet eg har gjeve den programvaren som blir utvikla i dette soneaspektet. Kapittel 7 bekriv utvlikingsprosessen av denne applikasjonen.

### Sone 3: CSCL/Testing

Denne sona/ fasen omhandlar empirisk evaluering av i)Agent-brukar grensesnittet, ii)Agent administrator grensesnitt og iii) Effekt av animert agent i forhold til popup og permanente vindauger i FLE.

### Sone 4: EUD/Testing

Den siste sona/fasen omhandlar brukbarheita til regelediteringsverktøyet som SA2 vert levert med. Testinga er knytt til ulike brukartypar. I tillegg skal ein skaffe seg innsikt i korleis brukarane forstår layout, design og struktur i oppsettet for regeleditering.

# 5 Metode

I dette kapittelet beskrive eg verktøy og metodar brukt i utviklinga av SA2 agenten, FLEA og RuleEditor. Nokre av verktøya er brukt allment, medan andre berre er brukt i enkelt tilfelle.

## 5.1 Programmeringsspråk

Programmeringsspråket Java er nytta som utviklingsplattform for agenten og regeleditoren. Java vart vald av fleire grunnar. Hovudgrunnen er gjenbrukspotensialet som ligg i eksisterande arbeid (Dolonen 2002). Då FLE3 er eit webbasert grensesnitt, har Java fordelar ved at det er plattformuavhengig slik at ein kan oppnå kompabilitet mellom ulike maskinplattformar (PC, Mac, Unix)

Python vart vurdert som eit alternativ til Java fordi det er utviklingsspråket både til Zope og FLE3. Argumenta for å ikkje bruke Python er knytt til tre kriterie: i)gjenbrukspotensialet, ii)Jython er eit modulspråk som overset mellom Python og Java og at iii)utviklar alt kjenner Java og er fortruleg med det.

MySQL er valt som databaseverktøy. Alternativet kunne til dømes ha vore MS SQL, PosgresSQL. MySQL vart funne tenleg fordi ein her hadde tilgang til gratis programvare, og den var tilgjengeleg ved mest alle lærestader. MySQL er brukarvenleg og stabil og brukar ikkje for mykje ressursar, sjølv under sterk belastning.

Jbuilder vart vald som plattform for utvikling av visuelt grensesnitt. Forutan at utviklar var godt kjend med dette verktøyet, var det viktig at JBuilder hadde ein del ekstra bibliotek i tillegg til JDK pakken til SUN. Biblioteka blei importerte til JDK når applikasjonen blei testa i Java 1.4.1.01. Både agent og regeleditor har blitt kompilert i begge kompilatorane (JBuilder og j2sdk1.4.1.01) og dette har fungert godt. I ein visuell kompilator (GUI builder) er det lettare å lage brukargrensesnitt i (samanl. med td. Emacs) då denne typen kompilatorar føyer seg betre inn under EuP prinsippa.

Sidan eg måtte editere ein del filer i FLE3, måtte eg og setje meg inn i syntaksen i Python. Programmeringa som vart gjort i FLE filene, var mest databasekall for å hente ut og leggje inn dei nye verdiane etter at kvar brukar la inn nye innlegg eller berre logga seg inn på systemet.

For å få integrert Agenten i FLE, måtte eg modifisere nokre filer i FLE3 kjernen. Desse filene er dei same som Dolonen (2002) modifiserte til agenten i si oppgåve. Eg har gjort ein del endringar av følgjande grunnar: For det første brukte eg ei anna databaseløysning, og for det andre trong den nye agenten meir informasjon frå systemet enn Dolonen(ibid.) sin agent gjorde. Grunnen til det utvida informasjonsbehovet er knytt til utvida delreglar. Dei modifiserte filene er Index.py, note.py. Det som er modifisert er gjengitt i Vedlegg B..

## 5.2 Design

I den prosessen med design og utvikling av to elementa mitt arbeid består av er det mykje teori som ligg bak utan at ein kan peike direkte på at ein spesifikk teori er brukt i ei eit spesifikt tilfelle. Mykje av den teorien som eg har tatt opp frå CSCW og CSCL fagfelta er brukt som underliggande materiale i utviklinga av begge applikasjonane. Awareness begrepa er noko av det som ligg mykje til grunn for designet i forhold til korleis agenten både oppfører seg og korleis den er blitt implementert i FLE. Usablilty begrepa til Nielsen (1993) er blitt brukt som verifisering av design og brukbarheiten av begge programma i eksperttestinga.

Wizard-teknikken er digitaliseringa sitt svar på den papirbaserte skjemautfyllinga. I staden for å ha eit skjema (form) der informasjon vert standardisert, deler wizard-teknikken opp "skjemakonstruksjonen" i seksjonar der systemet etterspør avgrensa informasjonsbitar i ei funksjonell rekkefølgje i separate skjermbilete. Framgangsmåten lettar informasjonsinn-fyllinga for brukarane. Installering av programvare med Windows Installer® er eit typisk døme på denne teknikken der brukarane vert leia gjennom installasjonen utan at dei treng å ta stilling til meir enn eitt informasjonsfragment om gongen.

For å animere agenten sitt reaksjonsmønster, blei eit enkelt GIF animeringsprogram brukt. Ved at agenten har fire svarmodusar, blei fire ulike aktivitetsnivå av animasjonen laga. Aktivitetsnivået til agenten kan variere på kvart av nivåa. Dei fire nivåa er også relatert til viktigheitsskategoriane til agenten. Desse kategoriane er også gitt med eksempel i same figur. Dess høgare kategori som er blitt utløyst, dess viktigare er beskjeden frå agenten, og dess meir aktivitet/ animering har agenten. Ved brot på regel i kategori 3, vil også agenten skifte farge mellom grønt og raudt. Grønt er den normalfarge, medan raudt er eit naturleg faresignal. Når regel på kategori 3 er broten er det svært viktig at brukarane tek omsyn til meldinga.

## 5.3 Evalueringsmetodar

I starten av arbeidet vart det planlagt ei større brukartesting av agenten i eit verkeleg scenario. DoCTA NSS sitt feltstudie var den mest sannsynlege settinga for ei slik større brukartesting. Då agenten ikkje var ferdig utvikla og stabil innan den tid, var det ingen moglegheit å ta den med i feltstudiet.

Usability refersanser til testinga

Regeleditoren har heller ikkje gjennomgått større empirisk testing i ei røynlege brukararsetting.

Agenttestinga, som er føretatt, er ei heuristisk evaluering med ulike medstudentar som har hatt erfaring innan slik evaluering. Det er og føreteke ekspert testing, der utviklar av den førre agenten har vore med og evaluert/ testa agenten ut frå kriterium som verka fornuftige å teste etter. Dei ulike aspekta som er blitt testa, har eg sett opp med dei aktuelle testformene i parentes.

- Design (heuristisk evaluering/ ekspert /medstudentar)
- Brukbarheitstesting (Ekspert testing)

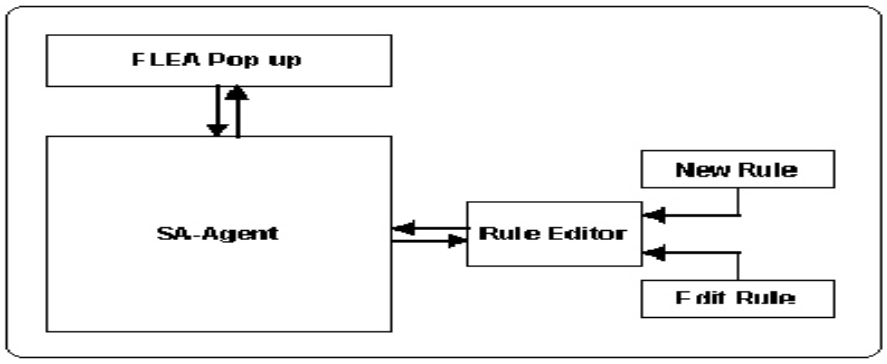
- Programmeringsstruktur og kommunikasjonsstruktur (Ekspert testing)

- Dialogform (Ekspert testing)

# 6 Agentutvikling

Eg beskriv først vidareutviklinga av SA agenten, der eg kallar den nye agenten SA2. Den animerte interaksjonsdelen av agenten har fått eit eige namn FLEA (Future Learning Enviroment Agent) Sjølve namnet og grunnteikninga til agenten har eg henta frå FLE3, men der er det visualiseringa eit stillestående GIF bilete og ein fast tekst. Det vart vurdert å laga ein ny figur til animeringa, men sidan eg ville utnytte gjenkjenningsfaktoren, brukar eg det same bildet som utgangspunkt. Dermed er det og naturleg å også bruke FLEA namnet slik UIAH har brukt det innafor FLE3.

Den andre delen av mitt utviklingsbidrag omhandlar RuleEditor (kapittel 7). Denne programvara er uavhengig av agenten, men er likevel ein viktig del av den. RuleEditor er eit forsøk på å lage eit høgnivå programmeringsspråk for reglane til SA2 slik at dei kan bli laga, redigerte eller sletta utan at agenten treng å bli restarta eller kompilert på nytt for å bli oppdatert med dei nye reglane. Strukturen er lagt opp slik at både brukarar og instruktørar/lærarar kan gjere desse endringane. Ei oversikt over korleis FLEA SA2 og RuleEditor heng samen, er vist i figuren under.




Figur 21Modell over strukturen over kva nye element eg har lagt til SA-Agent

Ein arbeidshypotese eg har jobba etter er relatert til Vygotsky sin teori om ZPD sona:

Ved hjelp av ein pedagogisk agent vil studentane kunne prestere betre fagleg, enn kva dei ville gjort viss dei hadde arbeidd åleine. Agenten skal kunne agere som ein meir kapabel medstudent eller lærar.

Eg trur ikkje at agenten fullt ut kan erstatte fordelane som menneskeleg rettleiing i grupper har, men at agenten skal kunne bidra til ein betre læringseffekt kan ein nok anta.

## 6.1 Føresetnadar for agenten

Rammeverket rundt SA2-agenten er det same som for SA-agenten som Dolonen (2002) arbeidde med. Ei rekkje av føresetnadane som var lagt opp for hans agent er også gyldige for mitt arbeide. Dette gjeld særleg scenariet Dolonen sette opp.

Dei pedagogiske føresetnadene for agenten baserer seg på "Knowledge Building" (Scardamalia & Bereiter 1993) "Progressive Inquiry" (Hakkarainen 1998), og "awareness" teoriane til Dourish & Belloti (1992). Regeleditoren RuleEditor, implementerer også pedagogikken til instruktørane gjennom dei editerbare reglane. Ved å la menneske stå for editeringa av reglane, og ikkje agenten sjølv, unngår eg problema knytt til maskinene sin manglande evne til å forstå sosiale prosessar, slik ein arbeider med dette innafor fagfeltet kunstig intelligens (KI, eng AI).

Mykje av arbeidet som er gjort før denne oppgåva er difor med på danne grunnlaget for utviklinga av SA2 . Dei 9 sub-funksjonane Dolonen la til i sin agent, er viktige element som er og blir brukt i mitt arbeid, men implementeringa av desse sub-funksjonane er modifisert. Dolonen identifiserte 3 problemområde for agenten: i)deltaking, ii)kategori og iii)ukommenterte poster. Eg har delt kvar av områda opp i fleire underkategoriar. Dei fleste er meir eller mindre ei spesifisering og oppdeling av Dolonen (ibid) sine 3 virkeområde, men eg har og lagt til eit nytt område eg har kalla "Innlogging". Dette sjekkområdet gir blant anna agenten høve til å ta kontakt med brukarar som ikkje så ofte er på logga eller ved er logga på for første gang. Eg ser på dette nye området som ein potensiell

informasjonskanal for råd og rettleiing frå agent til brukar. Særleg viktig er funksjonen som oppstartsrådgjevar for brukarane.

Kvar underdel av dei fire unike områda kallar eg ein primærvariabel. Kvar delagent for regelsjekking har ein primærvariabel den tek seg av. Når ein regel blir konstruert, vel brukaren først ut kva primærvariabel regelen skal leggje til grunn, deretter kva verdiar den aktuelle regelen skal ha. Dei primærvariablane eg har identifisert/sett opp i dette studie er:

1) Siste Innlogging.

2) Talet på innloggingar

3) Talet på poster.

4) Talet på usvarte poster.

5) Talet på problem

6) Talet på usvarte problem

7) Talet på problem.

Talet på primærvariablar er ikkje absolutt, og det finst mange av dei, men i dette arbeidet har eg berre tatt med dei som er opplista ovanfor. For å kunne utføre oppdateringar enkelt, blei desse primærvariablane lagt i eit eige databasefelt som vist i Tabell 5.

Ved å lage ein Primærvariabel for kvar delagent vil det også bli lettare å vidareutvikle agenten. Ved å leggje til ein ny primærvariabel, legg ein og inn ein ny delagent som tek seg av denne. Einaste internkoding som trengst, er eit kommandokall i RuleCoordinator Agent, Dette kallet er standardisert og likt for alle delagentane. Dette prinsippet medfører at SA2

agenten ikkje treng noko større omstrukturering for å bli utbygd seinare. Innlegging av kommandokall er vist i avsnitt 6.2.3.

### 6.1.1 Reglar

Agenten sine reglar består av 6 forskjellige variablar. Eg vil sette opp ein enkel skildring av dei forskjellige variablane i tabellform.

Tabell 4 Regelvariablane forklart. Samanstninga av kva parameter ein regel har som oppbygning

| | |
|---|---|
| Primærvariabel | Kva underdel av dei fire områda regelen skal omhandle |
| Verditype | Avgjer om verdiane i regelen skal reknast som absoluttverdiar- eller gjennomsnittverdiar |
| Typevariabel | Avgjer om talverdiane anten lik større eller mindre enn verdien agenten skal samanlikna regelverdien med, Denne verdien kan også komme som intervall |
| Talverdi | Verdien agenten skal sjekke sine innhenta data mot, talet på verdiar kan variere frå 1 til 4 verdiar, avhengig av kompleksiteten til regelen |
| Viktigheitskategori | Kva prioritet regelen har i forhold til andre reglar. Dess høgre kategori dess viktigare er regelen |
| Tekststreng (Frase) | Agentens melding til brukarane når den aktuelle regelen er broten. |

**Døme på ein regel:**

Brukar innput: Primærvariabel ”Sist innlogging”, Verditype:”Absoluttverdi”, Typevariabel: ”Større”, Verdivariabel:”5”, Viktigheitskategori: ”kat1”, Tekststreng. ”Det er lenge sidan du sist var innlogga, sjekk om det er kome postar du kan svare på sidan sist du logga inn”.

Agentoppførsel: Når ein brukar loggar inn etter 5 dagars fråvær, vil han få beskjeden ”

Målet med reglane er at dei skal vere redigerbare og tilgjengeleg for ulike program utan at det må endras på noko i strukturen. Eit eige klasseobjekt ville ikkje kunne tilfredsstille desse behova. Gjennom å legge reglane i ein databaseløysning, er det mulig å løyse begge dei behova eg har sett opp for reglane.

Teknisk ligg reglane som poster i ein regeltabell. Ved hjelp av denne tabellen er det enkelt å lage nye reglar eller redigere eksisterande reglar hurtig, effektivt og sikkert, fordi det er lettare å endre verdiar i ein database enn i kodefiler. Strukturen i databasen er vist i Tabell 5

Tabell 5 Databasefelt til Regel og Variabel. Oppsettet korleis reglane er lagra i databasen.

| | |
|---|---|
| Regel | Id, varnavn, verdi1tr, verdi2_tr, Verdi3_ntr, verdi4_ntr, frase, kat, gjabo, msli1, msli2 |
| Variabel (primærvariabel) | Id, navn |

### 6.1.2 Viktigheitskategorier

Gjennom konstruksjonen av reglane og gjennomgang av tidlegare arbeid, måtte eg lage ei prioriteringsrekkje for å kunne la agenten prioritera viktige reglar, men samtidig ignorere uviktige meldingar som har blitt vist før, for å ikkje få ein "information overflow"[5] hjå brukarane.

Etter ein del litteraturstudie, tankeverksemd, prøving og feiling, fann eg at 4 kategoriar var nok til å prioritere effektivt, samtidig som det ikkje blei for vanskeleg å velje kategori for dei brukarane som skal lage reglane. Dette litteratursøket omfattar blant anna kognitiv psykologi.

I lista under er dei ulike kategoriane beskrivne med døme på meldingar som agenten meiner vil passe innanfor kvar av kategoriane.

- Kategori 0: Veiledning. Kategorien skal innehalde reglar som er lett veiledning (generell). Hint og tips til korleis samabeid kan utførast best mogleg. Eller referanser til eksterne kjelder for informasjon. Eksempel på agentmelding "Til gruppearbeidet kan ein bruke Atekst eller Store Norske for å finne relevant informasjon".

[5] Information overflow er definert som å få inn meir informasjon enn kva ein greier å prosessere/behandle

- Kategori 1: Problem oppdagelse. Reglane som blir lagt til denne kategorien, kan vere av to typer: i)Hovudsakleg retta mot enkeltproblem, eller ii)Viktig generell informasjon. Eksempel på agentmelding for type i : "Du har vore lite pålogga i det siste, og det har komme mange nye innlegg som du ikkje har lest eller teke del i".

- Kategori 2: Spesifikt problem. Reglane i denne kategorien skal vere retta mot svært spesifikke problem, som kan utgjere trugsmål mot samarbeidet eller den totale progresjonen i gruppa. Eksempel på agentmelding: Du har posta mange fleire innlegg enn dei andre på gruppa, du bør kanskje trekke dei andre i gruppa meir med i diskusjonen".

- Kategori 3: Kritisk problem. Reglar som tilhøyrer denne kategorien, skal vere reservert for meldingar som skal medverke til å aktivere samarbeidet i ei gruppa der dette har brote saman ellerer i ferd med å bryte saman. Ein slik situasjon kan oppstå som dersom nokon i gruppa er "freeriders". Eksempel på agentmelding: Du bør svare direkte på andre sitt innlegg eller poste spørsmål som du sjølv lurer på".

I tillegg til kategoriseringsfaktoren var grunngjevinga for desse fire kategoriane at reglane kan ha forskjellige bruksområde ut frå den prosessen agenten skal støtte opp under. Eg har også delvis lenka desse kategoriane til dei forskjellige pedagogiske agenttypane som mellom anna Jondahl (2001) sette opp. Veiledniningskatagorien (Kategori 0) kan sjåast på som ein verktyagent. Dei tre andre kategoriane kan både vere til samarbeidsagent og kunnskapsbyggingsagent. Døma eg har satt opp, viser kategoriane som samarbeids-agentsdøme. Ein annan grunn til at denne katalogiseringa er tatt opp i samband med RuleEditor, er at instruktørane kan stille inn agenten med omsyn til kva katagoriar den skal oversjå, og kva den skal bli utløyst på.

## 6.2 Utvikling av FLEA og SA2 ($SA^2$)

FLEA og SA2 er to sider av same sak, der SA2 er grunnstrukturen i agenten, den inneheld reglane og algoritmane. FLEA er det visuelle grensesnittet brukarane vil oppleve når ein regel er blitt utløyst. FLEA er også bindeleddet mellom SA2 og RuleEditor, fordi det er mogeleg å starte RuleEditor frå FLEA.

Då dei to aspekta er ulike. Med om syn til teori og bakgrunn, beskriv eg FLEA og SA2 kvar for seg. Sidan begge elementa bruker same kode, vil kodeeksempla til FLEA vere beskrivne i SA2 interface, medan det visuelle til SA2 er beskrive i FLEA.

Når det gjeld skjermbilete av programmet, er alle bilete med referanse til versjon 0.x (v0.x) anten det er reine visuelle prototypar skapt i eit prototypingsverkty, eller delfungerande versjonar. Versjonar med versjonsnummer v1.x er fullimplementerte versjonar. RuleEditor og SA2/FLEA vart ikkje utvikla parallelt. Dei ulike versjonsnummera vil ikkje ha noko felles eller ha noko med kvarandre å gjere bortsett frå det å vise relativt kor mange gongar applikasjonen har gått gjennom iterasjonsprosessen.

### 6.2.1 Idé

Dolonen (2002) utvikla ein Studentassistentagent (Student Assistent agent) med definerte reglar. Også kjeldekoden til Dolonen (ibid) sin agent har vore tilgjengeleg for mitt arbeid.

Kjeldekoden har blitt vidareutvikla, slik at agenten har blitt utstyrt med editerbare reglar og har fått eit forbetra visuelt grensesnitt for kommunikasjon til brukarane. Ein del klassar er gjenbrukte anten direkte eller som namnegjenbruk. Nokre av klassene har fått endra delar av funksjonaliteten eller utvida funksjonalitet. Også kommunikasjonsstrukturen mellom objekta er delvis endra. Eg har arbeidd med evolusjonær prototyping (Figur 2) som overordna programmeringsstrategi. Utviklingsmålet med SA2 agenten har vore å legge til rette for ny funksjonalitet innebygd i rammeverket til den gamle SA-agenten.

### 6.2.2 Design av eit Brukar - Agent grensesnitt (FLEA)

Agenten har i hovudsak fått ei einvegs kommunikasjonsform mellom brukar og agent. Dette er fordi eg ikkje har sett det hansiktsmessig å lagge ein agent som kan motta forspursler slik Jondahl (2001) hadde i sitt forsøk.

