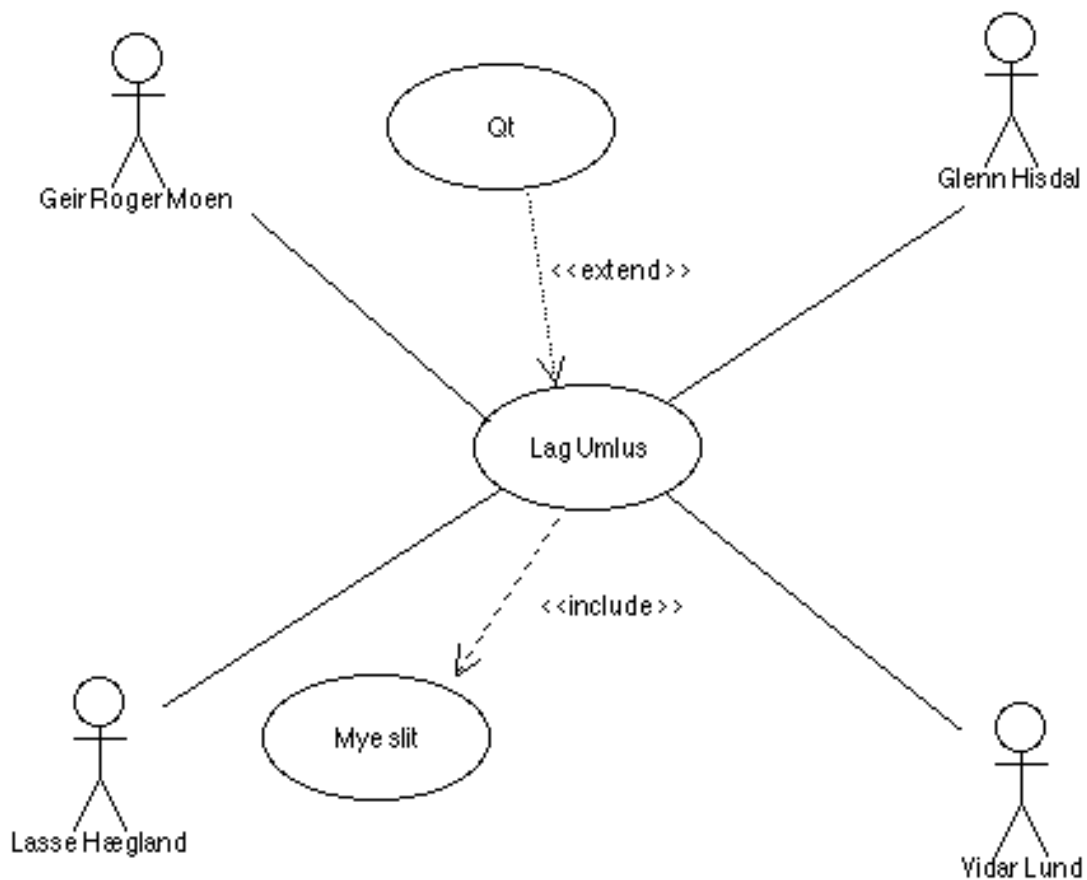


# UMLUS

Program for tegning av UML diagrammer



Hovedprosjekt ved Høgskolen i Bergen avdeling for  
ingeniørutdanning linje for data studieretning datateknikk  
for Glenn Hisdal Lasse Hægland Vidar Lund Geir Roger Moen  
Våren

## FORORD

Dette er en rapport for et hovedprosjekt ved Høgskolen i Bergen våren . Rapporten henvender seg til kontaktperson og andre lærere ved HiB samt sensor for hovedprosjektet

Vår gruppe er større enn en vanlig gruppe på hovedprosjektene dette fordi en på gruppen hadde søkt om hovedprosjekt på CERN og ikke ville binde seg til en annen gruppe. Alle fire var også interessert i en av oppgavene som skulle programmeres med Qt og vi var derfor en naturlig, dog litt stor, enhet.

Pga at vi var fire valgte vi å ta to hovedprosjekt: Sortering med QuickSort og dette. Et stykke ut i QuickSort-prosjektet viste det seg at en del av forutsetningene for oppgaven allerede var løst. Vi hadde da to valg: Omdefinere oppgaven eller legge prosjektet på is. Da vi hadde et annet prosjekt å falle tilbake på som var mer enn krevende nok, valgte vi det siste og kastet alle ressurser inn i jobbingen med UMLUS.

Ideen til dette hovedprosjektet kom fra Lasse Hægland mens Glenn Hisdal tidlig kom med forslag til navn: UMLUS som ble valgt med akklamasjon og ble programmets offisielle navn.

Prosjektet har en godt oppdatert webside:  
[http://solo.hib.no/ing\\_ghi/hovedprosjekt/uml/forside.php](http://solo.hib.no/ing_ghi/hovedprosjekt/uml/forside.php)

Gruppen ønsker å takke kontaktperson Carsten Helgesen for all veiledning i UMLs mystiske verden og Sven Olai Høyland for fleksibilitet når det gjaldt alle våre merkelige innfall og for alltid å være tilgjengelig for spørsmål.

Alle eksempler på UML i denne rapporten er selvfølgelig tegnet med UMLUS.



# HØGSKOLEN I BERGEN

Avdeling for Ingeniørutdanning

## TITTELSIDE FOR HOVEDPROSJEKT

<i>Rapportens tittel:</i> UMLUS - Program for tegning av UML-diagrammer	<i>Dato:</i> 17.06.2002
	<i>Rapportnummer:</i>
<i>Forfatter(e):</i> Glenn Hisdal, Lasse Hægland, Vidar Lund & Geir Roger Moen	<i>Antall sider u/vedlegg:</i> 34 (37)
	<i>Antall sider vedlegg:</i> 0
<i>Studieretning:</i> Datateknikk	<i>Antall disketter:</i> 1 CD
<i>Kontaktperson ved studieretning:</i> Carsten Helgesen	<i>Gradering:</i> Ingen
<i>Merknader:</i>	

<i>Oppdragsgiver:</i> Høgskolen i Bergen	<i>Oppdragsgivers referanse:</i>
<i>Oppdragsgivers kontaktperson:</i> Carsten Helgesen	<i>Telefon:</i>

<i>Sammendrag:</i> <p>UMLUS er et program for tegning, lagring og printing av UML-diagrammer.</p> <p>Programmet er programmert i C++ og benytter Trolltechs Qt-bibliotek for grafiske elementer. Dette gjør at kildekoden, problemfritt, kan kompiles både under Windows og Linux (og under Mac OS X for den saks skyld). De lagrede .uml-filene er lagret i XML-format, et relativt greit, lesbart tekstformat.</p> <p>Gruppen trengte et versjonskontrollsystem for å samordne arbeidet. Valget falt på CVS som ligger ved de fleste Linux-versjoner og i tillegg enkelt kan settes opp under Windows.</p> <p>Programmet er spesielt tilpasset faget LOD055 System- og programutviklingsmetoder, men kan selvfølgelig også brukes til å modellere andre større eller mindre programmeringsprosjekter.</p>
---

### Stikkord:

UML	Qt	CVS
-----	----	-----

Høgskolen i Bergen, Avdeling for ingeniørutdanning

Postadresse: Postboks 7030, 5020 BERGEN

Tlf. 55 58 75 00

Fax 55 58 77 90

Besøksadresse: Nygårdsgaten 112, Bergen

E-post: [post@hib.no](mailto:post@hib.no)

Hjemmeside: <http://www.hib.no>

# INNHOLDSFORTEGNELSE:

## INNHOLDSFORTEGNELSE:

### INNLEDNING

- ORGANISERING AV RAPPORTEN
- OPPDRAKSGIVER: HiB
- PROBLEMSTILLING
- UML UNIFIED MODELING LANGUAGE
- LØSNINGSFORSLAG

### KRAVSPESIFIKASJON

#### ANALYSE AV PROBLEMET

- AVGRENSING
- ORGANISERING

#### DESIGN

- PROGRAMMERINGSSPRÅK
- LAGRINGSFORMAT
- DESIGN

#### VERKTØY

- QT
- CVS (CONCURRENT VERSION SYSTEM)

#### IMPLEMENTERING

#### TESTING

- KJENTE FEIL ("KNOWN BUGS")
- MULIGE FREMTIDIGE PROBLEMER

#### BESKRIVELSE AV UTVIKLET SYSTEM I BRUK

#### EVALUERING AV PROSJEKTARBEIDET

##### EVALUERING AV PROGRAMMET

#### LITTERATUR

#### VEDLEGG:

#### STIKKORDREGISTER

#### APPENDIX A INNHOLD PÅ VEDLAGT CD

- WINDOWS
- LINUX (X )
- LINUX (PPC)
- UML FILER
- CVS
- DOC
- KILDEKODE

#### APPENDIX B SAMMENDRAG CVS LOGG

#### APPENDIX C GRUNNLEGGENDE OPPSETT OG BRUK AV CVS

- OPPSETT AV CVS OMRÅDE
- LEGG INN NYE PROSJEKTER
- CVSROOT MODULEN
- BRUK AV CVS
- OPPSETT AV CVS PÅ WINDOWS

#### APPENDIX D SKJERMBILDER

#### APPENDIX E VEIEN VIDERE

- VERSJON PRIMÆRT TING VI PLUKKET VEKK AV TIDSNØD
- VERSJON FORSLAG TIL UTVIDELSER
- SENERE VERSJONER?

# **1 INNLEDNING**

## **1.1 ORGANISERING AV RAPPORTEN**

### **1.1.1 Innledning**

Litt om rapporten prosjektet og oppdragsgiveren HiB

### **1.1.2 Kravspesifikasjon**

Hva skal programmet kunne gjøre?

### **1.1.3 Analyse av problemet**

Hvordan skal vi løse problemet og hvordan skal vi organisere oss?

