HOVEDPROSJEKT:

TITTEL

HEVN – M

Hvem Er hVor Når Mobil

FORFATTERE:

Arnstein Sveum
Anne Margrete Harbosen
Kosovare Krasniqi

DATO:

19.05.2003

SAMMENDRAG AV HOVEDPROSJEKT

Tittel:	HEVN – M	Nr. : 11
	Hvem Er hVor Når - Mobil	Dato : 19.05.03
Deltaker(e):	Arnstein Sveum	
	Anne Margrete Harbosen	
	Kosovare Krasniqi	
Veileder(e):	Rune Hjelsvold	
Oppdragsgiver:	Høgskolen i Gjøvik	
Kontaktperson:	Rune Hjelsvold	
Stikkord (4 stk)	Mobiltelefon, Web Service, SOAP, Java.	

Antall sider: 156	Antall bilag: 10	Tilgjengelighet (åpen/konfidensiell): Åpen

Kort beskrivelse av hovedprosjektet:

Utviklet et system for å få tilgang til lesing og oppdatering av data i HEVN-J sin database via mobiltelefon. HEVN-J er en eksisterende applikasjon som holder oversikt over ansatte i skolen.

Vi har utviklet modulene ” Min timeplan” og ”Finn ansatt”. Ansatte kan lese og oppdatere sin egen timeplan og se andres timeplan og personalia.

FORORD

Hovedprosjektet HEVN-M(Hvem Er hVor Når – Mobil) er et avsluttende prosjekt ved den treårige høgskoleingeniørutdanningen på Høgskolen i Gjøvik våren 2003.

Prosjektets deltakere har vært Arnstein Sveum, Anne Margrete Harbosen og Kosovare Krasniqi.

Prosjektideen var å benytte muligheten for programmering av applikasjoner på mobiltelefon for å gi mobil tilgang til lesing og oppdatering av data på en sentral tjener.

Ideen ble presentert for Høgskolen i Gjøvik ved Rune Hjelsvold, og prosjektgruppen utarbeidet et forslag om å benytte det eksisterende systemet HEVN ved HiG, og lage en løsning for tilgang til og oppdatering av data v.h.a. mobiltelefon. Forslaget vårt ble godtatt.

Vi vil takke følgende personer for hjelp og støtte under hovedprosjektet:

- Veileder og oppdragsgivers representant Rune Hjelsvold (for veiledning, hjelp, innspill og motivasjon)
- Øyvind Kolloen (for teknisk veiledning i MySQL og Java)
- Frode Haug (for informasjon om HEVN-J)
- Vaktmesterne ved HiG(for administrering og tilrettelegging av grupperom)
- Renholderne ved HiG(for renhold av grupperom)
- IT-tjenesten(for lån av PC'er og drift av nettverk)

Gjøvik 19.05.2003

Arnstein Sveum Anne Margrete Harbosen Kosovare Krasniqi

INNHOLDSFORTEGNELSE

Kapittel 1 INNLEDNING

1.1. PROBLEMOMRÅDE

I alle organisasjoner, store eller små, er det viktig å kunne ha oversikt over hvor de ansatte befinner seg til forskjellige tider og opplysninger om hvordan få tak i disse. I de siste årene har det vært mange mer eller mindre manuelle og systematiske løsninger på dette. En del organisasjoner fører dette rent manuelt, men for stadig flere har det vært naturlig å gjøre dette elektronisk da de har tilgang til PC koblet til nettverk.

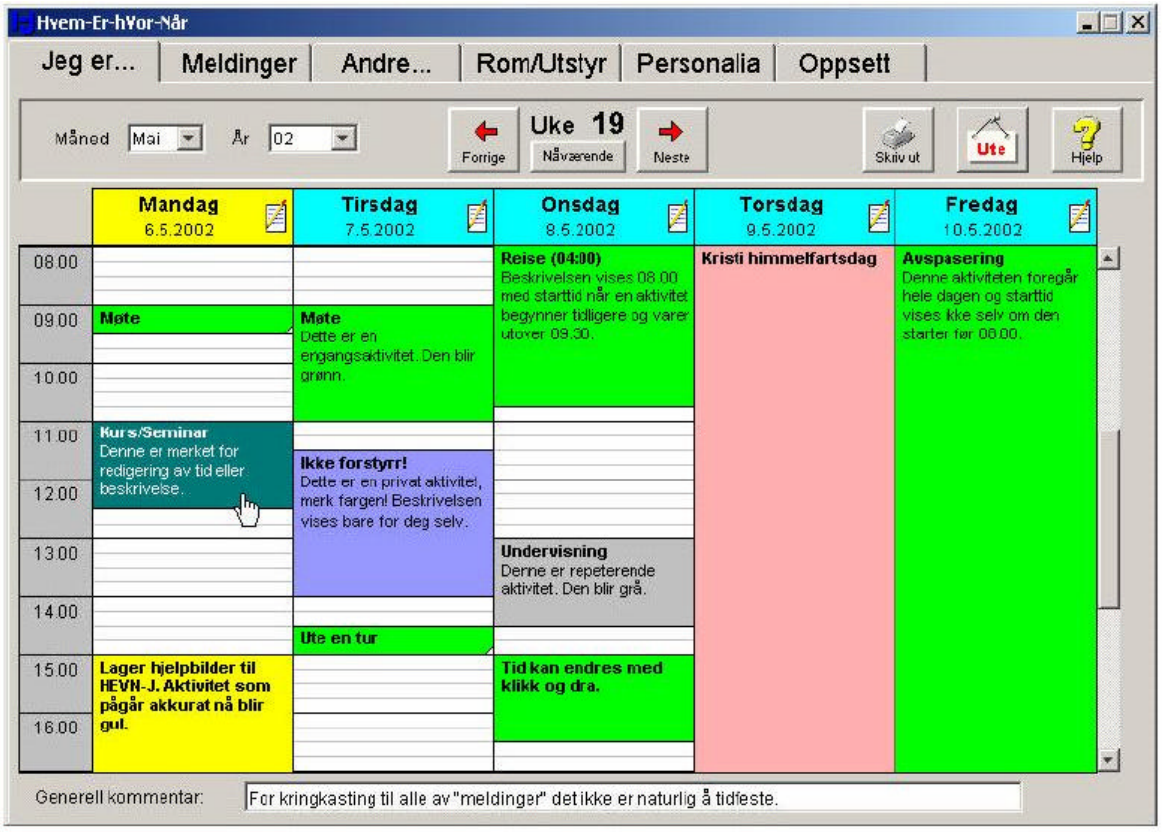
Et eksempel på en elektronisk tidsplanlegger er Microsoft Outlook. Den er kommersiell og man har tilgang til å lese andres timeplaner og mulighet for meldingsutveksling. Det finnes flere andre system med liknende funksjonalitet. Det har også vært vanlig de siste årene å registrere sin tilstedeværelse pr telefon. Ved å bruke koder på telefonen vil en oppringer få beskjed at vedkommende ikke er til stede på grunn av for eksempel møtevirksomhet.

Disse nevnte løsninger gjør at man blir avhengig av nettverkstilkoblet PC. (For å gjøre det telefonisk må man være på arbeidsplassen.)

På Høgskolen i Gjøvik brukes et program som kalles HEVN – Hvem Er hVor Når. Dette er en elektronisk tidskalender for de ansatte og har vært i bruk noen år allerede og er per i dag tilgjengelig på Høgskolens nettverk for ansatte. I tidskalenderen lager de ansatte sine egne timeplaner som gir en oversikt over hvor de befinner seg og hva slags gjøremål de har til enhver tid. Alle ansatte innenfor skolens nettverk har tilgang til å se disse timeplanene og eventuelt deres personalia. Ansatte har også oversikt over skolens ressurser som møterom, auditorier, teknisk utstyr osv, og mulighet til å reservere disse. Personalia for de ansatte ved skolen kan hentes ut, og meldingsutveksling mellom ansatte er også mulig. Dette forutsetter at alle har tilgang til PC.

HEVN består av en løsning med klientapplikasjoner på de ansattes arbeidsstasjon, som kommuniserer med en sentral database. I tillegg finnes verktøy for administrasjon og lignende.

HEVN er blitt utviklet som hovedprosjektoppgave ved HiG våren 1998. Den første versjonen ble kalt HEVN, og dette var en Windows-versjon utviklet i Delphi. I 2002 ble en ny plattformuavhengig versjon utviklet i Java. Navnet på denne versjonen ble HEVN-J, og det er denne som er i drift i dag. Det arbeides også med HEVN-WEB, som skal gi lesetilgang også for studenter via Internet. Dette er bakgrunnen for navnet på vårt prosjekt; HEVN-M, der M står for ”Mobil”. Figuren under viser det mest brukte skjermbildet i HEVN-J.



Figur 1-1 "Jeg er…" kortet i HEVN-J.

I de siste år har det vært en eksplosjonsartet anskaffelse og bruk av mobiltelefon. Det finnes gjerne flere mobiltelefoner i hver familie. Det er både eldre og yngre brukere. Teknologien på området blir bedre og bedre, og utviklingen går raskt. I den senere tid har det eksistert telefoner med tilgang til Internet. Dette blir mer og mer vanlig.

Denne prosjektetoppgaven går ut på å gjøre HEVN tilgjengelig uten PC, altså via mobiltelefon, PDA og lignende. Løsning innebærer å kunne oppdatere og lese informasjon fra HEVN-J sin database ved hjelp av Java-applikasjon på mobiltelefon o.l.

1.2. AVGRENSNING

Den viktigste grunnen til å utvikle HEVN – M er at man skal kunne få tilgang til HEVN-data uten å være avhenging av en nettverkstilkoblet PC. I enkelte tilfeller kan det være aktuelt å få tilgang til HEVN da man ikke har tilgang til PC. Hva er da mer naturlig enn å bruke mobiltelefon og få tilgang til HEVN med noen enkle tastetrykk?

Mobilnettverket ligger utenfor prosjektets ansvarsområde.

Oppdragsgiver ga uttrykk for at det burde være mulig å oversette til andre språk uten rekompilering, og uten særskilte programmeringskunnskaper. Slik oversettelse inngår ikke i prosjektet.

Med sikte på at mobiltelefon for å få tilgang til HEVN kun brukes i begrenset omfang, og at prosjektet kun varer et halvt års tid, vil det ikke bli tilgang til å bruke all funksjonalitet som HEVN har å tilby. Prosjektet avgrenses også til å benytte ferdige komponenter for brukergrensesnitt i tilgjengelige biblioteker.

1.3. OPPGAVEBESKRIVELSE

Informasjon om HEVN – *se pkt 1.1 og 2.1.1*

Prosjektideen var å benytte muligheten for programmering av applikasjoner på mobiltelefon for å gi mobil tilgang til lesing og oppdatering av data på en sentral tjener.

Ideen ble presentert for HiG v. Rune Hjelsvold, og prosjektgruppen utarbeidet et forslag om å benytte det eksisterende systemet HEVN ved HiG, og lage en løsning for tilgang til og oppdatering av data v.h.a. mobiltelefon. Vi fikk aksept for å benytte denne problemstillingen som bakgrunn for et mulig hovedprosjekt.

Vi mener dette er en aktuell problemstilling også i andre sammenhenger, der utnyttelse av mobilitet og datatilgang er sentralt, slik at dette er fremtidsrettet bruk av relativt ny teknologi. Dette har vi sett på som en utfordrende og spennende oppgave.

1.4. MÅLGRUPPE

Målgruppen for systemet er de som har tilgang til eksisterende HEVN, altså de ansatte ved Høgskolen.

Målgruppen for prosjektrapporten er brukere, studenter ved HiG og andre skoler med høyere utdanning innen IT, og andre interesserte innenfor næringslivet.

Prosjektrapporten inneholder en del teknisk informasjon til bruk for administratorer og driftsansvarlige. Dette kan også være til nytte for senere studentprosjekter.

Prosjektrapporten er også en dokumentasjon til oppdragsgiver over gruppas arbeid. Veileder og sensor vil legge denne til grunn for karaktersetting og er derfor en viktig målgruppe som det er tatt hensyn til under skriving.

1.5. FORMÅL MED PROSJEKTOPPGAVEN

Hovedmålet for prosjektet var å utvikle en løsning for tilgang til data i HEVN-J sin database, ved bruk av mobiltelefon som støtter Java-teknologi. Dette skal ikke være en erstatning for PC, men et hjelpemiddel for å kunne lese og oppdatere HEVN der PC ikke er tilgjengelig.

Gjennom prosjektet har vi også ønsket å heve vår kompetanse innen prosjektarbeid, systemutvikling, klient- og serverteknikk, databaseteknikk og programmering i Java.

1.6. EGEN BAKGRUNN OG KOMPETANSE

Alle studentene som har deltatt i dette prosjektet er tredje års datastudenter ved HiG. To av deltakerne går på driftslinja og har samarbeidet tidligere. Tredjemann går programmering.

To av studentene har ganske lang fartstid i arbeidslivet og har dratt med seg viktig lærdom og erfaring. En av studentene har rik og bred erfaring fra organisasjonsarbeid som har gitt viktige bidrag til prosjektet.

1.7. ARBEIDSFORMER

Gruppen fikk tildelt grupperom 038 i kjelleren på A-bygget. Det meste av arbeidet med hovedprosjektet har foregått der. I perioder der alle tre kunne jobbe med noe selvstendig måtte en av oss benytte en PC i storlabben, dette pga. at vi fikk tildelt kun 2 datamaskiner fra skolen. I tillegg har det meste av arbeidet med designdelen og framdriftsplanene foregått i labben pga. diverse systemutviklingsverktøy som kun finnes der, f.eks. Rational Rose – UML og Office Project.

Det har vært naturlig at deltakeren fra programmeringslinjen har stått for det meste av programmeringen, med unntak av noe GUI for at de andre deltakerne skulle kunne bli litt kjent med J2ME. Deltakerne fra driftslinjen har tatt seg av det meste av rapportskrivinger, møtereferater og systemutviklingen. Ved slutten av hver iterasjon har alle deltatt i rapportskriving for å få utfylt fullstendig.

Prosjektet ble gjennomført innenfor fastsatt (overordnet) tidsramme (se vedlegg A Gant-skjema) og ved hjelp av de tilgjengelige ressurser: Gruppedeltagernes arbeid med støtte av veileder og øvrige tilgjengelige ressurspersoner. Maskinvare og programvare som er tilgjengelig v.h.a. IT-tjenesten og for øvrig opensource-verktøy. Det var ønsket å låne en Java-telefon for å prøve ut applikasjonen ordentlig. Veileder skulle jobbe for å få til en avtale med Telenor FoU i Trondheim angående dette.

Siden det ikke ble noe av et slikt samarbeid, har vi benyttet mobilemulator for å kjøre applikasjonen på pc.

1.8. DOKUMENTSTANDARD OG ORGANISERING AV RAPPORTEN

Denne rapporten følger standarden for skrifttyper fastsatt i kravspesifikasjonen i *pkt. 2.1.2. i kapittel 2*.

Der vi mener det kan være ulike tolkninger av språklige uttrykk, eller at forskjellige uttrykk refererer til det samme, har vi forklart vår mening bak ordene i *pkt 2.1.6 i kravspesifikasjon*. F.eks. er både bruker og ansatt referanser til brukeren av systemet.

Definisjoner på forkortelser og teknologiske uttrykk finnes også under *punktet 2.1.6 i kapittel 2*.

Der vi refererer til ulike kapitler i rapporten, er kapittelnummeret benyttet. En referanse til dette punktet i dette kapittelet blir da som følger; *"se pkt.1.8 i kapittel 1."*

Referanser til litteratur- og ressurslisten er gjort på følgende måte: "[X]" bak det som refereres, der "X" henviser til nummeret foran kilden.

Henvisninger til de ulike vedleggene er som følger: *se vedlegg A Ganttskjema i iterasjon x*. Hvis henvisningen er til vedlegg i overordnet dokumentasjon vil notasjonen *se vedlegg A Ganttskjema i overordnet dokumentasjon* benyttes.

I samarbeid med veileder/oppdragsgiver valgte vi en iterasjonsmodell(inkrementmodell) for systemutviklingen hvor det i hver iterasjon(inkrement) skulle lages en eller flere funksjoner som ville ha en del felles. Derfor er alle kapitler bortsett fra nr 6 overordnet og gjeldende for alle iterasjoner.

Kildekode gjengitt i rapporten er skrevet med fonten Courier New og skriftstørrelse 10.

Figurer og bilder i rapporten er i fete typer og nummerert med kapittelnummer og figur/bildenummer, fortløpende i hvert kapittel. Første figur i kapittel 2 har følgende nummerering, Figur 2-1 "Navn eller forklaring til figur". Under iterasjonene vil i tillegg nummeret på aktuell iterasjonen fremgå av tall nr to. Eksempel: Bilde 6-2-2 "Navn eller forklaring til bilde". Dette er bilde i kapittel 6, iterasjon 2 og bilde nr 2.

Figurer og bilder i vedleggene blir nummerert med kapittelnummer først, evt. Iterasjonsnummer, bokstaven på vedlegget, og tilslutt figur/bildenummer. Eks. Bilde 10-A-1, det er et bilde i kapittel 10, vedlegg A og bilde 1. I iterasjonene vil også iterasjonsnummer være det andre sifferet i referansen, eks. Bilde 6-2-A-1. Dette er et bilde i kapittel 6, iterasjon 2, vedlegg A og bilde nr. 1.

Kapittel 6 er delt inn i tre deler, en for hver iterasjon. Hver del vil inneholde det som spesifikt tilhører den aktuelle iterasjon. Det være seg planlegging, design, koding og individuelle tester på funksjonene.

Denne rapporten er delt inn i 9 kapitler. Innholdet i disse kapitlene er som følger:

Kapittel 1: gir en kort introduksjon til prosjektet.

Kapittel 2: inneholder kravspesifikasjon. Den er delt i to deler. En del er overordnet og beskriver hele systemet i sin helhet. Del to er en mer detaljert spesifikasjon som gjelder for alle iterasjoner.

Kapittel 3: inneholder dokumentasjon for analyse og design som gjelder for alle iterasjoner. Her er det forklart hva systemet skulle gjøre og hvordan vi valgte å gjøre det.