#### *6.2.2.1 Agent grensesnitt*

SA2-agenten sitt grensesnitt skil seg frå Dolonen (2002) og Jondahl (2001) sitt visuelle grensesnitt, men er likevel basert på det eksisterande kodematerialet og datamaterialet. Det som skil mitt arbeide frå dei tidlegare arbeida er brukarkontakten. Korleis SA2 skulle ta kontakt med brukarane, har vore ein typisk evolusjon. Første papir-utkastet var ein popup-boks med tekst svært likt den førre agenten (SA) sitt grensesnitt. Etter kvart som ideane utvikla seg, vart den "animerte loppa" teken i bruk. Då denne loppa også var brukt i FLE3, ville det kunne skape ein attkjenning av objektet og dermed gje ein intuitiv innsikt i kva funksjon FLEA hadde som rettleiar og assistent..

Den første prototypen (Figur 22) av FLEA-grensesnittet som vart utvikla, viser korleis FLEA skulle komme opp når ein av reglane er brotne. Ei grunngjeving for å konstruere dette slik, var at agenten skulle fange merksemda til brukarane som då vil bli leia til å føre musepeikaren over den animerte agenten. Denne handlinga vil føre til at det kjem fram ein tekstboks under agenten med tekst som er relatert til det som utløyste agentreaksjonen.


Figur 22 Plassering av Agent popup (v.0.1) ut frå at den ikkje skal forstyrre brukarane altfor mykje.

Ved at den animerte figuren blei lagt inn i boksen, og ikkje over boksen, slik den først var tenkt, fall ideen om brukaraktivisering bort, trass i at det var ein svært interessant problemstilling. Tekstboksen fekk ein url-link med teksten:"Vil du vite mer" designa ut frå korleis interaktive menyar er. Denne linken skulle føre brukarane til ei anna intern FLE side, eller til ei ekstren side, alt etter kva som låg inne i den aktuelle regelen. Første

skjermbilde i RuleEditor kapittel 7, Figur 31 viser to tekstfelt: i) Kort tekst og ii) lang tekst. Tanken var at dersom brukaren ikkje forstod, eller han/ho ville ha meir kunnskap om problemet, kunne agenten referere til eksterne sider som A-tekst for informasjon. Den kunne og vise til fyldigare informasjon om korleis brukarane kunne fordele arbeidet mellom seg. Den interne visninga av lang tekst var tiltenkt at det skulle ha ein linkande visning slik som Chen og Wasson (2002) sin IA-agent presenterer sitt materiale(Figur 18 s46).

Ved hjelp av agenten skulle FLE få fram hjelpesider til brukarane. Dette hadde også ein hensikt ut frå at agenten sitt tekstvindauge har ein avgrensa storleik. Dermed blei talet på ord agenten kunne komme med i ei melding også sterkt avgrensa.

I den første versjon av grensesnittet, som vart fullstendig implementert (Figure 23), blei også høve til å redigere lagt inn gjennom linken "rediger". Denne linken startar RuleEditor, slik at den regelen som er utløyst, kan bli redigert. Førebels har alle brukarane høve til å redigere. Ved bruk av administrasjon av brukarrettane skal det vere mogeleg å tillate redigering for einskilde brukarar (administrator/instruktør/lærar), medan andre ikkje får tilgang til dette. Linken skal ikkje vere synleg for dei som ikkje har tilgang. Slik det er FLE har implementert tilgangskontroll av dei forskjellige funksjonane systemet har.


Figure 23 FLEA V 1.0 første fulle implementasjon av FLEA

Agenten vil forsvinne etter eit førehandsbestemt intervall, td. 5sek/ 10 sek / 15sek. / 25 sek. Dette vert bestemt av kva kategori som er blitt "trigga". Kategori 0 kan stå der i 5 sek mens kategori 3 blir ståande i 25 sek. Desse verdiane er sett opp som eksempel og vil vere justerbare i ein fullversjon av agenten. Dette fordi verdiar som ikkje er direkte begrunna ut frå pedagogiske eller MMI teoriar ikkje skal vere fastlagde i koden. RuleEditor applikasjonen tar seg av denne innstillinga.

Agenten skal registrere dersom den ofte viser same melding til same brukar. Agenten skal då vise denne meldinga. Her vert det differensiert mellom dei ulike viktigheitskategorieane. AgentFrame i Ruleeditor kan også skru av agenten sitt reaksjonsmønster på kategorinivå. Agenten får då ein tilnærma intelligent oppførsel. Ein vil då få ein agent som er tilpassa den einskilde brukaren i forsøket/undervsninga. Det er likevel ikkje mogleg å skru av kategori 3, som tilseier at meldinga er så viktige at den ikkje skal ignorerast, Dette fordi eg har definert Kategori 3 som ein regel som ikkje skal eller kan bli ignorert.

#### *6.2.2.2 Plassering AgentInterface*

Plasseringa av agenten er grunngjeve ut frå teoriar innanfor kognitive psykologi om merksemd, fokus og persepsjon. Fokus vil oftast liggje i midten av skjermen, og alt som er plassert lengre ut, må vere større enn det i midten for å bli observert like lett. Ei plassering nedst i høgre hjørnet vil difor verke meir diskret enn om agenten vart lokalisert til i fokusområdet.

Sjølve agentbilde og teksten glir ned frå øvre biletkant. Kor raskt agenten skal falle nedover skjermen kan regulerast. Eg har i utgangspunktet valt rask rørsle slik at agenten ikkje dominerer skjermen, men likevel tilstrekkeleg til at brukaren vert merksam på at det skjer noko og fokuserer agenten ved høve. For å fange – og halde på brukaren si merksemd, er agenten animert. Graden av animeringa vert regulert utifrå kva regel som er broten, og kva viktigheitgrad den har. For kategori 0 er det ein minimal animering, liten "vifting med armen" for å søke kontakt med brukar. Intensjonen er ein diskret kontakt: "Unnskyld, eg

vil gjerne seie noko". For dei neste kategoriane blir animeringa meir påtrengjande. Ved Kategori 3 "ropar" den animerte figuren (Figur 24) .

Figur 24 Stillbilde av FLEA i dei tre av dei fem versjonane original, beskjeden og overaktiv

Animeringa på agenten aukar drastisk gjennom at figuren flyttar seg opp og ned samtidig som den rører på armar og bein, eit skifte mellom grøn og raud farge på kroppen til FLEA er også lagt til. Gjennom all denne aktiviteten skal agenten gi brukaren ein peikepinn på kor viktig det er at han no les agenten si melding.

### 6.2.3 Utvikling av Student Assistent Agent 2 (SA$^2$)

SA2 agenten bygger på SA agenten til Dolonen.

#### *6.2.3.1 Beskrivelse av tekniske løysingar og val av design*

SA2 blei designa ut frå det å vere best mogleg editerbar både i oppførsel, reglar og vidareutvikling. Den indre programmeringsstrukturen er objektorientert, og agentens regelklassar har blitt bygd opp slik at alle delagentane har eit identisk rammeverk, og deretter blir kvar delagents spesialfunksjon lagt til.

Agenten har to kontrollobjekt med kvar sine funksjonar Coordiantor Agent (forkorta til CA) og Rule Coordinator Agent (forkorta til RCA).

Coordinator Agent koordinerer brukarinformasjon og trigging av FLEA mot brukarane. Rule Coordinator Agent styrer kontrollen over reglane og er den som identifiserer

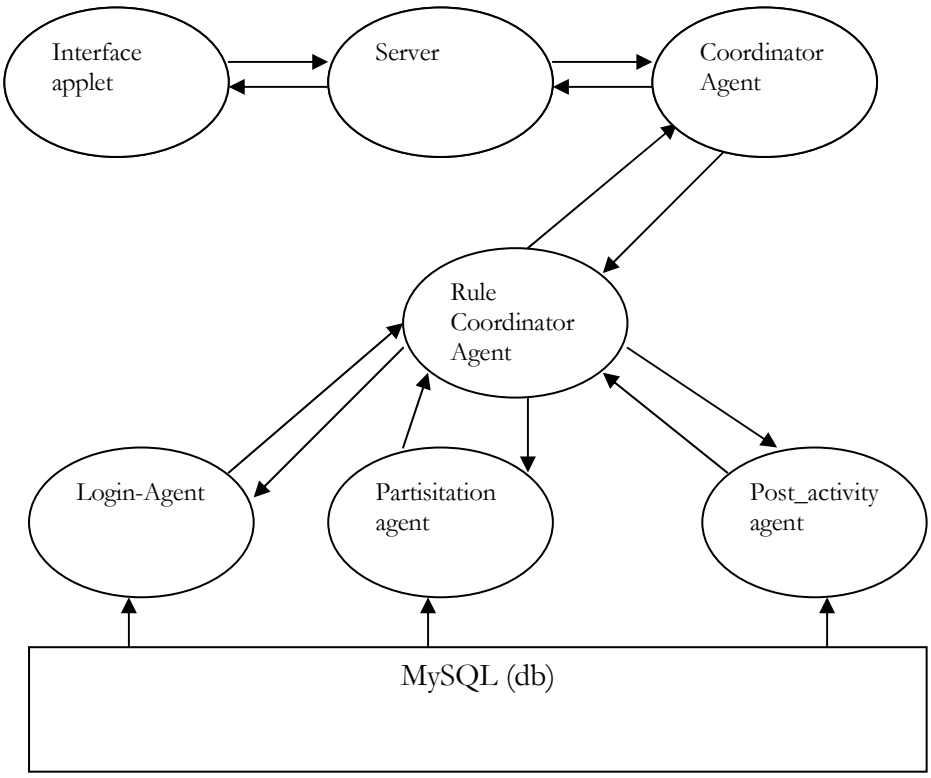
regelbrot. Ved regelbrot vil RCA gi melding til CA om at bruker ”NN” har brote regel nr ”n”. Sjølve utlysningsfunksjonen eller ”triggar”, som eg vil kalle den, har fire forskjellige nivå. Desse nivåa er definerte utifrå reglane sine fire viktigheitskategoriar.

RCA går regelmessig igjennom alle reglane for kvar einskild brukar for å registrere regelbrot. Ved brot blir opplysningane lagt i databasen, og søket går vidare. Når alle reglane er sjekka, vert neste fase initiert. Oppgåva no er sortering og trigging av eventuelle reglar som er lagt til i triggerdatabasen. Sorteringa av utløyste reglar blir prioritet etter viktigheitskategorien regelen ligg i, slik at Kategori 3 går før Kategori 0. Reglane blir utløyste med eit tidsintervall på 5 min. I den tida RCA triggar reglar, vil den ikkje sjekke innlogga brukarar for nye brot på reglane så lenge det finnes reglar som er registrert brotne reglar som ikkje er melde til brukar. Nyleg innlogga brukarar vil bli sjekka strakst dei har logga inn i FLE3.

RCA har også ein algoritme som sorterer reglane etter to kriterie, soft-tools og hard-tools. Desse kriteria er også relatert til viktigheitskategoriane. Agenten lagar ein database over kvar brukar og kva reglar denne brukaren har fått framvist. Dei to lågaste nivåa kan bli ignorerte av agenten dersom dei aktuelle meldingane vist fleire gonger. Grenseverdien er sett til 10 og ligg inne i systemet. Etter at triggerdatabasen er tømd, vil RCA gå i inaktiv modus (sovemodus) til neste aktiverings kommando frå CA kjem, eller at ein ny bruker har logga seg på. Då vil sjekking av reglane bli aktivert igjen.

#### *6.2.3.2 Indre kommunikasjonsstruktur i SA2 agenten*

Strukturen i SA2 Agenten er hierarkisk bygd opp med to sentrale element: i)appleten integrert i FLE3, som den visuelle delen av agenten og ii)Coordinator agenten som den styrande delen av agenten. Agenten blir kjørt som ein tråd (JavaThread) med gjennomkjøring etter eit fastlagt tidsintervall.




Figur 25 Indre klassestruktur av SA2 agenten. Dataflyt linjene vist i strukturen

Kvar av klassane i figuren vil bli nærare omtala med omsyn kva funksjonar den inneheld, og korleis dei fungerer i samanheng med resten av SA2 agenten. Eg vil her også drøfte likskap og skilnad mellom eige arbeide med SA2 agenten og Dolonen sitt arbeide med SA agenten.

*Interface – applet og FLEA*

FLEA er eit lite javascript som Interface objektet i SA2 kallar når ein regel er blitt broten. Kallet går gjennom ein javaapplet integrert inn i FLE3. Dette vert gjort ved eit appletkall frå html koden i FLE3. Sjølve appleten og oppsettet frå kallet er gjenbruk frå tidlegare arbeid. Skilnaden mellom SA2 og SA ligg i det at pop-up og det permanente vindauga er fjerna frå koden.

*Server*

Bortsett frå mindre modifikasjonar er servaren i agenten identisk med Dolonen sitt serverobjekt i SA agenten

*Coordiantor Agent*

Denne delen av agenten styrer kommunikasjonen frå server og til RuleCoordinator. Javatråden som aktiverer og deaktiverer agenten vert styrt frå dette objektet.

*RuleCoordinatorAgent*

RuleCoordinator Agent er eit nytt element som blei lagt til i SA2 agenten. Nokre av funksjonane har sitt grunnlag frå CoordiantorAgent i SA, men store delar av koden og strukturen er omarbeidd.

Gjennom oppbygginga av RuleCoordinator (RcA) blei det lagt vekt på at agenten skulle vere mogeleg å ekspandere slik at fleire delagentar og primærvaraiblar kunne leggast til utan å vere nøydd til å modifiserer strukturen i agenten. Dette var ein av grunnane til at RC-klassen blei laga. I SA-agenten låg denne administrasjonsfunksjonen internt i Coordinator Agent (CA).

Kall-linjene til delagenten er sett opp i linje 3 og 4 i figueren under. I linje 5 settes variabelen frå delagenten i ein boolsk sjekk. Dermed er ein ny delagent integrert i SA2 agenten.

```
01 else if (storleik ==0) {
02     boolean  start_sjekk= false;
03     svarLogin= L_agent.login(brukernavn);
04     svarDeltak = P_agent.login(brukernavn);
05     start_sjekk=(svarLogin=="trigget")|(svarDeltak ==   "trigget");
06
07       if   (start_sjekk == true) {
08          // hente data , behandle og slette frå trigga
09          // hent alle finn høgaste pri og utfør operasjonen
10          storleik = post_teller.dbCount3(tabell, db_tab, db_verdi);
11          int triggematrise[ ][ ]= new int [storleik][4];
12          trigget_regel[2]= -1;
```

```
13
14          try {  // Henter dei aktuelle verdiane frå trigga tabellen
```

Figur 26 kodedøme på korleis RuleCoordinator kallar delagentane og deretter sjekkar triggartabellen etter trigga reglar i det ein av delagentane har gitt positivt svar.

Eg har også utvikla eit skall for delagentar, slik at det skal vere enklast mogeleg å vedlikehalde, endre og lage nye delagentar for seinare studium/ forsøk. Dette skallet er lagt ved som vedlegg til oppgåva.

*Delagentar*

Delagentane er relaterte til reglane sine primærvaraiablar, der eg tilordnar ein primærvariabel til kvar delagent. I dette utviklingsarbeidet har eg berre tatt med to fungerande delagentar LoginAgent og Participation-postAgent, desse to er relatert til primærvariablane "Siste innlogging" og tal på usvarte postar. Kvar av delagentane blir i utgangspunktet sett opp med eigne tabellar i databasen. LoginAgent sin tabell er ein modifisert tabell av Dolonens logintabell.

Tabell 6 LoginAgent sin databasetabell, som held styr på antall innlogingar og dei to siste innloggingane til kvar brukar.

| Login | Name, ip, time, time2, antall |
|---|---|

Sjølve agenten er bygd opp rundt ein casestruktur der kvart av casane blir bestemt ut frå ein av variablane til regelen:  msli verdien (denne står for Mindre, Større, lik eller intervall). Figur 27 viser loginAgent sin case struktur der msli1 er satt til å vere "LIK".

```
01              case 2:   // msli1 = Lik
02                    brukernavnimat = brukarnavnmatrise[0][0];
03                    gammeltid =brukarnavnmatrise[0][1];
04                    nytid =brukarnavnmatrise[0][2];
05                    dag2=konv.gettimedag(gammeltid);
06                    dag1=konv.gettimedag(nytid);
07                    mnd2=konv.gettimemnd(gammeltid);
08                    mnd1=konv.gettimemnd(nytid);
```

```
09                      aar2=konv.gettimeaar(gammeltid);
10                      aar1=konv.gettimeaar(nytid);
11                      diff =regelcalc.check_login(dag1,dag2,
12                      aar1,aar2,mnd1,mnd2);
13                     if (tr_verdi1 == diff)  {
14                        svarmatrise[i][0]= regel_id;
15                        svarmatrise[i][1]= brukernavnimat;
16                        svarmatrise[i][2]= regelmatrise[i][4];
17                        svarmatrise[i][3]= "1";
18                        post_tell= post_tell +1;
19                        svar = "trigget"  ;
20                      }
21                      svar = "utrigget";
22                      break;
```

Figur 27 Kode til Case strukturen i LoginAgent. Case 2 msli = Lik.

Fordelane med denne strukturen er at koden innanfor kvar case er nesten identiske. Berre i linje 12 finn ein forskjell, og det går på om det er >,< eller = som vert sett inn. Ved intervall er det annleis, men skilnadene er ikkje så store. I dømet for LoginAgent er kontrollen av reglane enkel i den samanliknar verdien regelen har med den utrekna verdien. Er "if setninga" sann, vil svaret frå delagenten vere trigga og, dermed går informasjonen om brukar, og regel inn i ein eigen tabell: "Trigga"

Tabell 7 DB tabell Trigga, mellomlagring for alle utløyste reglar før dei blir vist til brukarane

| Trigga | Id, regelnr,user, nivaa,utloyst, funne |
|---|---|

.

Denne prosedyren gjer kvar delagent ved gjennomkjøring. Informasjonen i tabellen blir då behandla av RCA når den har fått melding om at ein regel er utløyst for den aktuelle brukaren den har sendt sjekk kall for.

## 6.3 Evaluering av Agenten

Den evalueringa som er blitt gjennomført i dette utviklingsarbeidet er blitt begrensa til eksperttesting. I designprosessen av agenten har ein god del bankenforliggande teori blirr

brukt som rammeverk for designet. Og ut frå mine tankar om korleis og kva agenten skal gjere har eg antatt eit reaksjonmønster frå brukarane.

### 6.3.1 Førehandstesting - Reaksjons mønster

Sidan eg ikkje har fått gjennomført noko større brukartesting av agenten som heilskap integrert i FLE3 slik som scenaria legg opp til, vil eg sette fram ein del hypotesar for evaluering av agenten og dens kunnskapsbyggingseffekt.

Eg antar at desse fire brukarreaksjonane på agenten vil vere aktuelle.

1) Observering av agenten → Lesing → Handling etter beskjed.

2) Observering av agenten → lesing → Ignorering.

3) Observering av agenten → ignorering.

4) Ignorering av agent.

Ut frå dei fire typane av reaksjonsmønster vil eg definere nummer 1 som det optimale og 4 som det minst optimale Eg antar ut frå tidlegare arbeider (Jondahl 2001 )at måten agenten presenterer seg og antall gonger den kjem opp for kvar brukar innanfor ein bestemt tidsperiode vil ofte besteme om det er 1, 2 eller 3 som blir det aktuelle responsmønster. Mellom type 1 og type 2 vil det ofte vere forståinga av beskjeden som vil gjere forskjellen. Reaksjonemønster 4 kan relaterast til feilplassering av agenten. Brukarane ser ikkje agenten når den dukkar opp og vil heller ikkje ta hensyn til beskjeden agenten bringer.

# 7 Utvikling av RuleEditor

## 7.1 Oversikt

I dette kapittelet blir andre delen av mitt utviklingsarbeid beskriven. I forhold til kva Figur 21 viser er dette kva som ligg til høgre for SA agenten, anmet på denne programvaren er RuleEditor.

### 7.1.1 Idè

Eit av problema med tidlegare utvikla agenter har vore at reglane for agenten sin oppførsel har lege fast i kjeldekoden, og dermed måtte ein ha programmeringskompetanse for å kunne editerer ein regel. Ei slik editering medfører at heile systemet må rekompilerast og startast på nytt for at endringane skulle bli registert og brukt av agenten. (Dolonen 2002). Eit av funna til Jondahl (2001) var at instruktørane som simulerte agentane fekk eit behov for å redigere eller lage nye svarmeldingar/responsmønster i ulike situasjonar under forsøket. Løysninga eg såg på dette problemet var å legge reglane i ein database i staden for at dei er lagt inn i koden. I og med at dette var noko som ikkje direkte var forsøkt før, og det var mange usikre aspekt rundt korleis designet og funksjonaliteten til RuleEditor skulle vere, blei veldig mykje av programmeringa gjort som bruk og kast prototyping.

