Forord.

TKIF er utviklet av tre studenter ved Høgskolen i Gjøvik (HiG) gjennom et hovedprosjekt våren 2003. Systemet er et støttesystem for forelesere i undervisningsøyemed.

Oppdragsgiver for prosjektet er Rune Hjelsvold ved HiG.

Formålet med rapporten er å beskrive hvordan prosjektgruppen har løst oppdraget og å hjelpe de som eventuelt vil bygge videre på dette prosjektet. Rapporten inneholder en beskrivelse av prosessen vi har gjennomført, med forklaringer, eksempler og annen dokumentasjon.

Vi vil benytte anledningen til å takke vår oppdragsgiver og veileder Rune Hjelsvold for god hjelp, spesielt under oppstart av prosjektet når vi jobbet for å sette oss inn i den nye teknologien som skulle benyttes, men også utover i prosjektløpet.

Ellers ønsker vi også å takke Øyvind Kolloen for positiv innstilling og hjelp underveis i prosjektet.

Høgskolen fortjener også takk for utlån av det utstyret vi trengte for å få et best mulig test og utviklingsmiljø.

Gjøvik 19.05.03

Torgeir Thunshelle	Roar S. Sollie	Arnstein Nydahl

Figurliste

1. Innledning

1.1. Oppgavedefinisjon

1.1.1. Bakgrunn

I dag er det er en trend at klassene det foreleses i på høgskole og universitetsnivå er relativt store. Dette fører ofte til at studentene blir passive og mange tør ikke stille spørsmål eller komme med kommentarer til foreleseren underveis. Læringsmiljøet er som regel bedre tjent med forelesninger hvor det kan føres en aktiv dialog mellom studenter og foreleser i timene. Slik kan flere sider av problemstillingene bli belyst og foreleser få tilbakemeldinger underveis, på hvorvidt studentene får med seg poengene.

Dagens teknologi med Personlige Digitale Assistenter (PDA) og trådløse nett gjør det mulig å etablere nye kommunikasjonsformer i en forelesningssal som kan supplere den verbale interaksjonen mellom foreleser og student. Det vil for eksempel være mulig å bruke PDAer tilkoplet et trådløst nett som mentometerknapper der studentene kan signalisere til foreleseren at han går for fort eller for sakte fram.

1.1.2. Beskrivelse av oppgaven

Oppgaven går ut på å lage et system som skal støtte en foreleser i kommunikasjonen med studentene i en forelesning.

Målet med TKIF er å supplere den muntlige kommunikasjonen mellom foreleser og studenter, med å gi en mulighet for anonyme tilbakemeldinger og besvarelser i tilnærmet sanntid til foreleser.

Oppdragsgiver ønsket i utgangspunktet en applikasjon til studenten og to til foreleseren. Til foreleseren var det ønskelig med en web-applikasjon til forberedelse og etterarbeid av forelesninger, for eksempel å lage ferdige oppgavesett og vurdere resultater av slike, og en applikasjon til bruk under forelesning for å motta svar og evalueringer fra studentene i tilnærmet sanntid.

Systemet skal være mest mulig plattformuavhengig og de enkelte applikasjonene i systemet skal også ha minst mulig avhengighet til hverandre hva gjelder utviklingsspråk og systemplattform.

Klientdelen av TKIF som skal ligge hos studentene skal kunne kjøres på både håndholdte PCer (PDA) og ordinære bærbare PCer, og kommunisere med foreleser over et trådløst nettverk.

De viktigste funksjonene i systemet er sending av informasjon fra studentapplikasjonen til foreleser og visning av denne informasjonen i tilnærmet sanntid hos foreleseren. På samme måte skal det være mulig å sende informasjon fra foreleserens applikasjon til studentene, også dette i tilnærmet sanntid.

1.1.3. Kompleksitet

Det ble tidlig klart at det mest interessante og også de største utfordringene med denne oppgaven ville ligge i selve systemets sammensetning, og ikke i å utvikle avanserte algoritmer for beregninger, søk og lignende.

Oppgavens egenart, sammen med oppdragsgivers krav til systemet, medførte at prosjektgruppen måtte sette seg inn i en rekke nye og ukjente teknologier. Dette innebar bruk av SOAP, XML, trådløs kommunikasjon og programmering for PDA. Oppdragsgiver stilte også som krav at når server har eller får data som skal sendes ut til foreleser eller student, så skal disse data sendes ut uten å måtte vente på forespørsel fra klientene.

For å kunne implementere SOAP og XML på PDA, som har begrenset både minne og prosessorkapasitet, brukes det egne versjoner spesielt tilpasset slike lettvektsenheter. Programmering for PDAene gav også andre begrensninger som bl.a. størrelse og oppløsning på skjerm.

1.1.4. Vår avgrensning

Kvaliteten på et system avhenger av de tre variablene tid, ressurser og hvor mye funksjonalitet som skal utvikles. Da tid og ressurser var gitt for dette prosjektet, prosjektet består av tre utviklere og skal være ferdig til en bestemt dato, var det bare funksjonaliteten prosjektgruppen kunne styre i vesentlig grad. Allerede i forprosjektet bestemte gruppen seg for å prioritere å lage en kvalitetsmessig solid plattform, hvor det skulle bli relativt enkelt å legge til ny funksjonalitet fremfor å prioritere *mye* funksjonalitet.

For å få innspill til funksjonalitet har prosjektgruppen benyttet seg av oppdragsgiver, men det er viktig å merke seg at systemet på ingen måte er ment å inneholde fult ut dekkende funksjonalitet for en foreleser. Dette krever testing over tid med systemet i reell bruk, for å få gode tilbakemeldinger fra brukere.

For å begrense oppgaven noe skal det ikke være mulig å holde flere forelesninger i samme fag på samme tid. Dette vil også være en naturlig forutsetning på de fleste høgskoler og universiteter og er besluttet i samråd med oppdragsgiver.

For å spare tid og ressurser besluttet prosjektgruppen å nedprioritere web-applikasjonen og i stedet legge inn den samme funksjonaliteten på forelesningsapplikasjonen.

1.2. Målgruppe

1.2.1. Målgruppe for TKIF

Målgruppe for TKIF vil være forelesere som ønsker å supplere den verbale kommunikasjonen med elevene, for å få flere og mer representative tilbakemeldinger og svar fra studentene, og gjennom dette øke kvaliteten på forelesningene.

1.2.2. Målgruppe for rapporten

Primært er rapporten skrevet for sensor og oppdragsgiver. Rapporten kan også være nyttig for fremtidige prosjekt som enten ønsker å utvide eller endre dette systemet, eller bare ønsker å benytte en del av den samme teknologien.

For de som ønsker å videreutvikle dette systemet kan kapittel 7 samt vedlegg M være spesielt nyttige.

1.3. Egen bakgrunn og kompetanse

Prosjektgruppen består av medlemmer fra hver av studieretningene for dataingeniør på Høgskolen i Gjøvik, herunder drift, programutvikling og systemutvikling. På denne måten har gruppen kompetanse både innen drift og oppsett av server, programutvikling i Java og C++, samt innen objektorientert systemutvikling.

Prosjektet tar, som nevnt tidligere, blant annet i bruk teknologiene SOAP, XML, samt kSoap og kXml variantene, hvilket var helt nye teknologier for alle i gruppen. Alle deltakerne i prosjektgruppen måtte sette seg inn i disse teknologiene tidlig i prosjektfasen, og lære å mestre dem underveis i prosjektløpet. Prosjektgruppen hadde også til dels liten kompetanse på flertrådsprogrammering, noe som også måtte tilegnes underveis. Fordi applikasjonen som kjører på studentklientene, ble valgt å utvikles i Java 1.1.8, måtte gruppen også sette seg inn i API-en til denne versjonen, spesielt gjaldt dette benyttelse av java.awt biblioteket i stedet for det mer vante swing biblioteket.

1.4. Organisering av rapporten

For å skille mellom begrepene som betegner servermaskinen og selve serverprogrammet i TKIF, er det gjennom hele rapporten konsekvent skilt mellom begrepene *server* og *TKIF-server*. Dette er gjennomført for å unngå forvirring rundt begrepene.

Kapittel 1 Innledning
Innledende kapittel med en overordnet beskrivelse av oppgaven med bakgrunn og egne avgrensninger. Kapittelet sier også noe om prosjektets utfordringer og gruppens sammensetning og kompetanse. Videre beskriver kapittelet kort hva de andre kapitlene inneholder.

Kapittel 2 Kravspesifisering
Inneholder systemkrav og eksempel på brukerkrav. Beskriver hvordan kravspesifiseringen ble gjennomført.

Kapittel 3 Design
Dette kapittelet inneholder beskrivelser av systemarkitektur, databasedesign og eksempler på klasse og kollaborasjonsdiagram.

Kapittel 4 Implementering
Beskriver valg av utviklingsverktøy og databaseplattform. Inneholder også eksempel på kode og beskrivelser av konkrete utfordringer vi har møtt på underveis.

Kapittel 5 Testing og kvalitetssikring
Dette kapittelet beskriver hva prosjektgruppen har utført for å ivareta den ønskede kvaliteten på systemet.

Kapittel 6 Installasjon
Hvordan konfigurere Java miljøet og installere systemet.

Kapittel 7 Videreutvikling
Inneholder en anbefaling for neste skritt i utviklingen, samt noen råd til hvordan videreutvikling bør gjennomføres.

Kapittel 8 Diskusjon
Egen vurdering av prosjektet og prosjektgruppens arbeid. Inneholder en vurdering av resultatet målt opp mot kravspesifikasjonen.

Kapittel 9 Konklusjon
Oppsummering av prosjektet og de erfaringer vi har gjort oss.

Kapittel 10 Litteratur
Oversikt over litteratur vi har benyttet underveis i prosjektet.

Kapittel 11 Vedlegg
Oversikt over alle vedleggene til prosjektrapporten.

2. Kravspesifisering

2.1. Innledning

Dette prosjektet hadde, som mange andre prosjekt, ingen klar kravspesifikasjon fra oppdragsgiver ved oppstart av prosjektet. Prosjektgruppen har ikke sett på det som verken spesielt for dette prosjektet eller som problematisk men snarer som en nyttig erfaring. Det viste seg også at selv om den viktigste funksjonaliteten ble beskrevet tidlig i prosjektet, ble mange av kravene endret noe etter hvert som funksjonaliteten ble konstruert og man fikk se det ferdige resultatet. Det vil også sannsynligvis komme flere nyttige endringsønsker og nye krav når TKIF blir testet ut i en reell situasjon og man får erfaringer med bruken av det.

I dette kapittelet beskrives systemkravene sammen med en oversikt over hvilke brukerfunksjoner som skal være med i TKIF. I tillegg er det en beskrivelse av hvordan kravspesifiseringsprosessen ble gjennomført.

Komplett kravspesifikasjon er lagt ved i vedlegg D.

2.2. Systemkrav

Kommunikasjon
Det har hele tiden vært et krav fra oppdragsgiver om at kommunikasjonen mellom studenter og foreleser skulle gå over trådløst nettverk. I tillegg var det også krav om at kommunikasjon inn til server skulle foregå over SOAP, samtidig som at server selv skulle kunne trigge en sending av data ut til studenter og forelesere.

Samtidige forelesninger
Det er ikke satt noen krav til antall samtidige forelesninger, bare at dette antallet skal kunne endres uten å måtte endre kildekoden. Antallet som er mulig vil avhenge av resursene på serveren som kjøres.

Prosjektet har sammen med oppdragsgiver satt som forutsetning at det ikke kan gå to forelesninger i samme fag samtidig.

Antall studenter
Heller ikke her er det satt krav fra oppdragsgiver om antall samtidig innloggede studenter, men antallet skal være mulig å endre uten å gå inn i kildekoden.

Meldingsformat
Oppdragsgiver ønsket også at meldinger som skulle sendes over socket, både til studenter og forelesere, skulle sendes som XML-formaterte dokumenter.

Klientplattform (student)
Applikasjonen skal kunne installeres på eksisterende operativsystem på PDA-en, og fortrinnsvis med en gratis Java VM.
Det var ønske fra oppdragsgiver at studentdelen av applikasjonen også skal kunne kjøres på en bærbar PC.

2.3. Brukerfunksjoner

Dette er et sammendrag av kravene som ble besluttet implementert i TKIF. For detaljert beskrivelse, se vedlegg D.

2.3.1. Foreleserapplikasjon

Se kapittel 3 i vedlegg D.