Kapittel 4: inneholder dokumentasjon over implementering. Gjelder for alle iterasjoner.

Kapittel 5: handler om hvordan arbeidet med testing og kvalitetssikring har blitt gjort under prosjektet.

Kapittel 6: inneholder dokumentasjon(kravspesifiasjon, analyse og design, implementering osv) for hver enkelt iterasjon.

Kapittel 7: innholder diskusjon av resultater.

Kapittel 8: inneholder konklusjon av prosjektet.

Kapittel 9: inneholder en litteraturliste- og ressursliste for rapporten.

Kapittel 10: inneholder oversikt over alle vedleggene til denne rapporten.

Kapittel 2 KRAVSPESIFIKASJON

2. KRAVSPESIFIKASJON

2.1. OVERORDNEDE FORHOLD

2.1.1. INTRODUKSJON

HEVN-M er en utvidelse av tidligere hovedprosjekter, HEVN og HEVN-J, ved Høgskolen i Gjøvik. (*se i innledning 1.1 for detaljer.*)

Pr i dag kan HEVN leses og oppdateres kun innenfor nettverket på HiG. Det arbeides med å gjøre HEVN tilgjengelig for lesing av data utenfor HIG, altså via Internet.

Oppdragsgiver, HiG, ønsker å gjøre HEVN tilgjengelig via mobiltelefon så systemet kan leses og oppdateres uten tilgang til PC. Veileder har kommet med innspill til kravspesifikasjonen og hvordan han ønsket systemet. I tillegg har vi brukt dokumentasjon fra de tidligere hovedprosjektene som et godt grunnlag for hva som finnes av funksjonalitet og hvordan databasen er beskrevet. For å gjøre HEVN tilgjengelig via mobiltelefon, vil det være andre krav som må spesifiseres. Derfor må vi lage en fullstendig kravspesifikasjon.

For å finne ut hvilke av HEVNs funksjonaliteter som er mest hensiktsmessig å gjøre tilgjengelig via mobil, vil det bli foretatt en spørreundersøkelse blant brukerne av HEVN. Underveis vil vi deretter avgjøre hvor mye av dette vi vil kunne klare. Valget av inkrementell systemutviklingsmodell gjør det enklere for oss å kunne avgjøre dette underveis i prosjektet. Vi tar ikke sikte på i dette prosjektet å gjøre HEVN-M komplett med alle HEVNs eksisterende funksjonaliteter.

DOKUMENTETS STRUKTUR

Kravspesifikasjonen i kapittel 2 er overordnet, dvs. at den gjelder for alle iterasjoner i prosjektet. Kravspesifikasjon for den enkelte funksjonalitet vil finnes i kapittel 6 for den aktuelle iterasjon.

Tekstdokumentene i prosjektet skrives i MS Office XP. Det er brukt Rational Suite Enterprise – Rational Rose Enterprise Edition som verktøy for UseCasediagram

Følgende standard er brukt under prosjektet:

Stil	Font	Størrelse	Attributter
Tittel	Times New Roman	19	Fet skrift, store bokstaver
Overskrift 1	Times New Roman	17	Fet skrift, store bokstaver
Overskrift 2	Times New Roman	16	Fet skrift, store bokstaver
Overskrift 3	Times New Roman	14	Fet skrift, store bokstaver
Overskrift 4	Times New Roman	12	Fet, store bokstaver, understreket
Overskrift 5	Times New Roman	12	Store bokstaver
Normal	Times New Roman	12	Ingen

2.1.2. SYSTEMETS OMGIVELSER OG BRUKERE

HEVN benyttes i dag som verktøy for tidsplanlegging og oversikt over ansattes aktiviteter og personalopplysninger, samt til reservering og oversikt over ledige ressurser ved HiG.

Mobilutgaven er laget for de ansatte ved skolen. Det er kun disse brukerne som skal kunne få tilgang til HEVN-databasen via mobiltelefonen, ved å oppgi sitt brukernavn og passord. Dette er av sikkerhetsmessige årsaker.

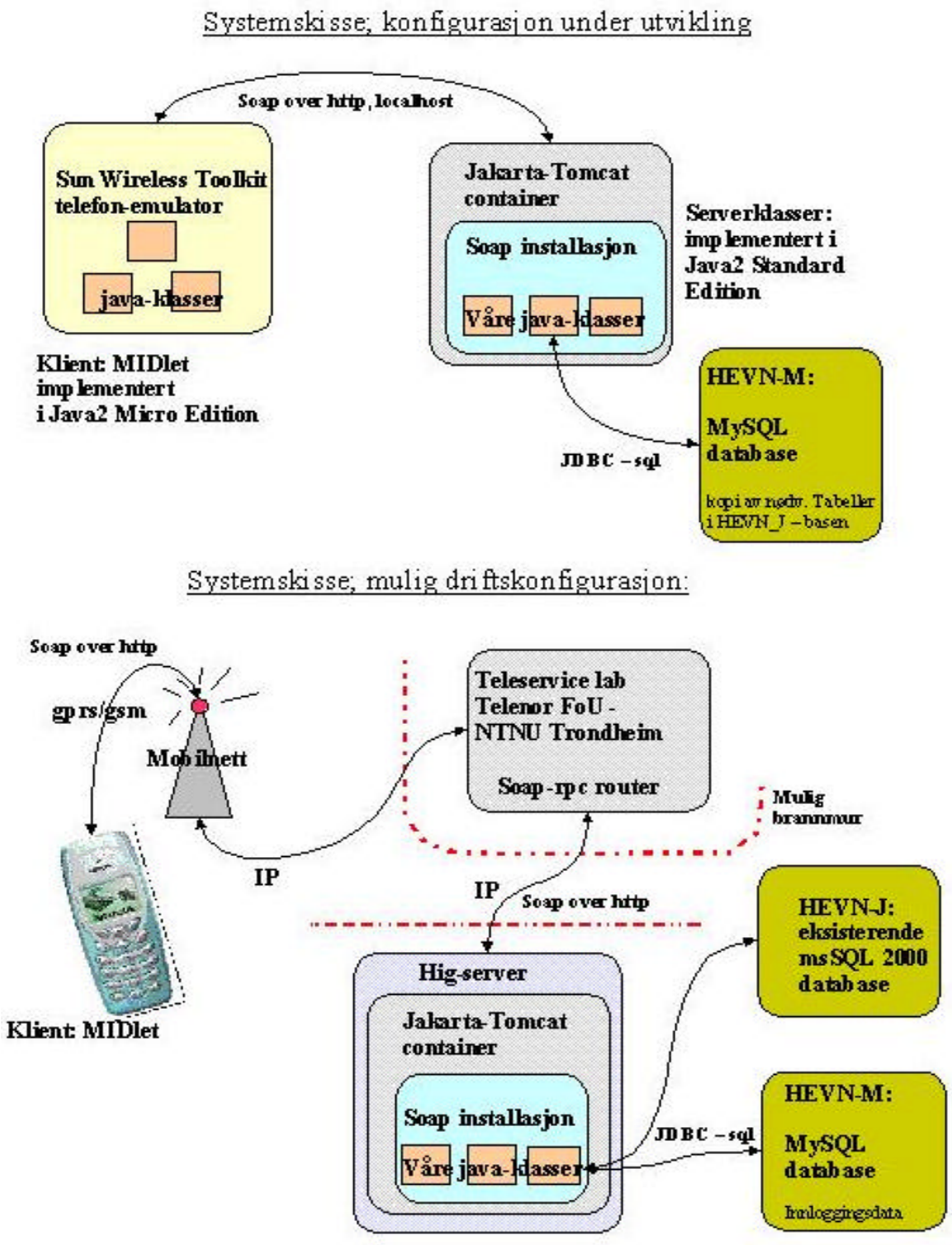
2.1.3. OVERORDNEDE KRAV TIL SYSTEMET

ARKITEKTUR

Løsningen skal implementeres som en Web Service, registrert på server hos NTNU i Trondheim. Dette innebærer at klienten (telefonen) sender forespørsel om data som et fjernmetodekall (RPC) til denne serveren, som i sin tur ber om data fra vår serverløsning – lokalisert på HiG. Disse dataene sendes begge veier med SOAP pakket xml.

Vår serverløsning skal da bestå av javaklasser i en Jacarta-Tomcat container med add-on for SOAP. Disse klassene vil benytte JDBC som grensesnitt mot HEVN-databasen.

Det kan også bli aktuelt å benytte en egen database for lagring av brukernavn, passord og brukerrettigheter, her kan MySQL være et godt valg.



Figur 2-1 Systemskisser

SIKKERHET

Oppdragsgiver har disse sikkerhetskravene til systemet:

- pålogging skal være nødvendig for å få tilgang til HEVN-M
- brukernavn, passord og rettigheter må tildeles hver bruker
- Det er ønskelig med https, men ikke et absolutt krav.

- i HEVN er det muligheter for å bestemme det man ønsker å gjøre tilgjengelig for andre via Web. Dette skal HEVN-M respektere. Hvis brukeren ikke ønsker å gjøre noe tilgjengelig via web, så skal det heller ikke være tilgjengelig via mobiltelefon.
- Passord skal ikke lagres i klartekst i databasen. Disse skal lagres kryptert (som en hashverdi).

YTELSE

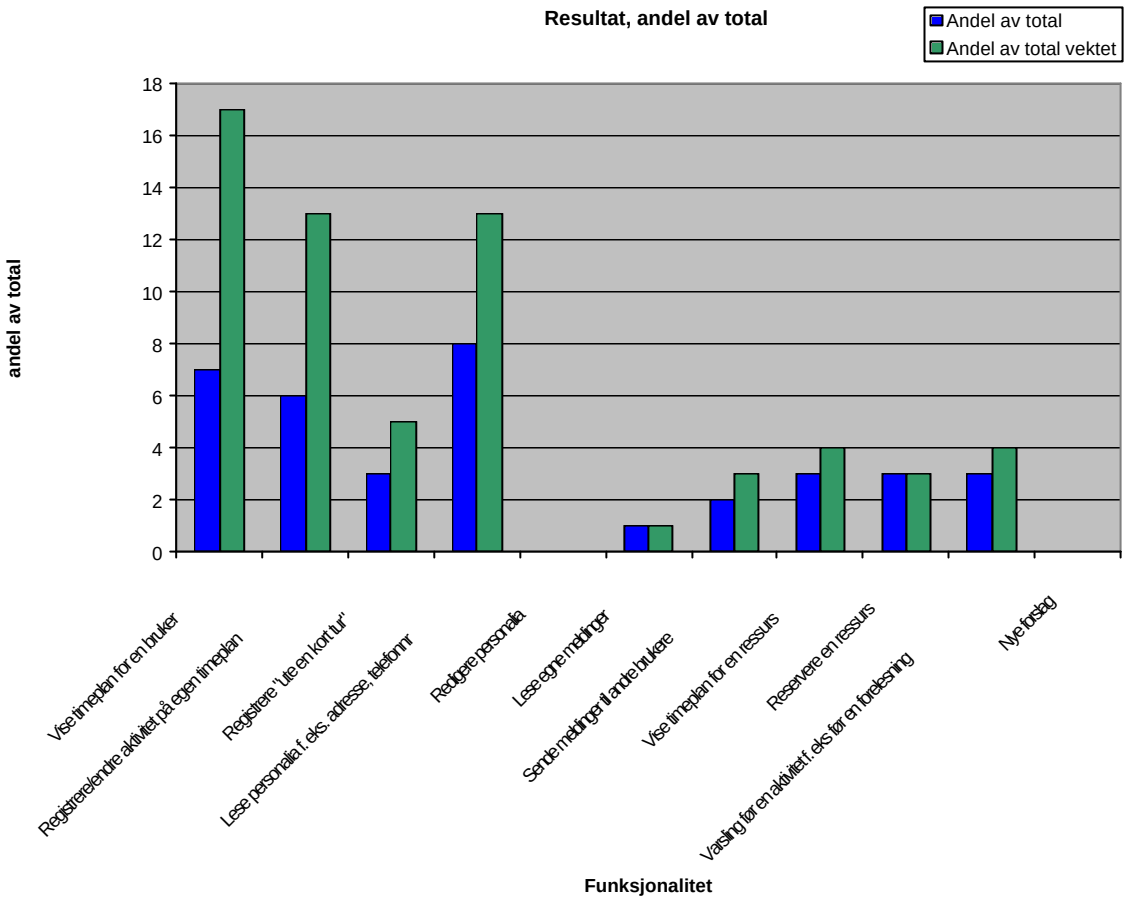
Systemets ytelse er begrenset på grunn av liten kapasitet på klientterminalen, og mulig begrenset båndbredde på nettverkstilkobling. Det er derfor viktig å utnytte de tilgjengelige ressursene på best mulig måte.

Oppdragsgiver har ikke stilt spesifikke krav til systemets ytelse. Brukerens oppfattelse av ytelse er allikevel en viktig faktor for brukervennligheten.

Klientapplikasjonens brukergrensesnitt skal utformes slik at det er responsivt, dvs. ikke ”fryses” ved kall til servertjenester og lignende.

FUNKSJONALITET

Krav til funksjonalitet er blitt kartlagt ut fra en uformell spørreundersøkelse blant brukere av HEVN. Vi stilte spørsmål om hvilke funksjoner i dagens løsning det ville være interesse for å ha tilgang til via mobiltelefon, og ba også om forslag til nye funksjoner. Responsen var ikke overveldende, så det statistisk grunnlaget er ikke så solid, men svarene gir et bilde av hvordan prioriteringen av de viktigste funksjoner bør være. Figuren under viser grafisk fordelingen av ønsker i rent antall og vektet etter prioritet:



Figur 2-2 Grafisk framstilling av spørreundersøkelsesresultat.

Vi er derfor kommet frem til denne prioriteringen av funksjonalitet for HEVN-M:

1. Lese egen timeplan
2. Vise timeplan for en gitt bruker
3. Endre, registrere opplysninger i egen timeplan
4. Lese personalia, f. eks adresse, telefonnr…
5. Registrere ”ute en kort tur”

6. Vise timeplan for en ressurs
7. Reservere – endre timeplan for en ressurs
8. Varsling før en planlagt aktivitet
9. Sende meldinger til andre brukere
10. Lese egne meldinger

I dette prosjektet tar vi i utgangspunktet ikke sikte på å gjennomføre hele prioriteringslista. I vår første iterasjon konsentrerer vi oss om kun nr 1 – ”Lese egen timeplan”, for å kunne få en viss oversikt over tidsbruk og arbeidsmengde ved fullstendig gjennomføring av en

funksjonalitet. Vi vil også finne ut hvordan J2ME fungerer og hvilke begrensninger den legger på oss.

Internasjonalisering

Det grafiske brukergrensesnittet på klientapplikasjonen skal benytte norsk som språk. Det skal være mulig å oversette til andre språk uten rekompilering, og uten særskilte programmeringskunnskaper. Slik oversettelse inngår ikke i prosjektet.

Systemutviklingsmodell

Vi har valgt å utvikle systemet etter en inkrementell modell, der vi i hver iterasjon klarlegger kravene for den funksjonalitet som skal implementeres. Kravspesifikasjon, design og implementasjon for hver iterasjon utarbeides som egne deler i et eget kapittel.

OPERASJON

Drift

Systemet skal under normal drift ikke kreve annet vedlikehold enn manuelt oppsett av brukernavn, passord og brukerrettigheter.

Det forventes at systemet skal ha tilgjengelighet tilnærmet på nivå med nettverkstilgang.

Det skal utvikles et verktøy for administrasjon av passord og brukerrettigheter.

Logging og feilrapportering

Systemet skal ha funksjonalitet for logging av hendelser på serversiden. Dette skal implementeres på en slik måte at forskjellige nivåer av logging kan velges uten å rekompilere kildekode. Logging skal kunne gjøres til valgbar fil.

Hvordan endring av loggenivå og loggfil utføres skal dokumenteres slik at driftspersonell kan utføre slike endringer uten spesielle verktøy.

Driftspersonell skal kunne få tilstrekkelig informasjon fra serverlogging på normalt driftsnivå.

Brukere skal på klientsiden få enkle men informative meldinger ved feil som oppstår, det kreves ikke logging på klientapplikasjonen.

Eventuell utvidelse med logging til et felles, sentralisert system er utenfor dette prosjektets mål.

Gjenervervelse etter feil

Det er ikke satt spesielle krav til gjenervervelse for serveren.

Feil på klienten skal ikke føre til uopprettelige, varige feil. Gjenervervelse skal kunne skje ved restarting av applikasjonen. Det bør tas hensyn til muligheten for feil på nettverk, strømforsyning, eller annet som fører til at klienten avslutter uten å logge av. Serveren bør slette sesjonsdata etter et gitt tidsintervall.

Oppstart og nedtaking

Det må utarbeides godt dokumenterte rutiner for opplasting av applikasjon på klient (telefon).

2.1.4. REFERANSER

1. http://java.sun.com/docs/codeconv/html/CodeConvToc.doc.html
2. MIDP 1.0: http://jcp.org/aboutJava/communityprocess/final/jsr037/index.html
3. CLDC 1.0: http://jcp.org/aboutJava/communityprocess/final/jsr030/index.html

2.1.6 ORDFORKLARINGER

“Bruker” og “ansatt” er referanser til brukere av HEVN.
”Klientapplikasjon”: Programvaren som kjøres på mobiltelefonen. Også kalt ”MIDlet” og ”klient”.
”Implementasjon”: Realiseringen av designet løsning. Også ”koding” er brukt.
”Iterasjon” og ”inkrement”: Ett gjennomløp i den inkrementelle systemutviklingsmodellen. En iterasjon er en frittstående versjon av systemet.
”Server”: Ofte brukt om våre klasser på tjeneren. Også brukt om hele oppsettet på tjenerenmaskinen.
”Hendelse” og ”aktivitet”: Brukes om et bestemt gjøremål i timeplanen. Eng: ”Task”.

2.2. DETALJERT KRAVSPESIFIKASJON

2.2.1. FUNKSJONELL SPESIFIKASJON

FUNKSJONELL STRUKTUR OG TVERR-RELASJONER

Datautveksling mellom klientapplikasjon og server utføres ved hjelp av SOAP. Dette betyr at denne kommunikasjonen benytter XML-formaterte fjernmetodekall og -returer. Dette gjelder generelt for all kommunikasjon mellom klient og tjener.