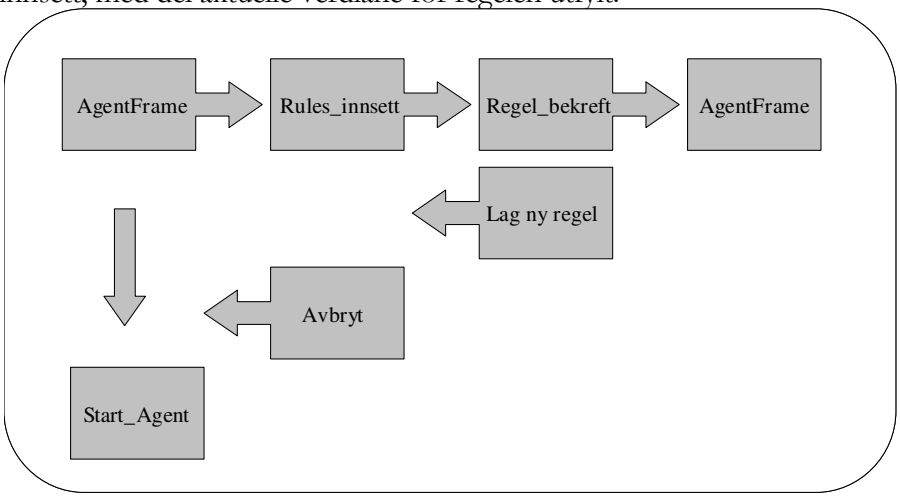
### 7.1.2 Struktur

RuleEditor er bygd opp som ein enkel Java applikasjon, uavhengig av resten av SA2 agenten. Ved å ha dette programmet som totalt uavhengig av SA2 agenten, vil ein unngå at feil i den eine applikasjonen påfører feil i den andre. Slike samanhengar fører til at agentserveren låser seg under oppdatering av reglane. Informasjonssendinga av reglar mellom RuleEditor og SA2 difor gjennom ein MySQL database etter følgjande struktur:

RuleEditor → SQL-database → SA2

*Sidenavigasjon*

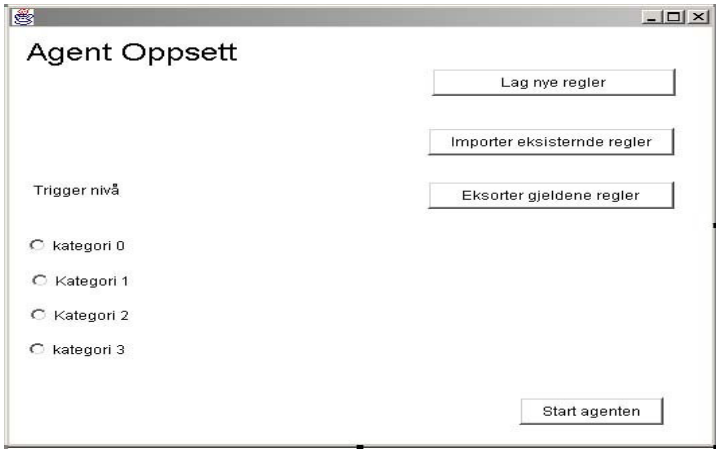
Designet for utfyllinga av nye reglar eller redigering av eksisterande er lagt opp etter wizard utfyllingsteknikken. Figur 28 viser korleis brukarane navigerer for å lage ein ny regel. Teknikken er valt fordi det skal vere enklast mogeleg å lage nye reglar eller editere eksisterande. Ved val av "Editere regel" når FLEA viser regelen, vil ein hoppe direkte inn i Rules_innsett, med dei aktuelle verdiane for regelen utfylt.




Figur 28 navigasjon i regeleditoren

## 7.2 Visuell del

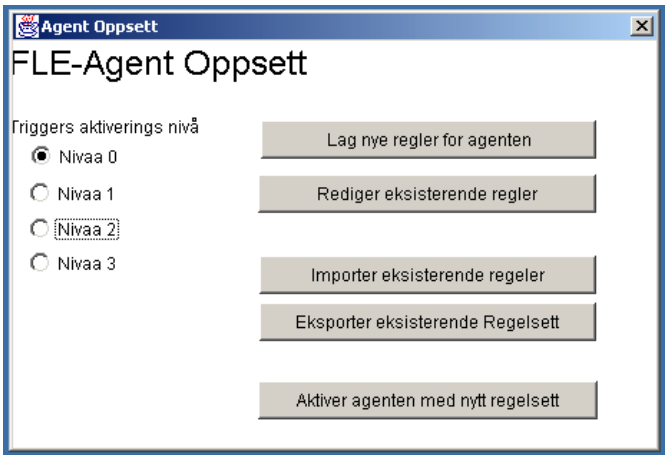
Editoren har eit hovudgrensesnitt som vert kalla Agent Oppsett. Dette hovudvindauget blei utforma for å ha tre funksjonar i) Lage reglane til agenten ii)Justere Triggernivået til agenten iii) Starte/restarte agenten. Figur 29 under viser det første utkastet til agentoppsettet.




Figur 29 RuleEditor hovudskjermbiletet v0.1 mockup

Designet skulle vere minimalistisk og enkelt, slik at brukarane lett fekk oversikten og kontrollen over korleis ein brukte dette verktøyet. For dette hovudgrensesnittet vart det vurdert som naudsynt å ha tilgang til funksjonane import- og eksport av regelsett. "Start agenten" - knappen blei lagt til som ein konsekvens av mi eiga testing av Dolonen sin agent. Dolonen sin agent måtte startes gjennom ein java kommando..

I versjon 1.0 (Figur 30) blei det gjort ein del endringar i tekstfrase, på dei forskjellige knappane og i storleiken på applikasjonsvindauga. Det var også lagt til eit alternativ "Rediger eksisterande regel". Denne funksjonen blei etterkvart sett på som like viktig som det å lage nye reglar. Sidan agenten skulle ha editerbare reglar, var det ein designfeil å ikkje ha med dette i førsteutkastet.



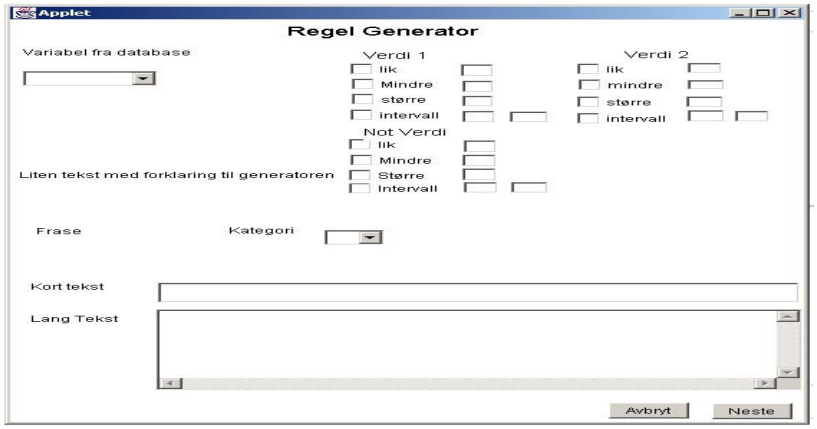

Figur 30 RuleEditor hovud skjermbilde V 1.0

Den nivåregulerte triggersettinga til agenten kan setjast til det ønska nivået alt etter kva behov ein har i den aktuelle brukarsettinga.

Skjemaet for å lage regler, eller editere eksisterande reglar, er strukturert etter eit wizard oppsett. I dette oppsette fyller ein ut delskjema for kvart nytt vindauge. Til slutt stadfester ein det endelege resultatet.

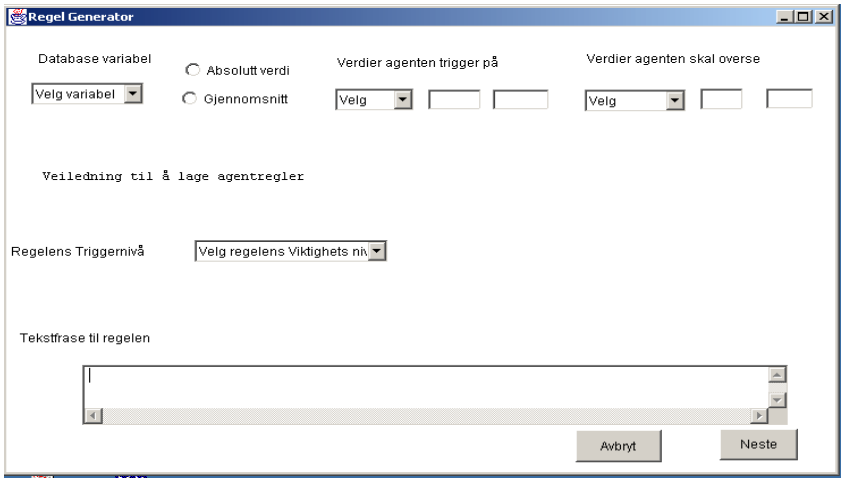
Input-boksen for regelgeneratoren har gjennomgått dei mest omfattande endringane frå mock up (versjon .0.1) til endeleg produkt. (Figur 31). Ein regel blir konstruert av eit sett med ei rekkje mulige verdiar og ein variabel. Førsteutkastet til regelkonstruktøren blei designa til å skulle ta i mot alle verdiane som kunne vere mogeleg uansett. Resultatet av dette valet var at dette grensesnittet (Figur 31) blei svært uoversiktleg og genrelt forvirrande.

Ved hjelp av blant anna eksperttest gjennomgjekk ein eit logisk oppsett av korleis reglar kunne bli konstruert, vart designet vidareutvikla. Resultatet av denne gjennomgangen var at talet på tekstboksar var redusert frå 27 til 4, og designversjon nr 2 (Figur 32) vart lagt fram.




Figur 31 RuleEditor regelgenerator skjermbilde v.0.1 Mockup.
Første degin av regeleditorens regelgenerator.

Vidare testing viste at ein aldri ville bruke meir enn ein boks for verdiar så lenge ein brukte *større*, *mindre* eller *lik* variablane. Det var heller ikkje nødvendig med to verditypar for ein regel.. I nokre tilfelle ville nok to separate verdiar vere til god nytte for brukaren, med for å forenkle systemet sin struktur, blei denne funksjonen ikkje teken med, og ein sat igjen med ein verdiboks og ein "not-verdiboks".

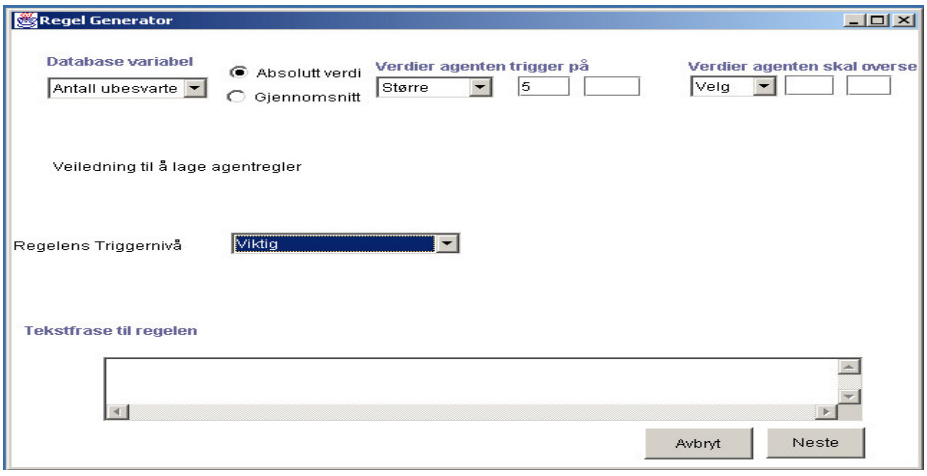



Figur 32 RuleEditor regelgenerator V1.0 Første fullstendige
versjon av regelgenrator layouten Med åpningsverdiar

Å ta ut eitt sett med variablar vil nok resultere i at visse reglar ikkje kan bli konstruerte, men truleg vil behovet for slike komplekse regelkonstruksjonar vere relativt lågfrekvent. Sett opp mot eit brukarvenleg grensesnitt, er difor høvet til å foreta visse redigeringar avgrensa.

På den neste versjonen som er vist i Figur 33 begynte det å vere meir kosmetiske og lesbarheits aspekt som blei endringane. Samt at strukturen om å starte øvst frå venstre og fylle ut/velje verdiar mot høgre blei meir forsterka.

Eit tanke som ikkje blei fullført, er rettleiingslinken "Veildening til å lage regler" som er på både i Figur 32 og Figur 33. Denne skulle gje ein liten pop-up boks med eksempel på korleis ein regel skulle kunne konstruerast og også ein liten definisjon av kvar av dei forskjellige triggerkategoriane komme opp, slik at brukarane fekk ein enkel intruduksjon til bruk av RuleEditor.




Figur 33 RuleEditor regelgenerator V1.1 Med delvis utfylt regel.

### 7.2.1 Feilhandsaming ved brukar-input

Dersom viktige element ikkje er fylt ut i regelgenerator-skjemaet, vil det komme ei tilbakemelding som viser kva post som ikkje er utfylt, denne tilbakemeldinga vil komme i ein standard popup-boks. Viss fleire poster manglar, vil ein berre få melding om den som

først blir identifisert. Ein mogleg funksjon som kan leggast til, er at det også vert vist kva poster som er feile eller mangelfulle, og at det vert gjeve rettleiande tips om korleis ein kan rette dette opp.

Oppsettet for feilhåndtering er heilt naudsynt då det er lett å gløyme eller oversjå enkelte boksar. Ved feilutfylling kan reglane feile i sjekkinga, noko som kan føre til systemsamanbrot for agentservaren. Sjekkinga av utfylte boksar vil difor også fungere som ei stabilitetssikring.

Etter at ein brakar har laga ein regel, får brukaren tilbakemelding om at "regel er lagra" og får høve til å ta eitt av tre val. Denne tilbakemeldinga til brukarane er viktig ut frå at systemet skal kunne gje tilbakemelding til brukarane om kva som skjer eller kva som har skjedd (Nielsen 2003WEB)

Når ein regel er godkjent, får brukarane opp Regelbrekreft-vindauge. Mockupversjoen og det endelege designet er mykje like, men knappane blei flytta ned for å vere meir i samsvar med wizardteknikken. Noko av funksjonaliteten er også endra. I mockupversjonen var eksporter-knappen tiltenkt å eksportere alle reglane som til då var blitt laga.


Figur 34 RuleEditor Regelbekreft v.0.1 Mockup


Figur 35 RuleEditor Regelbekreft v1.0

Det overordna designet til den siste versjonen er også i samsvar med resten av framene (rammene) som RuleEditor består av.

### 7.2.2 Regeleditoren sin programstruktur

Heile RuleEditor består av klassar som eg har delt opp i to hovuddelar. i)Visuelt element og ii)databehandlings element. Denne oppdelinga og namna på kvar av klassane som er presentert i Tabell 8.

Tabell 8 oversikt visuelle- og databehandlings-element

| Visuelle | Databehandling |
|---|---|
| AgenFrame | Admin_agent |
| Rule_innsett | Rule_editor |
| Rule_edit | Unfilled_stopp |
| Regel_bekreft | |
| Ufill-popup | |

Admin_agent er sjølve hovud modulen i RuleRditor. Elementet Rule_editor er konstruktøren og behandlinga av reglar som blir produsert.

# 8 Diskusjon

Ut frå problemstillinga til dette arbeidet satt eg opp to utviklingsmål; i) Agentfunksjonalitet i form av ein redigerbar agent og ii) Kommunikasjonsform i form at ein animert agent. Eg vil først diskutere aspekta rundt agentfunksjonaliteten i delkapittel 8.1, mens eg tar opp spørsmål rundt kommunikasjonsforma i delkapitel 8.2.

Diskusjonen vil basere seg på funn frå ekspert testinga samt eigne idear, då agenten ikkje har vore testa på noko brukargrupper i ei verkeleg setting.

## 8.1 Redigerbare agentar

Eg vil her drøfte kva nytte ein redigerbar agent kan ha for kunnskapsbyggingsprosessen i FLE3 , slik eg har implementert moglegeheiten for redigering av reglane til agenten.

Reglane til pedagogiske agentar har vore basert på eit statisk oppsett, anten direkte i kjeldekoden eller gjennom eigne databasar. Desse statiske regelsetta har den ulempen at viss det er ein regel som ikkje fungerer, må teknisk personell inn for å få den aktuelle regelen endra. Gjennom den løysninga eg har implementert, med eit enkelt webgrensesnitt over databasen, blir moglegheiten til å redigere reglar open for alle. Her er det opnar seg det nye moglegheiter med denne agenten. Alle kan ha mogelegheiten til å endre eller redigere agentens reglar. Også design av reglane kan bli testa ut ved å la elevane sjølv ha moglegheiten til å redigere reglane når agenten kjem med eit regelbrot.

### 8.1.1 Oppgåvespesifikt språk (TSL) vs Sluttbrukar programmering(EUP)

I designfasen av RuleEditor møtte eg ein del problemstillingar som eg vil drøfte ut frå dei val eg har gjort i dette arbeidet. Sentralt i dette avsnittet ligg argumentasjonen kvifor TSL er valt føre ei EUP tilnærming, ut frå ideane til Bush (1945) at eit teknologisk framskritt må vere meir brukarvenleg enn kva det forrige var for at det skal bli tatt i bruk.

Eg har prøvd å identifisere kva brukartypar som er tiltenkt rolla som brukarar av RuleEditor. Resultatet eg kom fram til var lærarar, vitskapleg ansatte og liknande. Dermed kan den aktuelle brukargruppa si erfaring innan programmeringsspråk spenne frå ingen erfaring til full forståing. Den store forskjellen i erfaringa til brukarane gjorde at eg begynte å helle mot eit TSL system framfor EUP. Eg måtte samanlikne fordelar og ulemper mellom dei to forskjellige bruksstrategiane for å sjå kva som var best eigna til denne typen bruk. EUP har fordelen med at det er eit kraftig verktøy, og muligheitane er alltid mange. Ein EUP editor kunne brukt Java Script til å konstruere reglane eller faktisk kunne konstruert eigne delagentar ut frå enkel kode. Ulempen er den store begrensninga eit spesifikt programmeringsspråk drar med seg, kor berre brukarar som kan det aktuelle programmeringsspråket er i stand til å konstruere eller redigere reglane. Resultatet av eit slikt program er at teknisk personell må inn kvar gong ein regel skal endrast, lagast eller slettast. Dermed ville utviklinga ikkje komme noko lengre enn kva den forrige agenten var. Fordelane ved å bruke ei slik løysning vil vere at måten reglar blir konstruert på ikkje er fastlagt, og kombinasjonane av moglege reglar vil vere neste ubegrensa.

TSL–løysninga som eg valte er heller ikkje optimal, for det første avgrensar den fleksibiliteten i konstruksjonen av reglar, slik at reglar kan ikkje bli designa i anna form enn kva oppsettet i RuleEditor tillet. Argumentasjonen min for å bruke TSL løysninga, sjølv om den er ei svakare og meir begrensa regelkonstruktør, er brukbarheiten til systemet. Den faktoren eg har lagt mest til grunn for designet er at brukarterskelen for bruk av RuleEditor skal vere så lav så mogeleg. Med TSL fokuset på objekt og tal verdiar vil det vere enklare for brukarane å forstå korleis ting skal konstruerast i ein TSL editor enn i ein EUP editor.

Det andre aspektet som var med i bilete når vala over korleis implementasjonen av RuleEdtor skulle bli avgjort var Mørch (1995) sin modell over tailoring gap. Mitt ønskje om at gapet mellom brukarane og systemet skal vere så lite så mogleg. Dermed beli det naturleg å velje ein tailoring strategi som ligg tett inntil korleis deisgnprosessen skjer og ikkje korleis programeringa frungere..

## 8.2 Animerte agentar vs. statiske agentar

Vesentlege problemstillingar knytt til det å designe ein pedagogisk agent var kva form kommunikasjonen mellom brukar og agent skulle få og korleis denne skulle fungere. I tidlegare arbeid starta ein gjerne med pop-up boksar, men gjekk etterkvert over til permanente vindauge ut frå omsyn til brukarane. Kritikken mot animerte agentar har vore stor. Blant andre kritiserer Scheniderman animerte agentar og hevdar at dei skapar falske forventingar til agenten som personifisert objekt (1992). Andre meiner derimot at denne effekten vil auke læringseffekten (Johnson 1998, Constantino-Gonzàles &Suthers 2001). Ved å bruke eit insekt (”bug”). Og ikkje eit menneske, som Johnson brukar i Adele og STEVE, som eit utgangspunkt for animeringa. Vil eg hevde at eit feltstudium kan vise at animerte ”bugs” kan auke effekten frå animerte agentar utan samtidig å aktivere den negative personifiseringseffekten Scheniderman fryktar. Det er gjort studie på det å bruke animerte bugs i den visuelle delen av agenten, ”Herman the bug” (Lester & Stone 1997) er eit døme på dette. ”Herman the bug” er ein teikneserieliknande (Figur 13) som rettleiar og rådgjev elevane innan fagfeltet biologi. Dette teikneseriefigur aspektet er det ikkje mange som har tatt opp. I teoristudiet var det berre Lester & Stone (1997) som har testa eit slikt animert grensesnitt. Dei fleste andre utviklings- og evalueringsprosjekta har testa animerte agentar forma som menneske eller at dei har hatt mennekjelege trekk (Lester m. fl.. 1997) .

## 8.3 Generelle aspekt med SA2 agenten

SA2 og RuleEditor er begge utvikla i Java. Fordelane og argumentasjonen med dette valet er grunngjeve i kapittel 5. I lys av Chen og Wasson (2002) sin agent og ein del andre agentutviklingsprosjekt (Johnson 1998, Lester & Stone 1997) , vil det vere naturleg å vurdere om fordelane ved ein overgang til Python og ei direkte integrering i FLE/Zope vil vere meir føremålstenleg for vidare agentutvikling

Dei visuelle effektane ein Java agent kan oppnå, slik både eg har gjort i mitt arbeide og måten Dolonen (2002) gjorde det på i sitt, vil ikkje kunne støtte brukarane på same måte som ”Herman the bug” (Lester & Stone 1997) er i stand til. At agent og

gruppevaresystemet ikkje har ei direkte implementering, men ligg som eit lag utanpå kvarandre kan ha utlemper i forhold til dei prosessane agenten skal støtte. Dette gjelder spesielt der agenten har behov for å reagere i augneblinket.