- Innlogging med brukernavn og passord.
  - Foreleser skal velge det faget han ønsker å forelese i før han supplerer med brukernavn/passord. Foreleser får kun velge blant de fag det ikke er innlogget andre forelesere i.
- Lage og redigere oppgavesett.
  - Et oppgavesett kan bestå av en til ti flervalgsoppgaver. Hver oppgave kan bestå av et til fem alternativer, hvorav et kan merkes som riktig. Oppgavesettene er koblet til faget foreleser er innlogget på.
- Vise kommentarer mottatt fra studentene i tilnærmet sanntid.
  - Kommentarer sendt fra studentapplikasjonen skal vises i en liste, uten at det kreves at foreleser må oppdatere listen.
- Sende URLer og kommentarer til studentene.
  - Foreleser skal ha en mulighet for å sende URLer og kommentarer til studentene fra foreleserapplikasjonen.
- Vise statistikk over evalueringer mottatt fra studentene.
  - Det skal være grafer som representerer faste evalueringsparametere. Disse parametrene er: Foreleser går for raskt frem, foreleser snakker for lavt, foreleser går for sent frem. Grafene skal vise statistikk over resultatene på evalueringene innenfor det inneværende tidsrommet, dvs. vise statistikk for eksempel inneværende halve minutt. Tidsrommet og oppdateringsfrekvens skal være mulig for driftsansvarlig eller operatør å endre. Se kapittel 3.3.6 i vedlegg D.
- Sende oppgavesett ut til studentene.
  - Foreleser skal kunne sende et oppgavesett til studentene. Hver slik utsending av et oppgavesett genererer en ny test, som skal inneholde resultatene av en gjennomføring. Testen skal inneholde oppgaveteksten og teksten til svaralternativene på det tidspunkt testen

ble generert. For hvert svaralternativ skal antallet som har svart det alternativet lagres. Testen skal lagres i databasen.

- Vise statistikk på besvarelser av oppgavesett.
  - Foreleser skal kunne hente opp enkelttester og se på statistikk over besvarelsene. Statistikken skal inneholde oppgavetekst, svartekster og prosentvis fordeling av valgte svaralternativer.
- Vise antall påloggede studenter.
  - Foreleser skal kunne se antall påloggede studenter. Dette skal oppdateres når studentene logger på og av systemet.
- Vise statistikk på besvarelser av spontane muntlige spørsmål
  - Foreleser skal kunne stille et muntlig spørsmål og angi to – tre svaralternativer. Studentene skal kunne svare på dette i applikasjonen og foreleser får da opp fordeling av statistikken på hvert enkelt svaralternativ. Statistikken skal nullstilles automatisk etter en gitt tid som skal være mulig å endre av driftsansvarlig eller operatør. Foreleser skal også selv ha mulighet til å nullstille statistikken manuelt. Dersom studentene endrer svaralternativer, skal applikasjonen registrere dette og oppdatere statistikken i henhold. Se kapittel 3.3.7 i kravspesifikasjonen i vedlegg D.

2.3.2. Studentapplikasjon

Se kapittel 4 i vedlegg D.

- Innlogging i forelesning.
  - Student skal velge fag blant de fagene hvor foreleser er logget inn.
- Automatisk mottak av utsendte oppgavesett.
  - Når foreleser sender ut ett oppgavesett, skal dette automatisk vises i studentapplikasjonen. Dersom studenten logger seg på etter at oppgavesettet er sendt ut, skal han kunne laste ned sist utsendte oppgavesett i applikasjonen.
- Automatisk mottak av utsendte URLer og kommentarer fra foreleser.
  - Ved utsending av URLer og kommentarer fra foreleser, skal disse dukke automatisk opp i studentapplikasjonen og samtidig lagres i en tekstfil.
- Skrive og sende kommentarer til foreleser underveis i forelesning.
  - Studentene skal ha mulighet til å sende fortløpende kommentarer/spørsmål til foreleser via studentapplikasjonen.
- Evaluere forelesningen underveis.
  - Det skal være lagt opp faste valg for evaluering av foreleser (går for raskt frem, for tregt frem eller snakker for lavt). Disse valgene skal være synlige i hele applikasjonen.
- Svare på oppgavesett.
  - Studentene skal kunne sende inn svar på enkeltoppgaver i

oppgavesettene. Dersom studentene endrer svaralternativ og sender besvarelsen på nytt, skal resultatet oppdateres i databasen.

- Svare på spontane muntlige spørsmål fra foreleser.
  - Dersom foreleser stiller muntlige spørsmål, skal studentene ha et fast skjema for besvarelse av muntlige oppgaver. Skjemaet skal inneholde tre valg for svaralternativer. Dersom studentene endrer svar, vil dette oppdatere statistikken vist på foreleserapplikasjonen.

2.4. Om kravspesifiseringsprosessen

2.4.1. Vurdering av brukerkravene

Vurdering av brukerkrav ble gjort på to nivåer. Et nivå hvor oppdragsgiver kommenterte brukerkravene etter at prosjektgruppen hadde detaljert beskrevet brukerkravene i usecasebeskrivelser. Det andre nivået var analysering og prioritering av de forskjellige brukerkravene i en risikoanalyse, se vedlegg J. Her ble kravene vurdert og vektet etter vanskelighetsgrad og antatt viktighet for bruker. De kravene som kom ut av analysen med høyest score, dvs. de som ble ansett for å være både viktige og vanskelige, ble prioritert først.

2.4.2. Hvordan vi spesifiserte brukerkravene

Ingen av brukerkravene var dokumentert på forhånd, så det ble mye opp til prosjektgruppen å utarbeide brukerkravene ut i fra samtaler med oppdragsgiver.

Brukerkravene ble spesifisert og dokumentert som usecasebeskrivelser. Først identifiserte vi en del alternativer for funksjonalitet som TKIF skulle tilby (usecase), deretter tok vi for oss hver enkelt funksjonalitet og dokumenterte interaksjonen mellom bruker og system (usecase beskrivelser). Hvert usecase beskriver et "happy day" scenario, som er hvordan flyten i hendelsesforløpet forventes å gå. Dersom det er andre alternativer til dette scenariet, er det dokumentert som alternativer.

2.4.3. Systemkravene

En del av systemkravene var lagt allerede i startfasen av oppdragsgiver. Dette var krav til bruk av SOAP og sanntids overføring av informasjon. Andre krav ble lagt til underveis, som for eksempel at overføring fra server til foreleser/student skulle være på XML-format.

3. Design

3.1. Innledning

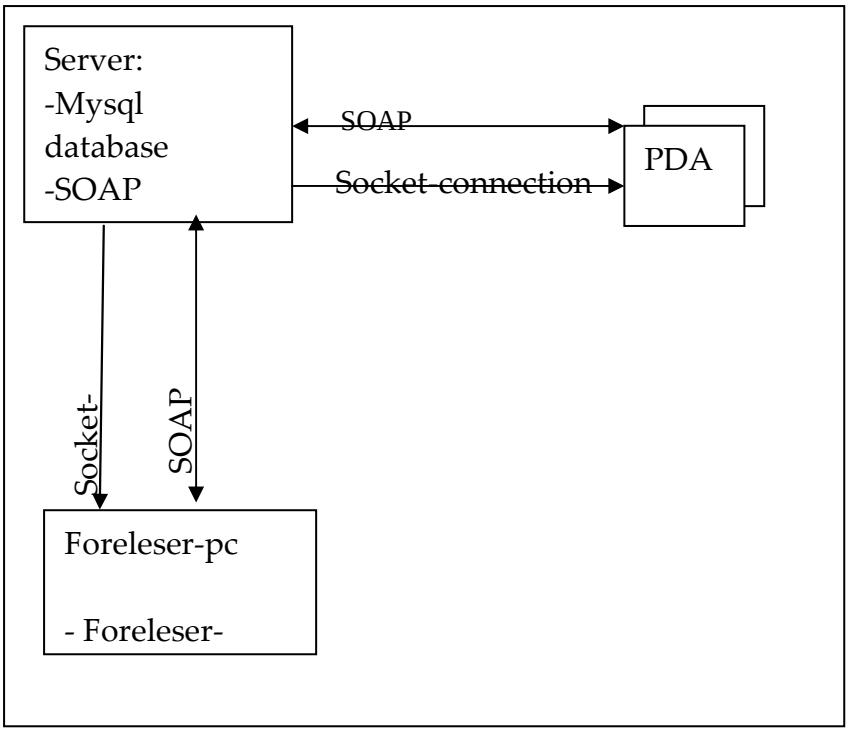
Kravspesifiseringsprosessen definerer hovedsakelig hva slags funksjonalitet TKIF skal tilby brukerne, men ikke hvordan dette skal implementeres. Det er dette som er formålet med designdokumentasjonen.

Vi har valgt å basere prosjektet på en 3-lags arkitektur hvor vi prøver å skille mellom brukergrensesnitt, applikasjonslag og databaselag.

Videre i dette kapittelet beskriver vi systemarkitektur, kommunikasjonsløsninger, eksempler på klassediagram, og databasearkitektur.

3.2. Systemarkitektur

Fordi dette er et studentprosjekt, ble det lagt vekt på å velge programvare som er fritt tilgjengelig, evt. programvare som Høgskolen i Gjøvik allerede har lisenser på.



Figur 1 Kommunikasjon PDA-Server og Foreleser-Server

3.2.1. Innledning

Systemet består av to typer klienter, en for studentene og en for foreleserne. Disse klientene kommuniserer med en server som koordinerer kommunikasjonen mellom forelesere og studenter, samt inneholder en sentral database. Databasen inneholder lagrede oppgavesett, resultater og registrerte foreleserbrukere.

3.2.2. Sikkerhet

TKIF krypterer ikke informasjonen som går over nettverket. Dette er gjort bevisst, ut i fra en vurdering om at det ikke blir overført spesielt sensitiv informasjon.

Innlogging
Prosjektgruppen vurderte kryptering av brukernavn/passord ved innlogging, ettersom overføring inn til server skjer i klartekst over HTTP. Dette kan gjøres ved å benytte Base64 koding av brukernavn og passord i SOAP headeren, eller implementere kryptering av SOAP meldingene. Kryptering av SOAP meldingene vil gjøre systemet tregere pga. økt prosesseringsbehov på klientene, samt større overhead i meldingene. Sylvain Benoist [Benoist] har skrevet en artikkel hvor han forklarer koding av kun brukernavn/passord i headeren vha. Base64 koding. Benoist hevder dette ikke er noen sikker måte å skjule passord på, da kun noen få linjer kode er nødvendig for å dekode en Base64 kodet string. Dersom det senere kommer krav om kryptering av brukernavn og passord (eller mer), anbefaler prosjektgruppen å vurdere bruk av overføring basert på SSL.

3.2.3. Kommunikasjon

TKIF baserer seg på faste IP-adresser både på server og klienter. Prosjektgruppen valgte å bruke faste IP-adresser, dvs. ikke DHCP, for å kunne ha bedre kontroll over hvem som logger seg på. Utdeling av adresser kan for eksempel skje som i dag, ved at studentene må kvittere dem ut hos IT-tjenesten. Det eneste som må konfigureres av TKIF under installasjon, er config.txt filene. Det finnes to config.txt filer, en som ligger på server og en som ligger på klientene. For mer detaljert informasjon ang. endringer av disse, se installasjonsveiledningen til TKIF i vedlegg O.

På grunn av kravet om at server skal kunne sende informasjon til klientene uten at de gjør forespørsler (se kapittel 2 i vedlegg D), valgte vi en løsning der all informasjon ut fra server til klientene går over socketforbindelser. Sending av informasjon til foreleserklienten kunne også ha vært løst med generering av SOAP-meldinger på server som sendes til foreleserapplikasjonen. Dette ville

imidlertid kreve installering av SOAP-container (Tomcat) på de PCer som skal kjøre foreleserklienten, noe prosjektgruppen vurderte å være uhensiktsmessig.

Når det ble bestemt at systemet skulle benytte både socketkommunikasjon og SOAP, bestemte vi oss for at all kommunikasjon inn til server skulle utføres med SOAP-kall, og at all kommunikasjon ut fra server skal gå over socket. Dette for å gjøre det lettere å få en oversikt over hvilket "medium" kommunikasjonen går i de forskjellige tilfellene.

Format på utgående meldinger fra server
I kapittel 2.4 i kravspesifikasjonen (vedlegg D) slås det fast at meldingene skal formateres som XML-dokumenter. Meldinger som skal sendes fra server til klientene blir derfor pakket inn i XML-dokumenter før de sendes over socketforbindelsen. Prosjektgruppen har unngått bruk av DTD og XML-skjema for validering av XML dokumentene som sendes ut fra server. Dette ble gjort for å redusere noe på kompleksiteten til systemet, da DTD og XML-skjema var noe prosjektgruppen ikke hadde kompetanse på i utgangspunktet.

Portnummer på socketforbindelsene
Portnummerene er initielt satt til 4998 og 4999 for hhv forelesere og studenter. Vi fant frem til ledige portnummer ved hjelp av en liste på iana.org [IANA]. Disse kan enkelt endres i konfigurasjonsfilene, se vedlegg O for detaljer.

Server - PDA
Kommunikasjonen mellom server og PDA foregår på trådløst nett over IEEE 802.11b. Dette ble valgt på bakgrunn av krav fra oppdragsgiver om trådløs tilknytning av studentene, jfr. systemkravene i kravspesifikasjon i vedlegg D.

Server – Foreleser
Denne kommunikasjonen foregår på "vanlig" LAN over TCP/IP. Dersom det er ønskelig kan også foreleseren tilknyttes ved hjelp av trådløst nett. Dette er bare et spørsmål om å installere og konfigurere den aktuelle PCen med trådløst nettverkskort.