### **1.1.4 Design**

Hvordan skulle programmet implementeres

### **1.1.5 Verktøy**

En kort innføring i Qt og CVS

### **1.1.6 Implementering**

Koden vår

### **1.1.7 Testing**

Hvordan testet vi ut programmet

### **1.1.8 Systemet i bruk**

Hvordan skal programmet startes og brukes?

### **1.1.9 Evaluering av prosjektarbeidet**

Hvordan gikk det? Hva har vi lært?

### **1.1.10 Evaluering av programmet**

Virker det ferdige programmet etter spesifikasjonene?

### **1.1.11 Litteraturliste**

Oppslagsverk vi har brukt for å kunne gjennomføre prosjektet

### **1.1.12 Vedlegg**

Kort liste av vedlegg

### **1.1.13 Stikkordregister**

For enklere å finne nøkkelord

### **1.1.14 Appendix A - Innholdsfortegnelse for vedlagt CD**

En enkel oversikt over hva som finnes på CDen

### **1.1.15 Appendix B - CVS logg**

En illustrasjon på hvor nyttig CVS var i utførelsen av prosjektet

### **1.1.16 Appendix C - Grunnleggende oppsett og bruk av CVS**

Hvordan setter man opp og bruker CVS

### **1.1.17 Appendix D - Skjermbilder**

Slik kan det se ut når du kjører UMLUS

### **1.1.18 Appendix E - Veien videre**

Nødvendige og mulige veier videre for UMLUS

## **1.2 OPPDRAGSGIVER: HiB**

Høgskolen i Bergen har            studenter og            ansatte

Består av    avdelinger

- Ingeniørutdanningen
- Lærerutdanningen
- Helse og sosialfag

Høgskolen ble etablert i            etter at    høgskoler ble slått sammen til en HiB er lokalisert    steder i Bergen

Avdeling Nygård er det som tidligere var Bergen Ingeniørhøgskole

Datautdanningen har            studenter hvorav    går i tredje og siste klasse

## **1.3 PROBLEMSTILLING**

Da vi hadde faget LOD    System og programutviklingsmetoder hadde vi innleveringer hvor vi skulle tegne UML diagrammer Til dette formålet finnes det en rekke verktøy feks ArgoUML Dia og Poseidon Disse viste seg da å havne i en av to kategorier: For dårlige eller for dyre Mange forkastet disse verktøyene og tegnet heller diagrammene i MS Word eller i verste fall for hånd

Idéen om å lage et eget program for å tegne disse diagrammene ble født den våren

Et annet problem var at de fleste programmer kun fantes for en plattform: Windows eller Linux (Unntaket er de som kjørte på Java men Javastøtten på maskinene på skolen kunne i beste fall kalles tvilsom ) Hvis vi laget et eget program måtte dette ihvertfall være multiplattform

### **1.3.1 LOD055 System- og programutviklingsmetoder**

Dette faget skal lære studentene sentrale metoder som nyttes ved analyse spesifisering design og testing av datasystemer Faget gir en innføring i det objektorienterte modelleringspråket UML En større prosjektoppgave gir trening i å skrive prosjekt og programdokumentasjon

## 1.4 UML - UNIFIED MODELING LANGUAGE

UML er et modelleringsverktøy som gir deg muligheten til å spesifisere visualisere og dokumentere programmeringsprosjekter og deres struktur og design. Dette gjør det enklere å sikre at programvaren som utvikles har god arkitektur allerede før første koden er tastet inn.

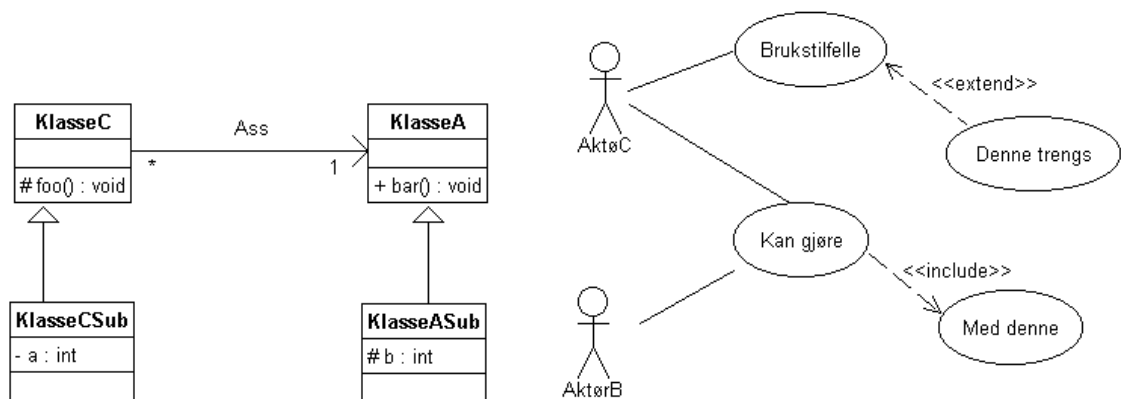
UML opererer med standard diagrammer fordelt i tre typer:

Strukturdiagrammer som inkluderer **Klasse**, **Objekt**, **Komponent** og **Deployment**diagram (har ikke funnet en god norsk oversettelse av sistnevnte).

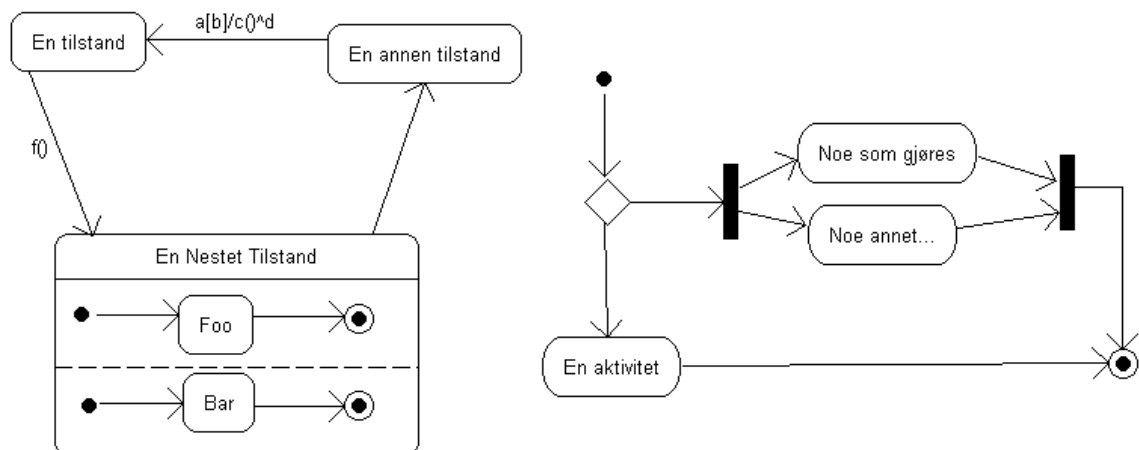
Oppførselsdiagrammer som inkluderer **UseCase**, **Sekvens**, **Samarbeid**, **Aktivitets** og **Tilstands**diagram.

Modelldiagrammer inkluderer **Pakkediagram**, **Subsystemer** og **Modeller**.

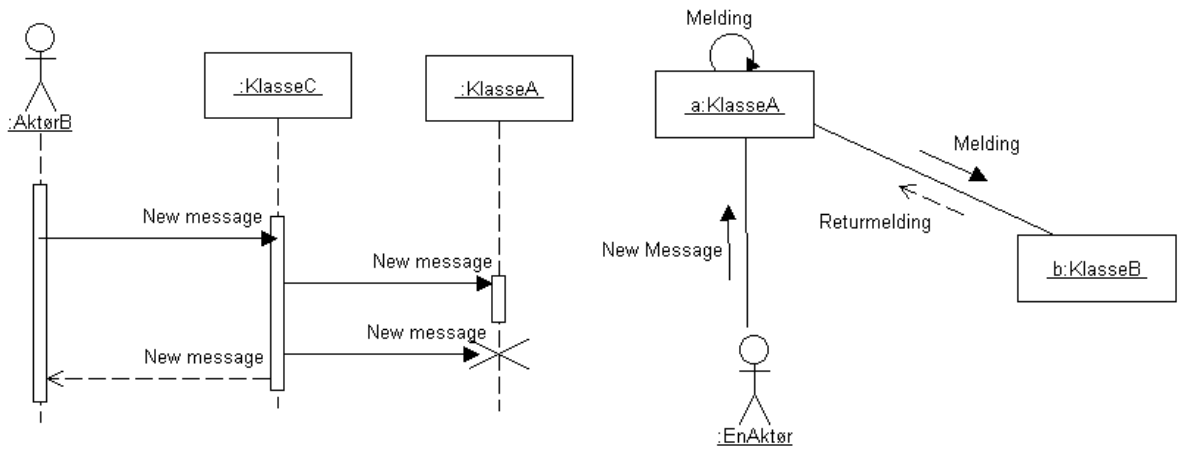
UML standarden blir ivaretatt av organisasjonen **OMG** (Object Management Group) som spesifiserer en del standarder for programvare utvikling. Mer om **OMG** og **UML** kan finnes på <http://www.omg.org/uml/>.