Eg meiner at ein systemintegrert agent vil kunne rettleie og rådgjeve brukarane betre enn kva ein ekstren agent kan ved at den systemintegrerte agenten overvakar brukarane i alle "trinn". Omdahl sette opp behovet for agentstøtte i postings augneblinken som eit viktig punkt (2002). Slik støtte har ein ekstern agent problem med å utføre, mens ein intern agent vil kunne gjere det.

## 8.4 Moglege forbetringar

Mange tekniske løysingar, idear og føresetnader har endra seg etterkvart som SA2 agenten og RuleEditor er blitt utvikla. Som vanleg i programvareutvikling blir aldri produktet fullkome, sett frå utviklars synspunkt. Det vil alltid vere mogeleg å forbetre utviklingsprosjekt, problem ein synest ein ikkje har løyst optimalt, eller noko ein kunne utforma betre. Dette gjeld også mitt utviklingsarbeide. Når ein, som i min situasjon, har ei avgrensa tid til rådvelde, må ein sette sluttstrek for utviklinga anten ein er nøgd eller ikkje. Eg vil her setje fram nokre punkt og idear for RuleEditor, SA2 agenten og FLEA som kan utviklast vidare i kommande arbeid. Sorteringa er gjort med grunnlag i dei to utviklingsspørsmåla eg hadde og den tekniske implementasjonen av SA2 agenten og RuleEditoren

### 8.4.1 Agentfunksjonalitet

Det første forbetringspotensialet ligg i RuleEditor. Ei direkte integrering i FLE, kan redusere talet på program frå tre (FLE, SA2 agenten og RuleEditor) kan dette reduserast ned til eit (FLE). Ved å kombinere Chen & Wasson (2002) sin agent med RuleEditor, kan instruktøren/læraren ha ein full oversikt over problema studentane har, og dermed kan raskt lage reglar for å løyse desse problema. Ved å ha regeleditoren som ei "side" i FLE, vil

kanskje RuleEditor bli bere ved at den blir lettare tilgjengeleg. Å kombinere IA agenten til Chen og SA2 agenten kan ha ein formidabel fo instruktørane/lærarane.

Det vil vere trong for å konstruere ein delagent, som hentar opp reglar som blir brotne ofte, utan at det blir gjort noko med, og som tar kontakt med administrator /instruktør i FLE for å få redigert den aktuelle regelen. Denne funksjonen kan nok Chen og Wasson (2002) sin agent gjere for SA2 agenten.

Det siste punktet for vidareutviklinga av RuleEditoren går på konstruksjonen av reglar, og behovet for meir avanserte reglar enn kva ein primærvariabel kan greie. Dersom ein har høve til å kombinere fleire primærvariablar, kan ein skape meir avanserte og spesifiserte reglar til agenten. Ei forventa negativ side med dette er at Regeleditoren blir meir avansert og med det truleg vanskelegare å bruke.

### 8.4.2 Kommunikasjonsform

Implementasjonen eg har gjort av det visuelle grensesnittet, baserer seg på at kommunikasjon mellom brukar og agent er ein pop-up men i ei animert form. Om denne løysninga er god eller ikkje gjenstår det å teste, men eg ser også andre former for kommunikasjon mellom brukar og agent ei form er at agenten har moglegheit til å poste kommentarar direkte i FLE sjølv. Eit eksempel på slik posting av innlegg oppstår når eit problem står usvart lenge. Agenten postar då eit innlegg definert ut frå ein "usvart problem"- primærvariabel regel, med oppmodning om å prøve å svare på denne posten. Ei slik tilnærming vil eg tru aukar merksemda og lysten til å lese meldinga frå agenten ved at ein "aktiverer" informasjonsboksen sjølv.

Eg har ikkje teke opp til diskusjon kvar agenten skal komme opp på skjermen, eller om den skal ha varierande plassering ut frå viktigheitsnivået på regelbrotet. Ei god løysing på dette spørsmålet er nært knytt opp til brukarevaluering / utprøving av ulike plasseringsversjonar.

Det bør og vurderast om agenten skal ha emosjonelle uttrykk slik Adele (Johnson 1999) har det. Ut frå desse rapportane viser det seg at effekten av agentar med emosjonelle utrykk gjev ein betre læringseffekt enn animerte agentar utan slik visning av "kjensler".

### 8.4.3 Agent implementering

Måten agenten er blitt utvikla på, har gjort at den ikkje har ein perfekt objektorientert struktur. Bruk og kast prototyping har vore flitting brukt, saman med at det har vore ein sterk moduloppbygging av agenten. Kvar modul (2-3 klassar) er blitt utvikla for seg sjølv og så sett inn i den eksisterande agenten. Denne samansyinga har ikkje alltid vore heilt ideell for stabiliteten og den overordna strukturen av agenten.

# 9 Konklusjon og vegen vidare

I problemstillinga sette eg opp to utviklingsmål for oppgåva, og gjennom diskusjonen har eg prøvd å sette søkelyset på forskjellige aspekt ved desse måla. Diskusjonen har eg og drøfta ein del av dei problemstiligane som har komme opp i utviklingsprosessen. Dei slutningane eg kan skissere opp er føljande:

Eg kan ikkje konkludere med at agenten har ein effekt for studentane når dei brukar gruppevaresystem som lærings- og samarbeidsarena. Tidlegare studier har vist at animerte agentar har ein positiv effekt på både samarbeids- og læringsprosessen til brukarane. Dermed håper eg at den måten eg implementerer redigerbare reglar i programvara på, vil vise at arbeidshypotesen min, om at ein animert redigerbar agent kan gje elevane eit ekstra løft i kunnskapsbyggingsprosessen kan bli empirisk verifisert gjennom eit større feltstudie.

Grunnen til at eg trur den kan bli empirisk verifisert ligg i at utviklinga av ei nøytral animert kommunikasjonsform som reduserer agentpersonifisering i høve til ordinære animerte agentar på same tid som ein utnyttar sentrale fordelar som animerte agentar har. Samtidig som der er redigerbar for brukarane.

Som ein avslutning vil eg sette opp nokre punkt for moglege framtidige arbeid med pedagogiske agentar i gruppevaresystem. Tre hovudområde skil seg ut. Ut frå den prosessen arbeidet har vore, er det eit informatikkfokus på desse tre områda. I tillegg kjem dei to områda er avgrensa arbeidet mitt opp mot i kapittel 4. Sidan dei områda er oppdefinerte i avsnitt 4.1.1 gjentar eg dei ikkje her.

1) Lage ein integrert agent med regel redigering direkte i FLE3, programmert i Python. Med det kan ein oppnå omgåande handling frå agenten, i staden for at agenten må hente ut data for så først analysere og reagere. Der også IA agenten til Chen (Chen & Wasson 2002) kan integrerast til eit einheitleg verktøy for lærarane saman med ein ny versjon av SA2 agenten.

2) Nye former for agentkommunikasjon, emosjonelle uttrykk, og varierande plassering i grensesnittet ut frå viktigheit til den brotne regelen.

3) Utvikling av ein redigerbar agent for andre virtuelle læremiljø, slik som Classfronter.

# REFERANSELISTE

Baggetun, R., Dolonen, J. and Dragsnes, S. (2001). Designing Pedagogical Agents for Collaborative Telelearning Scenarios. In: Bjørnestad, S., Moe, R., Mørch, A., Opdahl, A. (eds.): Proc. of IRIS24. Ulvik, Norway

Baggetun, R. 2001 Coordination Worck in Collaborative Telelearning. Maters Thesis, Department of Information Science, University of Bergen.

Bannon, L., Bjørn-Andersen, N., & Due-Thomsen, B. (1988) Computer Support for Cooperative Work: An Appraisal and Critique. In Proceedings EURINFO 88 - Information Technology for Organizational Systems, pp 297-303 (Amsterdam: North-Holland).

Bannon, L. 1992 Perspectives on CSCW: From HCI and CMC to CSCW* In Proceedings International Conference on Human-Computer Interaction (EW-HCI'92), St. Petersburg, Russia, August 1992 , pp 148-158. (Also "published" in electronic form with interactive discussion between readers and author on BCS HICOM electronic conference system, January, 1993)

Bereiter, C. & Scardamalia, M. (1993). Surpassing ourselves. An inquiry into the nature and implications of expertise. Chicago, IL: Open Court.

Beyer, B. K. (1985). Critical Thinking: What Is It? Social Education 49 (4), pp. 270-76.

Boehm, B 1988 The spiral model of software development and enchancement IEEE Computer. 21(5)

Bush, V 1945 As We May Think. The Atlantic Monthly; July 1945. Volume 176, No. 1; 101-108.

Brendshøi, A. 2003 Kunnskapsbygging i digitale læringsomgivelser Cand Ed. Oppgåve, Profesjonstrudiet i Pedagogikk UV-fakultetet, Universitetet i Oslo.

Chen, W., Mørch, A. & Wasson, B. 2003 "Pedagogical agent design for distributed collaborative learning." Internation Conference on Computers in Education. (2003) icce2002.massey.ac.nz/tutorial_2.html"

Chen, W & Wasson, B 2002 An Instructional Assistant Agent for Distrubuted Collaborative Learning. I Cerri, Gouarderes & Paraguacu (eds) ITS 2002 LNCS 2363 pp609-618, Springer Verfalg Berlin Heidelberg

Chen, W., J. Dolonen, Wasson, B.2003. Integrating Software Agents with FLE3**.** Designing for Change in Networked Learning Enviroments. Proceedings of the International Conference on Computer Support for Collaborative Learning 2003, Bergen, Kluwer Academic Publishers.

Constantino-Gonzàles & Suthers 2001 Coaching Collaboration by Comparing Solutions and Tracking Participation. In P. Dillenbourg, A. Eurelings, K. Hakkarainen (Eds.) European Perspectives on Computer-Supported Collaborative Learning, Proceedings of the First European Conference on Computer-Supported Collaborative Learning, Universiteit Maastricht, Maastrict, the Netherlands, March 22-24 2001, pp. 173-180.

Cyber, A. 1991 Customizing Application Programs, in Proc First International HCI`91 Worckshop (Moscow, Aug 5-9, 1991), pp 152-157

Dahlback N., Jønsson, A., Ahrenberg, L. 1993, Wizard of Oz studies: why and how. Proceedings of the 1st international conference on Intelligent user interfaces table of

contents Orlando, Florida, USA, Pages: 193 - 200 Year of Publication: 1993 ISBN:0-89791-556-9

Dahl O. J.& Nygaard K. 1965 SIMULA : a language for programming and description of discrete event systems : introduction and user's manual. Norsk Regnesentral.

Dolonen, J. A. (2002) ”The Development of a Pedagogical Agent System for Computer Supported Collaborative Learning”. Department of Information Science, UIB.

Dourish, P & Belloti, V. 1992 Awareness and Coordination in shared worckspace. Proccedings of CSCW`92 ACM Press, pp 107-114

Ehn, P. 1988 Ehn, P. (1988). Work-oriented design of computer artifacts**.** Falköping, Sweden: Arbetslivscentrum.

Ehn, P. 1992 "Scandinavian Design - on skill and participation" in Adler, P. & Winofgrad, T.: Usability - Turning technologies into tools, Oxford University Press, 1992

Ellis C..A Gibbs S.J & Rain G.L (1991) Groupware some issues and experience. ACM Communication of the ACM Vol 34 nr 1 1991

Ericsson, M. 1999 Supporting the Use of Design Knowledge - An Assessment of Commenting Agents, Department of Computer and Information Science, Linköpings universitet SE-581 83 Linkoping Sweden.

Fischer, G., Girgensohn, A.1990 End-User Modifiability in Design Enviroments, CHI`90 Proceedings, ACM 0-89791-345-0/90/0004-0183

Fisher, G. McCall, R. Mørch, A.. I. 1989: Design Environments for Constructive and Argumentative Design, in: Human Factor in Computing Systems, Proc. of CHI'89, ACM, 269-275.

Frønes, I (2002). Digitale skiller. Utfordringer og strategier. Faktabokforlaget

Geisler, C. 1994. Academic literacy and the nature of expertise : reading, writing, and knowing in academic philosophy. Hillsdale, N.J. : Lawrence Erlbaum, 1994.

Grudin, J. (1991) CSCW Introduction , ACM Vol 34 No 12 pp 30-34 Communication of the ACM

Grudin, J. (1994). Computer-Supported Cooperative Work: History and Focus, IEEE Computer vol. 27 no. 5 (May 1994), pp. 19-25. Available at http://www.ics.uci.edu/~grudin/Papers/IEEE94/IEEEComplastsub.html

Gutwin, C. Stark, G. and Greenberg, S. 1995. Support for workspace Awareness in Educational Groupware. Department of Computer Science, University of Calagry, Calgary, Alberta,Canada Carl Gutwin, Saul Greenberg: 1996 Workspace Awareness for Groupware. CHI Conference Companion 1996: 208-209

Gutwin, C. Greenberg, S. 1998 Design for Individuals, Design for Groups: Tradeoffs between Power and Workspace Awareness. CSCW 1998: 207-216

Hakkarainen, K. 1998. Epistemology of inquiry and Computer-supported collaborative Learning. doctoral dissertation, University of Toronto, Ontario, Canada.

Hakkarainen, K., Lipponen, L., & Järvelä, S. (2001). Epistemology of inquiry and computer-supported collaborative learning. In T. Koschmann, R. Hall. & N. Miyake, (Eds.) *CSCL2:* Carrying foward the conversation, (pp. 128-156). Mahwah, NJ: Erlbaum.

Jennings, N. R. Wooldrigde, M. J. 1996 Software Agents IEEE review Januar 1996. pp 17-20

Johansen, R. 1988 Groupware: Computer Support for Business Teams, Macmillan, New York, 1988

Johnson W.L., Haye-Rothes B. 1998 The first autonomus Agent Conference The Knowledge Engeenering Review 13(2)pp 1-6 1998

Johnson, W. L. 1999 Pedagogical Agents. Invited Paper at the International Conference on Computer in Eduaction.

Jondahl, S. (2001) Simulering av pedagogiske agentar i eit virtuelt læremiljø ved bruk av Wizard of Oz teknikken, Hovudfagsoppgåve Institutt for informasjonsvitenskap(IFI) UIB

Kaptelinin, V. 1996 Activity theory: Implications for human-computer interaction. In Nardi, B. A. (Ed.), Context and consciousness: Activity theory and human-computer interaction, 103-116. Cambridge, MA: The MIT Press.

Klann T. ed. EUD-Net 2002, workshop in Pisa IST Programme (IST-2002-8.1.2) D1.1 Roadmap Fraunehofer FIT.

Kligyte, G. (2001). I think I know what is good for you. User Interface design for a CSCL system. Master thesis at Media Lab, The University of Art and Design, Helsinki UIAH.

Koschmann, T. (1996). Paradigm shifts and instructional technology: An introduction. In T. Koshmann (Ed.) CSCL: Theory and practice of an emerging paradigm. pp. 1-23. Mahwah, NJ: Laurence Elmbaum Assocates.

Lester, J. C., and Stone, B. A. 1997. Increasing believability in animated pedagogical agents. In Proceedings of the First International Conference on Autonomous Agents, 16-21.

Lester J. C., Voerman J L.,. Towns, S. G Callaway C. B. 1997 Cosmo: A Life-like Animated Pedagogical Agent with Deictic Believability

Lieberman H. Paternò F. Repenning A. Wulf V. (2003) Perspective on End User Development CHI 2003 New Horisonts CHI workshop 2003 Ft Lauderdale USA

Ludvigsen, S., Mørch A. I. 2003 Categorisation in knowlegde building: Task specific argumentation in a co-located CSCL enviroment. In Wasson, B. Ludvigsen, S. & Hoppe, U. (eds) Designing for Change in Networked Learning Enviroments. Proceedings of the International Conference on CSCL 2003.

MacLean, A. Carter, K. Løvstrand, L. Moran T.s 1990 User-Tailorable System: Pressing the issues with buttons. CHI`90 Prossedings Rank Xerox EuroPARC.

Maes, P. 1994. Social interface agents: Acquiring competence by learning from users andother agents. In Etzioni, O., editor, Software Agents — Papers from the 1994 SpringSymposium (Technical Report SS–94–03), pp. 71–78. AAAI Press.

Malone T. Crowston K. 1994 The interdisciplinary study of coordination. *Computing Surveys*, 26(1), 87–119.

Mead G. H. 1934 Mind Self, and Society. Chicago: University of Chicago Press.

Misch, A 2001 Context-oriented communication support in a colloaborative learning enviroment. Kolombus is born: Futuer Learning Envorioment Tools, In S. Bjørnstad, R.E. Moe A.I. Mørch & A. Opdahl.(eds) Proceedings of the 24th Information Systems Reserch Smeinar in Scandinavia (Volume II pp.43-58) Department of Infroamtion Science, University of Bergen.

Mørch A. 1995 Three levels of End-User Tailoring: Customization, Integration and Extension. 3. Decennial Confrence Århus Denmark

Mørch, A. Omdahl, K. & Dolonen, J. 2004 Supporting Conceptual Awareness with Pedagogical agents, InterMedia, Univerity of Oslo – paper in production

Nielsen, J. (1989). What do users really want? Int.J. Human-Computer Interaction ,1 2, 137-147

Nielsen, J. (1992). The usability engineering life cycle. IEEE Computer 25, 3 (March), 12-22.

Nielsen, Jacob 1993 Usability Engineering Academic Press, Inc. ISBN 0-12-518405-0

Norman, D. A. (1988). The psychology of everyday things. New York: Basic Books.

Nygaard, K., 1992: "How Many Choices Do We Make? How Many Are Difficult?", pp. 52-59 in the Proceedings from that conference: "Software Development and Reality Construction", Floyd, C., Züllighoven, H., Budde, R., and Keil-Slawik, R., editors. Springer-Verlag, Berlin 1992.

Nwana H. S. (1996). Software agents: An overview. From: Knowledge Engineering Review, Vol. 11, No 3, pp. 1-40, Sept. 1996, Cambridge University Press, 1996.

Omdahl, K. 2002 "Designing pedagogical Agents forCollaborative Learning: An Empirical Study" Department of Information Science, UIB.

Paavola S Lipponen L. Hakkarainen K.. 2002 Epistemological Foundations for CSCL: A Comparison of Three Models of Innovative Knowledge Communities, UIAH, Finland

Popper K. 1972 Knowlegde objects. London UK Oxford University Press 1972

Preece, J, Rogers Y., Sharp, H., Benyon, D., Holland, S., Carey,Tom 1994 Human Computer Interaction, Addison Wesley, ISBN 0-201-62769-8

Pressman, R. 1994 Software Engineering "A Practitioner`s Approach" 3.Edition European adaption McGrawHill ISBN 007-707936-1

Shneiderman, B. (1992) Designing the User Interface: Strategies for Effective Human-Computer Intercation 2nd edn Reading MA. Addison-Wesley

Suchman, L. 1989 Notes on Computer Support for Cooperative Work. Working Paper WP-12, Dept. of Computer Science, University of Jyvaskyla, SF-40100, Jyvaskyla, Finland.

Sumner M. Klepper, R(1986) End-User Application Developmen, Practices, Policies and Organizational Impacts. ACM, Southern Illinois University, Edvardsville USA.

Vygotsky, L.S. (1978). Mind in Society. Cole, M., John-Steiner, V., Scribner, S., and Souberman, E. (Eds.). Harvard University Press, Cambridge, MA

Wasson, B. Guribye, F. og Mørch, A. Project DoCTA: Design and use of Collaborative Telelearning Artefacts. Forsknings og kompetansenettverk for IT i utdanning, nr. 5. Oslo: Unipub forlag; 2000. 380 s.

Wasson, B. Ludvigsen, S. 2003 Designing for Knowlegde Building. ITU rapport nr 19 ISBN82-7947-023-9

Wertsch, J. V, del Río, P. & Alvarez, A. 1995. Sociocultural studies of mind. Cambridge: Cambridge University Press.

Wood, David P & Kyo C. Kang: A Classification and Bibliography of Software Prototyping Software Engineering Istitute, Carnegie Mellon University. Pittsburgh Pennsylvania.

Wooldridge M. J. and Jennings N. R (1995). Intelligent Agents: Theory and Practice. In Knowledge Engineering Review 10(2), 1995.

Zimmerman, B. J. (1989). A social cognitive view of self-regulated academic learning. Journal of Educational Psychology, 81, pp. 329–339.

# WEBREFERANSAR

EUD http://giove.cnuce.cnr.it/EUD-NET/documents.htm

FLE 3 http://fle3.uiah.fi.