3.2.4. Server

Operativsystem
Ved valg av operativsystem vurderte vi Linux og Windows plattformer. Ettersom Linux er en typisk serverplattform, er kjent for å være meget stabil, og at det i tillegg finnes flere gratis Linux distribusjoner, valgte prosjektgruppen Linux som serverplattform. Med i vurderingen var også et

ønske fra prosjektgruppen om å få mer erfaring med bruk av UNIX operativsystem.

Databaseplattform
Vi valgte Mysql som databaseplattform fordi dette er en vel utprøvd og ikke minst gratis databaseplattform.

SOAP
Som SOAP container installerte vi Apache Tomcat og benyttet Apache SOAP v2.3.1 som Java-bibliotek for å lage SOAP tjenester.

TOMCAT
Variabelen "session-timeout" kan settes i filen "tomcat/conf/web.xml". Denne variabelen styrer timeout tiden på http sesjonene og er initielt satt til 30 minutter som standard. Denne kan justeres etter ønske.

Antall samtidige forbindelser
I kravspesifikasjonen (vedlegg D) er det beskrevet krav om at antall samtidig oppkoblede studenter og forelesere skal være mulig å endre uten å endre kildekoden. Dette er løst ved hjelp av konfigurasjonsfiler som applikasjonene leser ved oppstart. Dersom det er ønskelig å endre for eksempel maks antall samtidig påloggede studenter eller hvilken port disse skal benytte på socketforbindelsen, kan dette enkelt endres i konfigurasjonsfilen som er en tekstfil. For mer detaljert informasjon om dette, se installasjonsveiledningen i vedlegg O. Maksimalt antall studenter og foreleser er i utgangspunktet satt til hhv 1000 og 100.

3.2.5. Klient applikasjon (PDA)

Operativsystem
Prosjektgruppen avgjorde i dialog med oppdragsgiver at det ikke var fornuftig å kreve at studenter som ønsker å benytte seg av denne applikasjonen må installere et annet operativsystem enn det som er originalt på deres enhet.
Operativsystemet på PDA-ene innkjøpt til prosjektet var PocketPc 2002.

Java VM
For å kunne kjøre Java applikasjoner på PDA, trengs det en Java VM. I prosjektet installerte prosjektgruppen JeodeRuntime, et Personal Java kompatibelt kjøretidsmiljø, som fulgte med de innkjøpte PDA-ene. Det finnes flere andre Java VM-er som kan installeres, men prosjektgruppen lyktes ikke i å finne noen som var freeware og samtidig kunne kjøres på originalt

operativsystem. Hovedforskjellen fra 1.1.8 [1.1.8] til 1.3.1. [1.3.1] er at swing biblioteket ikke eksisterer i 1.1.8 og at man derfor må benytte seg mye av awt biblioteket til GUI programmeringen.

Java bibliotek
JeodeRuntime var bare kompatibelt med Java versjoner opp til 1.1.8, derfor er dette biblioteket benyttet under konstruksjon av studentapplikasjonen. PDA-er har merkbart mindre ressurser, bl.a. minne og prosessorkraft, enn vanlige datamaskiner. På grunn av dette valgte prosjektgruppen å legge inn lettvekts biblioteker for håndtering av SOAP og XML. Dette er kSoap og kXml fra Enhydra.org [Enhydra]. Disse bibliotekene er "open source" og spesialtilpasset J2ME teknologien [J2ME] og dermed mindre ressurskrevende.

Nettverk
For nettverkstilknytning av studentapplikasjonen benyttes Compaqs eget WL110 Wireless PC Card som installeres i "jakken" til PDAen.

3.2.6. Foreleser applikasjon (PC)

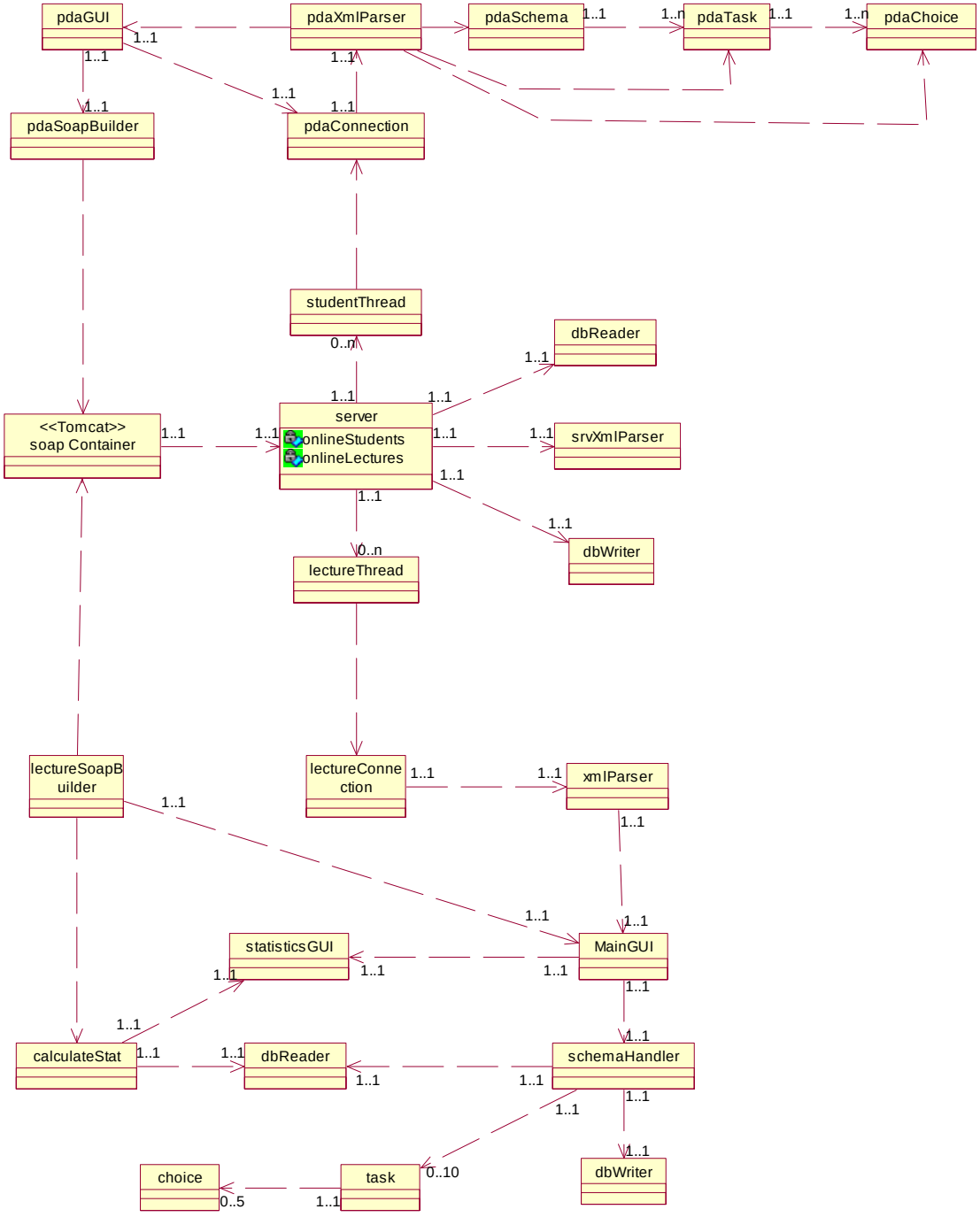
Operativsystem
TKIF er utviklet og testet på Windows XP, men er utviklet i Java 1.3.1 og skal kunne fungere på alle operativsystemer som kjører en J2SE kompatibel Java VM.

Oppdateringsfrekvenser på grafer
I grensesnittet for statistikk skal det i følge kravspesifikasjonen kapittel 3.3, være mulig å endre oppdateringsfrekvensene på visning av grafer over evalueringer og statistikk på tavlespørsmål. Dette kan gjøres ved å endre verdiene i konfigurasjonsfilen, se vedlegg O for detaljer.

3.3. Domenemodell

Domenemodellen (Figur 2) er en konseptuel modell av systemet, slik prosjektgruppen så det for seg tidlig i prosjektfasen, før designfasen begynte. Den ble videre oppdatert noe i designfasen etter hvert som nye problemstillinger meldte seg.

Målet med modellen er først og fremst å få visualisert systemets virkemåte og grovt hvilke komponenter systemet skal bygges opp av. Til tross for at domenemodellen har vært et hjelpemiddel i analysefasen, har vi valgt å sortere den under design da den forteller mer om hvordan systemet ser ut enn hva det gjør. Det er imidlertid viktig å merke seg at det ikke er noen direkte sammenheng mellom klassene i domenemodellen og klassene i designmodellen.



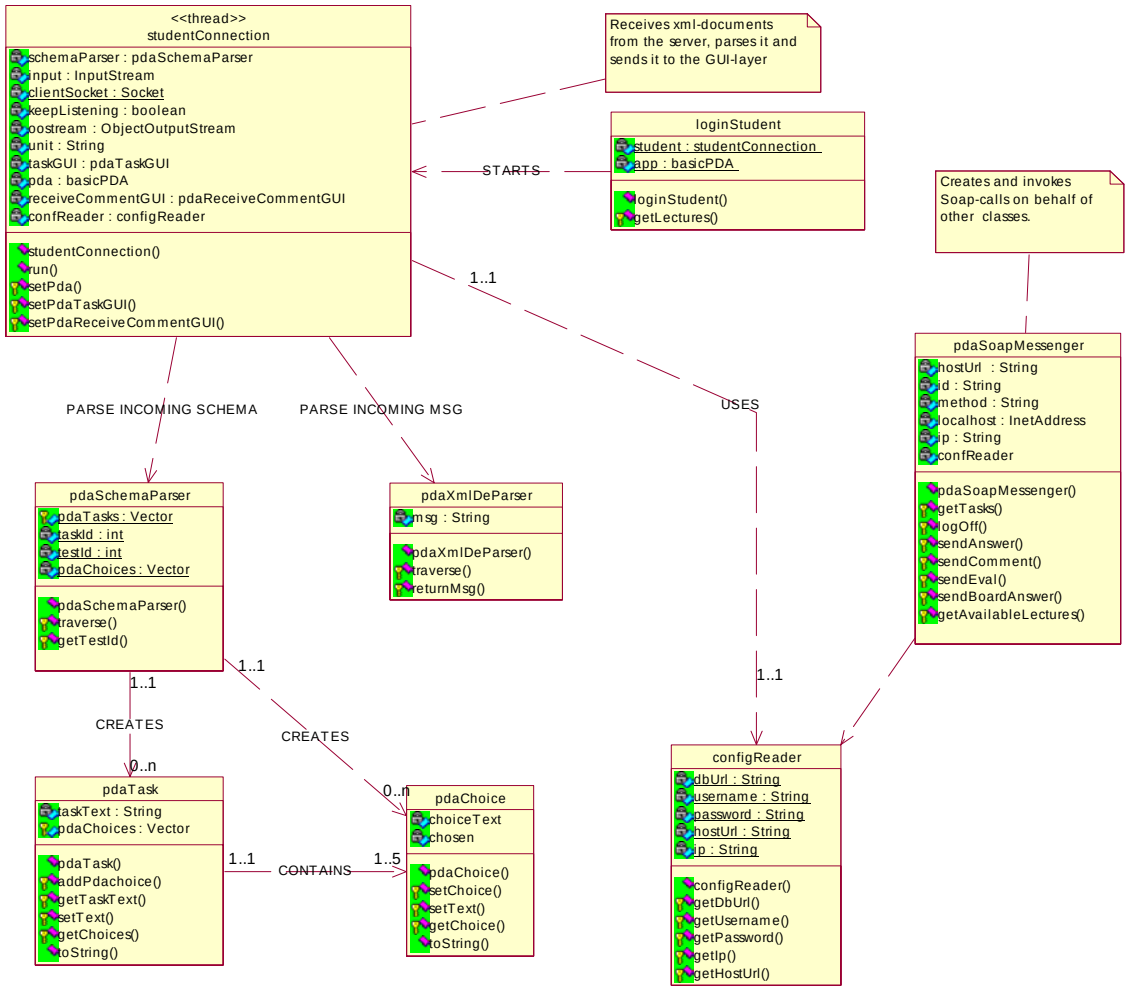
Figur 2 Domenemodell TKIF

Kort forklaring til domenemodellen (Figur 2 over**):**
Server:
Server består av klasser for lesing fra og skriving til databasen, klasse for generering av XML-dokumenter, klasser for å kommunisere med klientene over socket og en container som tar i mot innkommende SOAP-meldinger og mapper disse mot riktige klasser/objekter og deres funksjoner. All overføring av data ut til klientene fra server, går over socketforbindelsene, mens innkommende overføringer kommer som SOAP-meldinger over http til SOAP-containeren i Tomcat.

Klientapplikasjonene:
På klientene er det egne klasser for å håndtere socketkommunikasjonen og for å håndtere SOAP-kommunikasjonen med server. Klientapplikasjonene har også egne klasser for å parse de innkomne XML-dokumentene. I tillegg har disse applikasjonene også egen datastruktur for å representere oppgavesett. Denne strukturen består av klassene schema (container-klasse for oppgaver), task (oppgave) og choice (svaralternativ). Foreleser klienten har to GUIer, en for hovedapplikasjonen (mainGUI) og en for visning av sanntids informasjon fra studentene (statisticsGUI).