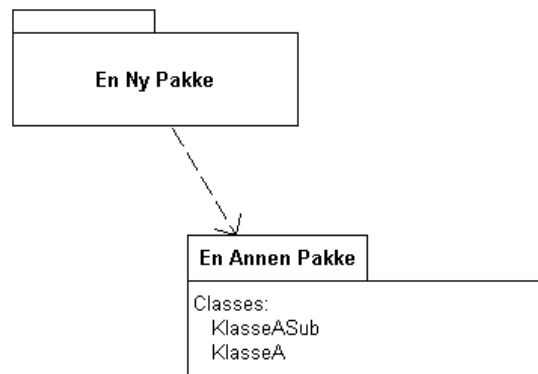
Eksempel på Klasse (t v) og UseCasediagram



Eksempel på Tilstands (t v) og Aktivitetsdiagram



Eksempel på Sekvens ( t v ) og Samarbeidsdiagram



Eksempel på Pakkediagram

## **1.5 LØSNINGSFORSLAG**

Vi ønsket å lage et brukervennlig program hvor man kan tegne lagre og skrive ut de for oss viktigste UML diagrammene: Klasse UseCase Sekvens Samarbeid Aktivitets Tilstands og Pakkediagram

Dette programmet skal kunne brukes først og fremst i faget LOD (Systemeringsfaget) men kan og være nyttig for større programmeringsprosjekter som feks i faget SOD Programmering av tekniske beregninger óg selvfølgelig andre hovedprosjekt ved datalinjen (HOD )

## 2 KRAVSPESIFIKASJON

Versjon \_\_\_\_\_ av programmet skal kunne tegne lagre og skrive ut UML diagrammer i henhold til standarden:

- Klasser med attributter metoder og assosiasjoner i Klassediagrammer
- Aktører og UseCases med ev extention points og assosiasjonene mellom disse
- Forskjellige tilstander og tilstandsoverganger i Tilstands og Aktivitetsdiagrammer
- Instanser av Aktører klasser og pakker i Sekvens og Samarbeidsdiagrammer
- Pakker med eller uten liste av innhold og assosiasjoner mellom disse i Pakkediagrammer

Programmet skal kunne kjøres på skolens maskinpark under Windows ( \_\_\_\_\_ ) og Linux

## 3 ANALYSE AV PROBLEMET

### 3.1 AVGRENSING

Vi ville lage et robust og oversiktlig program som enkelt kunne utvides med flere og andre muligheter. Et eksempel på dette var feks det å kunne bruke et prosjekt skapt i vårt program til å kunne få generert Java eller C kode.

Dette medførte at spesielt innsetting av klasser i prosjektet måtte behandles med omhu slik at de ble korrekt lagret i forhold til ting som assosiasjoner (spesielt arv) og attributter/metoder.

En annen ting som feks kan legges til senere er tegning av andre diagrammer både de UML diagrammene som vi ikke "trengte" i denne omgang (Objekt Komponent Deployment Subsystem og Modelldiagram) og andre typer diagrammer som (E)ER diagrammer til bruk i database design.

Vi har laget en liste av forslag (noen helt nødvendige andre mer svevende) til naturlige forbedringer og utvidelser av programmet i versjon og videre. Hele denne finnes i Appendix E.

### 3.2 ORGANISERING

Vi la opp til en anarkistisk arbeidsdeling hvor alle skulle ta del i all programmeringen.

Etter å ha blitt enige om en design for programmet skulle vi lage et skjelett med alle nødvendige filer. Så skulle vi utvikle klassediagrammet sammen (siden det skulle ligge i bunn for hele programmet) og utfra dette ene diagrammet utvikle de andre diagrammene. Først da skulle vi dele oss opp og ta hvert vårt diagram.

Lagring og utskrift lå implisitt i designet og skulle gå "automatisk".

Siden vi var en stor gruppe følte vi at vi hadde behov for et versjonskontrollsystem. Hvis fire personer skulle ha samordnet arbeidet i alle prosjektets filer ville vi brukt det meste av tiden bare på denne organiseringen, noe vi opplevde da vi programmerte en større oppgave i SOD sist høst. Uten et bedre system for dette ville prosjektet være dødfødt.

#### 3.2.1 ORIGINAL TIDSPLAN

Uke:				
Forarbeid				
Skjelett				
Klassedia				
Resten				
Testing				
Rapport				
Versjon				

## 4 DESIGN

### 4.1 PROGRAMMERINGSSPRÅK

Høsten 2004 skjedde det en omlegging av faget SOD 2004. Programmering med C og grafiske grensesnitt her ble det for første gang ved HiB undervist i Qt. To av gruppens medlemmer var studentassistenter i dette faget og måtte sette seg inn i Qt på egen hånd. De to andre på gruppen gjorde seg samtidig kjent med Qt på eget initiativ.

Qt er et grafisk bibliotek til C++ for flere plattformer og ble et naturlig valg siden en del av forutsetningen for programmet var at det skulle være brukervennlig og kunne kjøres på flere operativsystemer.

Et alternativ som knapt nok ble vurdert var å programmere i Java.

### 4.2 LAGRINGSFORMAT

Det første vi måtte avgjøre var om vi skulle lagre prosjektet samlet eller om vi skulle lagre hvert enkelt diagram og så ha en prosjektfil som holdt rede på hvilke diagram som var med i prosjektet. Vi gikk for en enkelt fil.

Så var det hvordan vi skulle lagre: Binær eller tekstfil. Vi gikk for en lesbar tekstfil og bruker XML format. XML er et velkjent, allment akseptert format som er strukturt og passet derfor godt til våre behov. Det finnes et eget standard format for lagring av UML prosjekter, men da dokumentasjonen av dette er på telefon katalog nivå, valgte vi å ikke sette oss inn i dette.

### 4.3 DESIGN

Når vi laget skjelettet delte vi alle objektene vi skal håndtere inn i passende kategorier og laget et arvehierarki basert på dette. I toppen ligger **Qt**, så har vi vårt **prosjekt** som inneholder de forskjellige diagrammene. Diagrammene kan tegne **figurer** som feks **klasser** og mellom klassene har vi **assosiasjoner**.

For detaljer i designen følger her et forenklet klassediagram som viser prosjektet vårt. Av hensyn til plassen er klassene vist her uten detaljer. Det fullstendige klassediagrammet vårt ligger på CDen under /UML filer/klassediagram uml



## 5 VERKTØY

### 5.1 QT

QT er et C++ klasse bibliotek for å lage grafiske programmer for Linux kommersielle versjoner av Unix Windows plattformer og Mac OS X. Det er selve QT biblioteket som er implementert for disse operativsystemene slik at etter å ha utviklet et program på en av disse plattformene er det bare å kompilere det på de andre.

Kjernegruppen i Trolltech begynte å utvikle QT i 1996 og stiftet Trolltech i 1997. Den første kommersielle versjonen ble utgitt i 1998. QT er i dag godt kjent i flere større utviklingsmiljøer.

Biblioteker for å lage grafiske programmer på flere plattformer bruker vanligvis en av tre metoder:

- **API layering:** Her legges en API på toppen av den enkelte plattforms API og så bruker den øverste den under til å "tegne" grafikken. Denne metoden er veldig enkel å programmere med, men siden den jobber med en API oppå en annen så blir disse programmene ofte trege.
- **API emulering:** Her blir plattformenes egne API'er emulert på de andre systemene. Det store problemet her er at det er sjelden enkelt å emulere en API i de enkelte systemene. Ofte kan en ende opp med at en kjører et program oppå en emulering som kjører oppå en emulering, dette medfører at en plukker med seg en del bugs som ligger i de forskjellige emuleringene nedover. Det er ikke uvanlig at en har et MFC lag oppå et Win32 lag oppå et Motif lag oppå Xt lag oppå et Xlib lag osv.
- **GUI emulering:** Her blir tegne og grafikk primitivene i de enkelte systemene brukt direkte av biblioteket. Den store fordelen her er at siden biblioteket jobber direkte mot det grunnleggende systemet så vil den jobbe en del raskere enn de bibliotekene som benytter flere API lag eller emulerer API'er. Den største ulempen er at med denne metoden så må alle mulige elementer som skal vises på skjerm råkodes for hver enkelt plattform, noe som medfører masse arbeid for de som lager biblioteket. QT bruker denne metoden.

## **5.2 CVS (Concurrent Version System)**

CVS er et versjonskontroll system som er mye brukt på prosjekter der flere utviklere skal jobbe sammen på et produkt

Når en bruker CVS ligger all kildekode (og evt. andre filer) til prosjektet på en servermaskin. Serverområdet blir kalt 'CVS Repository'. Alle utviklerne laster ned en kopi av koden fra CVS serveren og jobber hele tiden lokalt på denne kopien.

Koden blir lagret på serveren i et spesielt format slik at hver fil inneholder alle ulike versjoner av kildefilen. På denne måten kan en gå tilbake til tidligere versjoner av en fil hvis noe skulle skjære seg.

Når en ønsker å gjøre sine endringer tilgjengelige for de andre, laster en opp endringene en har gjort til serveren. Dette blir gjort med kommandoen 'cvs commit' når en står i mappen der kildekoden ligger.

For å oppdatere sin lokale kopi slik at den ikke blir utdatert i forhold til det som ligger på serveren, bruker en kommandoen 'cvs update'. Dette bør gjøres med jevne mellomrom, og spesielt før en sender opp endringer til serveren via 'cvs commit'. Hvis to utviklere har endret samme koden, vil 'cvs update' gi beskjed om dette, og en må inn og endre koden slik at rett versjon blir brukt videre. CVS vil legge begge versjoner inn i filen og markere dette med spesielle symboler.

CVS commit og CVS update er de viktigste og mest brukte kommandoene. For en oversikt over flere nyttige kommandoer og mer forklaring av bruk og konfigurering av CVS, se Appendix C.

## 6 IMPLEMENTERING

Vi har kodet            ikke blanke linjer (            kodelinjer totalt)

All kildekode er lagt ved på CDen

For nærmere beskrivelse av CDen se Appendix A

## 7 TESTING

Vi la i planleggingen opp til å la prosjektet følge en iterativ prosess hvor delene av programmet ble grundig testet etterhvert. Dette gikk forholdsvis bra, men pga tidspress på slutten ble grundig stresstesting av hele programmet på flere plattformer som feks Windows / nedprioritet.