Serverklassene kommuniserer med databasen ved hjelp av JDBC og SQL.

Applikasjonen vil bestå av en liste av valg – hovedmeny. Listen vil inneholde de aktuelle menyvalgene. Disse valgene blir beskrevet nærmere i *kapittel 6 - Iterasjoner*.

Databasen, som er en relasjonsdatabase, inneholder ulike tabeller som alle systemets deler henter informasjon ifra. Vi vil bruke en utviklingsdatabase som er en kopi av den reelle. Se nærmere spesifikasjon av eksisterende database i de tidligere hovedprosjekt fra 1998 – HEVN, Hvem-Er-hVor-Når og fra 2002 – HEVN-J, Hvem-Er-hVor-Når, samt *pkt. 2.3.1.5 i dette kapittelet*.

Se figur 2-1 Systemskisser for hvordan modulene henger sammen

DATASPESIFIKASJON OG DATAORDLISTE

DATA RAMMEVERK

Systemet skal håndtere data i kategoriene Hendelse, timeplan (liste over hendelser for en dag), personalia.

INPUT

På klientapplikasjon:

- Tekststrenger fra brukergrensesnitt.
- Tidspunkt – tidsintervall som ønskes vist; Tid/datoobjekt.
- Trykk på menyvalg, bekreftelser o.l.; Hendelser.
- Respons fra metodekall til server; SOAP-melding.
- Sesjonsdata fra server; informasjonskapsel(cookie).

På serversiden:

- Metodekall fra klient; SOAP-melding
- Svar på spørring mot database; Resultatsett.
- Sesjonsdata fra klient; informasjonskapsel

PROSESSERING

På klientapplikasjon:

- Formatere valgt tid/dato for sending til server.
- Bygge opp SOAP-melding med metodekall.
- Respondere på hendelser fra brukergrensesnitt.
- Håndtere serverinformasjon for sesjonshåndtering.

På serversiden:

- Parsing av SOAP-melding fra klient, utføre metodekall.
- Sjekke brukernavn og passord mot database.
- Forme spørringer i SQL ut i fra tilsendte data.
- Håndtere resultatsettet fra databasespørring, danne SOAP-melding med respons til klient.
- Returnere ønskede data til klient.
- Håndtere sesjonsdata

OUTPUT

På klientapplikasjon:

- Timeplandata til skjerm; Tekst/grafikk.
- Skjermmeldinger til bruker, krav om bekreftelser etc.; Tekst/Grafikk.
- Metodekall til server; SOAP-melding.

På serversiden:

- Respons på metodekall fra klient; SOAP-melding
- Spørring mot database; SQL
- Utskrift til logg; Tekst

2.3. BEGRENSNINGER

2.3.1. SOFTWARE DESIGN BEGRENSNINGER

SOFTWARE STANDARDER OG SPRÅK

Som programmeringsspråk på klient og server vil Java brukes fordi det er operativsystemuavhengig.

For kildekode følger vi konvensjonen som beskrevet i ”Code conventions for the Java Programming language”[1].

SOFTWARE GRENSESNITT

SQL via JDBC mellom serverklasser og database.

SOFTWARE PAKKER/VERKTØY

- Javaversjoner som skal brukes:

  J2ME 1.04: Java 2 Micro Edition
  J2SE 1.4.1_01: Java 2 Standard Edition

- KSOAP – Bibliotek med klasser for å kunne håndtere SOAP-meldinger i J2ME
- Apache soap 2.3 for håndtering av SOAP-meldinger på serversiden.

SOFTWARE KOMMUNIKASJONSSTANDARDER OG GRENSESNITT

Kommunikasjonen mellom server og klient foregår med standard SOAP. Det vil si at metoder i serverklassene gjøres tilgjengelig for fjernmetodekall fra klientene. http benyttes som transportprotokoll og dataene (metodekall og retur), formateres som XML.

DATABASE

Systemet skal benytte eksisterende MSSQL-2000 database for HEVN-J(se dokumentasjon for HEVN-J). Under utvikling vil vi av praktiske årsaker bruke en MySQL-database som kopierer tabellstrukturen i den eksisterende databasen.

Ved igangkjøring av systemet vil det måtte gjøres eventuelle endringer for å tilpasse drivere m.m. til MSSQL 2000. Dette ligger utenfor dette prosjektet. Det brukes en egen MySQL-database for lagring av passord og brukernavn for HEVN-M.

OPERATIVSYSTEM

Klientdelen krever ikke særskilt operativsystem, men den virtuelle maskinen KVM for java må være installert. Øvrige deler av systemet er plattformuavhengig, men krever støtte for SOAP.

TOLERANSER, MARGINER OG MULIGHETER/TILFELLER

Programløsningen må være slik at det i fremtiden er mulighet for utvidelse og implementering av nye moduler.

2.3.2. HARDWARE DESIGN BEGRENSNINGER

HARDWARE KRAV OG OMGIVELSE

Klient applikasjoner(på telefonen) implementeres som en såkalt ”MiDlet”. Dette er en applikasjonsstandard for bruk på små enheter med begrenset minne og prosessorkraft. Denne standarden setter bestemte krav til maskinvaren, bla. a:

- Min. 128kB ikke-flyktig minne for lagring av program.
- Min. 32kB flyktig arbeidsminne (RAM).
- Min. 8kB ikke-flyktig minne for datalagring.
- Skjerm med min. 96*54 punkters oppløsning.
- To-veis nettverkforbindelse.

Gjeldende versjon av MIDP[2] er 1.0. Denne profilen bygger på CLDC1.0[3]. Dette betyr at telefonen må støtte MIDP 1.0 [2] og CDLC-konfigurasjonen[3].

2.4. ASPEKTER OMKRING LIVSSYKLUS

2.4.1. MODUL- OG INTEGRASJONSTESTING

Vi vil bruke en inkrementell utviklingsmodell med tre selvstendige iterasjoner. Applikasjonene som blir laget i respektive iterasjoner vil integreres i systemet og testes underveis. Ved avslutning av hver iterasjon skal det lages en testplan i samråd med veileder eller en representant som veileder utnevner. Denne testplanen skal teste funksjonaliteten som er laget i de aktuelle iterasjoner. Det skal etter hver iterasjon lages en kjørbar versjon av systemet

Etter avslutning av siste iterasjon vil hele systemet utførlig testes og rettet for de feil og mangler som måtte finnes. Dette vil foregå fram til innlevering av prosjektrapporten.

I samråd med veileder er det bestemt at gruppemedlemmene utfører all testing.

2.4.2. KONFIGURASJONS- OG VERSJONSSTYRING

Det blir laget nye versjoner ved større endringer av dokumenter og diagrammer. Siden prosjektet er forholdsvis lite og dokumentmengden ikke så stor har vi valgt å håndtere versjonene med ukenummer eller dato, pluss nummerert fortløpende fra nr.1.

2.4.3. KRAV TIL SUPPORT, SERVICE OG VEDLIKEHOLD

Hvem som skal ha ansvaret for support, service og vedlikehold av systemet vil avgjøres av HIG som oppdragsgiver hvis det bestemmes å sette i drift systemet.

2.5. ASPEKTER OMKRING INSTALLASJON

2.5.1. HARDWARE INSTALLASJON

Systemet krever ikke spesifikk hardware, oppdragsgiver bistår ved installasjon.

2.5.2. OVERGANG/OMLEGGING

Systemet skal ikke erstatte et tilsvarende system, men må integreres med eksisterende HEVN-J. Dette ligger utenfor prosjektoppgaven, oppdragsgiver har ansvaret for dette hvis det i fremtiden velges å ta systeme t i bruk.

2.5.3. OPPLÆRING

Det bør utarbeides en enkel bruksanvisning for klientdelen.

2.6. UTGIVELSER UNDERVEIS (DELIVERABLES)

Etter hver iterasjon vil det bli laget en kjørbar versjon av systemet. Med kjørbar menes at versjonene skal kjøres i testomgivelser.

2.7. PROSJEKTSTYRING OG KVALITETSSIKRING

Prosjektets veileder fungerer også som representant for oppdragsgiver. Fremdrift og arbeidsfordeling fremgår av forprosjektrapporten. Validering, verifisering og akseptansetester utføres ved slutten av hver iteras jon og etter siste iterasjon..

Kapittel 3 ANALYSE OG DESIGN

3.1. HVA SKAL SYSTEMET GJØRE

3.1.1. GENERELT

Brukeren skal kunne logge seg på HEVN-M serveren ved hjelp av en klientapplikasjon, altså mobiltelefon. Fra serveren skal bruker kunne hente ut data fra HEVN-M databasen. Brukeren skal kunne lese, endre og slette sin egen timeplan (”Min timeplan”), og lese timeplan og personalia for andre brukere (”Finn ansatt”). Utviklingen av funksjonalitetene ”Min timeplan” og ”Finn ansatt” er beskrevet i iterasjon 1, 2 og 3.

3.1.2. ARKITEKTUR

Det planlagte samarbeidet med Telenor i Trondheim er ikke videreført. Systemet settes opp slik som beskrevet under kavspesifikasjon (*se 2.1.4 og fig. 2-1 utviklingskonfigurasjon).*

3.1.3. HEVN-M APPLIKASJONENS GRENSESNITT

Vi tar sikte på å lage systemet mest mulig brukervennlig siden dette er en forutsetning for at systemet skal bli brukt. Minne og skjermstørrelse vil gi oss en spennende utfordring i å vise brukere de nødvendige data som ønskes.

Det finnes mange mobiltelefoner på markedet som opererer med forskjellig operativsystem og viser informasjon på forskjellig måte. Dette er bestemt av systemet og lar seg ikke endre av oss. I iterasjonene vil vi vise forskjellige eksempler (telefoner) på dette.

Se i UseCasene og punktene om HEVN-M applikasjonens grensesnitt i hver enkelt iterasjon for detaljerte beskrivelser av systemet.

3.1.4. DATABASE

Det er ikke nødvendig for oss å designe en ny database siden applikasjonen vår skal kunne integreres med den eksisterende databasen til HEVN-J. Det hadde vært ønskelig med en kopi av den reelle HEVN-J databasen for å utvikle systemet vårt mot.

3.2. HVORDAN LØSTE VI DET

3.2.1. ARKITEKTUR

Ett av valgene vi sto overfor var valget mellom to forskjellige grensesnitt mot Databasen.

Valget sto mellom:

- å benytte SQL fra våre klasser og til DB, og resultatsett tilbake fra DB og til våre klasser.
- å implementere løsningen slik at vårt system opptrer som en HEVN-J klient mot DB, slik at det nye grensesnittet blir mellom vårt system og eks. HEVN-J server.

Vi kom fram til å benytte det første alternativet fordi dette er enklere og gir et standardisert, ryddig grensesnitt(SQL) mot serveren. Fagpersoner på HiG har også sagt seg enig i denne vurderingen. Løsningen vår benytter JDBC(Java Database Connectivity) mot SQL-serveren til HEVN-J.

Det er satt opp test- og utviklingsomgivelser på arbeidsstasjonene på grupperommet. Dette består av Jakarta Tomcat servletkontainer med Apache SOAP installert for drift av "Web Service"/server, Sun Wireless Toolkit med telefonemulator for utvikling av klient i Java2 Micro Edition, Sun Java2 Standard Edition SDK 1.4 for utvikling av øvrige klasser, og MySQL database for vår utviklingsversjon HEVN–databasen og for lagring av data for tilgangskontroll. (*Se også figur 2-1 i kapittel 2*.)

3.2.2. HEVN-M APPLIKASJONENS GRENSESNITT

Grovdesign av grensesnittet ble gjort ved hjelp av blyant og viskelær på papirlapper av skjermstørrelse. Grunnen til at vi benyttet denne "papirlappmetoden" er at det er lettere å forkaste forslag tegnet på papir enn om det skulle vært implementert. Eksempler på denne designmetoden finnes under Analyse og Design delen, *pkt. 1.2.1, 2.2.1* og *3.2.1 i kapittel 6*. Dokumentasjon av skjermbildene gjøres sammen med implementasjon og testing og ligger under Analyse og Design delen, *pkt. 1.2.2, 2.2.2* og *3.2.2 i kapittel 6*. Navigasjon i brukergrensesnittet er beskrevet i UseCase'ene.

Siden displayet på mobilen er liten og skjermstørrelsen varierer fra telefon til telefon, har dette selvfølgelig gitt oss en del begrensninger. Det er begrenset mengde informasjon som kan vises for bruker. Derfor har vi for eksempel valgt å vise timeplan kun for en dag. Datamengde som blir vist fullstendig på skjermen avhenger av størrelsen på hver enkelt skjerm. Bruker må derfor bla seg nedover for å se resten av dataene.

I J2ME ligger det et forhåndsdefinert bibliotek som har gitt oss begrensninger og lite frihet i å velge hvordan informasjon skal vises. Vi kan ikke styre prioriteringen på kommandoene (OK, AVBRYT osv), dvs. at plasseringen på skjermen er forhåndsdefinert. Hvis det er flere kommandoer enn det er plass til å vise, lager klienten kommandoen "menu" eller "options,

hvor kommandoene det ikke er plass til plasseres. Enkelte mobiltelefoner har forhåndsdefinert kommandovalg på selve telefonen. Da vil kommandofunksjonene legges til disse knappene. *Se også vedlegg B(iterasjon 1 og 2) HEVN-M GUI*

3.2.3. DATABASE

Det ble avklart med oppdragsgiver at vi ikke fikk tilgang til HEVN-J databasen. Vi har derfor benyttet dokumentasjonen av HEVN-J, og laget en kopi av de aktuelle tabeller og kolonner. Vi har benyttet identiske tabell- og kolonne- navn, slik at en integrering med det eksisterende systemet skal bli så enkelt som mulig. Detaljert beskrivelse av den ligger i *vedlegg D*

Kapittel 4 IMPLEMENTERING

4.1 GENERELT

Vi har benyttet ”Code conventions for the Java Programming language”[1], som veiledende standard for javakoden. I samråd med veileder/oppdragsgiver har vi valgt å bruke Engelsk som språk i kildekode og dokumentasjon av kode. Dette føltes naturlig, og har ikke vært noen belastning for prosjektarbeidet.

4.2 KODEEKSEMPLER

I dette avsnittet tar vi med noen eksempler på hvordan kildekoden ser ut. Kodeeksempler finnes også i beskrivelse av implementasjonen i de enkelte iterasjonene (*se kapittel 6*). For fullstendig kildekode *se elektronisk vedlegg på CD.*
Under følger koden til en metode fra klassen TaskSerializer. Denne metoden parser XML på serversiden, og returnerer en ”Bean” som Task-objektet bygges opp fra:

```
/**Implementation of the working method for serialization*/
public Bean unmarshall(String inScopeEncStyle, QName elementType,
                       Nodetsk, XMLJavaMappingRegistry xjmr,
                       SOAPContext ctx)
   throws IllegalArgumentException  {

   Element root = (Element)tsk;
   String name = root.getTagName();
   // Check for null
   if (SoapEncUtils.isNull(root)) {
     return new Bean(Task.class, null);
   }
   // The object to return
   Task ret = new Task();
   // Get data from SOAP and set fields in java object
   QName type = new QName("http://www.w3.org/2001/XMLSchema", "dateTime");
   Bean dateBean;
   NodeList list = root.getElementsByTagName("start");
   if (list == null || list.getLength() == 0) {
     throw new IllegalArgumentException("No 'start' element found");
   }
   Element el = (Element)list.item(0);
   dateBean = xjmr.unmarshall(inScopeEncStyle, type, el, ctx);
   Date d = (Date)dateBean.value;
   String typeName = d.getClass().getName();
   ret.setStart((Date)dateBean.value);

   list = root.getElementsByTagName("stop");
   if (list == null || list.getLength() == 0) {
     throw new IllegalArgumentException("No 'stop' element found");
   }
   el = (Element)list.item(0);
   dateBean = xjmr.unmarshall(inScopeEncStyle, type, el, ctx);
```

```
   ret.setStop((Date)dateBean.value);

   list = root.getElementsByTagName("type");
   if (list == null || list.getLength() == 0) {
     throw new IllegalArgumentException("No 'type' element");
   }
   el = (Element)list.item(0);
   if(el.getFirstChild() == null){
      ret.setType("");
   }
   else{
      ret.setType(((Text)el.getFirstChild()).getData());
   }
   list = root.getElementsByTagName("comment");
   if (list == null || list.getLength() == 0) {
     throw new IllegalArgumentException("No 'comment' element");
   }
   el = (Element)list.item(0);
   if(el.getFirstChild() == null){
      ret.setComment("");
   }
   else{
      ret.setComment(((Text)el.getFirstChild()).getData());
   }
   list = root.getElementsByTagName("isprivate");
   if (list == null || list.getLength() == 0) {
     throw new IllegalArgumentException("No 'isprivate' element");
   }
   el = (Element)list.item(0);
   ret.setIsPrivate(new Boolean(((Text)el.getFirstChild()).getData()));
   return new Bean(Task.class, ret);
}
```

Her følger ”run-metoden” som et eksempel fra klientklassen UserSearchForm, som bl.a. viser GUI for søk etter ansatt, gjør fjernmetodekall via en metode i Clien klassen, og håndterer returdataene fra kallet.

```
/**Run seperate <code>Thread</code> to perform search as a SOAP call
*  to the server class (performed with Client.doSoapCall).
*  Call method showResultList or show Alert to
*  display search results.
*/
public void run(){
   String[] foundUsers = null;
   // load a vector with the arguments
   try{
      Vector args = new Vector();
      args.addElement(fName.getString());
      args.addElement(lName.getString());
      // do the call, get the results
      Vector data = null;
      Object obj = client.doSoapCall("userSearch", args);
      if(obj instanceof Vector){
         data = (Vector)obj;
      }
      if(data == null){
         throw new Exception("Resultatvektor = ittno!"); //todo
      }
      // get results into array
      Enumeration resultObjects  = data.elements();
      int noOfTasks = data.size();
```

```
      // if match found, one or more...
      if(noOfTasks > 0 && keepOn) {
         foundUsers = new String[noOfTasks];
         int count = 0;
         while(resultObjects.hasMoreElements()){
            foundUsers[count] = (String)resultObjects.nextElement();
            count++;
         }
         showResultList(foundUsers);
      }
      // No match found
      else if(keepOn){
         Alert al = new Alert(null,
                              "Fant ingen brukere som passer til søket",
                              null, AlertType.ERROR);
         al.setTimeout(Alert.FOREVER);
         client.getDisplay().setCurrent(al, this);
      }
   }catch(Exception e){
      System.out.println("Feil: " + e.toString());
      e.printStackTrace();
   }
}// end run()
```