3.4. Klassediagram

Komplette klassediagrammer med forklaringer er lagt ved i vedlegg E. Vi valgte å legge klassene i tre forskjellige pakker. En pakke for server, en for studentapplikasjonen og en for foreleserapplikasjonen. Klassediagram for applikasjonslaget på studentapplikasjonen er vist i Figur 3 under.

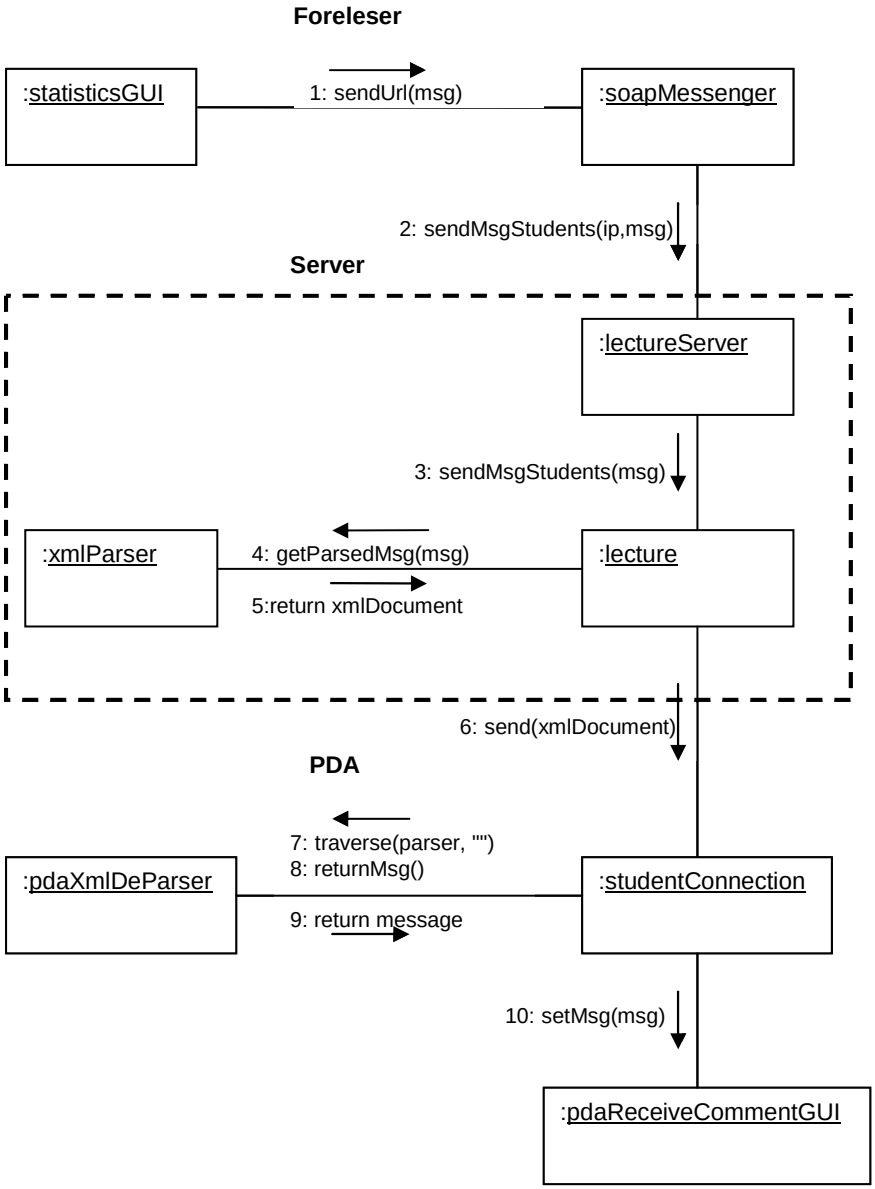


Figur 3 Klassediagram for applikasjonslaget på student applikasjonen

Kort forklaring
Vi ser klassen "studentConnection" som får alle innkommende XML dokumenter og videresender dem for konvertering avhengig av hvilket dokument de får inn. "StudentConnection" blir i sin tur startet av "loginStudent"-klassen som foretar innloggingen. "ConfigReader"-klassen henter opp IP-adressen og hostURL til SOAP-serveren fra en konfigurasjonsfil, slik at de andre klassene kan hente inn disse dataene ved behov. "PdaSoapMessenger" er klassen som håndterer all utgående kommunikasjon mot server, dvs. genererer og sender alle SOAP-kall.

3.5. Kollaborasjonsdiagram

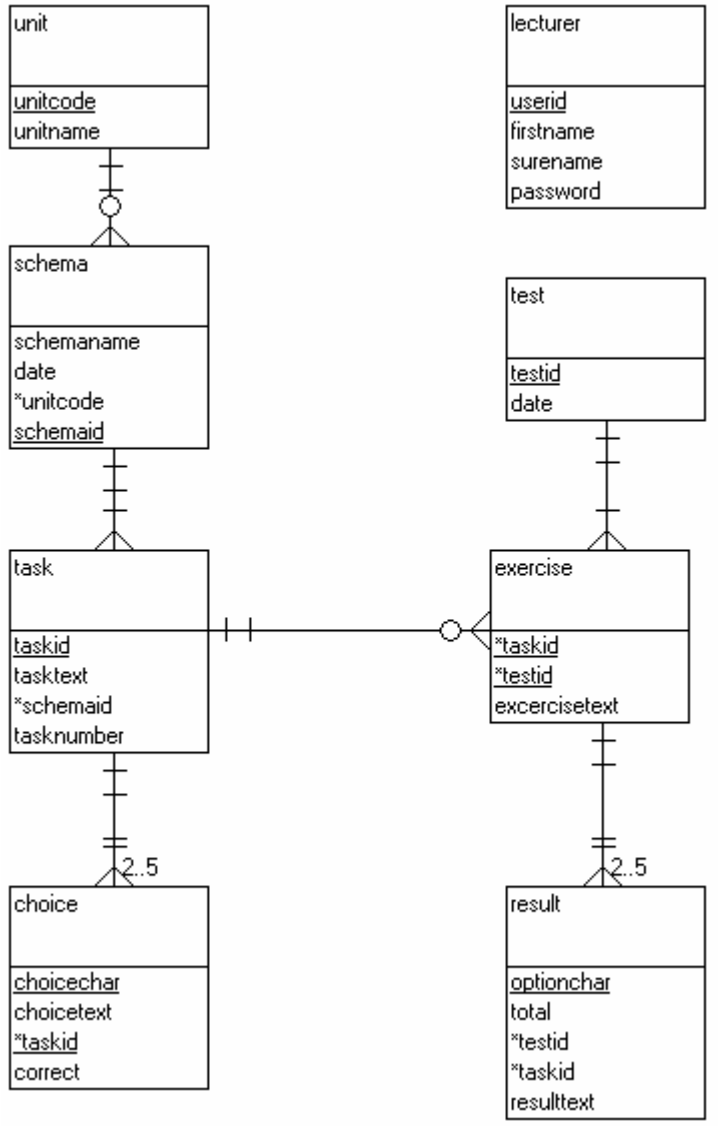
Kollaborasjonsdiagrammene, se vedlegg T, er tegnet for å gi en oversikt over hvordan de viktigste funksjonene brukerne har tilgang til er implementert i TKIF. Et eksempel på et slikt diagram er vist i Figur 4 under. Dette diagrammet viser hvordan funksjonen for å sende en Url fra foreleser til studentapplikasjonen er implementert.



Figur 4 Kollaborasjonsdiagram for sending av URL

3.6. Databasedesign

ER-modell og databaseskript er lagt ved i vedlegg F. Figur 5 under viser ER-modellen.



Figur 5 ER-modell

Et schema er et oppgavesett, og inneholder flere oppgaver (task). Hver oppgave inneholder oppgavetekst og to til fem alternativer (choice). Hvert alternativ består av bokstav (a til e), alternativtekst, samt en variabel for å indikere om alternativet er riktig eller galt.

Alle fag som skal kunne benytte dette systemet må registreres i tabellen unit med fagkode og fagnavn. Hvert fag (unit) kan inneholde flere oppgavesett.

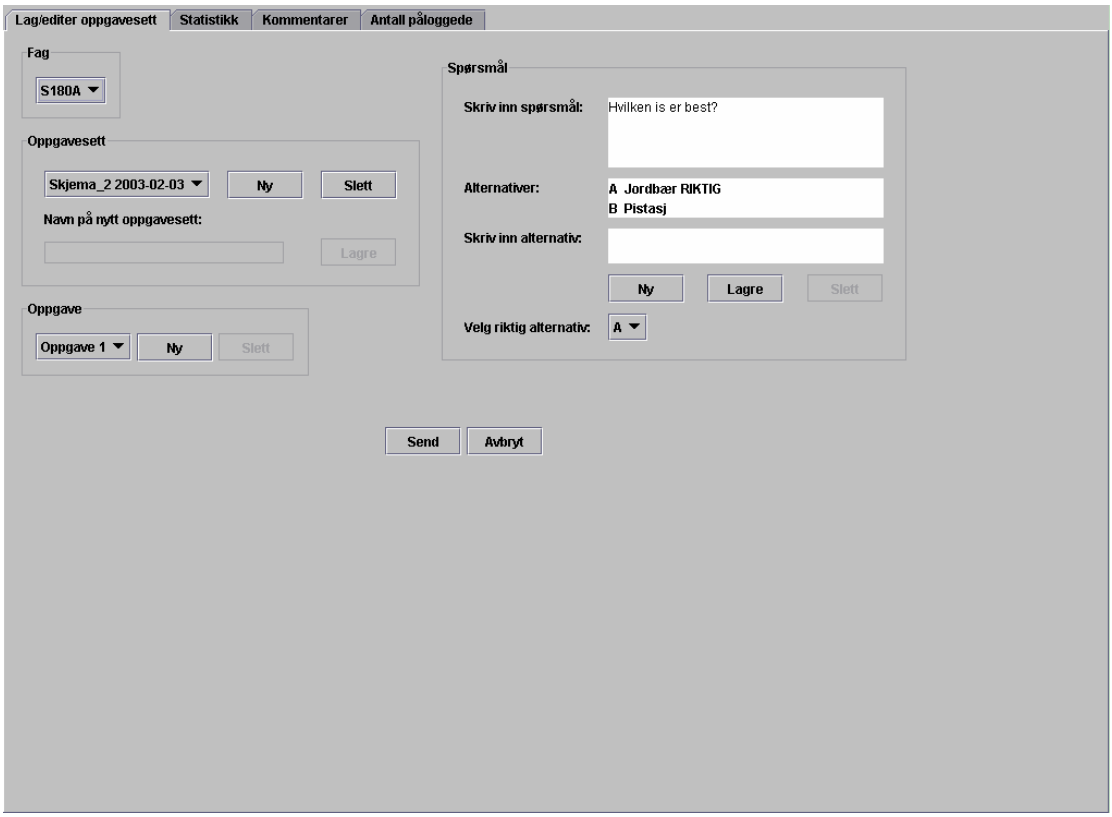
Testtabellen er en oversikt over alle prøver som er gjennomført. Hver gang et oppgavesett sendes ut til studentene, genereres det en ny test (prøve). Hver test inneholder en excercise for hver oppgave (task) og hver excercise

inneholder like mange resultat (result) som korresponderende oppgave inneholder alternativer. Ved generering av en ny test, lages det altså en excercise for hver task og det genereres riktig antall result. Result inneholder en variabel for å telle opp hvor mange som har svart det aktuelle alternativet på en oppgave. På denne måten kan samme prøve gjennomføres to eller flere ganger og man kan i etterkant sammenligne statistikken for de forskjellige gjennomføringene for eksempel for å se etter forbedringer.

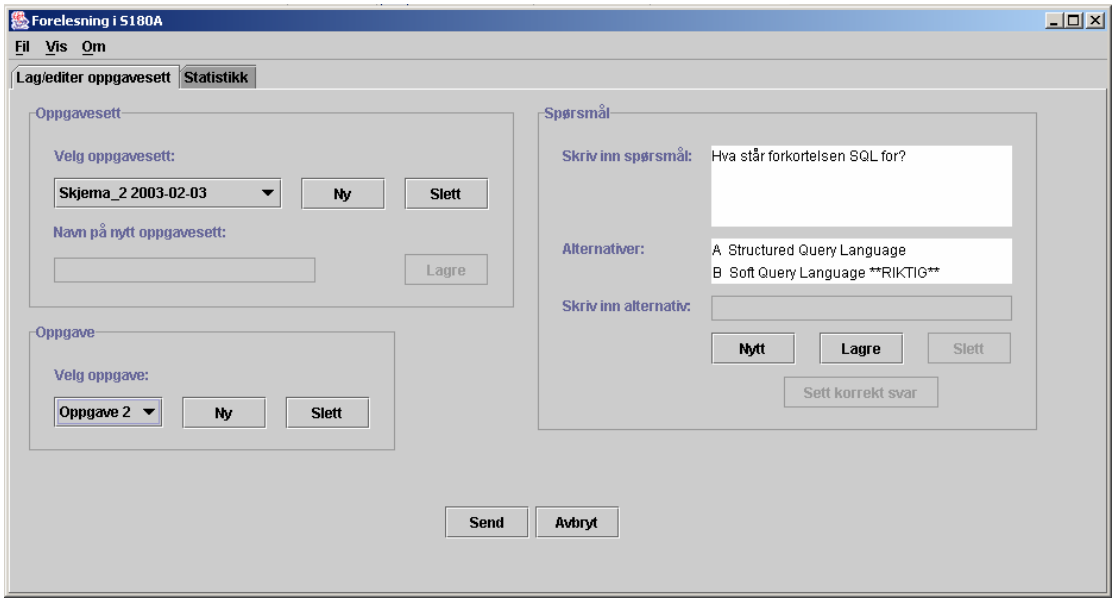
Tabellen lecturer inneholder navn og brukernavn/passord for de foreleserne som er registrert i TKIF. Tabellen brukes til validering av brukernavn passord ved pålogging av forelesere.