En viss gjennomgang av programmet fikk vi dog ved å tegne klassediagrammet til vårt eget prosjekt for å bruke det i presentasjonen og denne rapporten. Dette avdekket en del små, men irriterende feil i beta versjonen som vi da kunne fjerne til versjon

Ikke alle feil stod i vår makt å fjerne, enten fordi de ligger utenfor vår "rekkevidde" dvs at vi med god samvittighet kan skylde på feil i Qt biblioteket og håper at Trolltech fikser dette i fremtidige versjoner av Qt, eller rett og slett at vi ikke hadde tid til å fikse dem. Kjente feil i UMLUS er:

### 7.1 KJENTE FEIL ("*Known bugs*")

- "Printer Setup" menyen henter opp en meny med kun to valg: "Print" og "Cancel". "Print" skriver da ikke ut, men er egentlig "OK".
- Qt har problemer med å lese Printer Properties på lagrede prosjekt, og du må noen ganger kjøre "Printer Setup" fra "Project" menyen før du skriver ut (eller vil se rett visningsformat på skjermen).
- Hvis du bruker popup menyen som kommer opp ved høyreklikk til å slette et objekt, hender det at objektet er markert hvis du omgjør slettingen med "Undo".
- Vi har lagt inn få begrensninger på bruker input, så brukeren kan tegne metoder, attributter, assosiasjoner o.l med navn som ikke følger UML standarden, dette kan føre til feil i "Save" og "Load" funksjonene.
- Noen diagram kan lagre Undo informasjonen to ganger, slik at bruker må ta Undo to ganger.
- Vi er svært usikker på om Undo virker alle steder.
- Noen ganger vil assosiasjonene ikke tegnes korrekt opp mot klassene "sine", dvs at den ikke treffer kanten på klassen.
- Endene til assosiasjonene flyttes ikke med klassen, når klassen endrer størrelse pga endring av fontstørrelsen.
- Når "Moving Points in Associations On/Off" er på, vil punktene hvor assosiasjonene er forankret i klassen "sin" endres litt hver gang du har vært inni klassens "Properties" meny.
- I samarbeidsdiagram, hvis man setter inn en TOSELF assosiasjon og så endrer navn på klassen, vil ofte TOSELF assosiasjonen tegnes i feil retning.

### 7.2 MULIGE FREMTIDIGE PROBLEMER

Ting som kan framtidige utviklere bør være oppmerksom på:

- Om QPrinter sine enum'er blir endret i fremtidige versjoner av QT, så kan det bli problemer med innlasting av prosjekter lagret på gamle versjoner. Koden der dette blir brukt ligger spesielt i Diagram::load().

## 8 BESKRIVELSE AV UTVIKLET SYSTEM I BRUK

### 1 Systemkrav

Skjermopløsning: x eller bedre

RAM: Det som er anbefalt for operativsystemet

#### Windows:

Win (Win ) eller nyere Programmet er ikke testet på Win

Prosesor: anbefalt MHz eller bedre

#### Linux:

Qt eller bedre installert

### 2 Installasjon

#### Linux:

Pakk ut umlus\_v tar.gz der du vil ha programmet liggende

#### Windows:

Pakk ut umlus\_v.zip der du vil ha programmet liggende Pass på at qt mtedu.dll ligg i samme mappe som umlus.exe

### 3 Kjøre programmet

#### Windows:

Start umlus.exe

#### Linux:

Start umlus

## 4 Brukerinstruksjoner

### Åpningsdialog:

To valg:

- New Project: Lager nytt prosjekt
- Open Project: Åpner eksisterende prosjekt

### *Ved nytt prosjekt:*

Får en ny dialog der man skal skrive inn informasjon om prosjektet

### *Ved åpning av eksisterende prosjekt:*

Skal fullstendig sti til fil vises i editboksen Du kan bruke browse eller listeboksen til å velge filer

Trykk OK får å velge Cancel er i praksis det samme som New Project men du får ikke opp dialogen

### Programmet:

#### *Meny:*

- File
- Edit
- View
- Help

#### *Toolbar:*

- Open Project  
Åpner prosjekt
- Save  
Lagrer prosjekt
- Print  
Skriver ut aktivt diagram
- CLD  
Lager nytt klassediagram
- UCD  
Lager nytt UseCasediagram
- SED  
Lager nytt sekvensdiagram
- ACD  
Lager nytt aktivitetsdiagram
- PAD  
Lager nytt pakkediagram
- COD  
Lager nytt samarbeidsdiagram
- STD  
Lager nytt tilstandsdiagram

### ***File menyen:***

- New Project  
Lager nytt prosjekt Lukker det som er åpnet
- Edit Project  
Åpner dialog for å endre informasjonen om prosjektet
- Open  
Åpner dialogen for å åpne et prosjekt
- Save  
Lagrer prosjektet
- Save as  
Åpner dialogen for å bestemme navnet på prosjektfilen og hvor denne skal ligge
- New Package  
Lager en ny pakke
- New Diagram  
Åpner dialogen for å lage nytt diagram
- Close Diagram  
Lukker åpnet diagram
- Remove current diagram  
Sletter diagrammet fra prosjektet kan ikke ta undo på dette valget !
- Save as BMP  
Lagrer aktivt diagram som BMP
- Save as PNG  
Lagrer aktivt diagram som PNG
- Printer Setup  
Åpner dialogen for å sette printer innstillingene
- Print  
Skriver ut aktivt diagram
- Print All Diagrams  
Skriver ut alle diagrammer
- Recently Opened Projects  
Ny meny med de fire siste prosjektene som har vært åpnet

### ***Edit menyen:***

- Undo  
Angrer
- Moving Points in Associations On/Off  
Slår av/på automatisk oppdatering av assosiasjonene sine endepunkter ved endringer på klassene de er koblet mot
- Increase Font Size  
Øker fontstørrelsen
- Decrease Font Size  
Minker fontstørrelsen

### ***View menyen:***

- Packages  
Lukker/Åpner pakkevinduet Packages
- Diagrams  
Meny over diagram som er i prosjektet Også de som er lukket

### **Help menyen:**

- Known Bugs  
Informasjonsboks over alle de feilene vi kjenner til i programmet
- About Umlus  
Om programmet
- About QT  
Informasjon om Qt GUI biblioteket vi brukte for å lage programmet

### **CLD: navnet på diagrammet**

(Når du har åpnet diagram)

CLD: står her for klassediagram Kan og stå UCD SED

Har et spesielt valg

- Edit diagram  
Der du får opp dialogen med informasjon om diagrammet

En ny toolbar som er ulik for de forskjellige diagrammene

### **Klassediagram:**

- Klasse  
Oppretter ny klasse Må trykke i diagrammet for å sette den inn
- Diverse assosiasjoner  
Oppretter nye assosiasjoner Må trykke på fra klasse og til klasse for å sette assosiasjonen inn

En kan også trykke tilfeldige steder i diagrammet mellom klikk på fra og til klasse for å sette inn ekstra punkter på assosiasjonen

### **UseCaseDiagram:**

- Aktør  
Oppretter ny aktør
- UseCase  
Oppretter nytt Use Case
- Diverse assosiasjoner
- Boundary  
Oppretter nytt boundary område

### **Sekvensdiagram:**

- Aktør  
Oppretter en aktør instans
- Instans  
Oppretter en klasse instans
- Tidsboks  
Oppretter en ny tidsboks Må plasseres på tidsaksen til aktør eller klasseinstans
- Slutt  
Kryss som markerer at instansens levetid stopper Må plasseres på tidsaksen til klasseinstans
- Diverse meldinger

### ***Aktivitetsdiagram:***

- Start state  
Oppretter en startstate
- Stopp state  
Oppretter en stoppstate
- State  
Oppretter en tilstand
- Branch/Merge  
Oppretter en ny branch/Merge
- Aktivitet  
Oppretter en ny aktivitet
- Fork/Join  
Oppretter en ny fork/join
- Assosiasjon  
Setter inn en assosiasjon mellom de andre objektene
- Swimlane  
Oppretter en svømmebane

### ***Pakkediagram:***

- Pakke  
Oppretter en ny pakke
- Assosiasjon  
Assosiasjon mellom pakker

### ***Samarbeidsdiagram:***

- Aktør  
Oppretter en aktør instans
- Klasse  
Oppretter en klasse instans
- Assosiasjoner
- ToSelf melding  
Settes inn ved å klikke på den instansen som skal ha denne meldingen til seg selv
- Diverse meldinger  
Settes inn ved å klikke på assosiasjon de skal plasseres på

### ***Tilstandsdiagram:***

- States (start stopp )
- Branch/Merge
- Tilstand  
Oppretter en ny tilstand
- Parallell tilstand  
Oppretter en ny parallell tilstand
- Fork/Join
- Assosiasjon  
Assosiasjoner mellom de ulike figurene

## 9 EVALUERING AV PROSJEKTARBEIDET

Arbeidet med prosjektet gikk i starten litt sakte dette var fordi oppstarten av dette prosjektet for det meste bestod i å opprette et skjelett for dataprogrammet vårt og å utvikle det første diagrammet (Klassediagrammet) Mens Glenn jobbet med dette så jobbet de andre for det meste med det andre prosjektet vi hadde tatt på oss "Eksperimentering med QuickSort algoritmen" Dette andre prosjektet opptok da veldig masse av gruppens tid Etter et par uker oppdaget Vidar at det som var hovedidéen som QuickSort prosjektet bygget på allerede var implementert i GNU biblioteket

Etter at skjelettet var mer eller mindre ferdig og klassediagrammet var godt i gang kunne vi alle begynne for fullt å programmerere på Umlus Vi hadde til å begynne med planer om at alle på gruppen skulle fullstendig implementere forskjellige diagrammer Men i og med at vi var allerede langt bak tidsplanen og vi innså at dette ville innebære veldig masse koding la vi litt om på hvordan vi skulle jobbe Glenn jobba med å generalisere det som allerede var blitt gjort i Klassediagrammet Geir Roger og Lasse begynte å jobbe med de andre diagramma og Vidar begynte å jobbe med mer grunnleggende ting i diagrammet slik som Undo og lagring/lasting