For å gi et inntrykk av hvordan dataene formateres i XML, har vi også tatt med noen eksempler på utskrift av SOAP-kommunikasjonen. Dette er den XML'en som returneres som timeplandata for en gitt dag og bruker:

```
<?xml version='1.0' encoding='UTF-8'?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <ns1:getTasksForDayResponse xmlns:ns1="urn:no.hig.hevnm" SOAP-
        ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
          <return xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns2:Array"
          ns2:arrayType="ns1:task[3]">
              <item xsi:type="ns1:task">
              <start xsi:type="xsd:dateTime">2003-04-28T08:00:00.000Z</start>
              <stop xsi:type="xsd:dateTime">2003-04-28T09:30:00.000Z</stop>
              <type xsi:type="xsd:string">MÃ¸te</type>
              <comment xsi:type="xsd:string">PlanleggingsmÃ¸te for at, dataseksjonen</comment>
              <isprivate xsi:type="xsd:boolean">false</isprivate>
              </item>
              <item xsi:type="ns1:task">
              <start xsi:type="xsd:dateTime">2003-04-28T09:30:00.000Z</start>
              <stop xsi:type="xsd:dateTime">2003-04-28T11:45:00.000Z</stop>
              <type xsi:type="xsd:string">Ekskursjon</type>
              <comment xsi:type="xsd:string"></comment>
              <isprivate xsi:type="xsd:boolean">false</isprivate>
              </item>
              <item xsi:type="ns1:task">
                      <start xsi:type="xsd:dateTime">2003-04-28T11:45:00.000Z</start>
                      <stop xsi:type="xsd:dateTime">2003-04-28T13:00:00.000Z</stop>
                      <type xsi:type="xsd:string">Undervisning</type>
                      <comment xsi:type="xsd:string"></comment>
                      <isprivate xsi:type="xsd:boolean">false</isprivate>
```

```
            </item>
            </return>
        </ns1:getTasksForDayResponse>
    </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Her er et annet eksempel. Dette er XML'en som returneres fra serveren for å representere personalia for en valgt ansatt. Her har vi åpnet XML-dokumentet i en nettleser, og fikk dette skjermbildet:

Figur 4-1 xml -retur

Vi tar også med et eksempel på SQL. Eksemplet under viser SQL-filen som ble brukt for å sette opp tabellen "ansatt" i databasen:

```
CREATE TABLE Ansatt(
       login                VARCHAR(20)   NOT NULL,
       fnavn                VARCHAR(30)   NOT NULL,
       enavn                VARCHAR(50)   NOT NULL,
       tlf                  VARCHAR(8),
       jobbmail             VARCHAR(75),
       privtlf              VARCHAR(8),
       mobiltlf             VARCHAR(8),
       adresse              VARCHAR(50),
       postnr               CHAR(4),
```

```
       studtilgtimeplan    TINYINT,
       studtilgpersonalia  TINYINT
);