3.7. GUI design

Både under kravspesifiseringen og under design av brukergrensesnittet lagde vi mockups for å lettere visualisere funksjonalitet og utseende. Et eksempel på mockup er vist i Figur 6 under og er forslaget til skjermbildet av oppgaveredigering som er vist i Figur 7 under. Målet med mockupene var ikke å vise detaljert hvordan grensesnittet skulle se ut, men heller å klargjøre hvilke funksjoner bruker skulle ha tilgang til og hvordan det kunne se ut. Prosjektgruppen fant bruk av mockups meget nyttig, spesielt ved design av de mer kompliserte skjermbildene og funksjonene.



Figur 6 Mockup



Figur 7 Skjermbilde

Mocups og faktiske skjermbilder fra foreleser og studentapplikasjonene er lagt ved i hhv. vedlegg L og vedlegg K.

3.8. Bruk av verktøy til design

For design av klassediagram har prosjektgruppen benyttet Rational Rose Enterprise Edition. For å tegne mockups benyttet vi stort sett Delphi, men i mange tilfeller brukte vi også helt enkle blyantskisser for illustrering av skjermbilder.

For databasedesign benyttet vi Modelator v.4.0, som også genererte databaseskriptet for oppretting av databasen. Det er viktig å merke seg at skriptet ikke er ferdig til bruk etter autogenerering i Modelator, men må modifiseres. Modifiseringen gjelder syntaks på kommentarlinjer, innlegging av auto_increment på primærnøkler, evt. flere indekser, samt eventuelt mer kommentering utover det Modelator selv legger inn.

4. Implementering

4.1. Utviklingsspråk

Prosjektgruppen valgte å begrense valget av utviklingsspråk til de språk vi kjente til fra før. Dette ble gjort ut fra en vurdering om at prosjektet likevel medførte en del utfordringer med tanke på ukjente teknologier for prosjektdeltakerne, bl.a. bruk av SOAP, XML og programmering for håndholdte PCer.

Alternative språk som ble vurdert var C++, Java og Delphi.

4.1.1. TKIF-server

Valget falt på Java, fordi dette er et plattformuavhengig språk og TKIF-serveren kan da uten videre kjøres både på Linux og Windows installasjoner.

Ettersom ingen stabil versjon av Java 1.4.1 enda var utgitt for Debian sitt pakkesystem ved beslutningstidspunktet tidlig i prosjektfasen, falt valget på Java 1.3.1.

4.1.2. Foreleserapplikasjon

Vi valgte Java som utviklingsspråk av de samme grunner som for TKIF-serveren. I tillegg hadde prosjektgruppen allerede litt erfaring i utvikling av vindusapplikasjoner i Java i motsetning til i C++. For ikke å blande flere versjoner av Java, valgte vi også her versjon 1.3.1.

Vi valgte å ikke utvikle dette som en webapplikasjon ut fra ønsket fra oppdragsgiver om sanntidsoppdateringer uten bruk av timere. Kravet, som beskrevet i vedlegg D, var at TKIF-serveren skulle kunne ta initiativ til å sende data ut til klientene uten at klienten selv gjorde en forespørsel. En webapplikasjon kunne imidlertid ha håndtert de andre funksjonene som å lage/redigere oppgavesett og se på resultater fra gjennomførte prøver. En slik webapplikasjon kan være et nyttig tilleggsverktøy for forberedelse og etterarbeid av forelesninger.

4.1.3. Studentapplikasjon

Å velge utviklingsspråk for studentklienten, var det som var mest utfordrende. Her måtte vi i tillegg ta hensyn til relativt lav prosesseringskraft og lite minne. Det stod mellom C++ og Java, hvor C++ trolig ville gitt den raskeste applikasjonen og være det ”tryggeste” valget i så måte. Den siste tiden har det imidlertid kommet flere Javabibliotek spesielt beregnet for denne kategorien av lettvektsenheter (PDA og lignende), som vi ønsket å prøve. Videre er som nevnt Java et plattformuavhengig språk, noe som støtter

opp under kravet i kapittel 2 i kravspesifikasjonen (vedlegg D), og valget falt av disse grunner også her på Java.

4.2. Utviklingsverktøy

Tidlig i prosjektfasen gjorde prosjektet tester på bruk av Forte for Java, men maskinene prosjektgruppen har på utlån fra IT avdelingen var ikke kraftige nok til å kjøre dette programmet. Kildekoden er derfor skrevet i Editplus og Kate (Linux). Bruk av disse relativt enkle kodeeditorene var uproblematisk og gav kanskje også bedre oversikt under kodingen, enn avanserte verktøy som Forte.

For utvikling av database og databaseskript benyttet prosjektgruppen Modelator versjon 4.0 [Modelator]. Til tross for databaseskriptet genereres automatisk fra Modelator, må det foretas visse endringer i skriptet før det kjøres i databasen. Dette er endringer som blant annet å definere "auto_increment" i MySQL, avslutning av kommentarer, skrive ekstra kommentarer og oppretting av ekstra indekser.

4.3. Valg av databaseplattform

Prosjektet ønsket å benytte seg av en SQL-kompatibel databaseplattform, med støtte for datohåndtering og autogenerering av indekser.

Som nevnt tidligere, valgte prosjektgruppen MySQL som databaseplattform da dette er gratis programvare som tilfredsstiller de behov prosjektet hadde til databaseplattform.

4.4. Kodestandard

All koding og kommentering er utført på engelsk, for å gjøre kildekoden mest mulig internasjonal. Alle stringer som vises i applikasjonene er lagt i klassevariabler, slik at det er relativt enkelt å gå gjennom koden og oversette alt til et annet språk.

Prosjektet bestemte ingen fast struktur på navnsetting av variabler, funksjoner og klasser i startfasen (uten om at navnsetting skulle være engelsk). I etterkant ser vi at det nok burde ha vært gjort en bedre jobb på dette innledningsvis i prosjektet. Vi har imidlertid vært bevisste på å bruke god og forklarende navngiving, slik at kildekoden skal likevel være godt leselig.

Kildekoden er skrevet med tabulator som innrykk og vises best ved å sette tabulatoravstanden til 2 i editoren som brukes ved editering eller lesing av kildekoden.

4.4.1. Kommentering

Alle kildefilene ble kommentert etter samme mal. Øverst i hver fil ligger en standard heading og en kort beskrivelse av hvilken oppgave den klassefilen har. Alle funksjoner er også kommentert slik at alle argumenter og retur variable skal være dokumentert (med @params og @return). I tillegg er det også noe kommentering inne i funksjonene, der vi mener dette er nødvendig for å gjøre koden mer leselig.

Med denne måten å kommentere på, får man en god og oversiktelig dokumentasjon vha. javadoc. Komplett kildekodedokumentasjon generert av javadoc, ligger vedlagt på CD.

Heading i toppen av hver kildefil er skrevet etter følgende mal:

```
/**
* Classname:              "klassenavn"
* Packagename:            "pakkenavn"
* Version information:    "versjon av filen"
* Date:                   "dato sist modifisert"
* Author:                 "navn på medlemmene i prosjektgruppen"
* See project:            "link til prosjektets hjemmeside"
*
* "Kort beskrivelse av klassefilen".
*/
```

4.4.2. Eksempel på kode

Følgende kildekode er hentet fra klassen configReader (på studentklienten), som leser en konfigurasjonsfil med hostURL, IP-adresse og portnummer til serveren. Andre klasser benytter da denne klassen for å få tilgang til disse parametrene.

```
package no.hig.hovedprosjekt.pda;

import java.io.*;

/**
 *  Classname:                  configReader<BR>
 *
 *  Packagename:            no.hig.hovedprosjekt.pda<BR>
 *
 *  Version information:    1.0.0
 *
 *  Date:                   25.03.2003 <BR>
 *
 *  Copyright notice:       <BR>
 *
 *  Author:             Arnstein Nydahl, Torgeir Thunshelle og Roar S.
Sollie<BR>
 *
 *  See Project:
<AHREF="http://hovedprosjekter.hig.no/v2003/data/gruppe5/">
Trådløs kommunikasjon i klasserom </A>
 */

/**
* Class that reads a config-file consisting of the IP-address, hostURL and
portnumber
* to which the socketconnection is supposed to connect to.
*/
public class configReader{
   private static String ip, hostUrl, port;
   private final static String  filename = "config.txt";
```

```
  /**
  *
  * Constructor Initializes the dbReader and reads ip, and hostUrl from the
given file.
  *
  */
  protected configReader(){
    BufferedReader buffer;
    try{
    File file = new File(filename);
      if(file.exists() ){
        buffer = new BufferedReader( new FileReader(file) );
        ip = buffer.readLine();
        hostUrl = buffer.readLine();
        port = buffer.readLine();
      }
    }catch(Exception e){
      System.out.println("Error reading from file " + e);
    }
  }

  /**
  *
  *       @return the servers ip
  *
  */
  protected String getIp(){
   return ip;
  }

  /**
  *
  * @return the soap-hostUrl
  *
  */
  protected String getHostUrl(){
    return hostUrl;
    }
```

```
  /**
  *
  * @return the portnumber for the socketconnection
  *
  */
  protected String getPortnumber(){
    return port;
  }
}
```

4.5. Utfordringer underveis

Når vi ser bort fra generelle utfordringer, som å sette seg inn i ny teknologi, finne brukerkrav og gode designløsninger, støtte vi også på noen utfordringer ved selve kodingen av programfilene og de praktiske løsningene. I dette kapittelet forsøkes det å gi et bilde av de fleste av disse utfordringene.

4.5.1. Serverløsning

Prosjektet brukte mye tid på å få på plass en stabil og funksjonell TKIF-server, da dette dannet grunnlaget for å gjøre den videre utviklingen av funksjonalitet enklere.

Rekompilerte klassefiler

Noe av problemet i startfasen, var at vi ikke visste at Tomcat ofte måtte restartes etter rekompilering av kildefiler. Dette skulle i utgangspunktet heller ikke være nødvendig, da Tomcat er konfigurert til å automatisk laste rekompilerte klasser på nytt. Dette fungerer imidlertid ikke så lenge klassene for eksempel lytter på socket, eller er session/application objekter i SOAP-containeren. Dette kommer trolig av at objektene inneholder statiske variabler og socket oppkoblinger som de mister referansene til når de nye objektene blir lastet inn. For å unngå en del feilkilder innførte vi som rutine å restarte Tomcat for hver gang TKIF-serveren skulle kjøres etter rekompilering av kildefiler på serveren. Dersom applikasjonen ikke er startet mellom kompileringene vil det imidlertid ikke være noe problem for Tomcat å holde kontroll på de nye klassene.

SOAP-containeren og Java VM

Det er verdt å merke seg for fremtidige prosjekter at SOAP-containeren kjører sin egen Java VM. Dette betyr at man ikke kan starte TKIF-serveren "manuelt" ved å bruke "*java no.hig…klassenavn*" og så få tilgang til de samme objektene vha SOAP-kall. Dette er grunnen til at TKIF-serveren startes med innkommende SOAP-kall.

4.5.2. SOAP-kall på PDA

kSoap er beregnet for J2ME og J2SE og kunne således ikke direkte benyttes i Java 1.1.8. Problemet var at for å sette opp et kall over http, gjøres dette vha. klassen HttpTransportSE i J2SE. For å benytte denne i 1.1.8, måtte den modifiseres noe. Endringen kan sees i kildekoden til den modifiserte HttpTransportSE i vedlegg S.

Et annet problem på PDA-en er at det *kan* ta opp til to sekunder fra det settes opp et SOAP-kall til det kan settes opp et nytt. Prøver man å sette opp et nytt kall for tett etter det forrige, kastes en NullPointerException og kallet settes ikke opp. Prosjektgruppen har håndtert dette ved å gi tilbakemelding til bruker om at operasjonen ikke var vellykket og beskjed om å prøve på nytt. Dette er imidlertid ingen fullkommen løsning, da det forstyrrer brukeren unødig. Et forslag til hvordan dette kan utbedres er beskrevet i kapittel 7.2.1.