Terminologi http://portalen.bibliotekivest.no/terminologi.htm 30.09.03

http://www.millkern.com/mohnkern/faq/html.html

Nielsen web Jacob Nielsen http://www.useit.com/papers/heuristic/heuristic_list.html

30.april 2003

# VEDLEGG

- A Kodemateriell



- B FLE3 modifiserte filer

VEDLEGG A

**Vedlegg A 1 Admin_agent.java**

```
//Title:      Ped agent
//Copyright:  Copyright (c) 2003
//Author:     JanEirik
//Company:    Intermedia

// sjølve GUI adminet styrer applikasjoen som GUIet er.

import java.sql.*;
import javax.swing.UIManager;
import java.applet.*;

public class Admin_agent extends Applet{
  boolean packFrame = false;


  //Construct the application
  public Admin_agent() {
    agenframe frame = new agenframe();

    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from
their layout

    if (packFrame)
      frame.pack();
    else
      frame.validate();
    frame.setVisible(true);
  }

  //Main method
  public static void main(String[] args) {
    try  {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e) {
    }
    Admin_agent a = new Admin_agent();
  }

}
```

**Vedlegg A 2 agenframe.java**

```
//Title:      Ped agent
//Copyright:  Copyright (c) 2003
//Author:     JanEirik
//Company:    Intermedia

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.UIManager;
import java.lang.*;
import java.applet.*;
import java.awt.FileDialog;
import java.sql.*;

public class agenframe extends JFrame {
  JLabel jLabel1 = new JLabel();
  JButton Lagregler = new JButton();
  JButton importer = new JButton();
  JLabel jLabel3 = new JLabel();
  CheckboxGroup cbg = new CheckboxGroup();
  JButton startagenten = new JButton();
  JButton eksport_regler = new JButton();
  Checkbox kat0 = new Checkbox("Kategori 0", cbg, true);
  Checkbox kat1 = new Checkbox("Kategori 1", cbg, false);
  Checkbox kat2 = new Checkbox("Kategori 2", cbg, false);
  Checkbox kat3 = new Checkbox("Kategori 3", cbg, false);
  JButton rediger = new JButton();
  GridBagLayout gridBagLayout1 = new GridBagLayout();
  Konverter konv = new Konverter();


  public agenframe() {
    try  {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }

  private void jbInit() throws Exception {
    jLabel1.setFont(new java.awt.Font("Dialog", 0, 24));
    jLabel1.setText("FLE-Agent Oppsett");
    this.getContentPane().setLayout(gridBagLayout1);
    Lagregler.setText("Lag nye regler for agenten");
    Lagregler.addActionListener(new java.awt.event.ActionListener() {
```

```
      public void actionPerformed(ActionEvent e) {
        Lagregler_actionPerformed(e);
      }
    });
    importer.setText("Importer eksisterende regeler");
    importer.addActionListener(new java.awt.event.ActionListener() {

      public void actionPerformed(ActionEvent e) {
        importer_actionPerformed(e);
      }
    });
    jLabel3.setText("Triggers aktiverings nivå");
    startagenten.setText("Aktiver agenten med nytt regelsett");
    startagenten.addActionListener(new java.awt.event.ActionListener()
{

      public void actionPerformed(ActionEvent e) {
        startagenten_actionPerformed(e);
      }
    });

    this.getContentPane().setBackground(Color.white);
    this.setResizable(false);
    eksport_regler.setText("Eksporter eksisterende Regelsett");
    eksport_regler.addActionListener(new
java.awt.event.ActionListener() {

      public void actionPerformed(ActionEvent e) {
        eksport_regler_actionPerformed(e);
      }
    });


    this.setTitle("Agent Oppsett");

    kat0.setLabel("Nivaa 0");
    kat1.setLabel("Nivaa 1");
    kat2.setLabel("Nivaa 2");
    kat3.setLabel("Nivaa 3");

    //this.setSize(new Dimension(355,411));
    this.setSize(new Dimension(455,311));
    rediger.setText("Rediger eksisterende regler");
    rediger.addActionListener(new java.awt.event.ActionListener() {

      public void actionPerformed(ActionEvent e) {
        rediger_actionPerformed(e);
      }
    });
    this.getContentPane().add(jLabel1, new GridBagConstraints(0, 0, 2,
1, 0.0, 0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new
Insets(0, 0, 0, 239), 0, 0));
    this.getContentPane().add(kat2, new GridBagConstraints(0, 4, 1, 1,
0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(0, 14, 0, 57), 0, 0));
```

```
    this.getContentPane().add(kat1, new GridBagConstraints(0, 3, 1, 1,
0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(0, 14, 0, 57), 0, 0));
    this.getContentPane().add(kat0, new GridBagConstraints(0, 2, 1, 1,
0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(0, 15, 0, 56), 0, 0));
    this.getContentPane().add(jLabel3, new GridBagConstraints(0, 1, 1,
1, 0.0, 0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new
Insets(21, 0, 0, 0), 0, 0));
    this.getContentPane().add(kat3, new GridBagConstraints(0, 5, 1, 1,
0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(0, 14, 9, 57), 0, 0));
    this.getContentPane().add(Lagregler, new GridBagConstraints(1, 1,
1, 2, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(25, 33, 11, 45), 57, 0));
    this.getContentPane().add(importer, new GridBagConstraints(1, 5,
1, 1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(7, 33, 0, 45), 38, 0));
    this.getContentPane().add(eksport_regler, new
GridBagConstraints(1, 6, 1, 1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(6, 33, 0, 45), 16, 0));
    this.getContentPane().add(rediger, new GridBagConstraints(1, 3, 1,
1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(0, 32, 0, 45), 47, 0));
    this.getContentPane().add(startagenten, new GridBagConstraints(1,
7, 1, 1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(28, 32, 22, 45), 20, 0));

  }


  //Overridden so we can exit on System Close
  protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if(e.getID() == WindowEvent.WINDOW_CLOSING) {
      System.exit(0);
    }
  }

  void Lagregler_actionPerformed(ActionEvent e) {
      //kaller neste GUI (Regel generatoren)
      boolean packFrame = false;
    Rules_innsett edit_frame = new Rules_innsett();

  if (packFrame)
    edit_frame.pack();
  else
    edit_frame.validate();
```

```
  edit_frame.setVisible(true);
this.dispose();

}

void importer_actionPerformed(ActionEvent e) {
    String filnavn_inn;
    agenframe frame = new agenframe();
   FileDialog test =new FileDialog(frame, "load", 0);
   test.setVisible(true);
   filnavn_inn = test.getFile();
   //lage til lesing av fil
   //bruker  rule_editor til å laste inn regler

}

void eksport_regler_actionPerformed(ActionEvent e) {

   //skriver regelene til tekstfil
 agenframe frame = new agenframe();
   FileDialog test =new FileDialog(frame, "save", 1);
   test.setVisible(true);
   String filnavn_ut;
   filnavn_ut = test.getFile();
   //lage til skriving av fil fra database
   //bruker rule til eksoprt (kvart objekt blir ei tekst fil
   //sql oppkobling pluss filbehandling
}

void startagenten_actionPerformed(ActionEvent e) {
boolean packFrame = false;
String kat = new String();
String raakat = new String();
String query = new String();
int katop;
raakat = this.cbg.toString();
kat = konv.getstate(raakat);
System.out.println(raakat);
System.out.println(kat+"kat typen inn");
 this.dispose();
String url = "jdbc:mysql://localhost/agent_db";

  Connection con;
   try {
        Class.forName("org.gjt.mm.mysql.Driver");
   }

   catch(java.lang.ClassNotFoundException eie) {
        System.err.print("ClassNotFoundException: ");
        System.err.println(eie.getMessage());
   }


   try  {
        con = DriverManager.getConnection(url, "jankee", "kirsten");
        Statement stat = con.createStatement();
       query ="UPDATE kat SET nivaa = '"+ kat + "' where id = 1";
```

```
          System.out.println(query);
          katop = stat.executeUpdate(query);
          System.out.println("går ut av sql struktur oppdatering
fullført");
          con.close();
     }
     catch(SQLException ex) {
     System.err.println("SQL exsept2: " + ex.getMessage());
     }


  try {
// Crank up the server and wait for connection
SimpleServer server = new SimpleServer(8010);
        // System.out.println("Opprettet nytt Server objekt agentframe");
         server.waitForClients();
         System.out.println("startet wait for clients");
 }
      catch (Exception oops) {
// If there was an error starting the server, say so!
System.out.println("Got error starting server:");
oops.printStackTrace();
                                                                        }


    //starter agenten
  }

  void rediger_actionPerformed(ActionEvent e) {
   boolean packFrame = false;
    vis_alle_regeler edit_frame = new vis_alle_regeler();

  if (packFrame)
    edit_frame.pack();
  else
    edit_frame.validate();
    edit_frame.setVisible(true);
  this.dispose();

  }


}
```

**Vedlegg A 3 CoordinatorAgent.java**

```
import java.io.*;
import java.sql.*;
import java.net.*;
import java.util.Arrays;
import java.util.Random;

/**
 * Title:
 * Copyright:    Copyright (c) 2001
  * @author: Jan Dolonen // mod Jan Eirik Nævdal
```

```
 * @version 1.0
 * Description: Denne blir kalt opp fra SimpleServer uten argumenter.
 * Innehar metoder som  gjør diverse spørringer mot databasen.
 * Spørringene blir sendt som
 * strenger fra denne klassen. Svaret blir deretter sjekket opp mot
regler.
 * Hvis svaret bryter med reglene kalles Messages - klassen opp som
returnerer
 * et tekstlig svar som igjen blir sendt til SimpleServer.
 */

public class CoordinatorAgent {

  DbQuery dbquery = new DbQuery();
  Rulecoordinator rulesett = new Rulecoordinator();

  private String returnString = "";
  private String ip_no;
  private String name = null;
  private String j = null;
  private String t = null;
  private int pri;
  private int tooMuch, tooLittle, inq, addN, num, type, login = 0;
  private static int importance = 3; // compared to pri to decide of
the violation is an important one..


  public static void main(String[] args) throws IOException {

                          CoordinatorAgent hoved = new CoordinatorAgent();
                                                                  String vel;

  }

  public CoordinatorAgent() {
super();
  }
  // finner navn på bruker som har ip x.x.x.x
  public synchronized String fName(InetAddress ip){
                                                    ip_no = ip.toString();
    System.out.println(ip_no+"ipen som kommer inn til db i
coord.fname");
                                               j = dbquery.findName(ip_no);
    System.out.println(j + " kva navn som kommer ut av db");
                                                                    return j;
  }


  public synchronized String Coordinate(String brukernavn) throws
IOException {

                                                                j = brukernavn;
    System.out.println(j + brukernavn +"begge variabler som kommer inn
i Coord.Coordinate");
    returnString = regel(j);

    return returnString;
```

```
   }


  public synchronized String regel (String param) {
    String tuull = "";
    tuull = param;

    tuull = rulesett.regelsett_innlogging(param);


   return tuull;
  }


 }
```

**Vedlegg A 4 DBQuery.java**

```
import java.io.*;
import java.sql.*;
//import java.sql.Date;
import java.util.Date;
//import java.net.*;

/**
 * Title: DbQuery.java
 * Description: Denne klassen tar seg av all kontakt med databasen.
Resultatene
 * fra spørringene blir deretter sendt til Rules klassen der de blir
sjekket opp
 * mot reglene.
 * Copyright:    Copyright (c) 2001 //2003
 * Company:
 * @author: Jan Dolonen  // mod by Jan Eirik Nævdal
 * @version 1.1
 */

public class DbQuery {

  DbQuery db;
  private Connection con2 = null;
  private Connection conn = null;
  private Connection con4= null;
  String driver = "org.gjt.mm.mysql.Driver";
  String url = "jdbc:mysql://localhost/agent_db";
  String dbuser = "jankee";
  String dbpassword = "kirsten";
  //private Rules rules;
  private String name;
  private String navn;
  private ResultSet Ret;
  private Date test;
```

```
  private String query;
  private String text;
  private int i, j, k = 0;
 // String bruker_trigg[][] = new String [10][4]

  public DbQuery() {
  // tom konstruktør

  }
/**
 * Insert the method's description here.
 * Creation date: (28.11.01 21:25:19)
 * @param param java.lang.String
 */

  public synchronized String findName(String ip) {
try{
  System.out.println("I will now try to query..");
Class.forName(driver);
conn = DriverManager.getConnection(url, dbuser, dbpassword);
Statement s = conn.createStatement ();
//private String ip_addr =
String slash = "/";
int index = 0;
String nyIp = ip;
index = nyIp.lastIndexOf(slash);
String subIp = null;
subIp = nyIp.substring(index + 1);
subIp = "" + "'" + subIp + "'" + "";//System.out.println("subIp er: " + subIp);
//System.out.println("navnet er: " + name);
query = "Select name from login_no where ip_no = " + subIp ;
Ret = s.executeQuery(query);
if (Ret.next()) {
    navn = Ret.getString(1); // i = ant argumenter i alt minus brukerens.

}
 //System.out.println("navnet i findName er: " + navn);

Ret.close();
}
catch (Exception e){
System.err.println(e);
if (e instanceof SQLException)
System.err.println("SQL State: " + ((SQLException)e).getSQLState());
                  System.out.println("Something went wrong when trying
to query in findName()");
//           conn.close();
}
        finally {
            try { conn.close(); System.out.println("closing database
connection in findName"); }
            catch (Exception e) {}
        }

return navn;

 }
```

```
 public synchronized String findFrase(int regelid) {
  String frase = null;
  String query = null;
  int id = regelid;


  try  {  // henter loggin data
      i=0;
      con2 =  DriverManager.getConnection(url, dbuser, dbpassword);
      Statement stat2 = con2.createStatement();
      query = "Select frase from regel where id = " + "'"+id+"'" ;

      ResultSet variabel2=stat2.executeQuery(query);
      while(variabel2.next()) {
            frase = variabel2.getString("frase");
      }
      con2.close();
    }
    catch(SQLException ex) {
     System.err.println("SQL exsept hent login (hallo): " +
ex.getMessage());
    } // slutt db kall con3


 return frase;
 }

 public void db_post_slett(String db_tabell, int tabell_kol) {
  String tabell = db_tabell;
  int id = tabell_kol;
  String query = null;

  try  {  // henter loggin data
      i=0;
      conn =  DriverManager.getConnection(url, dbuser, dbpassword);
      Statement statn = conn.createStatement();
      query = "Delete from "+tabell+" where id = " + "'"+id+"'" ;

      ResultSet variabeln=statn.executeQuery(query);
     // while(variabeln.next()) {
      //      frase = variabeln.getString("frase");
      //}
      conn.close();
    }
    catch(SQLException ex) {
     System.err.println("SQL exsept hent login (hallo): " +
ex.getMessage());
    } // slutt db kall con3

 }

 public void db_nullstill_time2(String db_tabell, String db_verdi) {
  String tabell = db_tabell;
  String iduser = db_verdi;
  String query = null;
  String tid= null;
```

```
    try  {  // henter loggin data
      i=0;
      conn =  DriverManager.getConnection(url, dbuser, dbpassword);
      Statement statn = conn.createStatement();
      query = "Select time from "+ tabell +" where name = " +
"'"+iduser+"'" ;

      ResultSet variabeln=statn.executeQuery(query);
      while(variabeln.next()) {
            tid = variabeln.getString("time");
      }
      conn.close();
    }
    catch(SQLException ex) {
     System.err.println("SQL exsept hent login (hallo): " +
ex.getMessage());
    } // slutt db kall con3


    try  {  // henter loggin data
      i=0;
      con4 =  DriverManager.getConnection(url, dbuser, dbpassword);
      Statement stat4 = con4.createStatement();
      query = "Update "+ tabell +" Set time2 = "+"'"+ tid +"'"+" where
name = " + "'"+ iduser +"'" ;
      System.out.println(query);
      ResultSet variabel4=stat4.executeQuery(query);
     // while(variabeln.next()) {
      //      frase = variabeln.getString("frase");
      //}
      conn.close();
    }
    catch(SQLException ex) {
     System.err.println("SQL exsept oppdater logintabell: " +
ex.getMessage());
    } // slutt db kall con3

 }


}
```

**Vedlegg A 5 DBQueryCount.java**

```
//Title:        Ped agent
//Copyright:    Copyright (c) 2004
//Author:       JanEirik
//Company:      Intermedia
import java.io.*;
import java.sql.*;
import java.util.*;
```

```
import java.awt.*;
import javax.swing.*;
import java.lang.*;
import java.math.*;
import java.text.*;

public class DbQueryCount {
  Connection con = null;
  Connection con4 = null;
  String driver = "org.gjt.mm.mysql.Driver";
  String url = "jdbc:mysql://localhost/agent_db";
  String dbuser = "jankee";
  String dbpassword = "kirsten";

  public DbQueryCount() {
  //Tom konstruktør
  }

  public int dbCount3(String db_tabell, String db_tab, String
db_varverdi) {
   String tab = db_tabell;
   String dbfelt = db_tab;
   String dbverdi = db_varverdi;
   String query= null;
   int storleik= 0;

   try {
         Class.forName(driver);
     }

     catch(java.lang.ClassNotFoundException e) {
          System.err.print("ClassNotFoundException: ");
          System.err.println(e.getMessage());
     } // slutt henting



     try  {// henter storleik på matriser denne har where med og tar 3
varaibler

          con =  DriverManager.getConnection(url, dbuser, dbpassword);
          Statement stat2 = con.createStatement();

          query = "Select COUNT("+dbfelt+") FROM "+tab+" where
"+dbfelt+" = "+"'"+dbverdi+"'" ;
         // System.out.println("Select COUNT("+dbfelt+")  FROM "+tab+"
where "+dbfelt+" = "+"'"+dbverdi+"'");
          ResultSet variabel2 = stat2.executeQuery(query);  //"Select
COUNT(varnavn) FROM regel where varnavn = 'sist innlogging'");
          while(variabel2.next()) {
            storleik =variabel2.getInt("count("+dbfelt+")") ;
          }
     }
     catch(SQLException ex) {
          System.err.println("SQL exsept matrsie tell1: " +
ex.getMessage());
     } // slutt db kall  con
```

```
   return storleik;
  }

   public int dbCount2(String tabell, String db_variabel) {
   String tab = tabell;
   String dbfelt = db_variabel;
   String query= null;
   int storleik= 0;

   try  {// henter storleik på matriser i forskjellige størrelser
(denne tar 2 variabler inn

          con4 =  DriverManager.getConnection(url, dbuser,
dbpassword);
          Statement stat4 = con4.createStatement();
          query = "Select COUNT("+dbfelt+") FROM "+tabell+"";
          ResultSet variabel4 = stat4.executeQuery(query);//"Select
COUNT(name) FROM login");
          while(variabel4.next()) {
            storleik =variabel4.getInt("count("+dbfelt+")") ;
          }
          con.close();
    }
    catch(SQLException ex) {
          System.err.println("SQL exsept matrise tell2: " +
ex.getMessage());
    } // slutt db kall con4
   return storleik;
  }

  public int dbCount4(String db_tabell, String db_kolonneTell,String
db_kolonne_var, String db_varverdi) {
   String tab = db_tabell;
   String dbfelt = db_kolonneTell;
   String dbfelt2 = db_kolonne_var;
   String dbverdi = db_varverdi;
   String query= null;
   int storleik= 0;

   try {
          Class.forName(driver);
     }

     catch(java.lang.ClassNotFoundException e) {
          System.err.print("ClassNotFoundException: ");
          System.err.println(e.getMessage());
     } // slutt henting



     try  {// henter storleik på matriser denne har where med og tar 3
varaibler

          con =  DriverManager.getConnection(url, dbuser, dbpassword);
          Statement stat2 = con.createStatement();
```

```
          query = "Select COUNT("+dbfelt+") FROM "+tab+" where
"+dbfelt2+" = "+"'"+dbverdi+"'" ;
         System.out.println(query);
         // System.out.println("Select COUNT("+dbfelt+")  FROM "+tab+"
where "+dbfelt+" = "+"'"+dbverdi+"'");
          ResultSet variabel2 = stat2.executeQuery(query);  //"Select
COUNT(varnavn) FROM regel where varnavn = 'sist innlogging'");
          while(variabel2.next()) {
            storleik =variabel2.getInt("count("+dbfelt+")") ;
          }
     }
     catch(SQLException ex) {
          System.err.println("SQL exsept matrsie tell1: " +
ex.getMessage());
     } // slutt db kall  con

   return storleik;
  }



}
```