ALTER TABLE Ansatt ADD PRIMARY KEY (login);
ALTER TABLE Ansatt ADD FOREIGN KEY(postnr) REFERENCES Post(postnr);
```

Ytterligere eksempler på XML og SQL finnes vedlagt på CD.

4.3 VERKTØY

Vi har valgt å ikke bruke tunge utviklingsmiljøer som Sun Forte eller Borland JBuilder i dette prosjektet. Dermed har vi unngått å bruke mye tid på å lære å bruke verktøyene. Enklere verktøy fører også til et nærmere forhold til detaljene i implementasjonen. Vi har benyttet en enkel teksteditor(EditPlus), kommandolinjeverktøy fra Sun's java SDK 1.4, og Sun Wireless Toolkit for utviklingen. Vi har heller ikke benyttet særskilte verktøy for håndtering av XML. Databasen ble opprettet og manipulert med kommandolinjeverktøy og skriptfiler.

Mot slutten av prosjektet ble det benyttet en applikasjon med grafisk brukergrensesnitt for visualisering av databasen.("MySQLgui").

I forbindelse med testing under implementasjon, ble det mot slutten installert og brukt andre emulatorer. Blant andre for Nokia 3410 og 7210, Siemens S55 og "Nokia SDK for 60-series" (med Symbian OS).

4.4 ARKITEKTUR

Vi har satt opp test- og utviklingsomgivelser på arbeidsstasjonene på grupperom. Dette består av Jakarta Tomcat servletkontainer med Apache SOAP, slik som beskrevet i kravspesifikasjon (*2.1.4 og fig 2-1)*

Vi har benyttet en vanlig http-forbindelse, ikke https. Dette var heller ikke et absolutt krav.

Begrepet "Web Service" går i korthet ut på at klassemetoder blir gjort tilgjengelige for fjernmetodekall over http-protokoll, med SOAP som dataformat. SOAP er en standard for bruk av XML for metodekall og returdata.

Det krevde en del prøving, feiling og mye søking etter relevant dokumentasjon for å få arkitekturen på plass. Særlig kSOAP var beskjedent dokumentert. Vi fant imidlertid noe verdifull informasjon i artikler på Internet.(se litteraturliste)

4.5 KLIENT

Klientapplikasjonen er implementert som en ”MIDlet”, en applikasjon som kan kjøres på f.eks en telefon med java kompatibilitet. Se [2][3] i overordnet kravspesifikasjon. Her benyttet vi java2 Micro Edition (J2ME) som har API-er for slik utvikling. For å kunne benytte SOAP valgte vi kSOAP – biblioteket fra enhydra. Dette biblioteket inneholder klasser for å parse xml-dokumenter og for oppkobling og sending av fjernmetodekall over http.

For å slippe å sende/sjekke påloggingsdata for hvert serverkall hadde vi behov for å ta vare på http-sesjonsdata på klienten. Dette blir vanligvis ivaretatt av nettleseren ved ”vanlig” http-kommunikasjon. Her måtte vi finne en egen løsning. Vi valgte å lagre informasjonskapslene i http headeren fra serveren (cookies) som inneholder sesjons-id, og så sende med disse tilbake til serveren for å identifisere klienten ved senere metodekall. Denne kommunikasjonen foregår inne i kSOAP-klassene. For å få til dette måtte vi skrive om/utvide klassen HttpTransport i kSOAP.

Vi ser ut ifra testingen av systemet at XML parsing på klienten er svært ressurskrevende. Serialisering av objekter tar overraskende lang tid. Dette kan være mulig å optimalisere i en senere versjon av løsningen.

Det viste seg tidlig at J2ME har store begrensninger med hensyn til brukergrensesnitt. Vi mener allikevel at vi har klart å utnytte mulighetene best mulig. Dette er et typisk eksempel på hvorfor vi har kodet, testet, designet og kodet på nytt. En del av ideene og forslagene vi hadde i det opprinnelige GUI-designet lot seg rett og slett ikke implementere i J2ME. *Se vedlegg B*, bilder av brukergrensesnitt.

Klienten initialiseres ved hjelp av ”properties” lagret i en ekstern tekstfil. Denne inneholder navn på servertjeneste, adresse til server etc. Disse kan derfor endres på en enkel måte uten rekompilering av klientapplikasjonen.

Objekter som skal sendes som parametere eller returverdier i forbindelse med SOAP, må konverteres fra java-objekter til XML-datatyper (serialisering) og tilbake I kSOAP er det kun støtte for de enkleste datatypene som heltall og boolske verdier. Vi måtte derfor lage en egen klasse for serialisering av objekter av vår egen "Task" -klasse. Vi trengte også å sende en tabell (array) av slike Task-objekter. Dette løste vi med kode i klassen som utfører metodekallet. Task-klassen har bl.a to datofelter. Dette krevde også særbehandling, men her fantes det tilgjengelige klasser for serialiseringen i kSOAP-biblioteket.

Metodekallene fra klienten utføres i egen tråd, dette for at brukergrensesnittet skal være responsivt selv om kallet tar tid. Vi har laget en "progress bar" for å vise brukeren at noe skjer, og gi mulighet for å avbryte handlingen.

Data lagres i klientklassen, ikke i de grafiske komponentene. Dette for å skille data og presentasjon, og gi mulighet for å manipulere dataene uavhengig av brukergrensesnittet.

Samlingen av klasser må pakkes til en Java Archive fil(.jar). Denne filen sammen med en Java Application Descriptor(.jad) utgjør en "MIDlet-suite". Denne skal kunne lastes ned på en egnet telefon for installasjon. Dette ligger utenfor prosjektets arbeidsområde.

4.6 SERVERKLASSER

Det er utviklet en enkelt klasse med nødvendige metoder for å besvare fjernmetodekall. Denne klassen sørger for å laste databasedriver, bygge opp søkeuttrykk(sql) ut fra innparametre og foreta innhenting av data fra databasen via JDBC. Resultatet sendes så tilbake til klienten som returverdi.

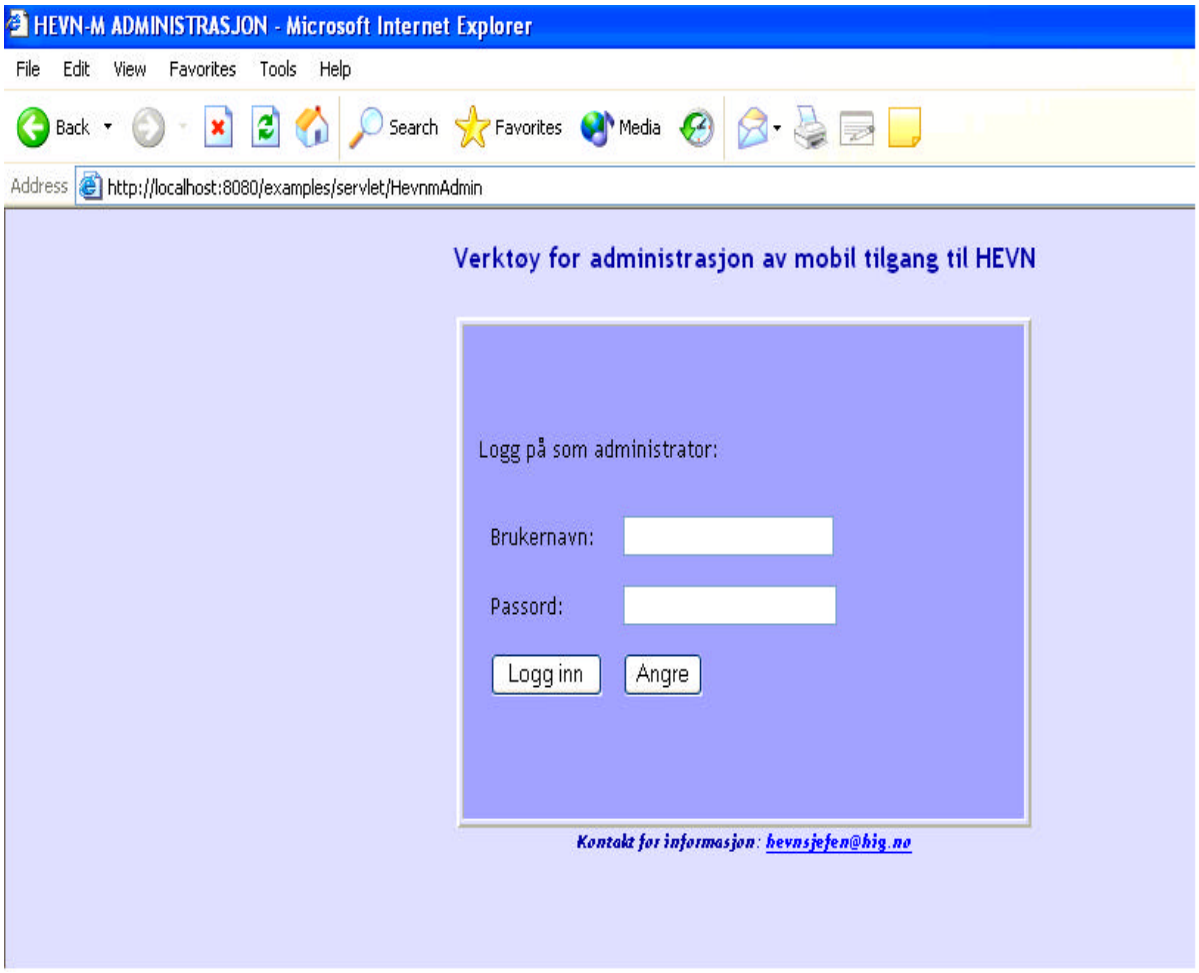
Håndteringen av sesjoner på serversiden løste vi ved å bestemme livstid (scope), for vårt objekt til "session" i distribusjonsbeskrivelsen (deployment descriptor). Objekter av klassen som besvarer et metodekall, vil da kun bli kontaktet av tilhørende klient. Sesjonsdata kan m.a.o lagres som variable i klassen. Nye klienter fører til opprettelse av et nytt objekt. Dette håndteres og konfigureres i Tomcat.

Som på klienten kreves det også her serialisering av objekter, men Apache SOAP har mye bedre støtte for ulike datatyper. For serialisering av Task-objekter benyttes samme klasse som på klienten.

Kryptering av lagrede passord i databasen ble gjort ved hjelp av "Message Digest" i java.security -pakken. Dette er en del av J2SE. Passordet sendes i klartekst fra klienten og krypteres v.h.a. SHA-1 algoritmen – en enveis hashfunksjon, før det sammenlignes med den lagrede verdien i databasen. Denne verdien er nødvendigvis kryptert på samme måte. Den lagrede verdien er en array av byte. For å få riktig verdi tilbake fra databasen måtte bytearrayen lagres som en såkalt "Blob", ellers ville dataene bli tolket som en tekststreng og forsøkt konvertert til tekstkarakterer. Ulempen ved en slik lagring er at selv den minste (TinyBlob) bruker 256 byte med plass.

Denne fremgangsmåten medfører at glemte passord ikke kan finnes igjen, det må legges inn nytt passord i slike tilfeller. Nye passord må legges kryptert inn i databasen. Vi måtte derfor utvikle en løsning for administrasjon av passord og tilgangsrettigheter. Se neste avsnitt.

4.7 ADMINISTRASJONS-SERVLET



Figur 4-2 administrasjons-servlet

En servlet er en javaklasse som besvarer kall over http fra en vanlig nettleser som klient. Responsen er en html side.

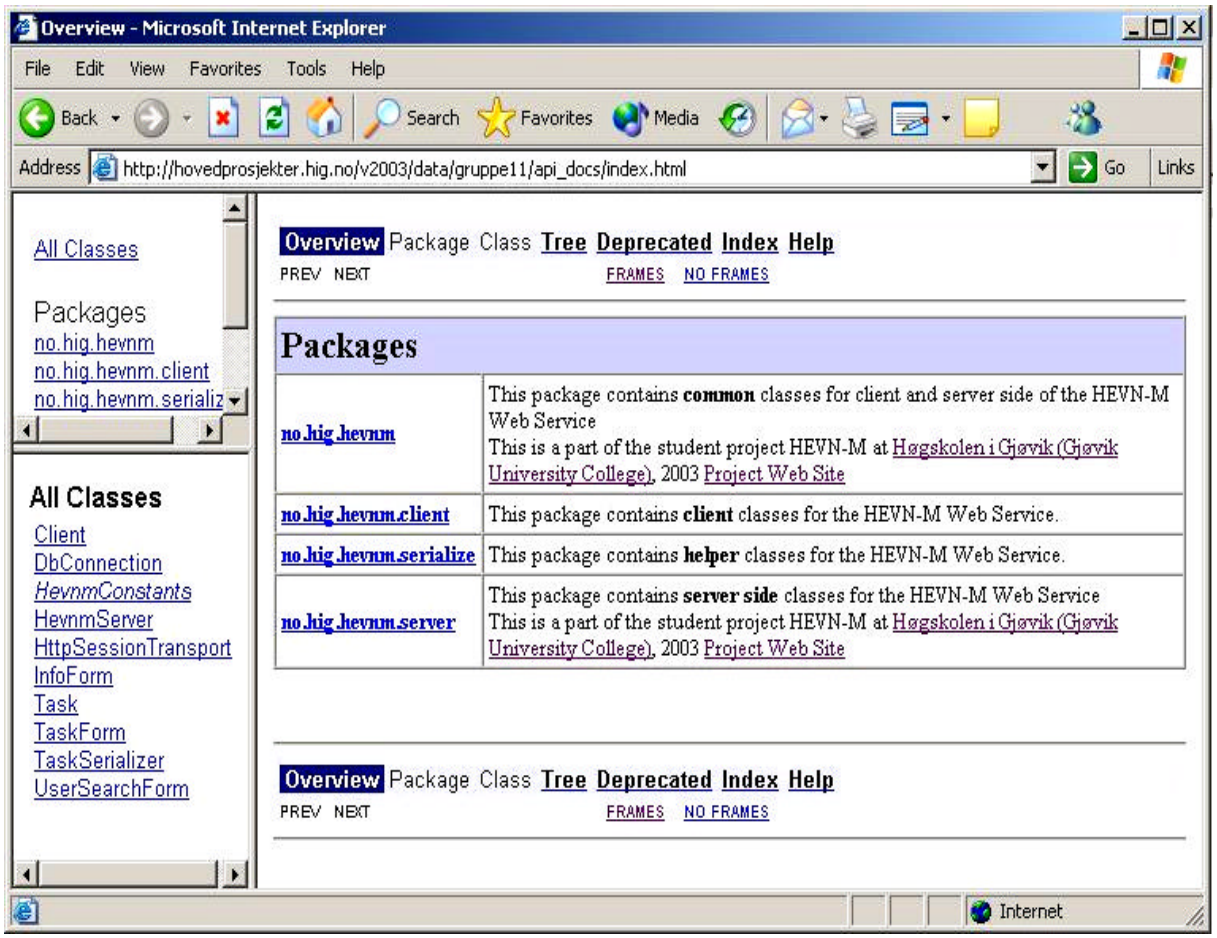
Den er utviklet for bruk under utvikling og testing av systemet. Denne virker, men må brukes "riktig"! Med dette menes at dersom systemet skal settes i drift, bør denne løsningen forbedres m.h.p. brukervennlighet og sikkerhet.

En forhåndsdefinert administrator kan logge seg på vår servlet med brukernavn og passord. Dette må i denne versjonen legges inn manuelt i databasen. Deretter søkes det etter aktuelle brukere blant registrerte brukere av HEVN tjenesten ved å skrive inn en del av personnavnet (ikke brukernavnet for nettverket). Det vises så en liste over treff. I denne listen markeres ønsket bruker. Det vises så en side der nytt passord legges inn. Passordet krypteres og lagres i databasen sammen med riktig brukernavn.

Det er altså ikke mulig å opprette en HEVN-M bruker som ikke allerede ligger i databasen over HEVN-J brukere. Under utvikling legges nye brukere inn direkte i databasen. Servleten har en medfølgende "properties-fil" med initialiseringsdata slik at den skal være mer fleksibel for fremtidig bruk. *Se diagram i vedlegg K.*

4.8 DOKUMENTASJON AV KODE

All javakode er dokumentert v.h.a. javadoc. Dette verktøyet genererer dokumentasjon på html format slik at den kan leses i en vanlig nettleser. Figur 4-3 viser et eksempel på denne dokumentasjonen vist i en nettleser. Kildekoden inneholder også vanlige kommentarer utover dette. Databasen er dokumentert med skriptfilene som er brukt, og utskrift fra kommandolinjeverktøy og grafiske verktøy.



Figur 4-3 javadoc i nettleser

Kapittel 5 TESTING OG KVALITETSSIKRING

5.1. METODER FOR Å SIKRE KVALITET

PROSESSKVALITET

Fremdrift og planlegging av arbeidsoppgaver er ivaretatt ved bruk av Gantt-skjemaer for fremdriftsplan. Det er ført prosjektdagbok under hele prosjektet.

Intern evaluering av gruppens arbeid har vært viktig for å sammenligne planlagt prosess med den faktiske gjennomføringen. Disse evalueringene ved avslutning av iterasjoner inngår i dokumentasjonen.

Inkrementell modell har gitt oss mulighet for å lære av problemløsning og mer eller mindre vellykkede fremgangsmåter i tidligere iterasjoner.

PRODUKTKVALITET

Det er ikke benyttet spesielle teknikker for kvalitetssikring utover testing etter plan i *pkt. 1.4.1(iterasjon 1)* og *pkt.2.4.1(iterasjon 2*). For detaljert dokumentasjon av testresultatene – *se vedleggene E i iterasjon 1 og 2.* All kode er også blitt testet som en integrert del av utviklingsarbeidet.

Arbeidet med kravspesifikasjon og design legges også til grunn for det ferdige produktets kvalitet. Endringer i forhold til kravspesifikasjon og kjente mangler ved utgitt versjon av systemet finnes i *vedleggene F i iterasjon 1 og 2.*

Kapittel 6 ITERASJONER

Dette kapitlet inneholder dokumentasjon for de enkelte iterasjonene. Vi har etter oppdragsgivers ønske forsøkt å levere en tilnærmet komplett rapport for hver iterasjon. Dette fører til at noe informasjon i ettertid er flyttet til overordnet nivå, andre punkter kan være gjentatt i alle iterasjoner. Resultatet kan noen steder virke lite leservennlig, men vi håper oversikten likevel er ivaretatt. Vi kommer tilbake til dette forholdet i evaluering av prosjektet.

1. ITERASJON NR.1

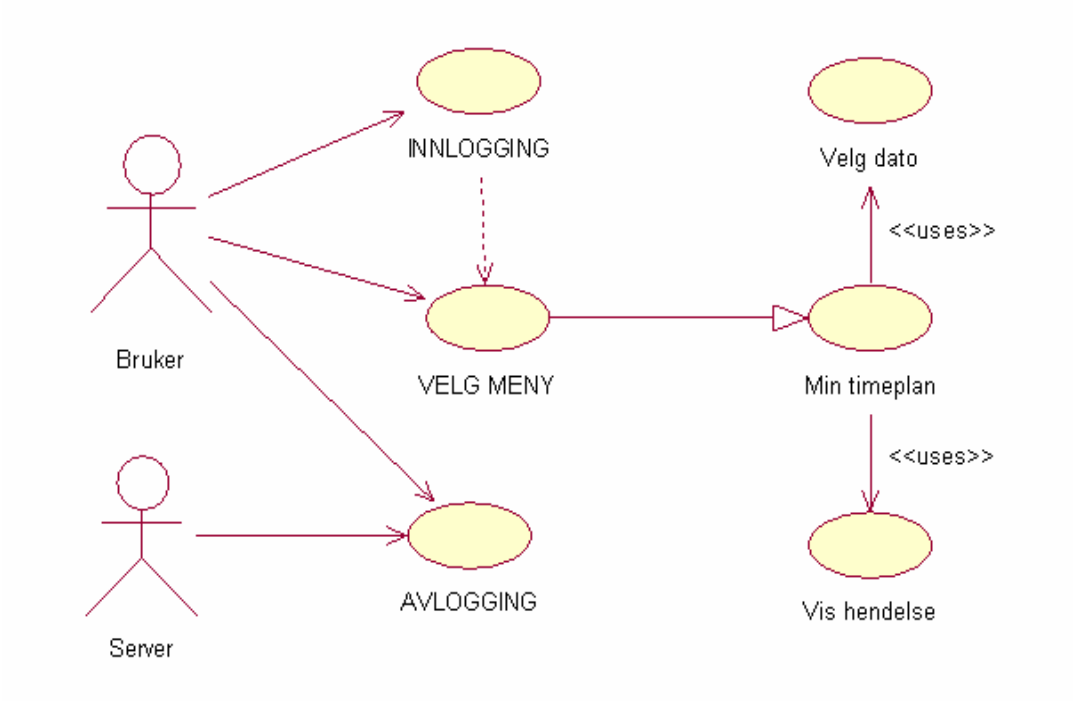
Denne iterasjonens innhold er å utvikle funksjonalitet for å kunne lese sin egen timeplan. Dette innebærer også pålogging.

1.1. KRAVSPESIFIKASJON

Dette er en detaljert kravspesifisering for de funksjonaliteter som tilhører denne iterasjonen. Dette er uttrykt ved UseCase-beskrivelsene Innlogging, Velg meny, Min timeplan, Velg dato, Vis hendelse og Avlogging.

1.1.1. FUNKSJONELL SPESIFIKASJON

USECASE DIAGRAM



Figur 6-1-1 Viser sammenhengen mellom UseCasene

USECASE BESKRIVELSER

Her følger den formelle beskrivelsen av hvert enkelt UseCase:

INNLOGGING

Mål: Identifisering og autentisering for å få tilgang til HEVN-M.

Aktør: Bruker

Beskrivelse: Bruker oppgir brukernavn og passord for å koble til HEVN-M tjenesten.

Prebetingelser:

- Må ha registrert brukernavn

- Må ha gyldig passord
- Tilgang til nettverk

Postbetingelser:
- Pålogget på systemet
- Sesjonsdata registrert på server
- Bruker skal ha melding om påloggingsstatus
- Går automatisk til neste UseCase hvis pålogging er OK.

Scenarier:
1. Bruker taster inn brukernavn, passord og OK
2. Klient sender data til server
3. Server sjekker brukernavn og passord
4. Server sender melding om status tilbake
5. Innlogging OK: Sesjonsdata lagres
6. Klienten viser påloggingsstatus for bruker
7. Hvis innlogging OK ? ”Velg visning”

Alternative scenarier:

5.1 Hvis innlogging ikke OK ? sesjon avsluttes

7.1 Hvis innlogging ikke er OK ? ”start”

Spesielle krav:

Brukernavn og passord sendes over http.

Passord lagres kryptert på server.

VELG MENY

Mål: Bruker velger en av HEVN-M’s tjenester

Aktør: Bruker

Beskrivelse: Hovedmeny som viser tilgjengelige tjenester i HEVN-M:
- ”Min timeplan” – vise egen timeplan for en valgt dag

Prebetingelser:
- Bruker er pålogget

Postbetingelser:
- Viser skjermbildet ”Velg dato” eller ”Avlogging”
- Ønsket tjeneste er lagret

Scenarier:

1. Bruker velger tjeneste og trykker OK

2. Klienten går til ”Velg dato”

Alternative scenarier:

1.1 Bruker velger ”AVSLUTT” ? Avslutter HEVN-M

MIN TIMEPLAN

Mål: Få en oversikt over dagens hendelser

Aktør: Bruker

Beskrivelse: Viser brukers hendelser en gitt dag.
Viser hendelsestype og dens start- og stopptid

Prebetingelser:

- Valgt tjeneste
- Valgt dato

Postbetingelser:

- Trykker ”OK” ? ”Vis hendelse” (Detaljbeskrivelse av den enkelt aktivitet)
- Trykker ”Avslutt” ? ”Avlogging”
- Trykker ”Tilbake” ? ”Velg dato”

Scenarier:

1. Presenterer liste over valgt datos ”hendelser”
2. Velger ”Hendelse” og trykker ”OK”
3. Går til valgt ”Hendelse”

Alternative scenarier:

1.1 Liste er tom – gi bruker melding om dette
2.1 Velger ”Avslutt” ? ”Avlogging”
2.2 Velger ”Tilbake” ? ”Velg dato”

VELG DATO

Mål: Velge ønsket dato

Aktør: Bruker

Beskrivelse: Bruker velger ønsket dato for visning

Prebetingelser:

- Bruker har valgt en tjeneste

Postbetingelser:

- Valgt dato lagret
- Trykker ”OK” ? valgt tjeneste(”Min Timeplan”)
- Trykker ”Tilbake” ? ”Velg meny”

Scenarier:

1. Presenterer dagens dato som standardverdi
2. Bruker velger datofelt – kalender vises

Alternative scenarier:

2.1 Trykker ”OK” – dagens dato lagres
2.2 Valgt dato lagres
- Bruker trykker ”Tilbake” ? ”Velg meny”

VIS HENDELSE

Mål: Få detaljert beskrivelse av en hendelse

Aktør: Bruker

Beskrivelse: Viser en brukers detaljerte beskrivelse av en valgt hendelse. Dette kan være start- og stopptid, hendelsestype og kommentarer.

Prebetingelser:

- Hendelse er valgt

Postbetingelser:

- Bruker har valgt
  - ”Avslutt”
  - ”Hovedmeny”
  - ”Tilbake”

Scenarier:

1. Presenterer dato om en hendelse
2. Velger
   - ”Avslutt”
   - ”Hovedmeny”
   - ”Tilbake”

Alternative scenarier:

1.1Liste er tom – gi bruker melding om dette
2.1Velger ”Avslutt” ? ”Avlogging”
2.2 Velger ”Hovedmeny” ? ”Velg meny”
2.3 Velger ”Tilbake” ? ”Velg dato”

AVLOGGING

Mål: Logge av HEVN-M og server

Aktør: Bruker og server

Beskrivelse: Logge av HEVN-M og server

Prebetingelser:
- Pålogget

Postbetingelser:
- Avlogget
  - Sesjon avsluttet

Scenarier:
1. Bruker får spørsmål om vedkommende virkelig vil logge av
2. "JA" – klient sender melding til server
3. Klient mottar melding om avlogging fra server og viser den til bruker
4. Avslutter klinetapplikasjonen

Alternative scenarier:

2.1 "Nei" ? "Velg meny"

1.1.1. TILGJENGELIGHET

Tilgjengeligheten for tjenesten "Min timeplan" er avhengig av at nettverket og systemet på serversiden er tilgjengelig, og at bruker er pålogget.

1.1.2. TESTING AV IMPLEMENTASJON

Funksjonaliteten skal testes kontinuerlig underveis i utviklingen av systemet. Disse testene utføres av gruppemedlemmene.

Det skal lages en løsning for å teste klientapplikasjon og servermetoder hver for seg. Dette utføres mot/med testklasser, med mulighet for generering av resultatfiler.

Ved avslutning av iterasjonen skal funksjonaliteten testes av gruppemedlemmene sammen med representant(er) for oppdragsgiver.

AKSEPTANSEKRAV

Disse punktene skal testes og godkjennes:

- Oppstart av klientapplikasjon (launch).
- Innlogging:
  - Med riktig/feil passord og riktig/feil brukernavn.
  - Med feil datatype som input.
  - Med for lange/korte tekststrenger.
- Valg av dato for visning.
- Valg av tid for visning.
- Presentasjon av dataene.
- Valg av detaljert visning av en aktivitet/gjøremål.
- Presentasjon av detaljerte opplysninger.
- Avslutning og angremuligheter.

1.2. ANALYSE OG DESIGN

1.2.1. HVA SKAL SYSTEMET GJØRE

I denne iterasjonen foreslo veileder at vi lager kun en funksjonalitet. Dette på grunn av at det er mye annet som skal gjøres før utviklingen kan starte. Vi må få på plass serverarkitektur, database og klientapplikasjon(telefonemulator) og få kommunikasjonen mellom disse til å fungere. Dette mente vi var lurt siden vi ikke har erfaring med dette og ikke vet det reelle tidsforbruket vårt.
Vi har valgt å implementere funksjonaliteten "Min timeplan", hvor ansatte skal kunne gå inn og se sin egen timeplan. Etter at bruker har valgt dato, skal en liste over denne dagens gjøremål og tidspunkt for disse vises. Brukeren skal kunne merke et gitt tidspunkt og gjøremål og få en mer detaljert oversikt – "Vis hendelse".

Ved første vellykkede innlogging skal brukernavnet lagres i klienten og automatisk vises ved senere innlogginger. Ved feil brukernavn og/eller passord skal det gis melding om dette og bruker får ny sjanse til pålogging.

Se detaljbeskrivelse i UseCase'ene for denne iterasjonen.

Når det gjelder "Min timeplan", er det foreslått at vi skal lage en liste over forhåndsdefinerte aktiviteter som bruker kan velge. Dette er enklere og raskere enn om bruker selv må skrive inn aktiviteter. I "Vis hendelse" skal bruker selv ha muligheten til å skrive inn det som ønskes innenfor en fastsatt grense. Feltet kan også stå tomt. Vi vil ikke her ta noe hensyn til hvordan dette blir vist på skjermen.

Klassestruktur og samhandling mellom objekter fremgår av klassediagram – *vedlegg C* og sekvensdiagram – *vedlegg D.*

Bilder 6-1-1 Papirlappmetoden – Første utkast av design på papir.

1.2.2. HVORDAN LØSTE VI DET

INNLOGGING/AVLOGGING

Bilde 6-1-1 Innlogging

Feltene brukernavn og passord skal skrives i samme skjermbilde. Hvis bruker taster inn feil tegn i minst én av feltene, vil bruker få mulighet til å taste inn på nytt. Meningen var at etter første vellykkede innlogging så skulle brukernavnet lagres i klienten. Dette fikk vi ikke tid til å gjøre i denne iterasjonen. Siden vi ikke anså dette som grunnleggende og viktig ble det nedprioritert og overført til neste iterasjon for at det ikke skulle påføre prosjektet ytterligere forsinkelser. Dette var noe veileder bifalte og var enig i.

VELG MENY

Bruker vil få en liste over de valg/retninger som er mulig. Etter en kort melding om vellykket pålogging, er dette det første skjermbilde som bruker får. Bildet viser tre valg kun for demonstrasjon, men det er kun ”Min timeplan” som fungerer foreløpig. Bruker kan avslutte HEVN-M ved å velge kommandoen ”Avslutt”.

Bilde 6-1-2 Hovedmeny

VELG DATO

Bilde 6-1-3 Velg dato. Eksempel på grafisk fremstilling på to forskjellige mobiltelefoner.

Forutsatt at "Min timeplan" er valgt, vil bruker få vist dagens dato som standardverdi. Bruker har mulighet til å velge annen dato. Som vi ser av bildene over har de forskjellige telefonene sin egen måte å håndtere dette på, avhenging av telefonenes implementasjon av MIDP (*Se pkt. 2.1.6 nr 2 i kapittel 2).*

MIN TIMEPLAN

Bruker får vist en liste over den valgte dagens aktiviteter og tidspunktene for disse. Det er ikke laget en forhåndsdefinert liste over aktiviteter. Dette vil bli tatt stilling til senere. Klientapplikasjonen kan avsluttes her, eller man kan gå tilbake og velge ny dato.

Bilde 6-1-4 Viser liste over aktiviteter en gitt dag

VIS HENDELSE

Her vises en mer detaljert beskrivelse av aktiviteten For å komme hit må bruker markere et tidsrom hun ønsker å se. Det er kun mulighet for å se en aktivitet av gangen. Hvis detaljer for annen aktivitet ønskes, må bruker velge ”Tilbake”, og markere en annen aktivitet først.

Bilde 6-1-5 Detaljert beskrivelse av en valgt aktivitet

1.3. IMPLEMENTERING

1.3.1 GENERELT

Arbeidet med koding, design og testing har gått nokså parallelt. Dette på grunn av nødvendigheten av å teste og gjøre forsøk for å finne ut hvordan de forskjellige delene av systemet skulle/kunne fungere sammen. Arkitekturen med trådløse klienter mot en ”Web Service” er relativt ny teknologi, og derfor ukjent - og spennende.

All koding er utført i forskjellige varianter av programmeringsspråket Java. Vi har i samråd med veileder valgt konsekvent bruk av Engelsk som språk i koding og dens dokumentasjon. *Se neste pkt*. Vi har videre etter beste evne fulgt java code conventions. *Se nr 1 i pkt 2.1.6 i overordnet kravspesifikasjon*.

For oversikt over klassestruktur og grafisk fremstilling av samhandlingen mellom objekter vises til *vedlegg C - klassediagram* og *vedlegg D - sekvensdiagram*.
For detaljert dokumentasjon vises til egen kodedokumentasjon og kildekode.

1.3.2 KODEEKSEMPLER

Eks 1: En del av koden for klassen ”Task”. Denne klassen representerer en aktivitet, og brukes i både klient og server. Det er en array/tabell av slike objekter som sendes tilbake fra serveren når bruker ber om timeplanen for en dag.
Eks 1:

```
package no.hig.hevnm;

import java.util.Date;

/**This class structures the data of a task (no: hendelse)
 * Objects of this class are used for representing taskdata on the
 *client and for serialized SOAP-messages
 * <HR>
 * This is a part of the student project HEVN-M at
 * <A HREF="http://www.hig.no">Høgskolen i Gjøvik</A>, 2003<BR>
 * <A HREF="http://hovedprosjekter.hig.no/v2003/data/gruppe11/">
 *Project Web Site</A>
 *@author Arnstein Sveum (arnstein.Sveum@hig.no)
*/
public class Task extends Object {
   private Date start;
   private Date stop;
   private String type;
   private String comment;
   private Boolean isPrivate;

   /**Default empty constructor*/
   public Task(){}

   /**Constructor with parameters
   *  @param sta The start of the task in ms from the 'epoch'.
   *  @param sto The stop of the task in ms from the 'epoch'.
   *  @param t The type of this task.
   *  @param c The owners comment to this task.
   *  @param ip Marking the Task 'isPrivate' if true
   */
   public Task(Date sta, Date sto, String t, String c, Boolean ip){
      start = new Date(sta.getTime());
      stop = new Date(sto.getTime());
      type = new String(t);
      comment = new String(c);
      isPrivate = new Boolean(ip.booleanValue());
   }
- - - - - - - - - - - -
```

Eks 2: En klasse som brukes for å lage bevegelse I skjermbildet som vises når klientapplikasjonen utfører operasjoner som tar noe tid (for eksempel pålogging, hente timeplandata). Denne klassen arver fra ”TimerTask” slik at run-metoden kan kalles med bestemte tidsintervall. Til dette brukes en Timer.