4.5.3. XML håndtering på studentapplikasjonen (PDA)

De kjente bibliotekene for XML håndtering i Java er alle relativt store og krevende når det gjelder minneresurser. For å lage en mest mulig effektiv applikasjon på PDAen brukte prosjektet tid på å søke etter XML-parsere som var mindre og mer effektive. Dennis Sosnoski har skrevet en artikkel på ibm.com [Sosnoski] om ytelse på XML-parsere. Her kommer det frem at pullparsere er overlegne bl.a. når det gjelder minnebruk og dokument byggetid. På grunn av dette fant vi frem til kXml [kXml], et Java-bibliotek beregnet for J2ME, hvor XML-parsing gjøres med en pullparser.

4.5.4. Norske spesialtegn

Det oppstod flere utfordringer med det å få frem de norske tegnene æ ø og å, som i enkelte situasjoner ikke ble vist riktig på skjerm hos PDA og foreleser klient.

Melding fra foreleser til student
Her oppstod det første problemet da æ, ø og å i meldinger, sendt fra foreleser via server til PDA, ikke ble vist riktig på PDA. For å løse dette måtte karaktersettet settes til UTF-8 på inputStreamReader klassen. Standard karaktersett på denne klassen i Java 1.1.8 er cp1252 som ikke håndterte de norske tegnene på PDAen.

Hente tekst fra databasen på server til en klient
I dette tilfellet, hvor aktuell tekst inneholdt et av de norske spesialtegnene, vil ikke disse komme riktig ut til klientene. Grunnen er at locale settingene på Linux må settes til no_NO. For å ta høyde for at dette ikke skal være

nødvendig, har prosjektgruppen omgått problemet ved å angi karaktersett (ISO-8859-1) når tekst hentes fra databasen og legges inn i Stringer.

4.5.5. Skjermstørrelse på PDA

Størrelsen på Compaq's iPaq, er 300 * 240. Dette gir en begrensning på hvor mye som kan vises i applikasjonen. For eksempel så er det ønskelig å se en hel oppgave med alle alternativer og send knapp i et skjermbilde. Dette begrenser mengden tekst som kan vises på oppgaven og alternativene betraktelig. For å unngå problemer med visning, som også er et krav i kapittel 2.5 i kravspesifikasjonen, har prosjektgruppen begrenset tekstlengden til 254 tegn i oppgaveteksten og 35 tegn i hvert alternativ. I tillegg kan man ikke legge inn mer enn 10 oppgaver, da dette vil ødelegge visningen av knappene for valg av oppgaver. Dette siste problemet har prosjektgruppen et forslag til løsning på i kapittel 7.2.2.

4.5.6. Innloggingstid

Tiden det tar fra applikasjonen startes til innloggingsbildet kommer opp, samt tiden fra login knappen blir trykket til man er innlogget tar noe lang tid på en PDA av den typen prosjektgruppen disponerer. Noe av grunnen er at Jeode (Java VM) bruker en del tid på å starte opp, men det er også tydelig at det å laste applikasjonen opp i minnet krever mye av PDAene.

Prosjektet har ikke prioritert å bruke tid på å se på effektivisering av oppstart/innloggingstid, da applikasjonen ellers har tilfredsstillende responstid.

5. Testing og kvalitetssikring

5.1. Innledning

Prosjektgruppen har lagt vekt på å konstruere et kvalitativt godt produkt med en stabil kjerne hvor det er enkelt å legge til ny funksjonalitet. Vi vurderte det slik fordi funksjonalitet får man best innspill på ved å teste ut systemet i forelesninger, noe prosjektgruppen har manglet både utstyr og tid til å gjennomføre.

Vi brukte flere teknikker for å innfri ønsket om høy kvalitet på produktet. Prosjektgruppen satset på testing, å finne beste praksis for utvikling av ny funksjonalitet, dokumentering, parprogrammering, samt konfigurasjonskontroll for å i møtekomme ønsket kvalitet.

5.2. Modultesting

Med modultesting menes test av en eller flere funksjoner eller klasser, før de integreres i eksisterende kode. Mange av disse testene spesifiserte vi på forhånd og kjørte etter at aktuell funksjon eller klasse var kompilert. Disse dokumenterte testene er beskrevet sammen med resultatet av testene i vedlegg R. Alle tester er ikke dokumentert, men å beskrive testscenarioet før koding har vist seg å være en god måte å jobbe etter og kan anbefales for videre utvikling.

5.3. Integrasjonstesting

Etter hvert som modultestene ble gjennomført og klassefilene så ut til å fungere, integrerte vi dem i eksisterende kode. Når dette var gjort, kunne vi teste funksjonaliteten som var utviklet og se om den fungerte sammen med det vi hadde utviklet så langt. Når det testes slik etter hvert som modulene integreres, er det ofte lettere å se hvilke moduler som fører til feilsituasjoner. På denne måten blir det også lettere å finne feilen og rette den opp.

5.4. Systemtesting

Med systemtesting mener vi testing av TKIF med begge klientapplikasjonene og TKIF-serveren. Det er ved disse testene hele systemet blir testet for å avdekke flest mulig feil og endringsønsker. I tillegg til at prosjektgruppen ved flere anledninger testet systemet etter hvert som vi endret på koden, ble det også foretatt en systemtest med eksterne testpersoner.

Systemtesten hvor vi testet TKIF på eksterne testpersoner, ble godt forberedt og dokumentert. Testen hadde til hovedhensikt å avdekke flest mulig feil i systemet, men også å undersøke brukervennligheten til systemet. Testpersonen fikk en kort introduksjon til hva TKIF var og hva som skulle skje

på testen, men ingen innføring i bruken av applikasjonen. Medlemmer fra prosjektgruppen fulgte med hver testdeltaker og noterte hvordan testene gikk. Testen hadde også en modul hvor deltakerne fritt kunne "klikke seg frem" i programmet for å prøve å fremprovosere flere feil. Til slutt fylte testdeltakerne ut et tilbakemeldingsskjema, hvor de kunne peke på problemer de støtte på, endringsønsker og andre kommentarer. For resultater på denne testen, se vedlegg R. Alle registrerte feil ble dokumentert i egne feilregistreringsskjemaer for senere oppfølging, se vedlegg Q.

5.5. Beste praksis

For å dra nytte av erfaringene underveis i utviklingen, søkte vi å finne en beste praksis for hvordan legge til ny funksjonalitet i applikasjonene. Denne praksisen er beskrevet og justert underveis i prosjektet, basert på egne erfaringer på hva som fungerte best for prosjektgruppen. Denne fremgangsmåten er dokumentert og beskrevet i vedlegg M som hjelp til eventuelle fremtidige prosjekter som ønsker å bygge videre på TKIF.

5.6. Dokumentering

Feil

Som nevnt over, er alle feil fra systemtesten dokumentert (vedlegg Q) i egne feilregistreringsskjema. I tillegg begynte prosjektgruppen også etter hvert å dokumentere feil vi oppdaget selv. For hver feil ble det fylt ut et feilregistreringsskjema, hvor feilen får en id, og så blir en kort beskrivelse av feilen med samme id lagt inn i en feilliste (tabell 1 under). I denne listen ser man også hvilke feil som enda ikke er rettet opp, og i hvilket feilregistreringsskjema man kan finne flere detaljer rundt feilen. Grunnen til at listen ikke er lenger kommer ikke av at det ikke ble funnet andre feil, men av at vi ikke startet med dokumentering av feil fra begynnelsen av prosjektet. Alle kjente feil ligger imidlertid i denne listen.

Feil nr:	Dato registrert:	Beskrivelse av feil	Dato rettet:	Godkjent:
01	16.04.03	Problemer med æøå overføring fra foreleser til PDA med send kommentar.	21.04.03	OK
02	21.04.03	- Får ikke opp øæå fra databasen og til forelesers spørsmål.	23.04.03	OK
03	02.05.03	Fikk ikke opp den nye statistikken som ble kjørt på det nye oppgave settet uten å restarte hele foreleser applikasjonen.	02.05.03	OK
04	02.05.03	Problemer med å logge av studenter når vi kjører applikasjonen på en PDA, men ikke fra en pc.		
05	02.05.03	Hente oppgave sett til PDA når man ikke får det knappen fungerte ikke, men fungerte får vi kjører det på pc.		
06	02.05.03	Ved trykking på Avbryt så manglet det at choice list skulle nullstilles.	02.05.03	OK

Tabell 1 Feilliste

GUI-beskrivelse
På bakgrunn av erfaringer fra systemtesten har prosjektgruppen utarbeidet en enkel beskrivelse av skjermbildene, hvor funksjonalitet og skjermbilder er beskrevet. GUI-beskrivelsen er lagt ved i vedlegg K.

Kommentering av kildekode
Det ble lagt vekt på å kommentere kildekoden etter hvert som den ble skrevet. Kommentering av koden har bl.a. som formål å gjøre koden mer leselig for andre, men det har også vært nyttig for prosjektgruppen når vi leser hverandres kode eller kildekode det er lenge siden vi jobbet med. For beskrivelse av hvordan koden er kommentert, se kapittel 4.4.

Klassediagram og kollaborasjonsdiagram
Klassediagram og kollaborasjonsdiagram ble tegnet parallelt med utviklingen, se for øvrig kapittel 3.3 og 3.5, samt vedlegg E og U. Etter hvert som applikasjonen vokste seg større og mer kompleks, fant vi disse diagrammene nyttige for å beholde oversikten. Filosofien bak dette var at det er lettere å lage strukturert og enhetlig kode dersom man har god oversikt over hvordan applikasjonen er bygd opp. I tillegg vil det også være raskere å sette seg inn i koden ved hjelp av disse diagrammene.

5.7. Konfigurasjonsstyring

Konfigurasjonsstyring skal hjelpe et prosjekt med håndtering av følgende situasjoner:
Forskjellige personer jobber med forskjellige deler av systemet samtidig.
Endringer av systemkomponenter under utvikling og drift.
Flere versjoner av et system.

Kort sagt kan man si at konfigurasjonsstyring skal hjelpe til å holde orden på endringer i systemet, slik at disse skjer kontrollert. Et verktøy for konfigurasjonsstyring kan hjelpe med ca 25 % av konfigurasjonsstyringen, mens resten må ivaretas vha fastlagte rutiner og prosedyrer. Prosjektet valgte å støtte seg til kun manuelle rutiner for konfigurasjonsstyring, fremfor å ta i bruk elektroniske verktøy som likevel ville kreve manuelle rutiner i tillegg.

Versjonskontroll
Vi vurderte behovet for verktøy som håndterte versjonskontroll til ikke å være stort nok til at vi ville tjene noe på å benytte dette. I et prosjekt på størrelse med TKIF med bare tre prosjektmedarbeidere, antok vi at det ville være relativt enkelt å holde orden på hvem som jobbet på hva til enhver tid. Skulle to personer likevel jobbe mot samme fil på et tidspunkt, ville både Kate og Editplus (editorene vi har benyttet) gi beskjed om dette.

Baselines
For å sikre at store endringer på systemet ikke ødela noe, og at man skulle ha mulighet til enkelt å gå tilbake til en stabil versjon, sidelagret vi såkalte baseline versjoner. Dette er komplette versjoner av systemet som ble "frosset" og lagret på egne kataloger for å ha noe å falle tilbake på.

5.8. Parprogrammering

Med parprogrammering menes det at to utviklere jobber sammen under programmeringen. Den ene fokuserer på selve kodeskrivingen for den aktuelle oppgaven de skal løse, samtidig som den andre kontrollerer det som blir skrevet og tenker på oppgaven i et større perspektiv. Dette er et av rådene fra systemutviklingsmodellen "Extreme Programming" som vi mener er både effektivt og gir høyere kvalitet på koden. På denne måten er det også alltid to utviklere som har inngående kjennskap til kildekoden skrevet vha parprogrammering.

Prosjektgruppen benyttet seg av denne metoden på de mest komplekse og vanskelige oppgavene, men vi programmerte også mye hver for oss på de enklere oppgavene. Parprogrammering har anslagsvis vært benyttet på omtrent 40 % av kodingen.

5.9. Sikkerhetskopiering

Før det ble tildelt prosjektområder på skolens servere, ble det foretatt daglig sikkerhetskopi av prosjektfilene. Sikkerhetskopiene ble lagt over til prosjektdeltagernes hjemmeområder på skolens server, hvor IT tjenesten på HiG tar daglig sikkerhetskopi.

Etter at vi fikk prosjektområdet tildelt, ble prosjektfilene flyttet over dit og det ble jobbet direkte mot prosjektområdet. Sikkerhetskopieringen var da håndtert helt og holdent av IT tjenesten. For sidelagring av forskjellige versjoner av TKIF, se avsnittet *Baselines* i kapittel 5.7.

6. Installasjon

Installasjonsveiledninger for TKIF er lagt ved i vedlegg N. Det finnes en config.txt fil til hver av applikasjonene (server, PDA og foreleser), som inneholder variabler som kan være aktuelt og endre, se vedlegg N for detaljer.