I og med at vi var en veldig stor gruppe og vi alle potensielt gjorde endringer i de samme filene så var det essensielt at vi hadde et versjonskontrollsystem Vi hadde allerede på forhånd bestemt oss for at dette var noe vi skulle bruke og hadde satt opp et CVS område på hjemmeområdet til Glenn på Solo CVS var noe som fungerte veldig bra oppgaven vår hadde nok aldri blitt ferdig om vi ikke hatt en eller annen form for versjonskontroll Etter dette hovedprosjektet mener gruppen at skolen bør sette opp et CVS område på for eksempel Solo eller en annen av skolens servere der dette prosjektet og eventuelle andre prosjekter som har mer enn én person kan ligge Dette vil også medføre at dette prosjektet lett kan følges opp av denne gruppen og eventuelt lett videreføres til en ny gruppe Se Kap 10 og app B og C for mer informasjon om CVS

Ettersom CVS var så essensielt for at arbeidet med prosjektet vårt skulle lykkes finner vi det noe rart at forskjellige versjonskontrollsystemer som CVS ikke inngår i pensum på dataingeniørstudiet At skolen ikke har et eget CVS område for elevenes prosjekter er òg beklagelig Vi mener at ved å gå igjennom feks CVS i undervisningen og latt elevene med behov for det bruke CVS i større prosjekter ville studentene lettere fungere i grupper på mer enn én person og ikke minst være bedre rustet for arbeidslivet etter skolen

## 10 EVALUERING AV PROGRAMMET

Vi føler at vi har laget et program som temmelig godt oppfyller kravene vi satte til det før prosjektet begynte

Det er en del ting vi ikke har fått til så bra som vi skulle ha ønsket og det er noen ting som vi måtte velge vekk grunnet tidsnød se Bugliste i kap og Versjon i App E

Programmet kommer i alle fall til å bli tatt i bruk av deltakerene på gruppen når vi senere skal systemere prosjekter

Som en kuriositet kan nevnes at en annen gruppe brukte UMLUS til å tegne UseCase diagrammene i presentasjonen av deres hovedprosjekt

## 11 LITTERATUR

- Perdita Stevens with Rob Pooley: Using UML Software Engineering with Objects and Components updated edition Addison Wesley
- Eriksson / Penker: UML Toolkit Wiley Computer Publishing
- Matthias Kalle Dalheimer: Programming with Qt O'Reilly
- Herbert Schildt: C/C Programmer's Reference Second Edition Osborne/McGraw Hill
- M Welsh M K Dalheimer & L Kaufmann: Running Linux O'Reilly rd edition
- TROLLTECHs online dokumentasjon av Qt: <http://doc.trolltech.com/>
- CVS online dokumentasjon: <http://www.cvshome.org/> og <http://linux.apus.sourceforge.net/> (oppsett)

## 12 VEDLEGG:

CD Se Appendix A

# STIKKORDREGISTER

API  
BMP  
Brukervennlig  
C  
CVS  
  add  
  checkout  
  commit  
  import  
  remove  
  update  
Diagram  
  aktivitet  
  klasse  
  pakke  
  samarbeid  
  sekvens  
  tilstand  
Feil  
GakkGakk  
GUI  
HiB  
Hovedprosjekt  
Java  
Kildekode  
Linux  
LOD   System og programutviklingsmetoder  
Mac  
Miljøvariabler  
OMG  
PNG  
Qt  
SSH  
Sti  
Trolltech  
UML  
UMLUS  
Unix  
Windows  
XML

## **APPENDIX A - INNHOLD PÅ VEDLAGT CD**

### ***Windows***

umlus.exe  
qt\_mtedu.dll  
README.txt  
HOWTO.txt

### ***Linux (x86)***

umlus  
README  
HOWTO  
LICENSE

### ***Linux (PPC)***

umlus  
README  
HOWTO  
LICENSE

### ***UML-filer***

Klassediagram uml  
Presentasjon uml

### ***CVS***

CVS Manual (f)  
CVSROOT (f)  
Prosjekt Log.txt

### ***Doc***

Hjemmesider (f)  
UML\_rapport.pdf

## **Kildekode**

### **Header-filer**

AboutDialog h	KlasseDiagram h	SekvensDiagram h
AktivitetKlasse h	KlasseDialog h	StandardAss h
AktivitetsAss h	Kontainer h	StandardKlasse h
AktivitetsDiagram h	Melding h	StateDialog h
AktorDialog h	MeldingsAss h	StateKlasse h
AktorKlasse h	Metode h	StateTransDialog h
ArvAss h	NestedDialog h	SwimlaneKlasse h
Assosiasjon h	NestedStateKlasse h	SyncKlasse h
AssosiasjonDialog h	OpningsDialog h	Tidsakse h
AvhengigAss h	Pakke h	Tidsboks h
BrukarAss h	PakkeAss h	TilstandsAss h
Diagram h	PakkeDiagram h	TilstandsDiagram h
DiagramDialog h	PakkeDialog h	TilstandsKlasse h
Diaholder h	PakkeVindu h	UmlListBoxItem h
Dialog h	Prosjekt h	UmlListView h
ExtendAss h	ProsjektDialog h	UmlListViewItem h
Figur h	RealiseringAss h	UseCaseDiagram h
IncludeAss h	SamarbeidsAss h	UseCaseDialog h
Includes h	SamarbeidsDiagram h	UseCaseKlasse h
Klasse h	SaveQuestionDialog h	

### **Source-filer**

AboutDialog cpp	KlasseDiagram cpp	SaveQuestionDialog cpp
AktivitetKlasse cpp	KlasseDialog cpp	SekvensDiagram cpp
AktivitetsAss cpp	Kontainer cpp	StandardAss cpp
AktivitetsDiagram cpp	Melding cpp	StandardKlasse cpp
AktorDialog cpp	MeldingsAss cpp	StateDialog cpp
AktorKlasse cpp	Metode cpp	StateKlasse cpp
ArvAss cpp	NestedDialog cpp	StateTransDialog cpp
Assosiasjon cpp	NestedStateKlasse cpp	SwimlaneKlasse cpp
AssosiasjonDialog cpp	OpningsDialog cpp	SyncKlasse cpp
AvhengigAss cpp	Pakke cpp	Tidsakse cpp
BrukarAss cpp	PakkeAss cpp	Tidsboks cpp
Diagram cpp	PakkeDiagram cpp	TilstandsAss cpp
DiagramDialog cpp	PakkeDialog cpp	TilstandsDiagram cpp
Diaholder cpp	PakkeVindu cpp	TilstandsKlasse cpp
Dialog cpp	Prosjekt cpp	UmlListView cpp
ExtendAss cpp	ProsjektDialog cpp	UseCaseDiagram cpp
Figur cpp	RealiseringAss cpp	UseCaseDialog cpp
IncludeAss cpp	SamarbeidsAss cpp	UseCaseKlasse cpp
Klasse cpp	SamarbeidsDiagram cpp	umlus cpp

## APPENDIX B - SAMMENDRAG CVS LOGG

For å demonstrere nytten vi hadde av CVS følger her et sammendrag av CVS logg filen. Den komplette loggen finnes under /CVS/Prosjekt Log.txt

Status ved innlevering av Hovedprosjekt

Antall filer på cvs området til prosjektet:

C kildekode:

C header filer:

Qt ui filer:

Qt prosjekt filer:

xpm bilder:

Totalt:

Den filen som ble endret flest ganger var Prosjekt.cpp

Den endte i revisjon (starter på )

Antall ganger cvs commit har blitt kjørt (endringer lastet opp til server):

Glenn Hisdal:

Geir Roger Moen:

Lasse Hægland:

Vidar Lund:

Totalt:

## **APPENDIX C - GRUNNLEGGENDE OPPSETT OG BRUK AV CVS**

Dette dokumentet vil hjelpe deg i gang med CVS

Det beskriver hvordan du enkelt kan sette opp et cvs område på serveren og starte et nytt prosjekt som bruker cvs

De nødvendige kommandoene for å jobbe med cvs vil bli forklart

### **OPPSETT AV CVS OMRÅDE**

For å ta i bruk cvs må en definere en mappe som skal fungere som rot for alle prosjekter som skal benytte cvs

Når en bruker cvs må en fortelle cvs hvor denne mappen er Dette gjøres enklest ved å definere variabelen CVSROOT

Hvis en bruker bash skallet blir kommandoen for dette 'export CVSROOT = sti til cvs rot '

Legg merke til at en må oppgi fullstendig sti til cvs rot

Når CVSROOT variabelen er satt må cvs området lages

Kommandoen for dette er 'cvs init'

CVS er nå klart til bruk og en kan legge inn de prosjekter en ønsker å benytte cvs for

### **LEGG INN NYE PROSJEKTER**

Den enkleste måten å legge inn nye prosjekter i cvs er å bruke kommandoen 'cvs import mappe author tag '

En lager da filene en ønsker å ha med i prosjektet lokalt Når en har fått ønsket struktur kan denne legges inn i cvs området

Eksempel:

Du har opprettet katalogen gakkgakk i /home/kvakk/

I denne katalogen har du lagt alle filene i prosjektet gakkgakk og ønsker å benytte cvs for å holde rede på koden

For å legge gakkgakk til i cvs området skriver du:

'cvs import gakkgakk dittnavn tag '

når du står i katalogen /home/kvakk/gakkgakk

tag kan settes til hva som helst f eks versjon\_

En kan nå laste ned koden til prosjektet gakkgakk vha cvs

## **CVSROOT MODULEN**

Alle CVS områder har den spesielle modulen CVSROOT. Denne inneholder konfigurasjonsfiler for cvs.

På vedlagte CD finnes en konfigurasjon som fungerer bra. Den er satt opp slik at den sender ut en e-post til alle utviklerne hver gang en commit blir utført. I tillegg vil den skrive informasjonen til en logfil.

Dette gjør det enkelt å holde seg oppdatert på hva som skjer med koden.

For å benytte denne konfigurasjonen gjør følgende:

`cvs checkout CVSROOT`

Kopier filene fra cd'en over i CVSROOT katalogen som ble opprettet.