```
   // Inner class for animating a gauge, used as a progress bar
   // The run() -method of this class is scheduled by a Timer.
```

```
private class ProgressBarAnimator extends TimerTask {
   private Gauge progBar;
   private String lab1, lab2;
   int level = 0;

   public ProgressBarAnimator(Gauge g, String str1, String str2 ){
      progBar = g;
      lab1 = str1;
      lab2 = str2;
   }
   // Animating the gauge
   public void run() {
      if(progBar.getValue() < progBar.getMaxValue()) {
         progBar.setValue(progBar.getValue() + 1);
      }
      else {
         progBar.setValue(0);
         progBar.setLabel(lab1);
      }
      if(progBar.getValue() %3 == 0){
         if( progBar.getLabel().equals(lab1) )
            progBar.setLabel(lab2);
         else
            progBar.setLabel(lab1);
      }
   } // end run()
} // end class updateProgressBar
```

1.4. TESTING

1.4.1 GENERELT

For generell beskrivelse av testing, *se pkt 1.1.2 i kravspesifikasjon*. Følgende endringer er gjort i det praktiske testarbeidet:

Testing er kun utført av gruppemedlemmene, dette i samråd med oppdragsgiver.
Det er ikke benyttet særskilte testklasser under avsluttende testing.
Avsluttende tester er utført etter testspesifikasjonene nedenfor.

1.4.2 TESTSPESIFIKASJON

Systemet testes i testomgivelsene på våre lokale arbeidsstasjoner med telefonemulator..
Detaljert spesifikasjon og testresultater finnes i *vedlegg E – Testplan og testresultat* for denne iterasjonen.

HVA SKAL TESTES

- Oppstart av klientapplikasjon (launch).
- Innlogging:
  - Med riktig/feil passord og riktig/feil brukernavn.
  - Med tomme tekststrenger.
  - Når server er utilgjengelig.
- Valg av dato for visning.

- Presentasjon av data for en dag.
- Valg av detaljert visning av en aktivitet/gjøremål.
- Presentasjon av detaljerte opplysninger.

- Avslutning, navigasjon og angremuligheter.
- Oppkobling når server ikke er tilgjengelig
- Avslutte server mens klienten er tilkoblet

1.4.3 TESTRESULTAT

Kommentarer til *Vedlegg E Testplan og testresultat*. Testing har foregått underveis i utviklingsarbeidet og har vært en viktig oppgave. Ved slutten av iterasjonen ble det laget en testplan for sluttesting av applikasjonene. Den ble utarbeidet litt vilkårlig mens vi navigerte oss gjennom de ulike funksjonaliteter. I denne iterasjonen var det ikke så mange funksjonaliteter som ble laget og det skulle ikke foretas noen endringer i databasen, så systemet var relativt ukomplisert. Testingen avdekket manglende feilhåndtering ved enkelte nettverksfeil. På grunn av forsinkelser bestemte vi å forskyve noen av de feil og mangler vi oppdaget til senere iterasjoner, *se Vedlegg F – Uløste oppgaver.* Utover dette møtte vi ikke så store problemer under sluttestingen og testingen ga oss ingen overraskelser som vi ikke kunne rette opp med en gang.

1.5. KONKLUSJON

1.5.1. DRØFTING AV FRAMDRIFTSPLAN FOR 1. ITERASJON

Det ble avvik i forhold til den planlagte framdriftsplanen på grunn av (at):

- Manglende erfaring gjør det vanskelig å planlegge tidsforbruk for de enkelte gjøremål.
- Vi måtte sette oss inn i mye nytt stoff i forbindelse med XML-SOAP og Web Service arkitektur.
- Vanskeligheter med å jobbe parallelt med programmering i forskjellige moduler pga. ulik programmeringskompetanse blant gruppens deltakere.

- Vi måtte i større grad enn planlagt lage testimplementasjoner for å klarlegge funksjonalitet før vi kunne gjøre mer fullstendig design.

Vi regner imidlertid med at mange av problemene som er løst i 1. iterasjon, vil spare oss for mye arbeid i senere iterasjoner. Dette fordi vi kjenner arkitekturen bedre, har lært mye om muligheter og begrensninger i J2ME og har lært bruk av diverse verktøy.

I denne iterasjonen har vi, i tillegg til det faglige, lært at det er vanskelig å planlegge prosessen på forhånd. Dette er en følge av vi ikke har tidligere erfaringer verken i store prosjektarbeid, eller forhåndskunnskap om tidsbruk for de forskjellige gjøremål.

Oversikt over tidsforbruket.

	Timer
Forprosjekt	119
Overordnet kravspek	57
1. iterasjon, kravspek	17
Utviklingsmiljø Implementering	13 129
Design GUI	59,5
Implementering GUI	37
Møtevirksomhet m/ veileder	25
Design, systemet Prosjektrapport, iterasjon 1	27,5 64
Sum	**548**

Pausene er ikke silt ut.

Se vedlegg A - Ganttskjema for å sammenligne den opprinnelige framdriftsplanen med den endelige.

1.5.2. EVALUERING AV ITERASJONEN

PRODUKT

Vi har fått systemet med klientapplikasjon på telefonemulator, egne Javaklasser distribuert som web-service og nødvendige programmer og biblioteker på plass. Dette var en krevende jobb fordi det var et "ukjent terreng" – det var mye nytt stoff som vi skulle sette oss inn i. Vi er derfor godt fornøyd med det foreløpige resultatet, selv om det mangler noe funksjonalitet på løsningen. Disse manglene anså vi ikke som grunnleggende viktig og ville ikke forskyve tidsplanen vår ytterligere. Vi valgte derfor å avslutte 1. iterasjon og vil i planleggingsfasen for 2. iterasjon avgjøre om disse manglene skal utbedres som en del av 2. iterasjon. *Se vedlegg F – Uløste oppgaver.*

Vi har også brukt mye tid på å lære oss bruk av verktøyene Office Project for Gantt-skjemaer og Rational Rose Edition for systemutviklingen. I tillegg har vi brukt mye tid på å lage en mal til rapportskriving. På grunn av systemutviklingsmodellen vår har det vært vanskelig å slavisk følge ferdiglagde maler. Oppdragsgiver ga oss litt frie hender angående dette, og vi har derfor laget en rapport mal som hittil fungerer. Denne er vi klar over kan endre seg i løpet av de neste iterasjonene.

PROSESSEN

Valg av en inkrementell systemutviklingsmodell har ført til at vi tidlig i prosjektet har et synlig produkt. Dette har vært motiverende for gruppa. Det har vært lærerikt i og med at vi har fullført en "runde". Vi kan nå tydeligere se hva som burde vært gjort annerledes og hva vi brukte tid på.

Vi føler at det er brukt litt mye tid på planlegging før vi visste nok om muligheter og begrensninger. For å få oversikt over hvordan systemet skulle fungere, var det vanskelig å følge standardmodellen med å planlegge, designe og kode. Vi begynte i litt feil ende med testimplementasjoner, for så å gå til planlegging og design. Deretter igjen gå til implementasjonsfasen. På den måten klarte vi å få tak på hvordan systemet skulle være.

Arbeidet med kravspesifikasjon har blitt gjort med litt "frie hender" fra oppdragsgiverens side.

Arbeidsfordelingen i gruppen har fungert godt i perioden, hovedsakelig er design og arbeidet med rapport og dokumentasjon utført av Harbosen og Krasniqi, mens det meste av implementasjon og koding er gjort av Sveum. Det er enighet om at denne fordelingen har fungert godt. Det har ikke oppstått kritiske problemer internt i gruppen eller i forhold til veileder/oppdragsgiver.

Gruppemedlemmene er enige om at egen og andres arbeidsinnsats er tilstrekkelig, men det bør nevnes at Sveum har utmerket seg. Han har gitt mye mer enn det som forventes av et 6 vekttalls prosjekt.

Se også statusrapport – Vedlegg E.

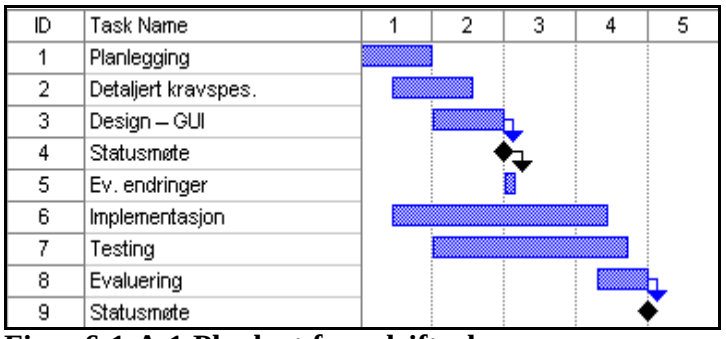
1.5.3. HOVEDKONKLUSJON

Gruppa er fornøyd med valg av systemutviklingsmodell. Den er motiverende på grunn av synlig resultater tidlig i prosjektet. Den gir også anledning til å gjøre forbedringer ved senere iterasjoner og er dermed utrolig lærende.

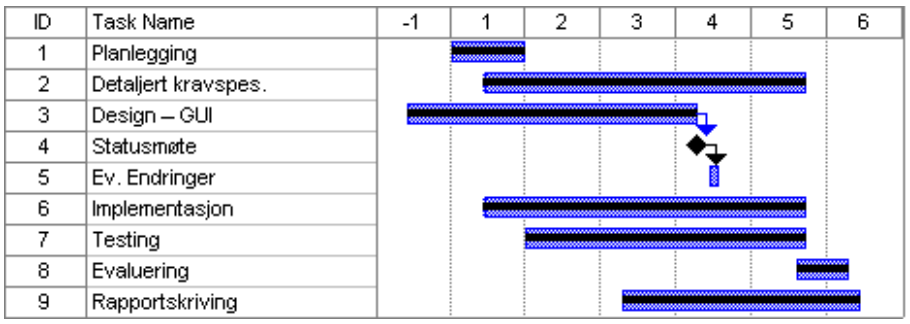
Til tross for at vi ikke klarte å løse absolutt alle de planlagte oppgavene så er vi fornøyd med resultatet. Gruppa har jobbet ordentlig i hele perioden og fungerer bra sammen.

1.6. VEDLEGG

A. GANTT-SKJEMA



Figur 6-1-A-1 Planlagt framdriftsplan



Figur 6-1-A-2 Reell framdriftsplan

B. HEVN-M GUI

I dette vedlegget viser vi de fleste av skjermbildene som inngår i denne iterasjonen. Vi henviser til tilhørende UseCase, og lar stort sett illustrasjonene tale for seg selv. Det er viktig å være klar over at det faktiske utseendet og navigasjonsmulighetene kan variere mye på forskjellige plattformer/telefoner.

Start av applikasjonen UseCase ”innlogging”

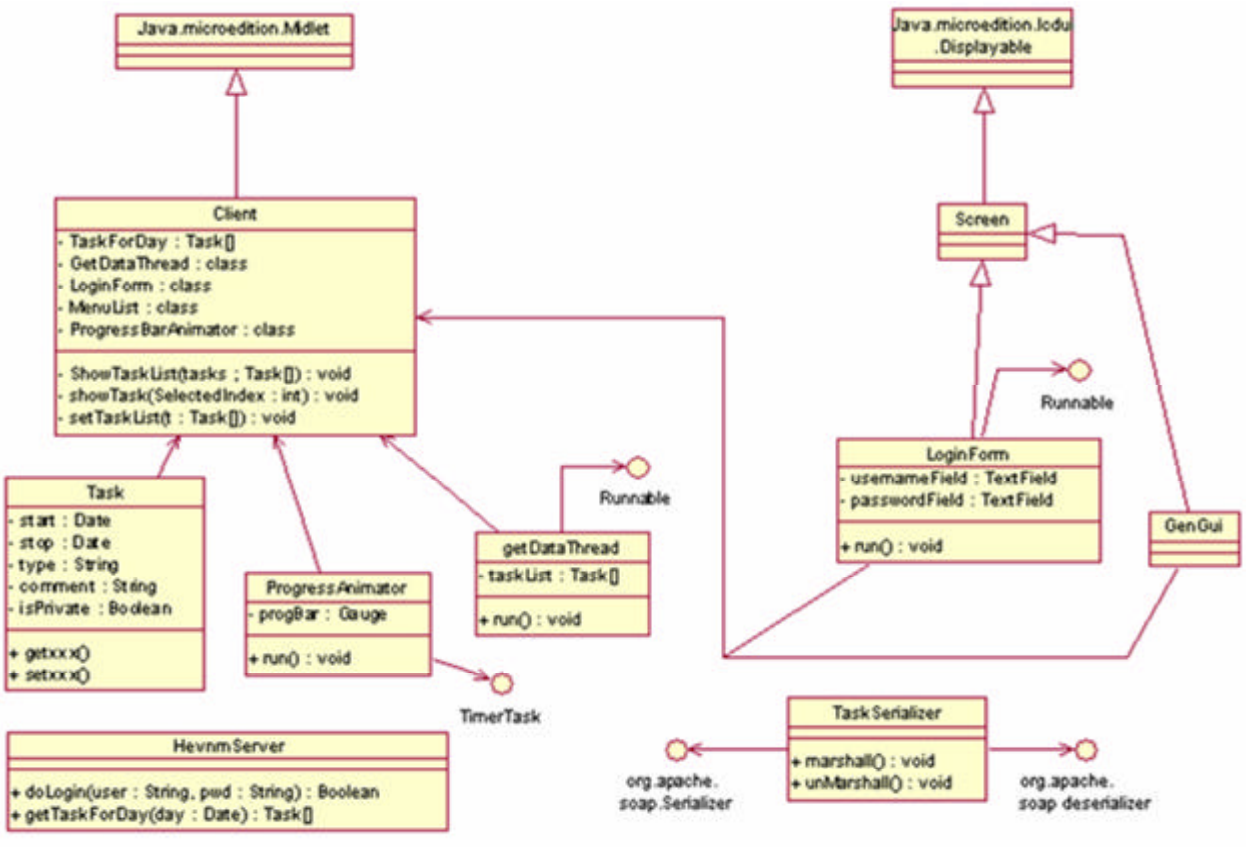
UseCase ”Velg meny” UseCase ”Velg dato”

UseCase ”Min timeplan” Animert progress-bilde

UseCase ”Vis hendelse”

C. KLASSEDIAGRAM

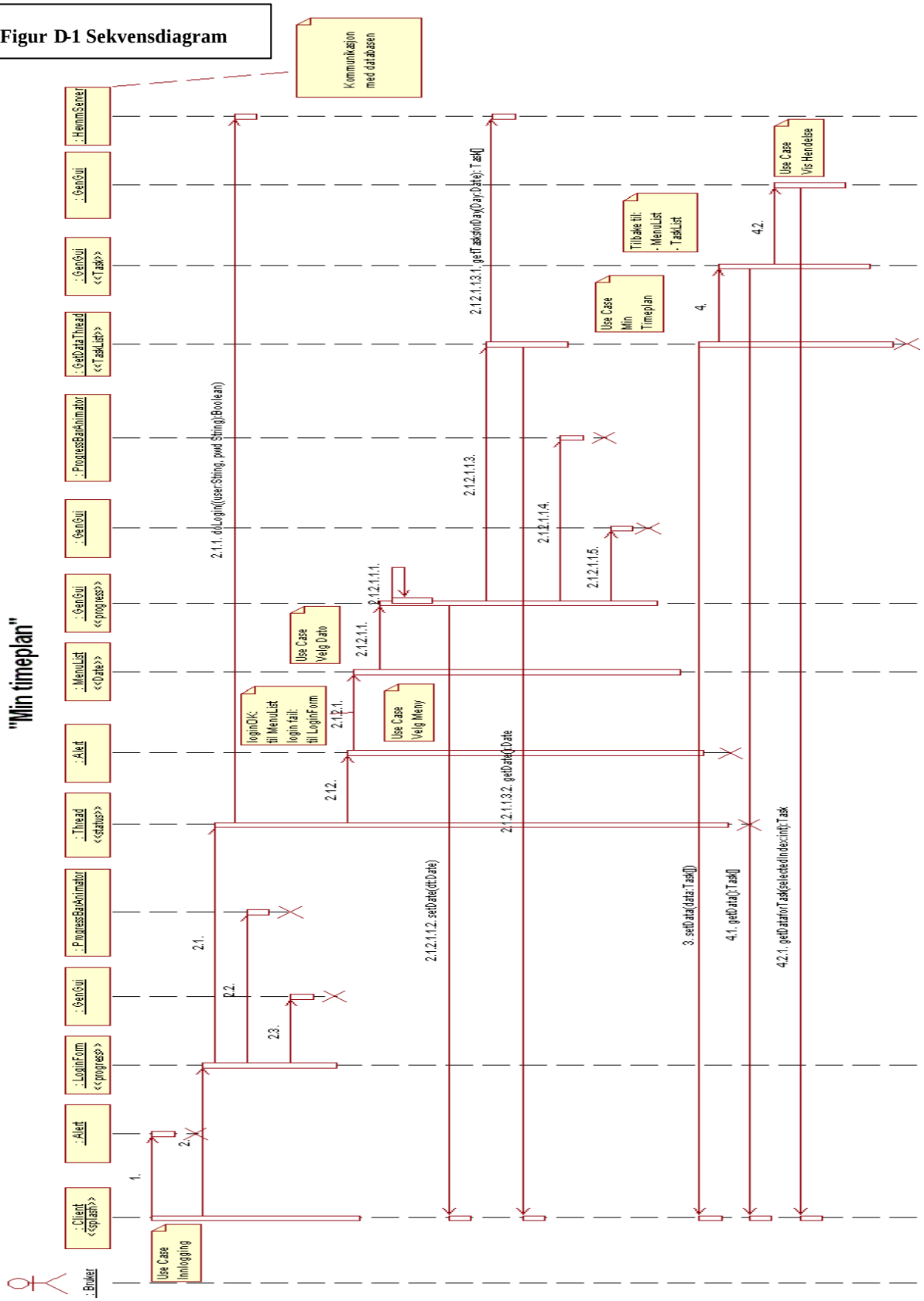
Klassediagrammet viser klassene som inngår i systemet, og forholdet mellom disse. For å forenkle diagrammet, er det benyttet ”lollipop” – notasjon for å vise implementasjon av et interface. Notasjonen ”GenGui” illustrerer de forskjellige klassene som inngår i J2ME sitt bibliotek av komponenter for brukergrensesnittet.



D. SEKVENSDIAGRAM

Figur D-1 viser et sekvensdiagram for de viktigste operasjonene i denne iterasjonen. Diagrammet viser livssyklus for objekter, og samhandlingen mellom disse. Det henvises i illustrasjonen til det aktuelle UseCase.

Figur D-1 Sekvensdiagram



E. TESTPLAN OG TESTRESULTAT

Min timeplan

Funksjon	Valg	Resultat
Innlogging	Avslutt	Avslutter HEVN-M
	Uten brukernavn og passord	Progressbar vises: Feilmelding
	Feil brukernavn(casesensitivt)	Progressbar vises: Feilmelding
	Feil passord	Progressbar vises: Feilmelding
	Riktig brukernavn og passord	Progressbar vises: oggingsmelding Meny
Meny	Avslutt	Avslutter HEVN-M
	Min timeplan	Velg Dato(med dagens dato)
Velg Dato	Avslutt	Avslutter Hevn--m
	Velger datofelt	Viser kalender
	Velger angitt dato	Progressbar vises: Ingen hendelser: Melding Reg. hendelser: Vises
Min timeplan	Tilbake	Velg Dato
	Avslutt	Avslutter Hevn--m
	Velger en hendelse	Vis hendelse
Vis hendelse	Tilbake	Min timeplan
	Avslutt	Avslutter HEVN-M
Progressbar ved innlogging	Avbryt	Innlogging
Progress før Vise hendelse	Kan ikke avbrytes	Kan ikke avbrytes
Server ikke tilgjengelig		Feilmelding: Server ikke tilgjengelig
Server stoppet mens klient var tilkoblet		Feil oppsto

F. ULØSTE OPPGAVER

Kjente feil og mangler ved utgitt versjon av 1. iterasjon

Brukernavnet lagres ikke på klienten for bruk ved fremtidig innlogging.

Det lot seg ikke gjøre å bla (skrolle) listen med hendelser slik at den mest aktuelle hendelsen ble automatisk valgt. Det hjelper ikke å sette det aktuelle feltet som ”selected”.

Logging fra serverklassen er ikke tilfredsstillende. Det gjenstår å konfigurere Tomcat riktig, og velge riktig loggenivå ut fra verdi i propertiesfil.

Det er ikke tatt hensyn til ønsket om enkel mulighet for oversettelse av tekst på klienten. J2ME har ikke støtte for slik internasjonalisering på samme måte som J2SE. Dette bør avklares med oppdragsgiver.

Det er fortsatt noe ”debug kode”, og unødig utskrift som ikke er fjernet fra kildekoden.

Endring av kSOAP klassen HttpTranspport ble gjort litt ”ufint”, ved å endre original kildekode og kompilere og legge klassen inn i biblioteket igjen – med samme navn. Dette bør gjøres ved å utvide klassen....

Det er manglende feilkontroll og ”exception” håndtering i klientapplikasjonen. Det burde gjøres en gjennomgang av alle feilmuligheter, og håndtere disse.

Aktiv avlogging fra klienten er ikke implementert.

For status på kjente feil og mangler ved avslutning av prosjektet, henvises til overordnet del av rapporten.

2. ITERASJON NR.2

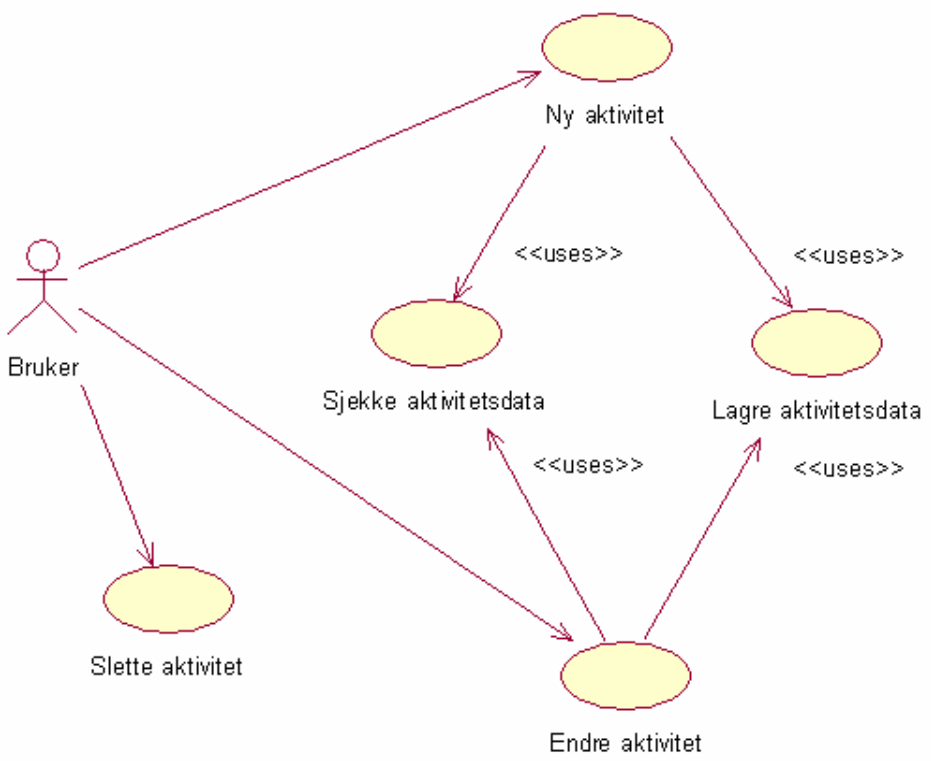
Denne iterasjonens innhold er utvikling av mulighet for å lagre nye aktiviteter, endre og slette eksisterende aktiviteter i egen timeplan, og å lese andre brukere sin timeplan.

2.1. KRAVSPESIFIKASJON

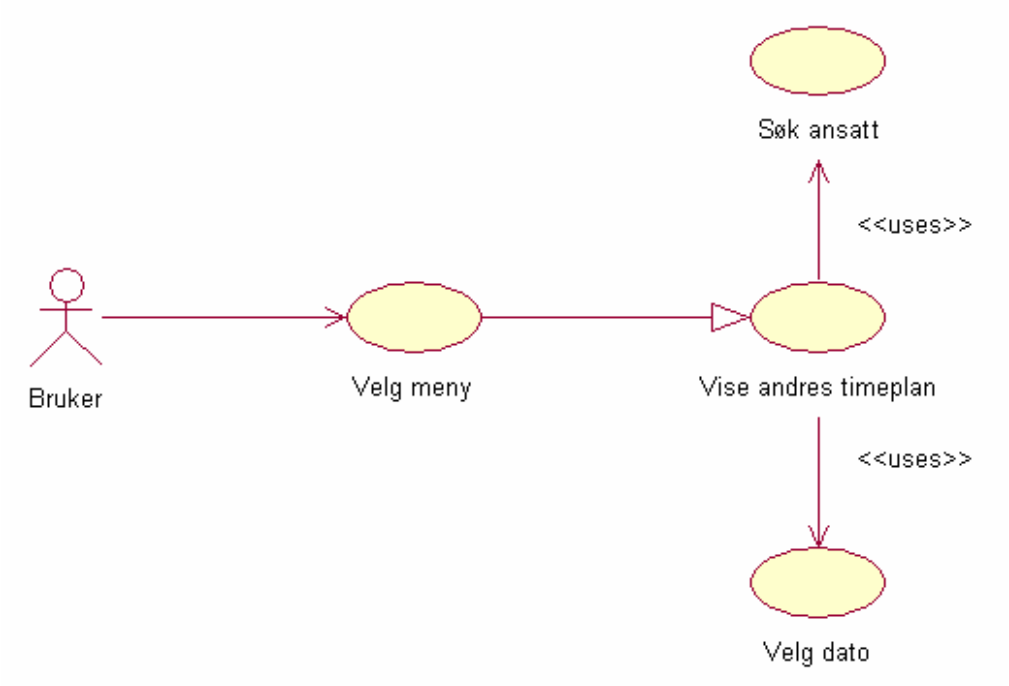
2.1.1. FUNKSJONELL SPESIFIKASJON

Dette er en detaljert kravspesifisering for de funksjonaliteter som tilhører denne iterasjonen. Dette er uttrykt ved UseCase-beskrivelsene ny aktivitet, endre aktivitet, sjekke aktivitetsdata, lagre aktivitetsdata, slette aktivitet, søk ansatt, velg meny og velg dato.

USECASE DIAGRAM



Figur 6-2-1 Viser sammenhengen mellom UseCasene



Figur 6-2-2 Viser sammenhengen mellom UseCasene

USECASE BESKRIVELSER

VELG MENY

Mål: Bruker velger en av HEVN-M's tjenester

Aktør: Bruker

Beskrivelse: Hovedmeny som viser tilgjengelige tjenester i HEVN-M:

- Min timeplan
- Andres timeplan

Prebetingelser:

- Bruker er pålogget

Postbetingelser:

- Viser "Velg dato", "Søk ansatt", "Avlogging"

Scenarier:

1. Bruker velger tjeneste og trykker OK
2. Klienten går til valgt tjeneste

Alternative scenarier:

2.1 Valgt: ”Min timeplan” ? ”Velg dato”.
2.2 Valgt: ”Andres timeplan” ? ”Søk ansatt”

MIN TIMEPLAN

Mål: Få en oversikt over dagens hendelser

Aktør: Bruker

Beskrivelse:

Viser brukers hendelser en gitt dag
Viser hendelsestype og dens start- og stopptid

Prebetingelser:

- Valgt en tjeneste
- Valgt dato

Postbetingelser:

- Trykker ”OK” ? ”Vis hendelse”
- Trykker ”Avslutt” ? ”Avlogging”
- Trykker ”Tilbake” ? ”Velg dato”.
- Trykker ”Ny aktivitet” ? ”Ny aktivitet”.
- Trykker ”Endre” ? ”Endre aktivitet”.
- Trykker ”Slette” ? ”Bekreft sletting.

Scenarier:

1. Presenterer liste over valgt dato’s ”hendelser”
2. Velger ”Hendelse” og trykker ”OK”
3. Går til valgt ”Hendelse”

Alternative scenarier:

1.1 Liste er tom – gi bruker melding om dette.
2.1 Velger ”Avslutt” ? ”Avlogging”.
2.2 Velger ”Tilbake” og ? ”Velg dato”.
2.3 Velter ”Ny aktivitet” og ? ”Ny aktivitet.
2.4 Velger ”Endre” og ? ”Endre aktivitet”.
2.5 Velger ”Slette” og ? ”Bekreft sletting”.

NY AKTIVITET

Mål: Registrere ny aktivitet i databasen.

Aktør: Bruker

Beskrivelse: Bruker registrerer ny aktivitet i ”Min timeplan” for en gitt dag.