Prosjektgruppen har ikke utarbeidet egne installasjonsveiledninger for Mysql, Tomcat og SOAP. For dette henviser vi til produsentenes egne installasjonsveiledninger. Det er imidlertid utarbeidet en hjelpeveiledning for installasjon av disse produktene på server, se vedlegg I, hvor det beskrives noen punkter det er spesielt viktig å merke seg.

6.1. Server

Server installeres med Apache http-server v.2.0 og Tomcat v.4.1 [Tomcat], Apache SOAP v.2.3.1. [SOAP], og MySQL [MySQL]. Serveren må også installeres med Java 2 kjøretidsmiljø [Sun]. TKIF-Serveren er utviklet i, og testet med Java 1.3.1. Se for øvrig vedlegg I, for tips til installasjonen.

6.2. PDA

For PDA-en kreves det en Java VM som håndterer Java 1.1.8 eller høyere. På PDA-en benyttet prosjektet Jeode Runtime fra Insignia. Denne følger med i den medsendte programvaren til Compaq iPaq H3950, som ble brukt til testing. Prosjektgruppen fant ingen andre gratis Java VM i løpet av research fasen i prosjektet, som kunne benyttes på iPaq.

6.3. Foreleserklient

Foreleserklienten krever JRE 1.3.1, samt diverse XML- og SOAP-pakker. Se vedlegg N for mer detaljert veiledning. Det er viktig å oppdatere classpath-variabelen og endre IP-adressen til server i config.txt (se installasjonsveiledningen i vedlegg O).

7. Videreutvikling

Etter å ha lagt ned mye jobb i utvikling av systemplattform og funksjonalitet i applikasjonene, har prosjektgruppen etter hvert funnet et mønster for hva vi synes er en god fremgangsmåte for å legge til ny funksjonalitet. Dette har vi beskrevet og lagt ved i vedlegg M.

7.1. Neste naturlige trinn

Til tross for at prosjektgruppen både har testet produktet under utvikling og gjennomført systemtest med andre testpersoner, har vi ikke hatt tid og ressurser til å teste systemet i en forelesning.

Det neste naturlige trinnet i videreutvikling av TKIF er derfor etter vår mening å teste systemet i forelesninger. Slike gjennomføringer fører til at systemet stresstestes, samt at det kan komme fornuftige krav og endringsønsker fra brukerne. Slike tilbakemeldinger er etter prosjektgruppens vurdering helt nødvendige for å kunne få et produkt med høy kvalitet og som kan tilfredsstille kommende brukere.

I tillegg bør det legges til funksjonalitet for å administrere foreleserbrukere. Dette kan lages som en helt enkel webapplikasjon.

7.2. Anbefalinger

7.2.1. Utførelse av SOAP-kall på PDA

Som nevnt i kapittel 4.5.2, bruker kSoap noe tid på å sette opp et kall til TKIF-serveren. For å ta høyde for dette kan alle SOAP-kall som feiler av denne grunn legges inn i en kø for senere eksekvering.

Et SOAP-kall kan også feile p.g.a. at databasen ikke er tilgjengelig eller at serveren ikke er tilgjengelig. Dette vil resultere i en IOException og kan håndteres på en annen måte dersom ønskelig. Eksisterende kildekode gir i dag brukeren beskjed om å prøve på nytt dersom SOAP-kallet returnerer en feilkode.

7.2.2. Visning av oppgaver på PDA

Siden skjermstørrelsen og oppløsningen ikke gir mulighet for visning av mer enn omtrent ti knapper for valg av oppgaver, kan man legge inn knapper for å bla i oppgavene.

8. Diskusjon

8.1. Innledning

I dette kapittelet har prosjektgruppen foretatt en egen vurdering av resultatet av prosjektet og prosjektgruppens fremdrift og arbeid.

8.2. Resultat

Ideen bak TKIF var å lage et system som hjelper studentene til å delta aktivt i forelesningene ved hjelp av mobile enheter og trådløst nettverk. Hovedmålet med prosjektet var å lage en plattform for et slikt system, hvor det skulle være enkelt å legge til ny funksjonalitet. Disse målene føler vi i all hovedsak at TKIF har oppnådd.

For at TKIF skal kunne hjelpe studentene i å bli mer aktive, må det være enkelt og intuitivt i bruk. Hvorvidt studentene faktisk blir mer aktive ved bruk av TKIF gjenstår å se, men tilbakemeldinger fra testene tyder på at systemet er meget enkelt å bruke. Systemet skal også støtte foreleserens behov, hvor et grunnleggende krav (kapittel 3.1 i kravspesifikasjon) har vært å ha minst mulig interaksjon (færrest mulig "museklikk") med applikasjonen under forelesning. Dette kravet er også tilfredsstilt, bl.a. ved hjelp av timere som oppdaterer grafer automatisk.

Oppstart og innlogging på student applikasjonen har vist seg å ta noe lang tid når applikasjonen kjøres på de PDAene prosjektet har hatt til disposisjon. Når brukeren er innlogget er imidlertid responstiden upåklagelig.

Under gjennomføringen av siste systemtest ble det oppdaget to feil på studentapplikasjonen når de kjøres på PDAene, som prosjektgruppen enda ikke har hatt tid til å løse. Ingen av feilene er kritiske og begge er beskrevet og dokumentert i vedlegg Q.

Kravspesifikasjonen ble skrevet ut fra hva som var ønskelig å lage i systemet, uten særlig tanke på hva det krevde av ressurser. Ut fra prioriteringer og samtaler med oppdragsgiver, har vi ikke utviklet webapplikasjonen pga. tidsaspektet. Dette forsvares også med at vi har implementert aktuell funksjonalitet for denne applikasjonen i foreleser applikasjonen.

Dersom man skal nyttiggjøre seg av et støttesystem under forelesninger, er det viktig at systemet er så enkelt at det ikke påvirker brukerens konsentrasjon i forelesningen. Derfor har vi jobbet for å lage så intuitive brukergrensesnitt som mulig, samt å automatisere enkelte funksjoner.

Tilbakemeldinger fra testene tyder så langt på at begge brukerapplikasjonene er meget brukervennlige.

Totalt sett føler prosjektgruppen at vi har kommet vel i havn med prosjektet og at forholdet mellom kravspesifikasjon og endelig produkt samsvarer godt.

8.3. Prosjektgruppens fremdrift og arbeid

Utviklingsmodell
Prosjektgruppen har fulgt en iterativ utviklingsmodell, hvor vi delte inn prosjektforløpet i mindre iterasjoner på 1-3 uker og grovfordelte aktiviteter. Ved slutten av hver iterasjon ble det avholdt et statusmøte. På statusmøtene oppsummerte vi hvor langt vi var kommet i forhold til fremdriftsplanen og foretok en justering dersom vi mente det var nødvendig. I tillegg ble også neste iterasjon planlagt i mer detalj enn hva fremdriftsplanen spesifiserte.

Denne måten å jobbe på førte til at vi hele tiden hadde kontroll med hvor langt vi var kommet i forhold til fremdriftsplanen. Dette gjorde det lettere å planlegge hvilke aktiviteter som måtte prioriteres og kraftsamles om.

Fremdriftsplan
Når prosjektet startet var prosjektgruppen optimistisk på hvordan fremdriftsplanen skulle være og hvor mye vi skulle få gjort. Kombinasjonen av at en stor del av teknologien vi ønsket å benytte var ny for oss og at vi har liten erfaring i estimering av utviklingsprosjekter, får ta det meste av skylden for at tidsplanen sprakk. Etter hvert som prosjektgruppen så at vi lå bak tidsplanen, la vi ned mer tid for å forsøke å kompensere for dette. I tillegg besluttet vi også å kutte noen av de minst viktige funksjonene, da vi ikke hadde noen illusjoner om at ting skulle ta vesentlig kortere tid mot slutten av prosjektet.

Det var hovedsakelig to oppgaver som tok lenger tid å implementere enn det prosjektgruppen hadde planlagt. Dette var selve serverplattformen med kommunikasjon til klientene, samt funksjonen for å lage og redigere oppgavesett. I det disse to utfordringene var løst, ble resten av funksjonaliteten stort sett implementert i henhold til planen. Grunnen til at det gikk noe raskere å implementere funksjonalitet mot slutten av prosjektet var at vi begynte å bli kjent med teknologiene og hadde utviklet en rutine for hva som var beste fremgangsmåte.

Til tross for at planlagte aktiviteter ble fullført innen tidsfristene i revidert fremdriftsplan, måtte det brukes mer tid enn estimert. En av grunnene til

dette kom av at ønskelige endringer for å gjøre koden mer robust oftere ble oppdaget mot slutten av prosjektet når systemet ble testet mer helhetlig.

Organisering
Organiseringen under jobbingen har vært preget av at vi har satt inn store ressurser på de oppgavene som vi anså som vanskelige, samt å være fleksible med omfordeling med tanke på de problemer vi støtte på underveis. Med store ressurser mener vi at vi jobbet i den form som heter "par programmering", som nevnt i kapittel 5.8, dvs. at to stykker sitter og jobber sammen med en oppgave. Dette har vi erfart er meget effektivt der oppgaven har vært vanskelig. I tillegg er det alltid to personer som kjenner til de delene av koden som utvikles på denne måten.

Alle i prosjektgruppa har kjennskap til alle delene i prosjekt. Dette er gjort bevisst slik at hvert enkelt gruppemedlem skal vite hvordan systemet fungerer. Selvfølgelig så ble det også en naturlig deling av arbeidsoppgavene, slik at noen har jobbet mer med for eksempel GUI og andre mer med kommunikasjonsdelen. Siden vi har prøvd å få til at alle skulle kunne litt om alt, har det vært en veldig fleksibel og variert arbeidsform for alle i prosjektgruppen. En annen stor fordel med dette er at vi lett kunne støtte hverandre hvis det kom opp både små og store problemer.

8.4. Kritikk av oppgaven

I dette kapittelet forsøker prosjektgruppen å være selvkritiske mht hvordan vi har jobbet gjennom hele prosjektforløpet. Enkelt sagt er dette momenter erfart under arbeidet med prosjektet som vi kanskje ville ha løst på en annen måte i dag.

Framdriftsplan
Å lage en god fremdriftsplan krever erfaring fra tidligere utviklingsprosjekter og jo mer erfaring man har, jo større blir sannsynligheten for at estimeringene i planen stemmer. Fordi studentprosjekter som regel ikke har denne erfaringen, er det naturlig at estimeringene ikke holder mål og at fremdriftsplanene sprekker. Dette tror prosjektgruppen at det er vanskelig å gjøre mye med, men det kan likevel gjøre grep i planleggingen for å kompensere for manglende erfaring. Ved å bryte opp innlæringsfasen i mindre biter kan det gjøre planen lettere å styre etter, da man tidligere oppdager avvik i forhold til planen. Dersom prosjektet underveis vet hvordan det ligger an i forhold til planen, kan det foreta justeringer underveis og på denne måten "redde" prosjektet. Derfor er det viktig å jevnlig vurdere fremdriften i forhold til planen.

Prosjektgruppen lagde tidlig en grov fremdriftsplan, hvor arbeidet ble delt opp i iterasjoner på en og to uker. Denne tidsplanen måtte revideres underveis, noe som i seg selv er naturlig, men det viste seg relativt tidlig at en del av estimeringen ikke holdt mål. Dette skyldes nok hovedsakelig at prosjektgruppen var uerfaren mht estimering av utviklingsprosjekter. Det har vært lærerikt å se hvor mye tid som går med til de ulike delene i et prosjekt, for eksempel planlegging, testing av ny teknologi, etc.

Ut i fra egne erfaringer, oppfordrer vi også fremtidige prosjekter til å vurdere fremdriften opp mot fremdriftsplanen hele tiden. For hver iterasjon oppsummerte vi hvor langt vi var kommet, planla neste iterasjon (se iterasjonsplan i vedlegg G) og vurderte om noe måtte kuttes i prosjektet. Dette fører til at man hele tiden har god kontroll med framdriften i prosjektet.

Brukermedvirkning
God kommunikasjon med oppdragsgiver og brukere er nok noe som er ønskelig for mange prosjekter, og TKIF er intet unntak i så måte. Vi mener det er spesielt viktig i et prosjekt som dette, hvor kravene til funksjonalitet/virkemåte ble avdekket underveis. For TKIF fungerte dette stort sett bra, men vi følte også at det kunne ha vært bedre på et par områder. Det er opp til prosjektgruppen å dra med oppdragsgiver og bruker så mye som nødvendig, og sørge for at de hele tiden har forståelse for hva som gjøres og hvorfor. Lykkes man i å kommunisere godt med disse viktige personene, er det større sannsynlighet for å få til et produkt som tilfredsstiller kundens krav.

Databaseoperasjoner
Oppdragsgiver ønsket at alle databaseoperasjoner skulle utføres på serveren. På grunn av en misforståelse mellom oppdragsgiver og prosjektgruppen var ikke prosjektgruppen klar over dette før mot slutten av prosjektet. For ikke å bruke tid på å gjøre om det vi hadde laget, besluttet oppdragsgiver at dette kun skulle gjøres gjeldende for den funksjonaliteten som gjensto å utvikle. Dette er grunnen til at databaseoperasjonene for å lage og redigere oppgavesett utføres på foreleserklienten, mens databaseoperasjonene for henting av statistikk ligger på TKIF-serveren.