**Vedlegg A 6 Diffcalc.java**

```
//Title:        aa
//Version:
//Copyright:    Copyright (c) 2004
//Author:       JanEirik
//Company:      Intermdia/IFI
//Description:  Your description

public class diffcalc {
  String stringmnd1, stringmnd2 = new String();
  int dag1,dag2,aar1,aar2, mnd1, mnd2;
  String type = new String();
  int diffaar, diffmnd, diffdag;
  int mnddager1, mnddager2, mndsum, mellomsum1,mellomsum2, mndteller,
i ;
  int svar= 0;
  Konverter omformer = new Konverter();
  public void  diffcalc() {
  //tom konstruktør
  }
  public int check_login(int dag1,int dag2, int aar1, int aar2, int
mnd1, int mnd2) {

System.out.println(dag1+","+dag2+","+aar1+","+aar2+","+mnd1+","+mnd2);
    diffdag=-1;
    diffmnd=-1;
    diffaar=-1;
    svar = -1;
```

```
    //Viss det har gått over eit år vil den returnere 365 uansett
    //konverter string mnd til integer
     if (aar1 == aar2) { // viss det er berre dager mellom i samme mnd

       if (mnd1 == mnd2) {

           diffdag = dag2-dag1;
           svar = diffdag;
           System.out.println( "diffdag : "+diffdag);
       }
       else if (mnd2 > mnd1) {
           diffmnd=mnd2-mnd1;
           if (diffmnd ==1) {
             mnddager1=  omformer.maanedsdager(mnd1, aar1);
             diffdag=mnddager1-dag1+dag2;
             svar=diffdag;
          // System.out.println(diffdag+ "diffdag");
           }
           else if (diffmnd >2) {
             // finn ut kva måned sist innlogging er og kor mange
dager det er
             mnddager1=  omformer.maanedsdager(mnd1, aar1);
              mellomsum1=mnddager1-dag1;
             mndteller=mnd1+1;
             for (i= 0; i==(diffmnd-2); i++) {
             mellomsum2= mellomsum2 + omformer.maanedsdager(mndteller,
aar1);
             }
             diffdag = mellomsum1 + mellomsum2;
             svar =diffdag;
            //System.out.println(diffdag+ "diffdag");
           }
           else if (diffmnd >6) {
              svar= 365;//setter inaktivitet i over 6 mnd lik eit år
           //System.out.println(diffdag+ "diffdag");
           }

       }

     }

     else if (aar1 < aar2) {  //
       diffaar = aar2-aar1;
       if (diffaar == 1) {
          if (mnd2 > mnd1) {
             svar = 365; // over eit år sidan innlogging
             //System.out.println(diffdag+ "diffdag");
          }
          else if (mnd2 == mnd1)   {
             svar=365;
            // System.out.println(diffdag+ "diffdag");
          }
       }
     }
     else  if (diffaar >2 ) { //viss det har gått over eit år.
       svar =365;
```

```
      //System.out.println(diffdag+ "diffdag");
     }

   return svar;
  }
}
```

**Vedlegg A 7 InterfaceAgent.java**

```
/**
 * Insert the type's description here.
 * Creation date: (29.08.2000 23:40:42)
 * @author:
 */

import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;


public class InterfaceAgent extends Applet implements Runnable,
ReadCallback {

protected TextArea ta;
protected Frame f;
private boolean isRunning = true;
private String t;
public Thread thread;
    private boolean running=true;
public Socket serverSock;
public DataOutputStream outStream;
public ReadThread reader;
    public static String host="129.240.213.120";//denne må endre til
ip-adressen til serveren
    public static int port=8010;
    private static int lengde = 100;
    private boolean important = false;
    public PopupAgent pop;



 public synchronized void dataReady(String str)

 {
   try {
readIn(str);
   }
   catch (Exception oida){

   }
```

```
 }


 public void disconnect()

 {

try {
reader.closeConnection();
stop();

} catch (Exception badClose) {

}

 }

 public void firePopUp(String melding) {

// fires pop up box.. which closes by itself..
t = melding;
pop = new PopupAgent(t);
pop.start();

 }

 public synchronized void readIn(String streng){

   try {
    System.out.println(streng+ " streng i readIn");
String t = streng;
// do an importance check to see whether or not the pop up box must fire..
//String imp = t.substring(0,3);
//important = imp.compareTo("NB!");
    important = false;

if (important == true){
      firePopUp(t);

      }
// this wraps the text so that it's not over the limit
// of the textarea used..
int check = 0;
int oldPos = 0;
String bb = " ";
int newPos = 0;
String newString = "";

while (t.length() > check){
newPos = t.lastIndexOf(bb, (lengde + oldPos));
newString = newString + (t.substring(oldPos,newPos) + '\n');oldPos = newPos;
check = oldPos + lengde;
}
t = newString + (t.substring(oldPos));
        ta.append(t);
```

```
        //running=false; // I dont want too many threads running...
}

   catch (Exception oi){
System.out.println("Feilmelding i readIn :"+oi);
   }

}


public void run()

{

while (running)


{

try {
       t = "" ;
       //System.out.println("så langt bra start run");
      sendString("Client Calling!");
      System.out.println("Klient ringer");
      ta.append(t);
     Thread.sleep(60000); //1000 = 1 sec.
      running = true;
      }
    catch (Exception oops) {
                      // If there was an error, print info and disconnect
    oops.printStackTrace();
    // System.out.println("så etter printstack");
     disconnect();
     running = false;

}

}

}

public synchronized void sendString(String str) throws IOException

{
try {
outStream.writeUTF(str);

}
catch (IOException ioe){
ioe.printStackTrace();

                                                                        }

}

public synchronized void stop() {
    try {
```

```
        Thread t=Thread.currentThread();
        t.interrupt();
        t = null;

    }
    catch(Exception e){
    }
}

public void start(){
thread = new Thread(this);
thread.start();


}

public void init(){

setName("AgentHandler");
setLayout(new FlowLayout());
ta = new TextArea();

this.add(ta);

try {
   // host ='129.240.213.120';
//port = '8010';
Socket serverSock = new Socket(host, port);
this.host = host;
this.port = port;
outStream = new DataOutputStream(serverSock.getOutputStream());
reader = new ReadThread(this, serverSock);
reader.start();
running = true;
}
                                                catch (java.io.IOException e) {
e.printStackTrace();
}

}
}
```

**Vedlegg A 8 Konverter.java**

```
//Title:        Your Product Name
//Version:
//Copyright:    Copyright (c) 2004
//Author:       Your Name
//Company:      Your Company
//Description:  Your description


import java.io.* ;
import java.lang.* ;
import java.util.*;

public class Konverter {
  String stmnd = new String();
  int intmnd ;

  public Konverter() {
   //tom konstruktør
  }

  public int maanadnr(String stmnd) {
    // Boolean rett = false;
     String Jan= "Jan";
     String Feb= "Feb";
     String Mar= "Mar";
     String Apr= "Apr";
     String May= "May";
     String Jun= "Jun";
     String Jul= "Jul";
     String Aug= "Aug";
     String Sep=  "Sep";
     String Oct = "Oct";
     String Nov = "Nov";
     String Des = "Des";


      if (Jan.equals(stmnd)== true) {
          intmnd =1;
      }
      else if (Feb.equals(stmnd)== true) {
          intmnd =2;
      }
      else if (Mar.equals(stmnd)== true) {
          intmnd =3;
      }
      else if (Apr.equals(stmnd)== true) {
          intmnd =4;
      }
      else if (May.equals(stmnd)== true) {
          intmnd =5;
      }
```

```
    else if (Jun.equals(stmnd)== true) {
        intmnd =6;
    }

    else if (Jul.equals(stmnd)== true) {
        intmnd =7;
    }
    else if (Aug.equals(stmnd)== true) {
        intmnd =8;
    }
    else if (Sep.equals(stmnd)== true) {
        intmnd =9;
    }
    else if (Oct.equals(stmnd)== true) {
        intmnd =10;
    }
    else if (Nov.equals(stmnd)== true) {
        intmnd =11;
    }
    else if (Des.equals(stmnd)== true) {
        intmnd =12;
    }


    return intmnd;
}//slutt mnd nr

public int maanedsdager(int maaned, int aar) {
    int count = -1;
    switch (maaned) {
      case 1:
      case 3:
      case 5:
      case 7:
      case 8:
      case 10:
      case 12:
        count = 31;
        break;

      case 4:
      case 6:
      case 9:
      case 11:
         count = 30;
         break;


      case 2:
         if (aar % 4 ==0)
             count = 29;
         else
            count =28;
        if ((aar % 100 == 0) & (aar % 400 != 0))
            count = 28;
      }
  System.out.println("inne i konverter mnddager"+ count);
```

```
  return count;
}  //slutt maanedsdager

public int gettimedag(String tid) {
  int dag=0;
  int svar;
  StringBuffer tid1 = new StringBuffer(25);
  StringBuffer mellom = new StringBuffer(4);
  tid1.insert(0,tid);
  //dag i måneden

  mellom.append(tid1.charAt(14));
  mellom.append(tid1.charAt(15));
  dag= Integer.parseInt(mellom.substring(0));

  svar = dag;
  System.out.println(dag);
return svar;
}  //slutt gettimedag
public int gettimeaar(String tid) {
  int aar=0;
  int svar= 0;
  StringBuffer tid1 = new StringBuffer(25);
  StringBuffer mellom = new StringBuffer(4);
  tid1.insert(0,tid);
  mellom.append(tid1.charAt(21));
  mellom.append(tid1.charAt(22));
  mellom.append(tid1.charAt(23));
  mellom.append(tid1.charAt(24));
  aar= Integer.parseInt(mellom.substring(0));
  svar = aar;
return svar;
} //slutt gettimeaar

public int gettimemnd(String tid) {
  int mnd=0;
  int svar;
  StringBuffer tid1 = new StringBuffer(25);
  StringBuffer mellom = new StringBuffer(4);
  tid1.insert(0,tid);
  mellom.append(tid1.charAt(17));
  mellom.append(tid1.charAt(18));
  mellom.append(tid1.charAt(19));
  stmnd=mellom.substring(0);
  mnd=maanadnr(stmnd);
  svar=mnd ;
return svar;
} //slutt gettimemnd


public int msli_tall(String msli) {
  int j= -1;
  int svar= 1;
  String Mindre ="Mindre";
  String Større = "Større";
  String Lik = "Lik";
  String intervall = "Intervall";
```

```
    String velg = "Velg regelens viktighets nivå";
   // j= 1;
    if (Mindre.equalsIgnoreCase(msli)) {
      j=0;
    }

    else if (Større.equalsIgnoreCase(msli)) {
      j=1;
    }

    else if (Lik.equalsIgnoreCase(msli))  {
      j=2;
    }

    else if (intervall.equalsIgnoreCase(msli)){
      j=3;
    }

    else if (velg.equalsIgnoreCase(msli)) {
     j= -1;

    }

    svar= j; // svar = J;  rett oppsett 1 er test variablene.
   // System.out.println(j+ " verdien av type i konvert");
    return svar;
  }//slutt msli_tall

  public String getstate (String knappstreng) {
    String streng = knappstreng;
    String nivaa = new String();
    StringBuffer nivaabuff = new StringBuffer(25);
    StringBuffer mellom = new StringBuffer(4);
    nivaabuff.insert(0,streng);

    mellom.append(nivaabuff.charAt(93));

    nivaa=mellom.substring(0);
    System.out.println(nivaa);
   return nivaa;
  }// slutt getstate
}
```

**Vedlegg A 9 LoginAgent.java**

```
import java.io.*;
import java.sql.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.lang.*;
import java.math.*;
import java.text.*;
```

```
public class loginAgent {
  //db oppsettsvariabler
  Connection con3 = null;
  Connection conn = null;
  Connection con = null;
  Connection con4 = null;
  Connection conn_trigga = null;
  String driver = "org.gjt.mm.mysql.Driver";
  String url = "jdbc:mysql://localhost/agent_db";
  String dbuser = "jankee";
  String dbpassword = "kirsten";
  int i, x, y;
  int j;
  int inttrigger1;
  int inttrigger2;
  String trigget = new String();
  int gammeltid;
  int nytid;
  int ok; //oppdatering av db ok.

  public void loginAgent() {

 // tom konstruktør
  }

  public String login(String kven)   {
    // nye objekt lag
    Konverter konv = new Konverter();
    diffcalc regelcalc = new diffcalc();
    DbQueryCount CountKall = new DbQueryCount();

    String b_user = kven;   // innparameteren frå serveren
    String querylogin = new String();
    String svar = new String();
    String verdi1= new String();
    String verdi2= new String();
    String kat= new String();
    String msli1= new String();
    String id = new String();
    int regelstorleik= 0;
    int brukarstorleik2 = 0;
    String time1 = new String();
    String time2 = new String();
    String brukernavn = new String();
    String gammeltid = new String();
    String nytid = new String();
    String regel_id = new String();;
    String triggerverdi = new String();
    String triggerverdi_interv = new String();
    String type = new String();
    String nivaa = new String();
    int mnd1, aar1, dag1,mnd2, aar2, dag2; //verdier fra tid via
stringbuffer
    String stmnd1,stmnd2 = new String();
```

```
    int tr_verdi1, tr_verdi2, diff;
    int post_tell=0;
    int triggerok;

    String brukernavnimat = new String();
    String query = new String();
    //faste verdier for dbquerycount
    String DB_felt = "varnavn";
    String tabell1 = "regel";
    String tabell2 = "login";

    String tekst_var = "sist innlogging";
    String DB_felt2 = "name";

    System.out.println(kven+ " :kven");
    System.out.println(b_user+ " :b_user");
     // driver til  mysql dben
     try {
          Class.forName(driver);
     }

     catch(java.lang.ClassNotFoundException e) {
          System.err.print("ClassNotFoundException: ");
          System.err.println(e.getMessage());
     } // slutt henting

     regelstorleik =  CountKall.dbCount3(tabell1, DB_felt, tekst_var);
     brukarstorleik2 = CountKall.dbCount2(tabell2, DB_felt2);

    // lager dei matrisene som skal vere med  for firskjellige svar og
sett.

    String regelmatrise[][] = new String[regelstorleik][5];
//matrise til regel
    String brukarnavnmatrise[][] = new
String[brukarstorleik2][3];//matriselogin datae
    String svarmatrise[][] = new String[regelstorleik][4];   //
svarmatrise  returnert til coordinator


    try {  // Henter dei aktuelle verdiane for å generee ein regel

          conn = DriverManager.getConnection(url, dbuser, dbpassword);
          Statement stat = conn.createStatement();
          ResultSet variabel = stat.executeQuery("select id,
verdi1_tr, verdi2_tr, kat, msli1 from regel where varnavn ='sist
innlogging'");
          i=0;
          while (variabel.next()) {
             id = variabel.getString("id");
             verdi1 = variabel.getString("verdi1_tr");
             kat = variabel.getString("kat");
             msli1 = variabel.getString("msli1");
             verdi2 = variabel.getString("verdi2_tr");
             // Innsetting til ei matrise for oppbevaring av data
             regelmatrise[i][0] = id;
             regelmatrise[i][1] = verdi1;
```

```
             regelmatrise[i][2] = verdi2;
             regelmatrise[i][3] =msli1;
             regelmatrise[i][4] = kat;
             System.out.println(id+" denne iden går inn i matrisa  (er
det her det bugger");
            i++;
          }
          conn.close();
    }
    catch(SQLException ex) {
       System.err.println("SQL exsept_i hent_regeleme:" +
ex.getMessage());
    } // slutt db kall conn


    try  {  // henter loggin data
      i=0;
      con3 =  DriverManager.getConnection(url, dbuser, dbpassword);
      Statement stat3 = con3.createStatement();
      querylogin = "Select name, time,time2 from login where name = "
+ "'"+b_user+"'" ;
      ResultSet variabel3=stat3.executeQuery(querylogin);
      while(variabel3.next()) {
            brukernavn = variabel3.getString("name");
            time1 = variabel3.getString("time");
            time2 = variabel3.getString("time2");

            brukarnavnmatrise[i][0]= b_user;
            brukarnavnmatrise[i][1]= time1;
            brukarnavnmatrise[i][2]= time2;
            i++;
          }
          con3.close();
    }
    catch(SQLException ex) {
     System.err.println("SQL exsept hent login (hallo her): " +
ex.getMessage());
    } // slutt db kall con3



     x= regelstorleik;
     i=0;
     //Svar = "";
     for (i = 0; i<x; i++) {
        post_tell= 0;
        regel_id =regelmatrise[i][0];
        triggerverdi =regelmatrise[i][1];
        triggerverdi_interv= regelmatrise[i][2];
        type =regelmatrise[i][3];
        nivaa = regelmatrise[i][4];

        tr_verdi1 = Integer.parseInt(triggerverdi);
        tr_verdi2 = Integer.parseInt(triggerverdi_interv);
        System.out.println("verdien frå db i tall " + tr_verdi1 );
        j =konv.msli_tall(type);
```

```
        svar = null;


            switch(j) {

                case 0:    // msli1 = Mindre
                  System.out.println("inne i case1 "+ regel_id); // 2
                    brukernavnimat = brukarnavnmatrise[0][0];
                    gammeltid =brukarnavnmatrise[0][1];
                    nytid =brukarnavnmatrise[0][2];
                    dag2=konv.gettimedag(gammeltid);
                    dag1=konv.gettimedag(nytid);
                    mnd2=konv.gettimemnd(gammeltid);
                    mnd1=konv.gettimemnd(nytid);
                    aar2=konv.gettimeaar(gammeltid);
                    aar1=konv.gettimeaar(nytid);

                    diff
=regelcalc.check_login(dag1,dag2,aar1,aar2,mnd1,mnd2);
                    System.out.println(" alle sjekkete users og derira
diff: "+diff+", "+ brukernavnimat );

                    if (tr_verdi1 > diff)  {

                      svarmatrise[i][0]= regel_id;
                      svarmatrise[i][1]= brukernavnimat;
                      svarmatrise[i][2]= regelmatrise[i][4];
                      svarmatrise[i][3]= "1";
                      post_tell= post_tell +1;

                      svar = "trigget"  ;
                      System.out.println("Bruker : " + brukernavnimat
+" har trigget regel "+regel_id+", ");
                    }

                  // }
                  svar = "utrigget";

                   break;

               case 1:   // msli1 = Større,  setter denne opp  defult
                         //  til testing av regelen
                    System.out.println("inne i case1 "+ regel_id); //2
                    brukernavnimat = brukarnavnmatrise[0][0];
                    gammeltid =brukarnavnmatrise[0][1];
                    nytid =brukarnavnmatrise[0][2];
                    dag2=konv.gettimedag(gammeltid);
                    dag1=konv.gettimedag(nytid);
                    mnd2=konv.gettimemnd(gammeltid);
                    mnd1=konv.gettimemnd(nytid);
                    aar2=konv.gettimeaar(gammeltid);
                    aar1=konv.gettimeaar(nytid);

                    diff
=regelcalc.check_login(dag1,dag2,aar1,aar2,mnd1,mnd2);
                    System.out.println(" alle sjekkete users og derira
diff: "+diff+", "+ brukernavnimat );
```

```
                    if (tr_verdi1 < diff)  {

                      svarmatrise[i][0]= regel_id;
                      svarmatrise[i][1]= brukernavnimat;
                      svarmatrise[i][2]= regelmatrise[i][4];
                      svarmatrise[i][3]= "1";
                      post_tell= post_tell +1;

                      svar = "trigget"  ;
                      System.out.println("Bruker : " + brukernavnimat
+" har trigget regel "+regel_id+", ");
                    }

                  // }
                  svar = "utrigget";
                break;

               case 2:   // msli1 = Lik
                  System.out.println("inne i case1 "+ regel_id); // 2
                     brukernavnimat = brukarnavnmatrise[0][0];
                     gammeltid =brukarnavnmatrise[0][1];
                     nytid =brukarnavnmatrise[0][2];
                     dag2=konv.gettimedag(gammeltid);
                     dag1=konv.gettimedag(nytid);
                     mnd2=konv.gettimemnd(gammeltid);
                     mnd1=konv.gettimemnd(nytid);
                     aar2=konv.gettimeaar(gammeltid);
                     aar1=konv.gettimeaar(nytid);

                 diff
=regelcalc.check_login(dag1,dag2,aar1,aar2,mnd1,mnd2);
                    System.out.println(" alle sjekkete users og derira
diff: "+diff+", "+ brukernavnimat );

                    if (tr_verdi1 == diff)  {

                      svarmatrise[i][0]= regel_id;
                      svarmatrise[i][1]= brukernavnimat;
                      svarmatrise[i][2]= regelmatrise[i][4];
                      svarmatrise[i][3]= "1";
                      post_tell= post_tell +1;

                      svar = "trigget"  ;
                      System.out.println("Bruker : " + brukernavnimat
+" har trigget regel "+regel_id+", ");
                    }

                  // }
                  svar = "utrigget";

                  break;

               case 3:     // msli1 = Intervall
                    System.out.println("inne i case1 "+ regel_id); // 2
                     brukernavnimat = brukarnavnmatrise[0][0];
                     gammeltid =brukarnavnmatrise[0][1];
                     nytid =brukarnavnmatrise[0][2];
```

```
                    dag2=konv.gettimedag(gammeltid);
                    dag1=konv.gettimedag(nytid);
                    mnd2=konv.gettimemnd(gammeltid);
                    mnd1=konv.gettimemnd(nytid);
                    aar2=konv.gettimeaar(gammeltid);
                    aar1=konv.gettimeaar(nytid);

                    diff
=regelcalc.check_login(dag1,dag2,aar1,aar2,mnd1,mnd2);
                    System.out.println(" alle sjekkete users og derira
diff: "+diff+", "+ brukernavnimat );

                    if ((tr_verdi1 < diff) & (tr_verdi2 > diff)) {

                      svarmatrise[i][0]= regel_id;
                      svarmatrise[i][1]= brukernavnimat;
                      svarmatrise[i][2]= regelmatrise[i][4];
                      svarmatrise[i][3]= "1";
                      post_tell= post_tell +1;

                      svar = "trigget"  ;
                      System.out.println("Bruker : " + brukernavnimat
+" har trigget regel "+regel_id+", ");
                    }

                  // }
                  svar = "utrigget";
                  break;

              case 4:    // viss det er ein feil i systemet msli1 er
ikkje definert.
                  y=0;
                  svar = null;
                  System.out.println("null verdi inn i msli1 variablen,
feil i regel " +regel_id );

                  break;

            }
          // System.out.println(svar + brukernavn +" kven som har
trigga " );  //kommer ein gong

     }

    //dytte ut dataene i regelmatrisa til db for sjekking i
rulecoordinator
    String bruk= null;
    String rege_id= null;
    String nivaaet= null;
    String trigget = null;

    for (int b=0; b<regelstorleik; b++) {

            rege_id = svarmatrise[b][0];
            bruk = svarmatrise[b][1];
            nivaaet = svarmatrise[b][2];
            trigget = svarmatrise[b][3];
```

```
            if (bruk != null) {

              try {  // setter inn regelne og brukernavn som har
trigga mot regel

                conn_trigga = DriverManager.getConnection(url, dbuser,
dbpassword);
                Statement stat = conn_trigga.createStatement();
                query = "INSERT into trigga VALUES (NULL,'"+ rege_id +
"', '" + bruk  + "', '" + nivaaet  + "', '" + null + "','"+ null+"')";
                triggerok = stat.executeUpdate(query);
                conn_trigga.close();
              }
              catch(SQLException ex) {
                System.err.println("SQL exsept til triggermatrise: " +
ex.getMessage());
              } // slutt db kall
            }
         //
System.out.println(rege_id+","+bruk+","+nivaa+","+trigget);

    }  // ferdig for-sløyfe
   return svar;   // returnerer trigget eller null
  }
}
```