Prebetingelser:

- Bruker har valgt ”Ny aktivitet” i UseCaset ”Min timeplan” eller ” Vis hendelse”. Dvs. at timeplan for aktuell dag er lagret i minne.

Postbetingelser:

- Databasen er oppdatert. Vise bekreftelse.
- Trykker ”Tilbake” ? ”Min timeplan” eller ”Vis hendelse”.
- Trykker ”Avslutt” ? ”Avlogging”
- Trykker ”Hovedmeny” ? ”Velg meny”

Scenarier:

1. GUI viser skjermbilde med tekst/dato/tid felter for å legge inn data om den nye aktiviteten. Trykker ”OK”.
2. UseCase ”Sjekke aktivitetsdata”.
3. Be om bekreftelse
4. UseCase ”Lagre aktivitetsdata”

Alternative scenarier:

1.1 Se postbetingelser for navigasjon.
3.1 Konflikt med tidligere lagret data:
- Bruker velger: ”Overskrive” eller ”Avbryt”.

3.1.1 ”Overskrive” ? Gå til pkt. 4
3.1.2 ”Avbryt” ? Gå til pkt. 1

Spesielle krav: Brukeren skal kun ha adgang til data om egne avtaler

SJEKKE AKTIVITETSDATA

Mål: Bruker skal gjøres oppmerksom på konflikter med tidligere lagrede aktiviteter. Unngå å sende ”ufullstendige” data til server.

Aktør: ”Ny aktivitet”, ”Endre aktivietet”.

Beskrivelse:

- Sjekke at GUI-felter ikke inneholder ”tom streng”. (Kun tillatt for kommentar – detaljert beskrivelse).

- Sjekke om det oppstår konflikter med eksisterende aktivit eter.

Prebetingelser:
- Bruker har valgt ”OK” i ”Min timeplan” eller ”Vis hendelse”.

Postbetingelser:
- Bruker blir gjort oppmerksom på eventuelle feil i data. Intern melding om konflikt (håndteres i ”Ny aktivitet”/”Endre aktivitet”).

Scenarier:
1. Sjekke om det er innhold i alle feltene unntatt ”kommentar” – detaljbeskrivelse.
2. Sjekke om det er konflikt mellom nye aktivitetsdata og tidligere lagrede aktiviteter.
3. Ingen konflikt: ? pkt. 3 i ”Ny aktivitet”/”Endre aktivitet.

Alternative scenarier:
2.1 Funnet tomme felter: feilmelding ? ”OK” ? ”Ny aktivitet” eller ”Endre aktivitet”.
3.1 Konflikt: ? pkt 3.1 i ”Ny aktivitet”/”Endre aktivitet”.

LAGRE AKTIVITETSDATA

Mål: Oppdatere databasen med nye data for en aktivitet. Oppdaterer klienten med korrekte data.

Aktør: ”Ny aktivitet”, ”Endre aktivitet”.

Beskrivelse:
- Hente data fra GUI, sende til server som oppdaterer databasen.
- Returnere OK: true/false for status på oppdatering.
- Bruker får bekreftelse/feilmelding.
- Klienten henter oppdatert versjon av dagens timeplan fra server.

Prebetingelser:
- Data som ønskes lagret er tilgjengelige i klientminne.
- Data er sjekket for konflikter: UseCase ”Sjekke aktivitetsdata”.

Postbetingelser:
- Database og klient er oppdatert med samme data.
- Bruker har fått bekreftelse/feilmelding.

Scenarier:
1. Klient sender data til server.
2. Server lager sql-uttrykk, og lagrer data.
3. Server sjekker om oppdatering var OK.
4. Returnere ”OK-melding” til klient: ”true”.

5. Klient ber om nye data fra server, og oppdaterer minne.
6. GUI viser bekreftelse på lagring sammen med ”progress skjerm”.
7. GUI viser (oppdatert) timeplan.

Alternative scenarier:

4.1 Oppdatering ikke gjennomført:
- returnere melding ”OK = false”.

Spesielle krav:
4.1.2 Klient beholder gamle data.
4.1.3 GUI viser feilmelding.
- Bruker trykker ”OK” ? pkt. 7

ENDRE AKTIVITET

Mål: Bruker skal kunne endre en eksisterende aktivitet.

Aktør: Bruker

Beskrivelse: Bruker skal kunne endre dato, tidspunkt, aktivitetstype og kommentar. Endringen vil gjelde kun en aktivitet.

Prebetingelser:

- Bruker har valgt ”Endre” i UseCaset ”Min timeplan”. Dvs. at timeplan for aktuell dag er lagret i minne.

Postbetingelser:
- Databasen skal være oppdatert
- Bruker skal ha bekreftelse på endring, evt. feilmelding hvis endring ikke fullføres.
- Timeplandata i klienten oppdatert med nye data fra databasen.
- Velger:
  - ”Hovedmeny”
  - ”Tilbake”
  - ”Avslutt”

Scenarier:
1. GUI viser skjermbilde med editerbar tekst- dato/tid - felter som inneholder dataene til den aktuelle aktiviteten. Bruker foretar ønsket endring, trykker ”Lagre".
2. UseCase ”Sjekke aktivitetsdata” (? ingen konflikt).
3. Be om bekreftelse.
4. UseCase ”Lagre aktivitetsdata”.

Alternative scenarier:
2.1 Velger ”Avbryt” ? ”Min timeplan”, ingen endring utføres.

3.1 Konflikt med tidligere lagret aktivitet. Bruker velger ”Overskrive” eller ”Avbryt”.
3.1.1 Velger ”Overskrive” ? pkt. 4.
3.1.2 Velger ”Avbryt” ? pkt. 1.

Spesielle krav: Brukeren skal kun ha adgang til data om egne avtaler

SLETTE AKTIVITET

Mål: Slette en tidligere lagret aktivitet

Aktør: Bruker

Beskrivelse: Brukeren velger å slette en registrert aktivitet. Data om aktiviteten fjernes fra databasen og klienten oppdaterer lokale data.

Prebetingelser:
- Bruker har valgt ”Slette” i ”Min timeplan” eller i ”Vis hendelse”.

Postbetingelser:
- Aktiviteten slettet.
- Database og klient er korrekt oppdatert.
- Bruker har fått bekreftelse/feilmelding.
  - ”Avbryt” ? ”Min timeplan”
  - ”OK” i bekreftelse/feilmelding ? ”Min timeplan”.

Scenarier:
1. GUI ber om bekreftelse før sletting ? ”OK”.
2. Klienten ber server om å slette aktuell aktivitet.
3. Server lager sql – uttrykk og sletter aktiviteten i databasen.
4. Se pkt. 4 - 4.1 osv. i UseCase ”Lagre aktivitetsdata”.

Alternative scenarier:
1.1 Liste er tom – gi bruker melding om dette
2.1 Bekreftelse ikke gitt, tilbake til ”Min timeplan”.
2.2 Velger ”Tilbake” og ? ”Velg dato”.

Spesielle krav: Brukeren skal kun ha adgang til data om egne avtaler

ANDRES TIMEPLAN

Mål: Få oversikt over en annens timeplan for en gitt dag.

Aktør: Bruker

Beskrivelse:

- Viser gitt brukers timeplan for en gitt dag.
- Viser aktivitetstype og dens start- og stopptid.

Prebetingelser:

- ”Vise andres timeplan” er valgt i hovedmenyen
- UseCase’ene ”Søk ansatt” og ”Velg dato” er gjennomført

Postbetingelser:

- Trykke ”Avslutt” ? ”Avlogging”
- Trykke ”Tilbake” ? ”Søk ansatt”

Scenarier:

1. Presenterer liste over valgt datos ”aktiviteter”
2. Velger ”Tilbake”

Alternative scenarier:

1.1 Liste er tom – melding om dette til bruker
1.2 Ingen tilgang til timeplan – melding om dette til bruker
2.1 Velger ”Avslutt”

SØK ANSATT

Mål: Finne en ansatt i databasen

Aktør: Bruker

Beskrivelse: Bruker oppgir ett søkeuttrykk for etternavn og/eller fornavn. Får deretter opp en liste med aktuelle treff. Blant disse velges et navn.

Prebetingelser:

- ”Vise andres timeplan” er valgt i hovedmenyen.

Postbetingelser:

- Unikt og korrekt brukernavn er valgt og lagret i midlet.

Scenarier:

1. GUI viser tekstfelter for inntasting av søkeuttrykk for etternavn og fornavn. Trykker ”OK”.
2. Sjekke søkeuttrykkene før sending. Ikke tillatt med tomme strenger i både etternavn og fornavnsfeltene.
3. Søkeuttrykkene sendes til server som lager sql-uttrykk og henter aktuelle brukernavn
4. Server returnerer String array med brukernavnene.
5. GUI viser liste over brukernavnene. Bruker velger en av disse og trykker ”OK” eller

”SELECT”.
6. Valgt navn lagres i midlet. Lista slettes.
7. GUI viser ”Velg dato” – se eget UseCase.

Alternative scenarier:

1.2 Trykker ”Avslutt” ? ”Avlogging”
1.3 Trykker ”Tilbake” ? ”Velg Meny”
3.1 Søkeuttrykk ikke godkjent. Gi melding ? pkt.1
4.1 Server finner ingen aktuelle brukernavn. Returnerer tom array.
4.1.1 GUI viser ”Ingen funnet”. ”OK” ? pkt.1.
5.1 Bruker velger ”Nytt søk” ? pkt.1

2.1.2. DRØFTING AV ALTERNATIVER

Synkronisering av data ved endringer i databasen(slette, endre, ny):

Ulike alternativer for synkronisering er vurdert. Det tar lang tid å hente nye data fra databasen, men det er også viktig med konsistente data. Hvis klienten skal sørge for å foreta de samme endringer i interne data, som tilsvarende data i databasen, kan feil oppstå. Dette fører til færre serverkall, men også til mer prosessering på klient.

Vi har valgt å hente nytt datasett fra databasen når en vellykket oppdatering er utført. Hvis oppdateringen ikke gjennomføres beholder klienten sine data uendret.

Begrunnelse og konsekvenser:

- Sikrer konsistente data
- Eventuelle endringer i rekkefølgen av aktiviteter gjøres på server v.h.a. sortert utvalg fra databasen.
- Økt ventetid og nettverkstrafikk
- Sårbart for nettverksfeil.

Brukers mulighet til å avbryte en endring(slette, endre, ny):

Problemer kan oppstå dersom bruker gis mulighet til å avbryte prosessen med å gjennomføre en endring i databasen etter at kallet til server er startet. Vi har ikke kontroll på hvor langt denne prosessen er kommet, så vi må anta at endringen er utført når kallet blir gjort.

Vi har valgt å ***ikke*** gi bruker mulighet til å avbryte en endring. Brukeren får valg om å avbryte før prosessen startes. Ved feil (nettverk – i/o) må klienten håndtere dette.

2.1.3. TESTING AV IMPLEMENTASJON

Funksjonaliteten testes som en integrert del av arbeidet med kodingen. Dette skal gruppemedlemmene gjøre.

Ved avslutning av iterasjonen vil gruppemedlemmene i samråd med veileder lage en testplan som skal teste nye funksjonaliteter og endringer av eksisterende funksjonaliteter forårsaket av iterasjon 2.

AKSEPTANSEKRAV

Disse punktene skal testes og godkjennes:

- Andres timeplan
  - Søke etter en ansatt
  - Velge ønsket ansatt og dato
  - Hente data/visning

- Endre aktivitet
  - Dagens timeplan lagres i minnet
  - Endre egen timeplan
  - Oppdatert timeplan vises

- Ny aktivitet
  - Lagre en ny aktivitet på egen timeplan

- Slette aktivitet
  - Slette alle opplysninger om en gitt aktivitet

2.2. ANALYSE OG DESIGN

2.2.1. HVA SKAL SYSTEMET GJØRE

I iterasjon 2 har vi valgt å lage to funksjonaliteter som skal fungere fullstendig. Det er å endre egen timeplan og se andres timeplan.

”Min timeplan”, blir utvidet fra den første iterasjonen. I den utvidete versjonen skal klienten i tillegg til å kunne se egen timeplan, endre og lagre ny aktivitet. Brukeren skal kunne bestemme om aktiviteten han/hun registrerer, skal være privat eller offentlig. Det vil si om aktiviteten skal vises for andre brukere eller ei. Det skal lages ei liste over forhåndsdefinerte gjøremål, som f. eks. møte, undervisning, annet etc. Det skal vurderes hvilke aktiviteter som skal forhåndsdefineres.

I ”Andres timeplan” skal klienten kunne søke på kun/minst en bokstav i fornavn og/eller etternavn til en ansatt ved HiG og få vist timeplanen for den valgte personen en gitt dag. Det er kun offentlige aktiviteter som skal vises.

Bilder 6-2-1 Papirlappmetoden– Første utkast av design, på papir.

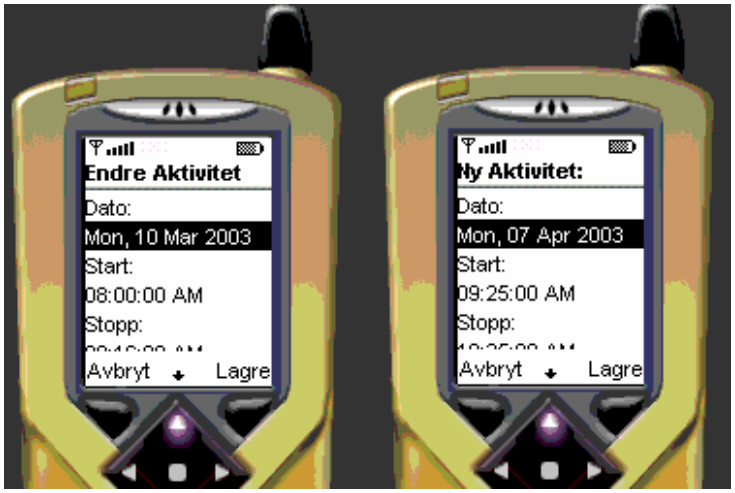
2.2.2. HVORDAN LØSTE VI DET

DRØFTING AV ALTERNATIVER:

Tester viser at det er begrensede muligheter for oppsett av skjermbildet for endring og ny aktivitet. Vi hadde ønsket å gi brukeren en kort liste med forhåndsdefinerte valg, inkludert ”annet”, for å bestemme aktivitetstype. For å oppnå dette må vi lede brukeren gjennom mange menyvalg. Dette ble vurdert nærmere etter tester av alternativer under

implementasjonen. Vi valgte å forenkle dette ved å bruke et enkelt tekstfelt hvor brukeren skriver inn ønsket aktivitetstype.

MIN TIMEPLAN



Bilde 6-2-2 Endre/Ny aktivitet

Fra "Meny" kan brukeren velge å se sin egen timeplan, "Min timeplan". Denne funksjonaliteten har blitt videreutviklet i denne iterasjonen. I iterasjon 1 kunne brukeren kun se sin egen timeplan. Nå har brukeren flere valgmuligheter. Brukeren kan lagre en ny aktivitet og i tillegg endre og slette lagrede aktiviteter.

ANDRES TIMEPLAN

Funksjonaliteten for å få vist andre ansattes timeplan en gitt dag.

Bilde 6-2-3 Andres timeplan

Fra ”Meny” kan brukeren velge ”Andres timeplan” og får da opp et skjermbilde; ”Søk ansatt”. Her vises to felter som skal fylles ut med fornavn og/eller etternavn. På grunn av brukervennligheten har vi valgt å lage systemet slik at bruker kan taste inn for eksempel kun en bokstav i en av eller begge feltene. Resultatet skal da inneholde liste over de navn som inneholder gitte bokstaver i henholdsvis en av eller begge feltene. Deretter kan brukeren velge en person og få vist timeplanen for den valgte personen. Andre brukere har ikke tilgang til å se de detaljerte hendelsene i timeplanen til en annen bruker.

2.3. IMPLEMENTERING

2.3.1 GENERELT

Kodingen av funksjonaliteten i 2. iterasjon har i stor grad bestått av tillegg, endringer og utvidelser av kildekode fra 1. iterasjon. Endringer i eksisterende kode er gjort der det i ettertid har vist seg å finnes enklere måter å samle og utføre lignende oppgaver på. Å utvikle kode for å kunne lagre til databasen har vært det mest arbeidskrevende innen denne iterasjonen. Vi har implementert databaseoppdateringer på en slik måte at dette skal være konsistent med funksjonaliteten i HEVN-J (dvs at nye hendelser kan overskrive tidligere registrerte hendelser, men slik at overlapping ikke finner sted).

Hva som er blitt gjort med ”Uløste oppgaver ” fra iterasjon 1 (Vedlegg F):

kSOAP-klassen HttpTransport er blitt omskrevet og gitt nytt navn; HttSessionTransport.java.

Aktiv avlogging er fortsatt ikke implementert, men sesjonshåndteringen i Tomcat fjerner ubrukte sesjoner etter en gitt tid(settes opp i konfigurasjonen av Tomcat). Dette fungerer da som avlogging.

De øvrige punktene på listen av uløste oppgaver vil bli videreført til i neste iterasjon.

2.3.2 KLIENT

Klientapplikasjonen er utvidet med en ny klasse med brukergrensesnitt for å representere en hendelse med editerbare felter. Denne klassen sørger også for lagring i en egen tråd. Likeledes er en klasse for søk blant ansatte (brukere av hevn-j) utviklet. Se kodeeksempel 1:

kodeeksempel 1