Konfigurasjonsstyring
Konfigurasjonsstyring var, som nevnt i kapittel 5.7, basert på manuelle rutiner og prosedyrer. Selv om prosjektet bestod av kun tre personer, hadde vi ved et par anledninger problemer med at to personer jobbet mot de samme filene. Dette trenger ikke bety at man bør benytte egne verktøy til versjonshåndtering, men det viser at selv for små prosjektgrupper er det viktig at det lages en gjennomtenkt plan for konfigurasjonsstyringen.

Testing
Målet er ikke å vise at programmet er feilfritt, hvilket i praksis er vanskelig å gjøre kun ved hjelp av testing. Testing kan bare avdekke feil, og kan vanskelig garantere at programmet ikke inneholder flere feil. Det finnes imidlertid metoder for å teste deler av applikasjoner, for ut fra resultatene på den delen av koden estimere hvor mye feil det er i hele applikasjonen. Prosjektet har ikke benyttet noen slike estimeringsmetoder, men bare hatt som mål for testingen å finne og fjerne så mange feil som mulig før programmet tas i bruk.

Tidlig i prosjektfasen ble ikke modultestene planlagt og dokumentert, men bare gjennomført ettersom klassefilene ble skrevet. Når vi ser tilbake på testingen gjennomført i prosjektet, burde vi startet tidligere med planlegging og dokumentering av tester. Alle tester kunne vært skrevet på forhånd slik som utviklingsmodellen XP beskriver. Ved å skrive testene på forhånd blir man også mer oppmerksom på hva funksjonene og modulene, som testene skrives for, skal gjøre.

8.4.1. Batterikapasitet på PDA

En interessant oppdagelse vi hadde under testing var batterikapasiteten på Compaq iPaq. Disse PDAene som prosjektgruppen disponerer vil være lite egnet som klienter i en forelesning. Dette kommer av at nettverkskortet tømmer jakken for batteri lenge før det har gått en skoletime. Etter at prosjektgruppen har brukt PDAene i fire måneder, tar det bare 15-20 minutt å tømme et fult oppladet batteri når nettverkskortet står i og kommuniserer med aksesspunktet. Dette betyr at batterikapasitet må vurderes nøye når man velger enheter til bruk i dette systemet.

8.5. Veileder/oppdragsgiver

Professor Rune Hjelsvold har vært både oppdragsgiver og veileder for TKIF under hele prosjektfasen.

Samarbeid
Kontakten med oppdragsgiver og veileder har stort sett vært lite formalisert. Det har vært noen avtalte møter, samt en skriftlig statusrapport som ble sendt i en periode da veileder/oppdragsgiver var utenlands. Utenom møtene og statusrapporten, hadde prosjektgruppen bare uformelle konfereringer muntlig og pr e-post underveis i prosjektet. Dette har fungert tilfredsstillende sett fra prosjektgruppens ståsted.

Prioritering av funksjonalitet og brukerkrav
Prosjektgruppen spesifiserte tidlig, i samarbeid med oppdragsgiver, flere aktuelle brukerfunksjoner som var ønskelig å implementere i systemet. Når vi

syntes at tilstrekkelig antall var definert, listet vi opp, foretok en risikovurdering (se vedlegg J) og prioriterte alle funksjonene for å finne ut hva vi skulle starte med. Prioriteringen vurderte vi ut fra en kombinasjon av antatt vanskelighetsgrad, samt antatt viktighet for bruker.

I denne prosessen foretok vi egne vurderinger både på vanskelighetsgrad og viktighet for bruker. Viktighet for bruker baserte vi mye på egne oppfatninger og noe ut fra hva vi hadde oppfattet under samtaler med oppdragsgiver. Etter oppsummering av prosjektet ser vi imidlertid at prosjektgruppen burde ha involvert oppdragsgiver mer for å gi han mulighet til å bestemme graden av viktighet for de forskjellige funksjonene.

Rollene
Prosjektgruppen oppdaget etter hvert at det i noen tilfeller var vanskelig å skille mellom hva som var oppdragsgivers krav/ønsker og det som var veileders anbefalinger/tips. Med dette mener vi ikke at oppdragsgiver/veileder har vært uklar, men heller at vi ikke har vært oppmerksomme nok på å skille disse to rollene fra hverandre. Dette resulterte ved et par anledninger til misforståelser da vi tolket noe som egentlig var oppdragsgivers ønske, som et tips fra veileder og endret da på brukerkravet (som vi trodde bare var tips fra veileder).

Prosjektgruppen tror ikke akkurat dette vil være noe stort problem ute i arbeidslivet, da de fleste reelle prosjekter har oppdragsgiver og veileder atskilt. Ute i reelle prosjekter kan man likevel støte på lignende utfordringer når det gjelder forholdet til oppdragsgiver. Dette kan for eksempel være at en oppdragsgiver ikke peker ut representative brukere, at brukerrepresentantene ikke har nok tyngde og myndighet hos oppdragsgiver, eller at det ikke er utpekt faste representanter i det hele tatt.

Når vi ser tilbake på kommunikasjonen mellom prosjektgruppen og oppdragsgiver ser vi at vi burde håndtert den annerledes. Prosjektgruppen burde vært klarer på å formalisere kravspesifikasjonen bedre. Dette kunne vært gjort slik at kravspesifikasjonen ble oppdatert etter hvert endringsønske og at oppdragsgiver da burde godkjenne elle forkaste den aktuelle endringen. På denne måten får prosjektgruppen en kravspesifikasjon de enkelt kan forholde seg til.

Vår anbefaling til kommende hovedprosjekter hvor oppdragsgiver og veileder er samme person, er at de har dette i hodet ved møter med oppdragsgiver og veileder. Dersom det er praktisk mulig, kan man ved hvert møte bestemme hvilken rolle oppdragsgiver/veileder skal ha i det spesifikke møtet og heller holde et nytt møte etterpå hvor personen tar den andre rollen.

I tillegg til dette bør også dokumentproduksjonen formaliseres tidlig i prosjektet, slik at både oppdragsgiver og veileder kan komme med eventuelle korreksjoner tidlig i prosjektfasen. Det krever da mindre av prosjektgruppen å korrigere utviklingen når korreksjonene kommer tidlig i prosjektforløpet.

9. Konklusjon

Vi har arbeidet med å lage et system som skal supplere den verbale kommunikasjonen mellom foreleser og studenter i forelesning.

I TKIF har prosjektgruppen tatt i bruk flere teknologier vi på forhånd var ukjente med. Det har vært meget lærerikt å sette seg inn i så mye nytt, men vi har også lært mye om hvordan sette seg inn i ny teknologi på egenhånd. I tillegg til at flere av teknologiene var ukjente for prosjektgruppen, benyttet vi oss også av relativt ny teknologi for håndtering av XML og SOAP på lettvektsenheter som PDAer.

TKIF gir mulighet for sanntids kommunikasjon, hvor foreleser hele tiden kan se statistikk på hvordan studentene oppfatter forelesningen i øyeblikket. Foreleser kan kontinuerlig lese kommentarer studentene har sendt og samtidig se statistikk over faste evalueringsparametere. Foreleser kan både stille spontane enkeltspørsmål og sende ut ferdiglagde oppgavesett for så å se på statistikk over besvarelsene. Det har blitt lagt vekt på å lage et brukervennlig system, for at systemet ikke skal ta for mye fokus under forelesninger. Studentene kan hvor som helst i applikasjonen formidle en av de faste evaluerings-parametrene, for eksempel at foreleser går for fort frem. Informasjon som sendes fra foreleser, vil automatisk dukke opp hos studenten.

Skulle prosjektgruppen gjentatt utviklingsarbeidet, ville vi satset på mer formalisert dokumentproduksjon og verifisering opp mot oppdragsgiver og veileder tidlig i prosjektfasen.

Til tross for at prosjektgruppen var sammensatt av medlemmer fra alle tre studieretningene, har vi alle utviklet våre kunnskaper innen både andres og egne fagområder. For første gang i skolesammenheng har vi kombinert kunnskap fra flere fagområder og gjennomført et prosjekt helt fra ide fasen, gjennom kravspesifiseringen, design og til tilnærmet ferdig produkt. Dette og det å finne frem til og tilegne seg lærdom på egenhånd har vært både utfordrende og lærerikt, og er en meget realistisk og nyttig erfaring vi tar med oss ut i arbeidslivet.

10. Litteratur

10.1. Referanser

[1.1.8] Sun Microsystems, Inc. "Java 1.1.8" http://java.sun.com/products/archive/jdk/1.1/index.html [05 februar 2003].

[1.3.1] Sun Microsystems, Inc. "Java j2sdk1.3.1". http://java.sun.com/j2se/1.3/docs/index.html [05 mai 2003].

[Benoist] Benoist, Sylvain. 2 July 2002. "Security with Apache SOAP" http://www.soapuser.com/sb_02jul02.html [02 April 2003]

[Enhydra] Enhydra Community. Ca 2000. http://enhydra.org/ [09 mai 2003]

[IANA] Internet Assigned Numbers Authority http://www.iana.org/assignments/port-numbers [15 februar 2003].

[J2ME] Sun Microsystems, Inc. http://java.sun.com/j2me/ [13 februar 2003].

[kSoap] Enhydra community. Ca 2000. http://ksoap.enhydra.org/software/documentation/api/ [14 april 2003]

[kXml] Enhydra community. Ca 2000. http://kxml.enhydra.org/software/documentation/apidocs/ [14 april 2003].

[Modelator] Metodedata AS. http://www.metodedata.no/modelator.html

[MySQL] MySQL v.4.1. http://www.mysql.com/ [15 april 2003].

[SOAP] Apache SOAP. Ca 1999. http://ws.apache.org/soap/index.html [04 mai 2003].

[Sosnoski] Sosnoski, Dennis M. "A look at features and performance of XML document models in Java". September 2001. http://www-106.ibm.com/developerworks/xml/library/x-injava/ [07 mars 2003].

[Sun] Sun Microsystems, Inc. http://java.sun.com/

[Tomcat] Apache Tomcat v.4. Ca 1999. http://jakarta.apache.org/tomcat/index.html [17 februar 2003]

10.2. Bibliografi

[Allamaraju] Allamaraju, Subrahmanyam, " Professional Java server programming J2EE 1.3 Edition", ISBN 1-86100-537-7

[Apache srv] Apache Server v.2.0 http://www.apache.org/

[Bowen] Rich Bowen, Daniel Lõpez Ridruejo, Allan Liska , "Apache administrator's handbook" ISBN 0-672-32274-9

[Brogden] Bill Brogden, "SOAP Programming with Java", ISBN 0-7821-2928-5

[DuBois] Paul DuBois "MySQL" ISBN 0-7357-0921-1

[FortuneCity] FortuneCity.com, Inc. ca. 2000. "Introduction to XML and XML With Java". http://members.fortunecity.com/seagull98/XmlTutorial.html [21. februar 2003].

[Goodwill] James Goodwill, "Apache Jakarta-Tomcat, ISBN 1-893115-36-4

[Haustein] Stefan Haustein. "KXML and Java". 15 februar 2001. http://www.microjava.com/articles/techtalk/kxml?content_id=1030 [19 februar 2003].

[Horstmann] Cay S. Horstmann, Gary Cornell, “Core Java 1.1”, “Volum I Fundamentals” ISBN 0-13-766957-7

[Hyde] Paul Hyde, “Java thread programming” ISBN 0-672-31585-8

[Mobile Læringstjenester] Kjetil Haraldstad og Daniel Rustad. ”Mobile Læringstjenester”. ca. 2002. http://student.iu.hio.no/hovedprosjekter/2002/data/33/ [15. januar 2003]

[Sommerville] Ian Sommerville ”Software Engineering”, 6 Edition, Sybex, ISBN 0-201-39815-X

11. Vedlegg

Vedlegg A: Ordforklaringer.
Vedlegg B: Forprosjektrapport.
Vedlegg C: Prosjektavtale.
Vedlegg D: Kravspesifikasjon.
Vedlegg E: Klassediagram.
Vedlegg F: Databasedesign.
Vedlegg G: Fremdriftsplan og iterasjonsplaner.
Vedlegg H: Ressursforbruk.
Vedlegg I: Hjelpeveiledning for serverinstallasjon.
Vedlegg J: Risikoanalyse.
Vedlegg K: GUI-beskrivelser.
Vedlegg L: Mockups på skjermbilder.
Vedlegg M. Beste praksis.
Vedlegg N: SOAP Deployment descriptorer.
Vedlegg O: Installasjonsveiledning TKIF.
Vedlegg P: Eksempel på prosjektdagbok
Vedlegg Q: Feildokumentasjon.
Vedlegg R: Test dokumentasjon.
Vedlegg S: HttpTransportSE.java
Vedlegg T: Kollaborasjonsdiagram
Vedlegg U: XML formater
Vedlegg V: CDROM med kildekode, klassediagram (Rational Rose fil), databaseskript, installasjonsfiler TKIF, Prosjektrapport (pdf), JRE, MySQL, Apache m/Tomcat og Soap for serverside.