Editor filen `loginfo` og skift ut domene med ønsket domene og e-post adressene med de du ønsker skal motta meldinger etter commit. Skift også ut navnet på log-filen med ønsket navn (fullstendig sti).

Kjør `cvs update` når du står i CVSROOT katalogen.

Kjør `cvs add` fil liste på alle filer med `?` foran.

Sjekk at alle filene i CVSROOT har `execute` flagget satt.

Kjør `cvs commit`.

## **BRUK AV CVS**

Her kommer en kjapp innføring i de viktigste kommandoene for bruk av cvs  
Før en kan begynne må en sette CVSROOT variabelen på samme måte som nevnt over

Hvis en sitter på en annen maskin enn den cvs området ligger på må en benytte en eller annen for innlogging på server maskinen for å få tilgang til filene

Den sikreste måten er å benytte SSH (secure shell) En må da ha en konto på server maskinen og den må være satt opp til å godta innlogging via ssh

På den lokale maskinen må en ha installert cvs programmet og en ssh klient

For å fortelle cvs at en skal benytte ssh må variabelen CVS\_RSH settes til ssh

I bash gjøres dette slik: 'export CVS\_RSH ssh'

Når en benytter ssh må også CVSROOT variabelen endres litt

'export CVSROOT :ext: dittnavn @ maskin domene : sti til cvs rot '

### **cvs import**

Legger til nye moduler i cvs treet Se beskrivelse over

### **cvs checkout**

Denne brukes for å laste ned filene til et prosjekt første gang En kan senere bruke cvs update for å laste ned endringer

En må oppgi hvilken modul (prosjekt) en ønsker å laste ned

Eksempel:

'cvs checkout gakkgakk'

Du vil nå få opprettet katalogen gakkgakk i arbeidskatalogen din og den vil inneholde alle filene i modulen gakkgakk

### **cvs commit**

Oppdaterer filene på serveren slik at dine endringer kommer med

cvs vil starte i den katalogen du står i og sende opp endringer i den og alle underkataloger

Hvis du oppgir et filnavn vil cvs sende opp endringer i bare den filen

For å oppgi en log melding (som vil bli skrevet i en cvs log) bruk opsjonen

m melding

Hvis denne ikke er oppgitt vil cvs starte en editor der du får skrive inn denne

## **cvsv update**

Sjekker om noen av filene på cvs serveren er nyere enn din lokale kopi. Hvis det er tilfelle vil dine filer bli oppdatert. cvs update vil skrive ut en bokstav ved siden av filnavnet til hver fil den sjekker. Disse bokstavene beskriver status til filen. Mulige bokstaver er:

**U:** En nyere versjon eksisterte på serveren og din lokale kopi har blitt oppdatert. Dette skjer for alle nye filer på serveren og filer som er oppdatert på server men som du ikke har endra lokalt.

**P:** Samme som U men cvs sender en patch i stedet for hele filen. Bruker mindre båndbredde.

**A:** Filen har blitt lagt til av deg vha cvs add og vil bli sendt opp til serveren neste gang du kjører en cvs commit.

**R:** Filen har blitt fjernet av deg vha cvs remove og vil bli fjernet fra serveren neste gang du kjører cvs commit.

**M:** Filen er endret lokalt av deg. M kan indikere en av to ting: Du har endret filen men ingen andre har det. Eller både du og andre har endret filen men cvs greide å slå sammen endringene uten konflikter. CVS vil skrive ut litt ekstra informasjon hvis det slår sammen endringer.

**C:** Både du og andre har gjort endringer og cvs greide ikke å slå sammen endringene skikkelig. For å rette opp i dette må du åpne filen i en editor og se etter strengen. Etter denne strengen blir navnet på filen vist igjen etterfulgt av din versjon av koden. Etter din versjon kommer strengen etterfulgt av koden som lå på serveren. Til slutt kommer Du må nå finne ut hvilken kode som skal være med og fikse dette manuelt. Etterpå kan du sende filen opp til serveren vha cvs commit.

**?:** Filen er i arbeidskatalogen din men ikke på serveren (cvs kjenner ikke til filen).

## **cvsv add**

Legger til nye filer eller kataloger på cvs området. Filene blir sendt opp på serveren neste gang du kjører cvs commit.

## **cvsv remove**

Sletter filer fra cvs området. Filene må slettes fra arbeidskatalogen før denne kommandoen blir kjørt.

Filene vil bli slettet fra serveren neste gang du kjører cvs commit.

For en mer grundig beskrivelse av CVS se CVS manualen på vedlagte CD eller på <http://www.cvshome.org/>

## **Oppsett av CVS på Windows**

### **Windows og CVS**

For å bruke CVS trenger man en CVS klient. Eksempler på klienter for Windows er klienten fra Cyclic. Denne er tekstbasert og må integreres i MS DOS ledetekst. Den kan lastes ned fra <http://www.cvshome.org/dev/codewindow.html>

En annen klient er WinCvs, denne er grafisk. Vi fikk ikke denne til å virke, men hvis andre vil teste denne kan den lastes ned fra <http://www.cvsgui.org/>

I noen tilfeller må en kunne logge på en server for å få tak i det sentrale CVS området. Da bør en bruke SSH og trenger en SSH klient. Dette kan lastes ned fra nettet, f.eks. OpenSSH fra <http://www.networksimplicity.com/openssh/>. Denne er tekstbasert. En annen er den grafiske SSH Secure Shell Client som kan lastes ned fra <http://www.ssh.com/products/ssh/download.cfm>

### **Miljøvariabler**

For å få CVS til å virke må vi sette en del miljøvariabler. Klienten fra Cyclic er avhengig av at klienten er med i søkestien, dvs. at PATH miljøvariabelen har med mappen hvor CVS klienten ligger. Dette gjøres forskjellig under ulike versjoner av Windows.

I Windows / må en inkludere variabler i PATH i autoexec.bat. En bør være forsiktig når man editerer autoexec.bat ellers kan installerte programmer slutte å fungere; i verste fall vil ikke Windows starte opp.

Begge CVS klientene trenger SSH og derfor må også stien til SSH klienten settes i PATH. Med OpenSSH må en gjøre dette selv, mens SSH Secure Shell Client greier dette selv. Når man installerer SSH Secure Shell Client får en spørsmål om man vil integrere det i PATH'en, da bør man svare ja.

Slik kan det se ut i autoexec.bat når du setter PATH variabelen:

```
set PATH PATH ;d:\cvs \
set PATH PATH ;C:\Program Files\NetworkSimplicity\ssh
```

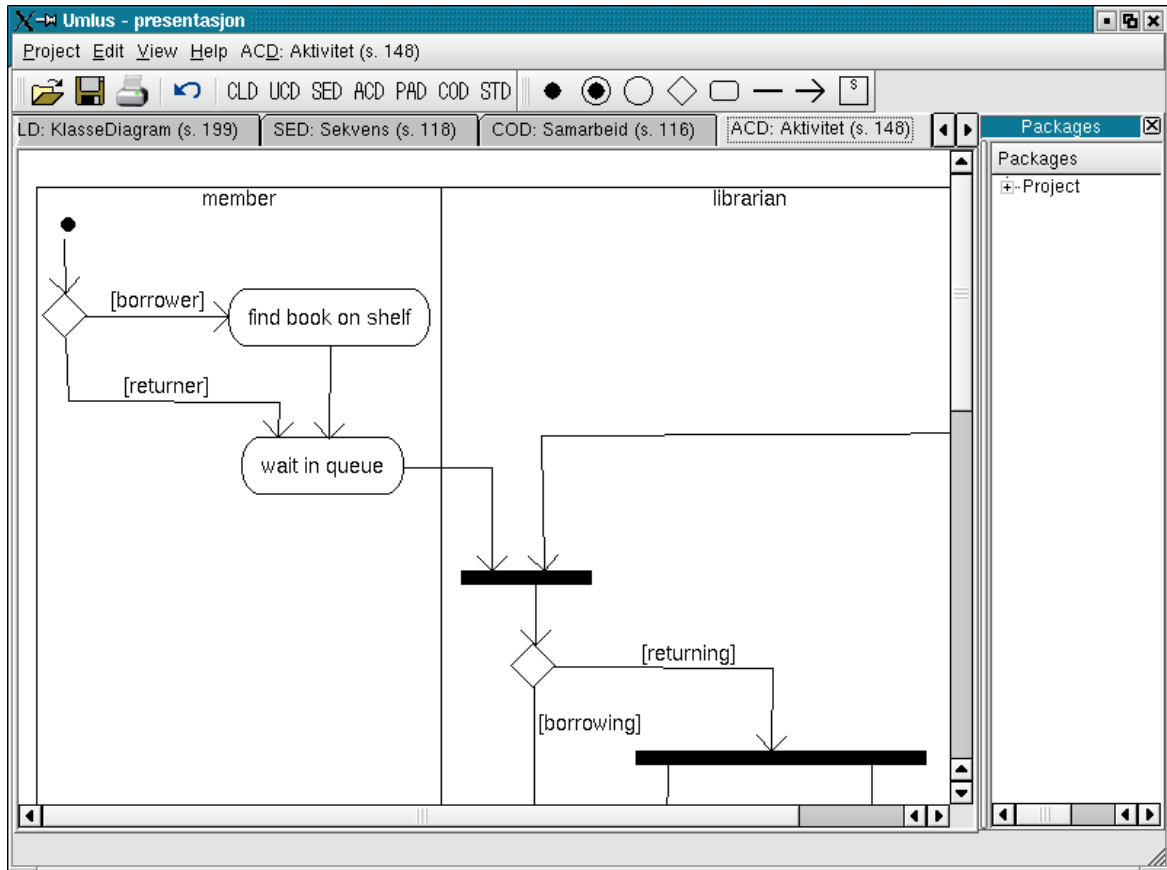
I Win og WinXP så må du sette miljøvariablene i System. Du finner de antagelig under avansert. Der får du opp en hel liste over miljøvariabler. Der må du legge dem inn på denne formen:

```
;D:\cvs ;C:\Program Files\NetworkSimplicity\ssh
```