**Vedlegg A 10 Readcallback.java**

```
/**

 * Implements a callback interface for the ReadConn class

 */

public interface ReadCallback

{

/**

 * Called when there is data ready on a ReadConn connection.

 * @param str the string read by the read thread, If null, the

 *      connection closed or there was an error reading data

 */

public void dataReady(String str);
}
```

**Vedlegg A 11 ReadThread.java**

```
import java.net.*;
import java.lang.*;
import java.io.*;

/**
 * A thread dedicated to reading data from a socket connection.
 */
public class ReadThread extends Thread
{
protected Socket connectionSocket; // the socket you are reading from

protected DataInputStream inStream;     // the input stream from the socket

protected ReadCallback readCallback;

/**

 * Creates an instance of a ReadThread on a Socket and identifies the
callback

 * that will receive all data from the socket.

 *

 * @param callback the object to be notified when data is ready

 * @param connSock the socket this ReadThread will read data from

 * @exception IOException if there is an error getting an input stream

 *          for the socket

 */

public ReadThread(ReadCallback callback, Socket connSock)

throws IOException

 {
System.out.println("du er i ReadThread konstruktøren");
connectionSocket = connSock;

readCallback = callback;

inStream = new DataInputStream(connSock.getInputStream());
System.out.println("du er i ReadThread konstruktøren 2");

}
/**
 * Closes down the socket connection using the socket's close method
 */
protected void closeConnection()
```

```
{
  try {
connectionSocket.close();
} catch (Exception oops) {
}
interrupt();
                                                                        }
/**
 * Continuously reads a string from the socket and then calls
dataReady in the
 * read callback. If you want to read something other than a string,
change
 * this method and the dataReady callback to handle the appropriate
data.
*/
public void run()

{
while(!Thread.interrupted())

{
try {

          // readUTF reads in a string
System.out.println("du er i ReadThread run tråd not interupt");

String str = inStream.readUTF();
          // Notify the callback that you have a string
//System.out.println("R.T kva denne får inn : " +str);
readCallback.dataReady(str);
         System.out.println("blir readtread ferdig eller venter den
evig");
}

catch (Exception oops)

{

// Tell the callback there was an error
System.out.println("killing the thread feil som har skjedd: "+ oops);
readCallback.dataReady(null);
closeConnection();

}

}

}
}
```

**Vedlegg A 12 Regelbekreft.java**

```
//Title:        Ped agent
//Copyright:    Copyright (c) 2003
```

```
//Author:       JanEirik
//Company:      Intermedia

import java.awt.*;
import javax.swing.*;
//import com.borland.jbcl.layout.*;
import java.awt.event.*;
import javax.swing.UIManager;

public class Regelbekreft extends JDialog {
  JPanel panel1 = new JPanel();
  JButton Nyregel = new JButton();
  JButton meny = new JButton();
  JLabel jLabel1 = new JLabel();
  JButton Eksporter = new JButton();
  GridBagLayout gridBagLayout1 = new GridBagLayout();

  public Regelbekreft(Frame frame, String title, boolean modal) {
    super(frame, title, modal);
    try  {
      jbInit();
      pack();
    }
    catch(Exception ex) {
      ex.printStackTrace();
    }
  }

  public Regelbekreft() {
    this(null, "", false);
  }

  void jbInit() throws Exception {
    panel1.setLayout(gridBagLayout1);
    this.getContentPane().setBackground(Color.white);
    Nyregel.setText("Lag ny regel");
    Nyregel.addActionListener(new java.awt.event.ActionListener() {

      public void actionPerformed(ActionEvent e) {
        Nyregel_actionPerformed(e);
      }
    });
    meny.setText("Gå til meny");
    meny.addActionListener(new java.awt.event.ActionListener() {


      public void actionPerformed(ActionEvent e) {
        meny_actionPerformed(e);
      }
    });

    jLabel1.setFont(new java.awt.Font("Dialog", 0, 24));
    jLabel1.setText("Ny regel lagret");
    Eksporter.setText("Eksporter Regler");
    Eksporter.addActionListener(new java.awt.event.ActionListener() {

      public void actionPerformed(ActionEvent e) {
```

```
        Eksporter_actionPerformed(e);
      }
    });
    panel1.setBackground(Color.white);
    getContentPane().add(panel1);
    panel1.add(jLabel1, new GridBagConstraints(0, 0, 2, 1, 0.0, 0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new
Insets(53, 104, 0, 0), 8, 0));
    panel1.add(Nyregel, new GridBagConstraints(1, 1, 1, 1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(61, 12, 224, 0), 25, 0));
    panel1.add(Eksporter, new GridBagConstraints(0, 1, 1, 1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(61, 2, 224, 0), 0, 0));
    panel1.add(meny, new GridBagConstraints(2, 1, 1, 1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(59, 22, 224, 2), 11, 0));
  }

  void meny_actionPerformed(ActionEvent e) {
    boolean packFrame = false;


  //Construct the application

    agenframe frame = new agenframe();

    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from
their layout
    if (packFrame)
      frame.pack();
    else
      frame.validate();
    frame.setVisible(true);
    this.dispose();
  }

  void Nyregel_actionPerformed(ActionEvent e) {
   //kaller neste GUI (Regel generatoren)
      boolean packFrame = false;
    Rules_innsett frame = new Rules_innsett();

  if (packFrame)
    frame.pack();
  else
    frame.validate();
    frame.setVisible(true);
  this.dispose();

  }
    //skal skrive ut elementene fra databasen til ei fil
  void Eksporter_actionPerformed(ActionEvent e) {

  }

protected void processWindowEvent(WindowEvent e) {
```

```
    super.processWindowEvent(e);
    if(e.getID() == WindowEvent.WINDOW_CLOSING) {
      System.exit(0);
    }
  }
}
```

**Vedlegg A 13 Ruleeditor.java**

```
//Title:        Ped agent
//Version:
//Copyright:    Copyright (c) 2003
//Author:       JanEirik
//Company:      Intermedia
//Description:  Your description

//tar seg av innputen frå admin GUI og sjekker dataene og laster dei
inn i datbasen

import java.io.*;
import java.sql.*;
import java.lang.*;

public class Rule_editor {
   String  str1, str2, str3, str4, str5, str8, str9, str10, str11,
str12, str13  = new String();
    private String query = new String();
   int str1int, str2int, str3int, str4int,str5int, str8int, str9int,
regel;

  public Rule_editor( String str1, String str2, String str3, String
str4, String str5, String str8, String str9, String str10, String
str11, String str12, String str13) throws IOException{
  System.out.println("str1 : "+str1);
  System.out.println("str2 : "+str2);
  System.out.println("str3 : "+str3);
  System.out.println("str4 : "+str4);
  System.out.println("str5 : "+str5);
  System.out.println("str8 : "+str8);
  System.out.println("str9 : "+str9);
  System.out.println("str10 : "+str10);
  System.out.println("str11 : "+str11);
  System.out.println("str12 : "+str12);
  System.out.println("str13 : "+str13);


  //sql dytting

    String url = "jdbc:mysql://localhost/agent_db";
    Connection con;
     try {
          Class.forName("org.gjt.mm.mysql.Driver");
     }
```

```
     catch(java.lang.ClassNotFoundException e) {
          System.err.print("ClassNotFoundException: ");
          System.err.println(e.getMessage());
     }


     try  {
          con = DriverManager.getConnection(url, "jankee", "kirsten");
          System.out.println("kontakt ok_sist");
          //con.close();
          System.out.println("går inn i sql strukturen i
rules_editor");



          Statement stat = con.createStatement();
         query ="INSERT into regel VALUES (NULL,'"+ str10 + "', '" +
str1 + "', '" + str2 + "', '" + str3 + "', '" + str4 + "', '" + str12
+ "', '" + str11 +"', '" + str5 + "', '" + str8 + "', '" + str9 +"',
'" + str11 + "')";

          regel = stat.executeUpdate(query);
          System.out.println("går ut av sql struktur opplasting
fullført" + regel);
          con.close();
     }
     catch(SQLException ex) {
     System.err.println("SQL exsept2: " + ex.getMessage());
     }

  }

}
```

**Vedlegg A 14 RuleCoordinatoor.java**

```
//Title:        Ped agent
//Version:
//Copyright:    Copyright (c) 2003
//Author:       JanEirik
//Company:      Intermedia
//Description:  Your description

//lager eit objekt av regelsettet , ut fra databasen


import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
//import com.borland.jbcl.layout.*;
```

```
import java.io.*;
import java.sql.*;
import java.sql.Date;
import javax.swing.UIManager;
import java.util.*;
import java.lang.*;

//import com.borland.jbcl.control.*;


public class Rulecoordinator {
 loginAgent L_agent = new loginAgent();
 DbQueryCount post_teller = new  DbQueryCount();
 DbQuery db_call = new DbQuery();
 int trigget_regel[] = new int[3];
 Connection con3 = null;
  Connection conn = null;
  Connection con = null;
  Connection con4 = null;
  Connection conn_trigga = null;
  String driver = "org.gjt.mm.mysql.Driver";
  String url = "jdbc:mysql://localhost/agent_db";
  String dbuser = "jankee";
  String dbpassword = "kirsten";
  int i =0;
  int mellomid =0;
  String tabell2 = "login";


  public void Rulecoordinator() {
  //tom konstruktør
  }


 /* public String regelsett_poster(String param) {
    // tester av antall poster
    String svar_post = new String();
    return  svar_post;
  }*/



  public String regelsett_innlogging(String name) {    // denne som
blir prototypen på regeltrigging og sjekking
  //tar seg av total innlogging og sist innlogging
    System.out.println("Skal kjøres kvar gong treaden våkner");
     String query = null;
    //init del av login sjekken
    String brukernavn = name;
    String svar = new String();
    String svar_innlogging = new String();
    int storleik = 0;
    int id, regelid, nivaa;

    // dbvariabler
    String tabell = "trigga";
    String db_tab = "user";
```

```
    String db_verdi = brukernavn;

    // jobb delene av login-sjekk
    storleik = post_teller.dbCount3(tabell, db_tab, db_verdi);    //
finn antall poster som er trigga

     if (storleik>0) {
      // sjekk gamle poster
        int triggematrise[][]= new int [storleik][4];
        trigget_regel[2]= -1;
        try {  // Henter dei aktuelle verdiane for å generee ein regel

          conn = DriverManager.getConnection(url, dbuser, dbpassword);
          Statement stat = conn.createStatement();
          query = "select id,regelnr, nivaa from trigga where user =
'"+brukernavn+"'";
          ResultSet variabel = stat.executeQuery(query);
          i=0;
          while (variabel.next()) {
             id = variabel.getInt("id");
             regelid = variabel.getInt("regelnr");
              nivaa= variabel.getInt("nivaa");


             triggematrise[i][0] = id;
             triggematrise[i][1] = regelid;
             triggematrise[i][2] = nivaa;


            i++;
          }
          conn.close();
        }
        catch(SQLException ex) {
          System.err.println("SQL exsept_i hent trigga regeler:" +
ex.getMessage());
        } // slutt db kall trigger tabell

        for (int j=0; j<storleik; j++) {

            if  (triggematrise[j][2]> trigget_regel[2]) {
               trigget_regel[0]  =   triggematrise[j][0];
               trigget_regel[1]  =   triggematrise[j][1];
               trigget_regel[2] =   triggematrise[j][2];
            }
        }
       // dbslett
       mellomid=trigget_regel[0];
       int mellomregelid = trigget_regel[1];
       db_call.db_post_slett(tabell,mellomid);
       //db_call.db_nullstill_time2(tabell2,brukernavn);
       svar_innlogging =db_call.findFrase(mellomregelid);
     }


    // viss tabell er tom for poster med aktuelt brukernavn
     else if (storleik ==0) {
```

```
        svar= L_agent.login(brukernavn);
        if (svar == "trigget")  {
          // hente data , behandle og slette utførte operasjoner frå
trigga
          // hent alle finn høgaste pri slett denne frå trigga og
utfør operasjonen
          storleik = post_teller.dbCount3(tabell, db_tab, db_verdi);
          int triggematrise[][]= new int [storleik][4];
        trigget_regel[2]= -1;
        try {  // Henter dei aktuelle verdiane for å generee ein regel

          conn = DriverManager.getConnection(url, dbuser, dbpassword);
          Statement stat = conn.createStatement();
          query = "select id,regelnr, nivaa from trigga where user
='"+brukernavn+"'";
          ResultSet variabel = stat.executeQuery(query);
          i=0;
          while (variabel.next()) {
             id = variabel.getInt("id");
             regelid = variabel.getInt("regelnr");
              nivaa= variabel.getInt("nivaa");


             triggematrise[i][0] = id;
             triggematrise[i][1] = regelid;
             triggematrise[i][2] = nivaa;


            i++;
          }
          conn.close();
        }
        catch(SQLException ex) {
          System.err.println("SQL exsept_i hent trigga regeler:" +
ex.getMessage());
        } // slutt db kall trigger tabell

        for (int j=0; j<storleik; j++) {

            if  (triggematrise[j][2]> trigget_regel[2]) {
               trigget_regel[0]  =   triggematrise[j][0];
               trigget_regel[1]  =   triggematrise[j][1];
               trigget_regel[2] =   triggematrise[j][2];
            }
        }
       // dbslett
       mellomid=trigget_regel[0];
       int mellomregelid = trigget_regel[1];
       //db_call.db_post_slett(tabell,mellomid);

       svar_innlogging =db_call.findFrase(mellomregelid);



        }
```

```
        else if ( svar == null) {
            svar_innlogging = "ingen";
        }

    }
    return  svar_innlogging;
  }



 /* public String regelsett_problem(String  param) {
        //Tester typer av poster (kan lage fleire versjoner eller ein
generell
    String svar_problem = new String();
    return svar_problem;
  }*/

}
```

**Vedlegg A 15 Rule_edit.java**

```
//Title:        Ped agent
//Copyright:    Copyright (c) 2003
//Author:       JanEirik
//Company:      Intermedia
//rein input fra brukere  parser infoen til rule_editor

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import javax.swing.UIManager;
import java.sql.*;
import com.borland.jbcl.layout.*;

public class Rules_edit extends JFrame  {
  JLabel jLabel1 = new JLabel();
  Choice choice1 = new Choice();
  JTextField jTextField1 = new JTextField(null);
  Choice choice2 = new Choice();
  TextArea textArea1 = new TextArea("Several notes have been posted
since you were last logged in. Please make an effort to answering some
of them",300,63,3);
  Button Avbryt = new Button();
  Button Neste = new Button();
  JLabel jLabel4 = new JLabel();
  JLabel jLabel8 = new JLabel();
  JTextField jTextField2 = new JTextField(null);
  JTextField jTextField3 = new JTextField(null);
  JLabel jLabel2 = new JLabel();
  Choice choice3 = new Choice();
```

```
  Choice choice4 = new Choice();
  JTextField jTextField4 = new JTextField(null);
  //ImageControl imageControl1 = new ImageControl();
  CheckboxGroup cbg1 = new CheckboxGroup();
  Checkbox checkbox1 = new Checkbox("Absolute value", cbg1, false);
  Checkbox checkbox2 = new Checkbox("Average", cbg1, false);
  JLabel jLabel3 = new JLabel();
  XYLayout xYLayout1 = new XYLayout();   //eng
  //Checkbox checkbox1 = new Checkbox("Absolutt verdi", cbg1, false);
  //Checkbox checkbox2 = new Checkbox("Gjennomsnitt", cbg1, false);
  //Image flea = new Image;

  public Rules_edit(){
    try  {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }

  public static void main(String[] args) throws IOException{
    Rules_innsett rules_innsett1 = new Rules_innsett();
  }
```

**Vedlegg A 16 Rules_innsett.java**

```
//Title:        Ped agent
//Version:
//Copyright:    Copyright (c) 2003
//Author:       JanEirik
//Company:      Intermedia
//Description:  Your description

//rein input fra brukere  parser infoen til rule_editor

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
//import com.borland.jbcl.layout.*;
import java.io.*;
import javax.swing.UIManager;
import java.sql.*;
import com.borland.jbcl.layout.*;
//import com.borland.jbcl.control.*;

public class Rules_innsett extends JFrame  {
    JLabel jLabel1 = new JLabel();
    Choice choice1 = new Choice();
    JTextField jTextField1 = new JTextField(null);
    Choice choice2 = new Choice();
    TextArea textArea1 = new TextArea();
```

```
    Button Avbryt = new Button();
    Button Neste = new Button();
    Label label1 = new Label();
    JLabel jLabel4 = new JLabel();
    JLabel jLabel8 = new JLabel();
    JTextField jTextField2 = new JTextField(null);
    JTextField jTextField3 = new JTextField(null);
    JLabel jLabel2 = new JLabel();
    Choice choice3 = new Choice();
    Choice choice4 = new Choice();
    JTextField jTextField4 = new JTextField(null);
    JTextArea jTextArea1 = new JTextArea();
    CheckboxGroup cbg1 = new CheckboxGroup();
    Checkbox checkbox1 = new Checkbox("Absolutt verdi", cbg1, false);
    Checkbox checkbox2 = new Checkbox("Gjennomsnitt", cbg1, false);
    XYLayout xYLayout1 = new XYLayout();
    unfilled_stop stopper = new unfilled_stop();
    String str1, str2, str3, str4, str5, str6, str7, str8, str9,
str10, str11, str12, str13, overgang = new String();



    public UFill_popup frame;

    public Rules_innsett(){
      try  {
        jbInit();
      }
      catch(Exception e) {
        e.printStackTrace();
      }
    }

    public static void main(String[] args) throws IOException{
      Rules_innsett rules_innsett1 = new Rules_innsett();

    }

    private void jbInit() throws IOException{


      jLabel1.setText("Database variabel");
      this.setTitle("Regel Generator");
      this.setSize(new Dimension(739, 480));
      this.getContentPane().setLayout(xYLayout1);
      this.getContentPane().setBackground(Color.white);
      jLabel4.setText("Verdier agenten skal overse");
      label1.setText("Regelens Triggernivå");
      jTextField1.setText(null);
      jTextField2.setText(null);
      jTextField3.setText(null);
      jTextField4.setText(null);
      textArea1.setText(null);
      String databaseverdi= new String();
      Avbryt.setName("Avbryt");
      Avbryt.setLabel("Avbryt");
      Avbryt.addActionListener(new java.awt.event.ActionListener() {
```

```
      public void actionPerformed(ActionEvent e) {
        Avbryt_actionPerformed(e);
      }

    });

    Neste.setName("Neste");
    Neste.setLabel("Neste");
    Neste.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(ActionEvent e)  {
      try {
         Neste_actionPerformed(e);
      }
      catch (IOException a) {
                  System.err.print("Something went wrong in messages..");
                                                                        }
    }

  });

    //liten hardkoding for choisemenyene
    //her skal eigentleg eit database kall vere og så skal det bli
lagt inn i choice menyen
     //vil nok også vere ei if eller while sløyfe
     jLabel8.setText("Tekstfrase til regelen");
    jLabel2.setText("Verdier agenten trigger på");
    jTextArea1.setText("Veiledning til å lage agentregler");


    choice1.add("Velg variabel");

    //sql kommando til henting
   String url = "jdbc:mysql://localhost/agent_db";

    Connection con;
     try {
          Class.forName("org.gjt.mm.mysql.Driver");
     }

     catch(java.lang.ClassNotFoundException e) {
          System.err.print("ClassNotFoundException: ");
          System.err.println(e.getMessage());
     }

     try  {
          con = DriverManager.getConnection(url, "jankee", "kirsten");
          System.out.println("kontakt ok_i rules_insett");
          //con.close();

          Statement stat = con.createStatement();
          ResultSet variabel = stat.executeQuery("select navn from
variabel");
          while (variabel.next()) {
             databaseverdi = variabel.getString("navn");
            choice1.add(databaseverdi);
```

```
        }
        con.close();
    }
    catch(SQLException ex) {
    System.err.println("SQL exsept2: " + ex.getMessage());
    }

    // innsetting av kategoritype
     this.choice2.add("Velg regelens Viktighets nivå");
     this.choice2.add("Opplysninger");
     this.choice2.add("Mindre viktig");
     this.choice2.add("Viktig");
     this.choice2.add("Svært viktig");

     //valg fra verdi
     this.choice3.add("Velg");
     this.choice3.add("Lik");
     this.choice3.add("Større");
     this.choice3.add("Mindre");
     this.choice3.add("Intervall");

     this.choice4.add("Velg");
     this.choice4.add("Lik");
     this.choice4.add("Større");
     this.choice4.add("Mindre");
     this.choice4.add("Intervall");


    this.getContentPane().add(textArea1, new XYConstraints(67, 344,
623, 62));
    this.getContentPane().add(jLabel8, new XYConstraints(12, 308, -1,
-1));
    this.getContentPane().add(jLabel1, new XYConstraints(28, 24, -1, -
1));
    this.getContentPane().add(jTextArea1, new XYConstraints(32, 143,
571, 58));
    this.getContentPane().add(label1, new XYConstraints(4, 217, -1, -
1));
    this.getContentPane().add(choice2, new XYConstraints(166, 217,
171, -1));
    this.getContentPane().add(jLabel4, new XYConstraints(512, 24, -1,
-1));
    this.getContentPane().add(Avbryt, new XYConstraints(502, 410, 76,
33));
    this.getContentPane().add(Neste, new XYConstraints(629, 409, 69,
32));
    this.getContentPane().add(jTextField2, new XYConstraints(612, 63,
38, 22));
    this.getContentPane().add(jTextField4, new XYConstraints(670, 63,
42, -1));
    this.getContentPane().add(choice4, new XYConstraints(510, 64, 90,
-1));
    this.getContentPane().add(jTextField3, new XYConstraints(429, 64,
50, -1));
    this.getContentPane().add(jTextField1, new XYConstraints(372, 64,
47, -1));
```

```
    this.getContentPane().add(choice3, new XYConstraints(290, 63, 70,
-1));
    this.getContentPane().add(checkbox1, new XYConstraints(157, 33, -
1, -1));
    this.getContentPane().add(checkbox2, new XYConstraints(154, 62, -
1, -1));
    this.getContentPane().add(choice1, new XYConstraints(22, 57, 100,
-1));
    this.getContentPane().add(jLabel2, new XYConstraints(292, 28, -1,
-1));


    // comp2D.drawImage();

    //get this.getContent.Pane.choice1 = sql kommando retrive
databasefelt X
   }

   public void paintComponent(Graphics comp) {

    Toolkit kit = Toolkit.getDefaultToolkit();
    Image flea = kit.getImage("liten-bug.gif");

    Graphics2D comp2D = (Graphics2D) comp;
    comp2D.drawImage(flea, 400,200, this);

   }


  //Overridden so we can exit on System Close
  protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if(e.getID() == WindowEvent.WINDOW_CLOSING) {
      System.exit(0);
    }



    }
      //avbryt går til hovudmeny agentoppsett
  void Avbryt_actionPerformed(ActionEvent e) {
    boolean packFrame = false;
    agenframe frame = new agenframe();

    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from
their layout
    if (packFrame)
      frame.pack();
    else
      frame.validate();
    frame.setVisible(true);

    this.dispose();

  }
   //går vidare til lagring av regel
```

```
  void Neste_actionPerformed(ActionEvent e) throws IOException{

    boolean packFrame = false;

   // int str5;
    //boolean str5 =false;
    boolean failed = false;
    boolean stat1 = false;
    boolean stat2 = false;
    boolean ok = false;
    String oks = null;
        str5 = null;
        str1 = this.jTextField1.getText();
        str2 = this.jTextField2.getText();
        str3 = this.jTextField3.getText();
        str4 = this.jTextField4.getText();
        str8 = this.choice3.getSelectedItem();
        str9 = this.choice4.getSelectedItem();
        str10 = this.choice1.getSelectedItem();
        str11 = this.choice2.getSelectedItem();
        str12 = this.textArea1.getText();
        str13 = "0";
        stat1 = this.checkbox1.getState();
        stat2 = this.checkbox2.getState();
        System.out.println(stat1+","+ stat2);
        if (stat1 = true)  {
        str5 = "1";
        }
        else if (stat2 = true) {
        str5 = "2";
        }
      oks = stopper.behandle_uutfylt(str1, str2, str3,str4, str5,
str8, str9, str10, str11, str12, str13);
     // System.out.println(oks +"oks variabel");
      if (oks == null) {

        //try {
         //Rule_editor regelinn = new Rule_editor(str1, str2,
str3,str4, str5, str8, str9, str10, str11, str12, str13);

         Regelbekreft frame = new Regelbekreft();
         //Validate frames that have preset sizes
         //Pack frames that have useful preferred size info, e.g. from
their layout
         if (packFrame) {
          frame.pack();
         }
         else {
          frame.validate();
          frame.setVisible(true);
          this.dispose();
         }

        //}
       // catch (IOException a) {
          // System.err.print("Something went wrong in rule_innesett..");
       // }
```

```
      }
      else {
      // System.out.println(oks);
       firePopUp(oks);

      }
 }

 public void firePopUp(String melding) {
   boolean packFrame = false;
   //System.out.println(melding);
   frame = new UFill_popup(melding);


   if (packFrame)
      frame.pack();
    else
      frame.validate();
    frame.setVisible(true);

// fires pop up box.. which closes by itself..
//String t = melding;

 }
}
```

**Vedlegg A 17 ServerConn.java**

```
import java.io.*;
import java.net.*;

/**
 * This class represents a server's client. It handles all the
 * communications with the client. When the server gets a new
 * connection, it creates one of these objects, passing it the
 * Socket object of the new client. When the client's connection
 * closes, this object goes away quietly. The server doesn't actually
 * have a reference to this object.
 * Just for example's sake, when you write a server using a setup
 * like this, you will probably have methods in the server that
 * this object will call. This object keeps a reference to the server
 * and calls a method in the server to process the strings read from
 * the client and returns a string to send back.
 */

public class ServerConn extends Object implements ReadCallback

{
protected SimpleServer server;

protected Socket clientSock;

protected ReadThread reader;
```

```
// protected AltAgentClient reader;

protected DataOutputStream outStream;
protected InetAddress me;

 public ServerConn(SimpleServer server, Socket clientSock) throws IOException

{
      System.out.println("har kommet inn i serverConn");
 this.server = server;

  this.clientSock = clientSock;
  me = clientSock.getInetAddress();
  System.out.println("InetAddress :" + me);

  outStream = new DataOutputStream(clientSock.getOutputStream());

  reader = new ReadThread(this, clientSock);

 reader.start();

 }
/**
 * This method received the string read from the client, calls
 * a method in the server to process the string, and sends back
 * the string returned by the server.
 */
public synchronized void dataReady(String str)

 {
 if (str == null)
{
System.out.println("er i dataready");
disconnect();

return;

}

  try {
   System.out.println("didn't kill");
      outStream.writeUTF(server.processString(str, me));
   }

  catch (Exception writeError) {

writeError.printStackTrace();

   disconnect();

   return;
 }
 }
/**
 * This method closes the connection to the client. If there is an
error
```

```
 * closing the socket, it stops the read thread, which should
eventually
 * cause the socket to get cleaned up.
 **/

 public synchronized void disconnect()

 {

try {
         System.out.println("disconnection fra server");
   reader.closeConnection();

  } catch (Exception cantclose) {

reader.interrupt();
reader = null;

  }

 }
}
```

**Vedlegg A 18 SimpleServer.java**

```
import java.io.*;
import java.sql.*;
import java.net.*;
import java.util.Vector;


/**

 * This class implements a simple server that accepts incoming

 * socket connections and creates a ServerConn instance to handle

 * each connection. It also provides a processString method that

 * takes a string and returns the reverse of it. This method is

 * invoked by the ServerConn instances when they receive a string

 * from a client.

 */


public class SimpleServer extends Object

{
```

```
protected ServerSocket listenSock;
protected Vector liste;
BufferedReader min; // min står for mail - in
PrintWriter mout; // mout står for mail - out
Socket ms; // ms står for mail - socket
protected InetAddress clientIp;
private String t = null;
   String trigget = new String();


public SimpleServer(int listenPort)

throws IOException

 {

// Listen for connections on port listenPort

  listenSock = new ServerSocket(listenPort);

 }
 public static void main(String[] args)

 {

 try {

// Crank up the server and wait for connection

SimpleServer server = new SimpleServer(8010);
System.out.println("Opprettet nytt Server objekt");
// legge til her en metode for mail utsending til de som ikke har
postet noe på kjempe lang tid?
server.waitForClients();
         System.out.println("startet wait for clients");


} catch (Exception oops) {



// If there was an error starting the server, say so!

 System.out.println("Got error starting server:");

 oops.printStackTrace();

 }
 }

 public synchronized String processString(String inStr, InetAddress a)
throws IOException

 {
 clientIp = a;
```

```
  t = "";
   CoordinatorAgent rule = new CoordinatorAgent();    //coord agent sartat
// System.out.println("blir prosessString Starta og kva element gjer
det");
 t = rule.fName(clientIp);   //ip henta
//System.out.println("blir prosessString Starta og kva element gjer
det"+ t);

if (t == null){
t="";

}

      else  {
 trigget = rule.Coordinate(t);
//sender regelsjekk med brukernavn frå tilbake om nokon av regalne er
trigga

         }


return trigget;

 }


public void waitForClients()

 {

 while (true)

 {

  try {

// Wait for the next incoming socket connection
System.out.println("accepting new connection Server klar");
Socket newClient = listenSock.accept();


// Create a ServerConn to handle this new connection
ServerConn newConn = new ServerConn( this, newClient);
System.out.println("new ServerConn");

 } catch (Exception badAccept) {
badAccept.printStackTrace();

// print an error, but keep going

 }

}

 }
```

```
}
```

**Vedlegg A 19 Ufill_stopp.java**

```
public class unfilled_stop {
  Konverter konv = new Konverter();

  public unfilled_stop() {

  }

  public String behandle_uutfylt(String str1, String str2, String
str3,String str4, String str5, String str8, String str9, String str10,
String str11, String str12, String str13) {
    String svar = null;
    int msli_int1 =-1;
    int msli_int2 =-1;
    msli_int1 = konv.msli_tall(str8);
    msli_int2 = konv.msli_tall(str9);
    String varnavn = "velg variabel";
    String katnavn = "Velg regelsens viktigheitsnivå";
    int bad = -1;
    String test = null;
    String frasenull = "";

    if (str12.equalsIgnoreCase(frasenull) ) {
      System.out.println("problem med frase" );
      svar = "du har gløymt å sette kva frase regelen skal ha" ;
    }

    if (str11.equalsIgnoreCase(katnavn)) {
        System.out.println("problem med katnavn");
        svar = "du har gløymt å sette kva triggernivå regelen skal ha"
;
    }
    if (str1.equalsIgnoreCase(frasenull) ) {
        System.out.println("stoppet med  ikkje tall i boks 1");
          svar = " du har glemt å sette  tall i boks 1";
    }
    if (msli_int1== -1) {
          System.out.println("stoppet med msli = -1");
          svar = " du har glemt å velge om tallene skal vere absolutt
eller gjennomsnitt";

    }

    // tar dei første feilpostene
    if (str10.equalsIgnoreCase(varnavn))  {
        System.out.println("stoppet med feil varnavn");
        svar = "du har glemt å velge variabel for regelen";
    }
```

```
    return svar;
  }

}
```

**Vedlegg A 20 Vis_alle_regler.java**

```
//Title:        Ped agent
//Version:
//Copyright:    Copyright (c) 2004
//Author:       JanEirik
//Company:      Intermedia
//Description:  Your description


import java.awt.*;
import javax.swing.JFrame;
//import com.borland.jbcl.layout.*;
import java.sql.*;
import java.awt.event.*;
import java.io.*;

public class vis_alle_regeler extends JFrame {

  Choice choice1 = new Choice();
  Label label1 = new Label();
  Button button1 = new Button();
  Button button2 = new Button();
  String id_regel = new String();
  GridBagLayout gridBagLayout1 = new GridBagLayout();



  public vis_alle_regeler() {
    try  {
      jbInit();
    }

    catch(Exception e) {
          e.printStackTrace();
    }

    String db_variabel = new String();
     boolean tomt = false;
     String databaseverdi;;
     String databaseverdi2 = new String();

     choice1.add("Velg variabel");
     //sql kommando til henting
```

```
     String url = "jdbc:mysql://localhost/agent_db";
     //  String url = "jdbc:mysql://localhost/agent_db";
heimevariabel
     Connection con;
     try {
          Class.forName("org.gjt.mm.mysql.Driver");
     }

     catch(java.lang.ClassNotFoundException e) {
          System.err.print("ClassNotFoundException: ");
          System.err.println(e.getMessage());
     }

     try  {
          con = DriverManager.getConnection(url, "jankee", "kirsten");
          System.out.println("kontakt ok_i vis_alle_regeler");


          Statement stat = con.createStatement();
          ResultSet variabel = stat.executeQuery("select id, varnavn
from regel");
          System.out.println("kommer inn i sql statusen_henting");
          while (variabel.next()) {
             databaseverdi = variabel.getString("id");
             databaseverdi2 = variabel.getString("varnavn");

            choice1.add(databaseverdi +"," + databaseverdi2);
         }
         con.close();
     }
     catch(SQLException ex) {
     System.err.println("SQL exsept_i heting_til_choise: " +
ex.getMessage());
     }


 }

     //get from  get_rules_db eller Rule  med opplesing av alle regler

  private void jbInit() throws Exception {
    this.getContentPane().setBackground(Color.white);
     this.getContentPane().setLayout(gridBagLayout1);
    // this.setSize(new Dimension(300,372));
    this.setSize(new Dimension(300,372));

    label1.setFont(new java.awt.Font("Dialog", 0, 16));
    label1.setText("Velg Regel å redigere");
    button1.setName("Neste");
    button1.setLabel("Neste");
    button1.addActionListener(new java.awt.event.ActionListener() {

      public void actionPerformed(ActionEvent e) {
        Neste_actionPerformed(e);
       }
    });
```

```
    button2.setName("Avbryt");
    button2.setLabel("Avbryt");
    button2.addActionListener(new java.awt.event.ActionListener() {

      public void actionPerformed(ActionEvent e) {
        Avbryt_actionPerformed(e);
      }
    });
    this.getContentPane().add(choice1, new GridBagConstraints(0, 1, 2,
1, 1.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.HORIZONTAL,
new Insets(19, 6, 0, 56), 202, 0));
    this.getContentPane().add(label1, new GridBagConstraints(0, 0, 2,
1, 0.0, 0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new
Insets(18, 26, 0, 99), 0, 0));
    this.getContentPane().add(button2, new GridBagConstraints(0, 2, 1,
1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(122, 6, 114, 0), 50, 0));
    this.getContentPane().add(button1, new GridBagConstraints(1, 2, 1,
1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(123, 28, 114, 63), 53, 0));
  }
  void Avbryt_actionPerformed(ActionEvent e) {
  boolean packFrame = false;
    agenframe frame = new agenframe();

    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from
their layout
    if (packFrame)
      frame.pack();
    else
      frame.validate();
    frame.setVisible(true);

    this.dispose();

  }

  void Neste_actionPerformed(ActionEvent e) {
  String endring;
  char tempvar;
  String db_variabel = new String();
     boolean tomt = false;
     String id_regel = new String();
     String dbverdi1 = new String();
     String dbverdi2 = new String();
     String dbverdi3 = new String();
     String dbverdi4 = new String();
     String dbverdi5 = new String();
     String dbverdi8 = new String();
     String dbverdi9 = new String();
     String dbverdi10 = new String();
     String dbverdi11 = new String();
```

```
     String dbverdi12 = new String();
     String query = new String();
    /**
      endring = this.choice1.getSelectedItem();
     tempvar = readChar(endring);
     while (tempvar  ",")  {
       id_regel = tempvar;
       tempvar = readchar(endring);
     }**/
     id_regel ="'1'";
    // choice1.add("Velg variabel");
     //sql kommando til henting

     String url = "jdbc:mysql://localhost/agent_db";

     Connection con;
     try {
          Class.forName("org.gjt.mm.mysql.Driver");
     }
     catch(java.lang.ClassNotFoundException xe) {
          System.err.print("ClassNotFoundException: ");
          System.err.println(xe.getMessage());
     }
     try  {
          con = DriverManager.getConnection(url, "jankee", "kirsten");
          System.out.println("kontakt ok_i vis_alle_regeler_
neste.button");

          Statement stat = con.createStatement();
          query =  "select id, varnavn, verdi1_tr, verdi2_tr,
verdi3_ntr from regel where id = " + id_regel ;
          ResultSet variabel2 = stat.executeQuery(query);


         System.out.println("forbi executeQuery");

            while (variabel2.next()) {
             dbverdi1 = variabel2.getString("id");
             System.out.println("kommer forbi id");
            dbverdi2 = variabel2.getString("varnavn");
             System.out.println("kommer forbi varnavn");
             dbverdi3  = variabel2.getString("verdi1_tr");
            System.out.println("kommer forbi vedi2");
            dbverdi4  = variabel2.getString("verdi2_tr");
             System.out.println("kommer forbi verdi2");
             dbverdi5  =  variabel2.getString("verdi3_ntr");
             System.out.println("kommer forbi verdi3");
          System.out.println(dbverdi1 + dbverdi2 + dbverdi3 + dbverdi4
+ dbverdi5 );
            System.out.println("kommer forbi print");
         }
         con.close();
     }
     catch(SQLException ex) {
     System.err.println("SQL exsept_ i henting til neste frame: " +
ex.getMessage());
     }
```

```
   boolean packFrame = false;  //framevariabel
   Rules_edit frame = new Rules_edit();

    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from
their layout
    if (packFrame)
      frame.pack();
    else
      frame.validate();
    frame.setVisible(true);
    this.dispose();

   }
   protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if(e.getID() == WindowEvent.WINDOW_CLOSING) {
      System.exit(0);
    }

  }

  //public vis_alle_regeler() {

}
```

VEDLEGG B

**Vedlegg B 1 FLE.PY**

```
def redirect_to_webtop(self, REQUEST):
     """Redirect to webtop."""
     uname = str(REQUEST.AUTHENTICATED_USER)
     if hasattr(self.fle_users,uname):
         ip_address = REQUEST['REMOTE_ADDR']
         #navdal - here is the opening of the database connect
         # the module of MySQLdb has to be imported.
         import MySQLdb
         from time import localtime, strftime
         import time
         login_time   =   strftime("%H:%M:%S   %a,   %d   %b   %Y",
      localtime(time.time()))
         db_conn  =  MySQLdb.connect(host="localhost",  user="jankee",
      passwd="kirsten", db="agent_db")
          time2 = login_time
          antall = "1"
          c4 = db_conn.cursor()
          c3 = db_conn.cursor()
          c1 = db_conn.cursor()
          c = db_conn.cursor()
          c2 = db_conn.cursor()
          c2.execute("Select   time   FROM   login   Where   name   =
      ('%s')"%uname)
          result = c2.fetchmany(1)
          for record in result:
          time2 = record[0]
          #antall = antall +"1"
          #db_conn.commit()
          c3.execute("DELETE   FROM   login_no   WHERE   ip_no   =
      ('%s')"%ip_address)
          c4.execute("INSERT        INTO        login_no        VALUES
      ('%s','%s','%s')"%(uname, ip_address, login_time))
          c1.execute("DELETE FROM login WHERE name = ('%s')"%uname)
          db_conn.commit()
          c.execute("INSERT  INTO  login  VALUES  ('%s','%s','%s','%s',
      %s)"%(uname, ip_address, login_time, time2, antall))
          #c.execute("INSERT         INTO         login         VALUES
      ('%s','%s','%s')"%(uname, ip_address, login_time))
          db_conn.commit()
          c2.close()
          c1.close()
          c.close()
```

```
        db_conn.close()
        return REQUEST.RESPONSE.redirect(
        self.state_href(REQUEST, 'fle_users/' + uname + '/webtop/'))
```

**Vedlegg B 2 NOTE.PY**

```
REQUEST.RESPONSE.redirect(self.state_href(REQUEST,
                '../../index_html'))
        elif post:
            self.do_publish()
            import string
            #NB! HERE THE DATA IN FLE WILL BE RECORDED IN A DATABASE..

            import MySQLdb
            from time import localtime, strftime
            dbpath = repr(REQUEST.URL2)
            index1 = (string.rfind(dbpath, '/courses/') + 9)
            index2 = (string.rfind(dbpath,'tmp_objects') - 1)
            dbpath2 = dbpath[index1:index2]
            index1 = (string.rfind(dbpath2, '/') + 1)
            foreldreid = dbpath2[index1:index2]
            ntime    =    strftime("%H:%M:%S    %a,    %d    %b    %Y",
      localtime(time.time())) # time when note is created
            notatid = self.get_id()
            forfatter = self.get_author()
            ttiden = self.get_tt_id()
            subjekt = self.get_subject()
            db_conn = MySQLdb.connect(host="localhost", user="jankee",
      passwd="kirsten", db="agent_db")
            cursor = db_conn.cursor()
            #cursor.begin()
            fnam = "out.txt"
            file = open(fnam,"w")
            file.write(notatid)
            file.write(forfatter)
            file.write(ttiden)
            file.write(subjekt)
            file.write(ntime)
            file.write(foreldreid)
            file.close()
            cursor.execute("INSERT              INTO              note
      VALUES(%s,'%s','%s','%s','%s',%s)"%(notatid,  forfatter,  ttiden,
      subjekt, ntime, foreldreid))
            #db_conn.commit() # releasing the locks
            cursor.close()
```

```
    db_conn.close()
    REQUEST.RESPONSE.redirect(self.state_href(REQUEST,
        # Ugly hack.
        '../../%s/index_html'%self.get_id()))
else:
    raise 'FLE Error', 'This should not happen.'
```