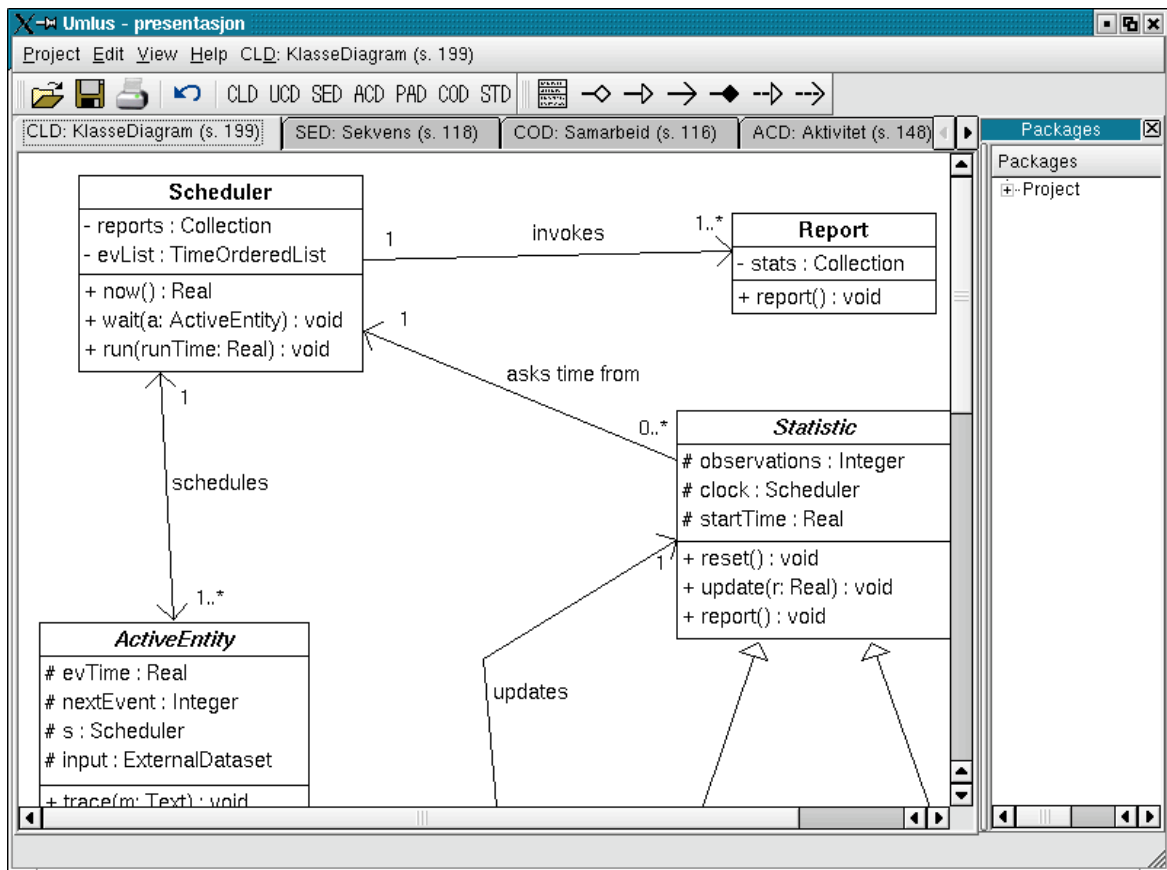
PATH er en semikolon separert liste.

En må muligens legge til flere miljøvariabler for CVS\_RSH og CVSROOT.

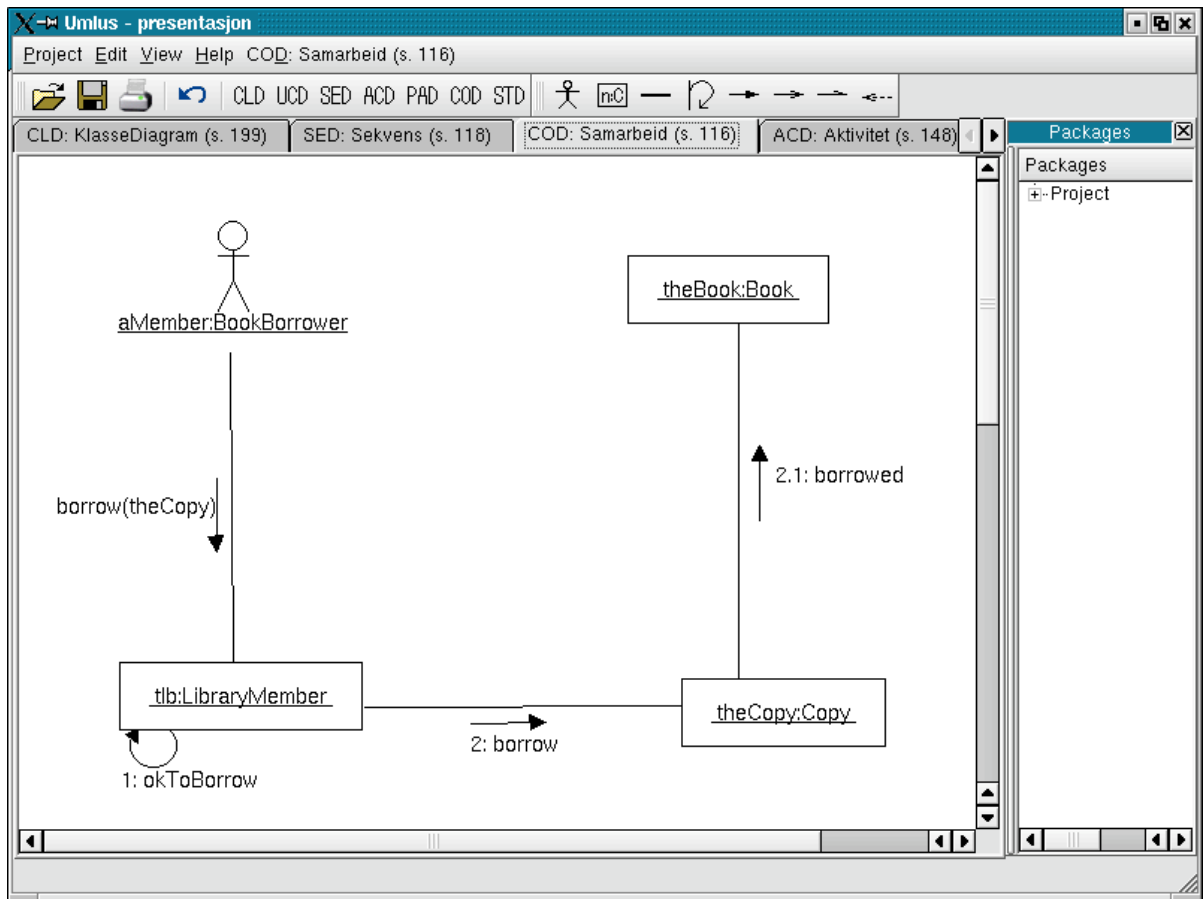
## APPENDIX D - SKJERMBILDER



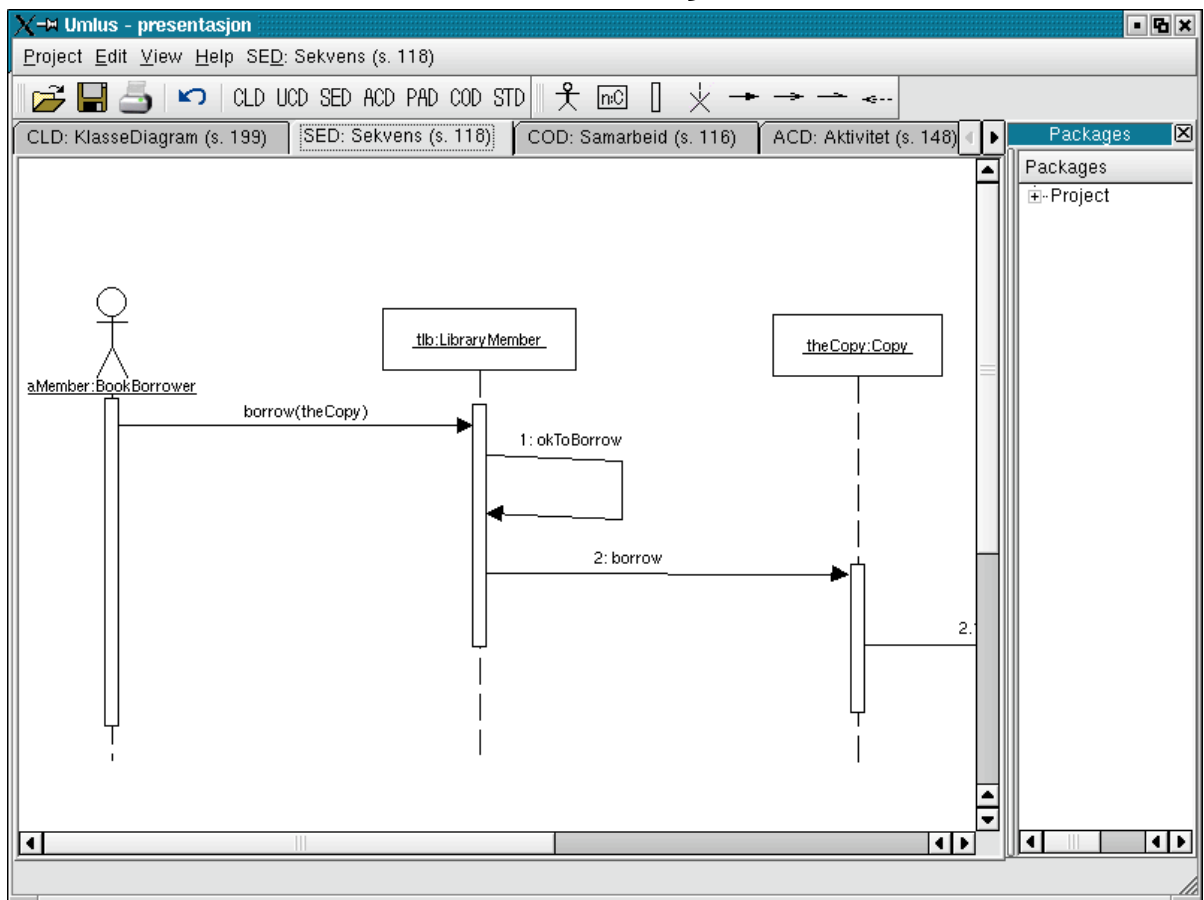
Aktivitetsdiagram



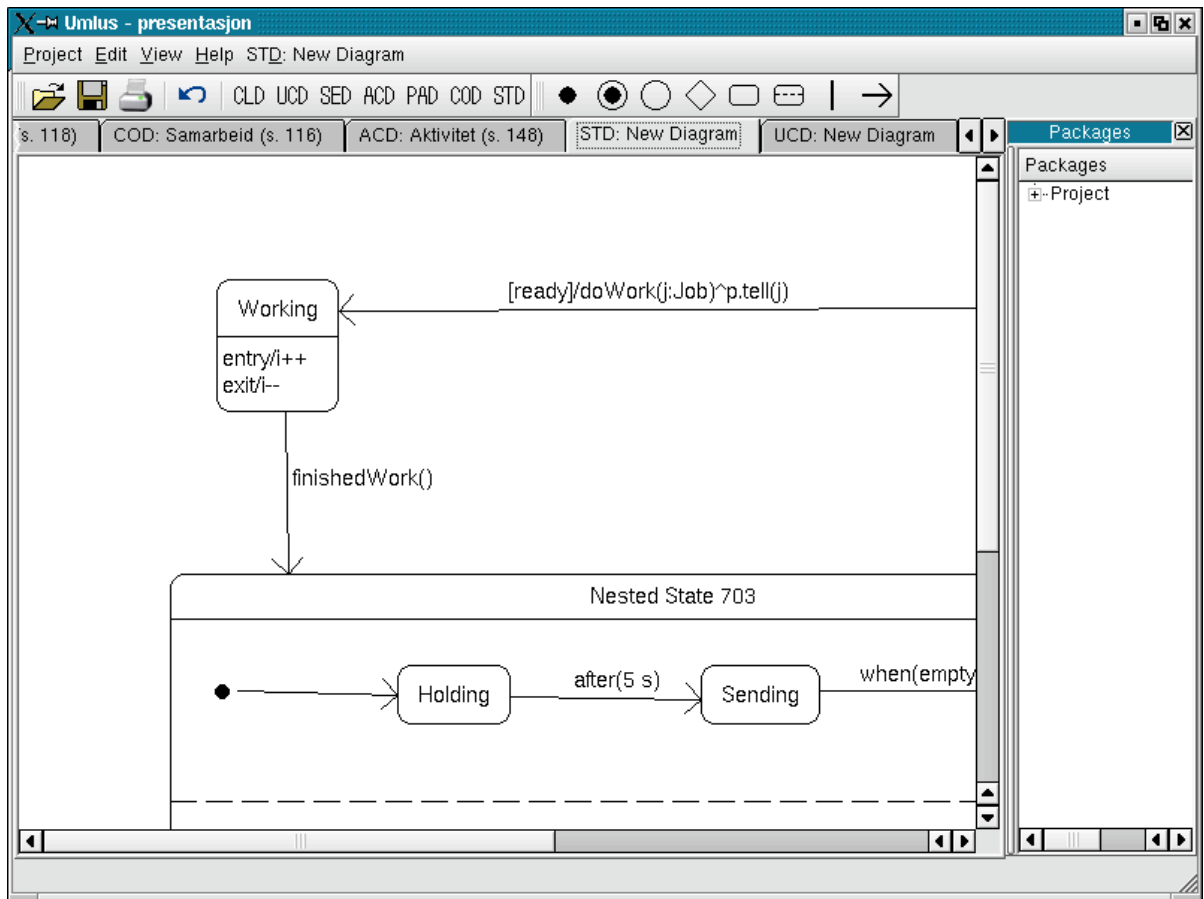
Klassediagram



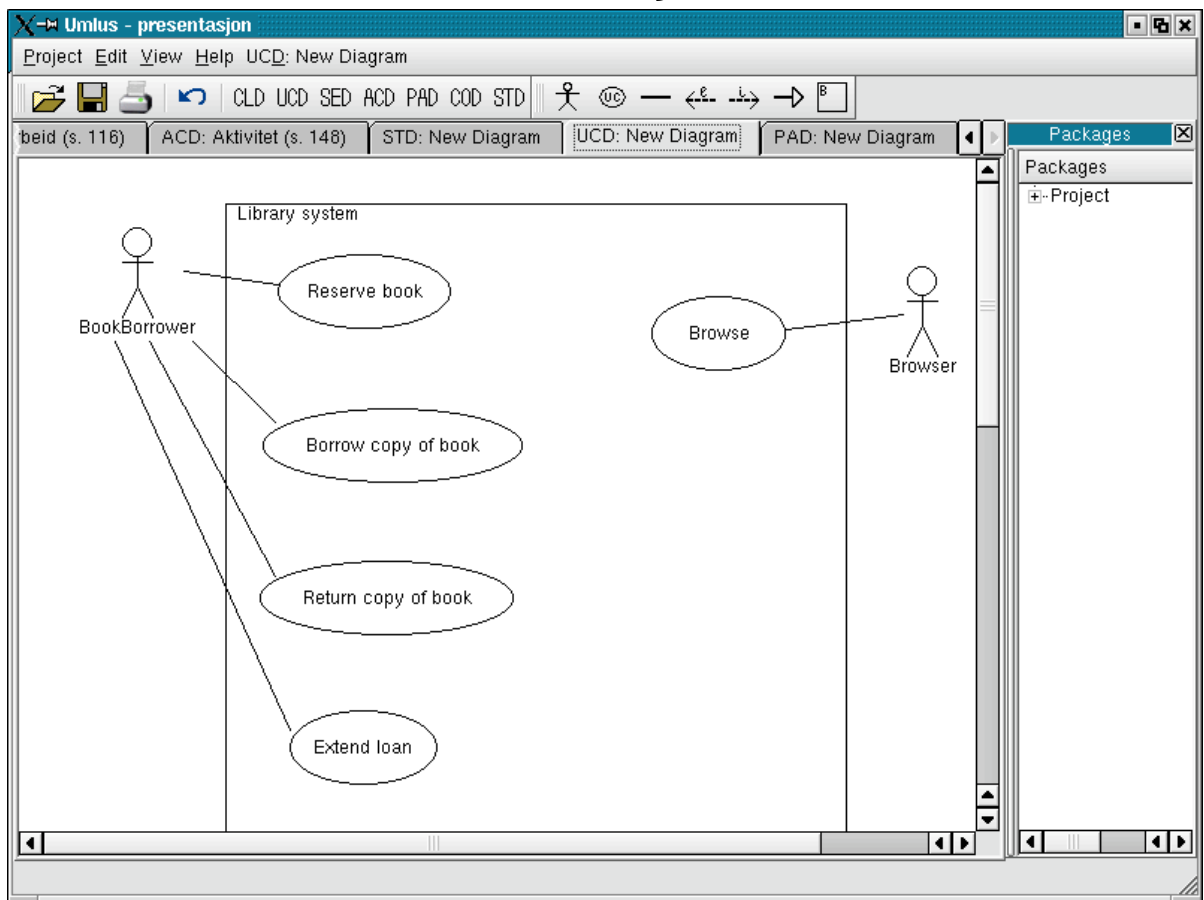
Samarbeidsdiagram



Sekvensdiagram



Tilstandsdiagram



Usecasediagram

## APPENDIX E - VEIEN VIDERE

Merk at dette primært er våre forslag visjoner og ønsker om ting som burde være med i programmet og må på ingen måte ansees som den Eneste Rette Vei

### ***Versjon 2.0 - Primært ting vi plukket vekk av tidsnød***

- Fikse alle kjente bugs (kap 1)
- Dra ut et rektangel med musen for å merke flere objekter samtidig
- Gruppér funksjon gruppér deler av tegningen slik at hvis du flytter en så følger resten med
- Cut'n'paste
- Bedre hjelpe system
- Mulighet for å skrive ut Notat/Spesifikasjons informasjonen
- La brukeren selv kunne få bestemme font type
- implementere undo på resten av prosjektet strukturen er allerede lagt opp
- Rekursjonsbokser i Sekvensdiagrammet
- Når du skal tegne flere klasser som skal arve fra samme superklasse så kan det være greit å kunne slå sammen piler
- Zoom funksjon
- drag'n'drop innad i listviewet
- Autotegning av klassediagrammer for pakker skal benytte en RyddDiagram() metode (som selvfølgelig må implementeres)
- oppdatering av assosiasjonenes ankerpunkt ved endring av fontstørrelse
- ikke vis klasse i prosjekt når prosjekt ikke er hjemmepakke
- Subklasse QListBox slik at den kan ta pekere for å gjøre feks søking mer effektivt
- Mulighet til å endre rekkefølge på metoder og attributter eventuelt kunne velge at disse skal sorteres f eks alfabetisk

### ***Versjon 3.0 - Forslag til utvidelser***

- ER diagrammer og annet for Database design
- Generere kode i java / c++ ut fra klassene som er definert i prosjektet tilsvarende generere SQL kode ut fra (E)ER diagrammer
- Import / export funksjon opp mot standard lagringsformat for UML
- tegne et prosjekt basert på eksisterende kode

### ***Senere versjoner?***

- Flytdiagrammer og andre diagrammer for Elektronikk
- Andre aktuelle diagramtyper
- lage begrensninger slik at bare ting som er lovlig kan tegnes da nærmer det seg et "intelligent" program