```
/**Default constructor: initialize form with fields to enter a new task
*  @param title The title of the form
*  @param mid The midlet who shows this form
*/
public TaskForm(String title, Client mid){
   super(title);
```

```
      this.owner = mid;
      long offset = 2 * 3600000; // shortcut to set gmt + 2 ...
      long localTime = new Date().getTime() + offset;
      Date toDay = new Date(localTime);
      TimeZone tz = TimeZone.getTimeZone("GMT");
      dateField = new DateField("Dato: ", DateField.DATE, tz);
      dateField.setDate(toDay);
      startField = new DateField("Start: ", DateField.TIME, tz);
      startField.setDate(clearDate(toDay));
      stopField = new DateField("Stopp: ", DateField.TIME, tz);
      stopField.setDate(clearDate(new Date(toDay.getTime() + 3600000)));
      typeField = new TextField("Type: ","Møte", 30, TextField.ANY);
      commentField = new TextField("Kommentar: ","", 100, TextField.ANY);
      isPrivateChoice = new ChoiceGroup(null, ChoiceGroup.MULTIPLE);
      isPrivateChoice.append("Privat", null);
      .......
   }
```

Vi har fordelt oppgaven med validering av inputdata mellom klient og server for å unngå unødig ventetid på "umulige" lagringsønsker. Klientapplikasjonen foretar sjekking før lagring av en endret/ny hendelse for å avdekke konflikter men tidligere lagrede hendelser. Brukeren gis mulighet for å overskrive.

Resursbruken på klienten begynner å bli problematisk på grunn av veldig lang ventetid for vanlige operasjoner. Dette må det tas hensyn til i den videre utviklingen.

2.3.3 SERVERKLASSER

På serversiden av systemet er den klassen som implementerer Web Service løsningen utvidet med metoder for endringer av databasen. Dette ble gjort ved å kode metoder for tilgang via SOAP for endring, nyregistrering og sletting. Det ble også laget tilsvarende "interne" metoder. Ved ny/endre må dette noen ganger utføres som en serie databasekall. Disse er utført i en transaksjon slik at dersom feil oppstår blir det utført "rollback" til databasens tilstand før endringen startet. Metodene som er tilgjengelige for fjernkall via SOAP benytter de "interne" metodene for å utføre endringene på databasen. En metode ble laget for om nødvendig å lage plass til ny/endret hendelse. Se kodeeksempel 2:

kodeeksempel 2

```
   /**Check for conflicts with existing taskdata. Make room for the new
*  task by delete or moving start and/or stop times of existing data.
   *  @param t A Task object, new or updated
   *  @param tasks A array of the task of the actual day.
   *  @param con A open Connection object.
   */
   private void makeRoom(Task t, Task[] tasks, Connection con)
      throws Exception {
      // if possible conflicts
      if(tasks != null && tasks.length > 0){
         // check all tasks
```

```
         for(int i=0;i<tasks.length;i++){
            // old task the same or inside new task on both sides
if(tasks[i].getStart().getTime() >= t.getStart().getTime()&&
tasks[i].getStop().getTime() <= t.getStop().getTime() ){
               dbDelete(tasks[i], con);
               continue;
            }
            // old stop inside new task
            if(tasks[i].getStop().getTime() > t.getStart().getTime() &&
               tasks[i].getStop().getTime() <= t.getStop().getTime() ){
               tasks[i].setStop(t.getStart());
               dbUpdate(tasks[i], con);
            }
```

Server er fortsatt implementert som en klasse med metoder som offentliggjøres for SOAP kall.

Det later ikke til å være problemer med hastighet eller minnebruk på serversiden.

2.4. TESTING

2.4.1 GENERELT

For generell beskrivelse av testing, *se pkt 2.1.3 i kravspesifikasjon* for denne iterasjonen.. Følgende endringer er gjort i det praktiske testarbeidet:

Testing er kun utført av gruppemedlemmene, dette i samråd med oppdragsgiver.
Det er ikke benyttet særskilte testklasser under avsluttende testing.
Avsluttende tester er utført etter testspesifikasjonene nedenfor.
Testresultatene finnes i vedlegg E for denne iterasjonen.

2.4.2 TESTSPESIFIKASJON

Testing av systemet foregår i testomgivelser på våre lokale arbeidsstasjoner med telefonemulator.

HVA SKAL TESTES

- Valg i hovedmeny:
  - ”Min timeplan”
  - ”Andres timeplan”

- ”Min timeplan”:

  - Valg av ny aktivitet
  - Legge inn ny dato, starttid, stopptid, type, kommentar og privat.
  - Utelate alle/deler av ovennevnte elementer
  - Presentere oppdateringer: timeplan for en dag samt kommentarer.
  - Vise oppdateringer for andre brukere hvis privat er merket/ikke merket.

  - Valg av endre aktivitet

  - Dagens timeplan lagres i minnet
  - Endre registrert dato, starttid, stopptid, type, kommentar og privat.
  - Utelate alle/deler av ovennevnte elementer
  - Presentere oppdateringer: timeplan for en dag samt kommentarer.

- ”Andres timeplan”:

  - Søke bokstaver i kun etternavn, kun fornavn,
  - kombinasjon av etternavn/fornavn og begge felter utelatt.
  - Presentere oppgitt brukes timeplan for en dag.
  - Avslutning, navigasjon og angremuligheter.

Se testdokumentasjon i vedlegg E Testplan.

2.4.3 TESTRESULTAT

Kommentarer til Vedlegg E Testplan og testresultat. Testing har også i denne iterasjonen foregått vært en integrert del av utviklingsarbeidet. Ved slutten av iterasjonen ble det laget en testplan for sluttesting av applikasjonene. Da vi nå trenger å endre data i databasen, er mulighetene for feil, og konsekvensen av dem, større en i 1. iterasjon. Testingen avdekket at lagringsrutinene var mangelfulle. Det var særlig i forbindelse med konfliktsjekking (med tidligere aktiviteter) problemene oppsto. Vi ser at det blir nødvendig med sjekking også på serveren. Retting av disse feilene forskyves til senere iterasjoner, se Vedlegg F – Uløste oppgaver.

2.5. KONKLUSJON

2.5.1. DRØFTING AV FRAMDRIFTSPLAN

I denne iterasjonen har vi klart mye bedre å følge vår planlagte framdriftsplan enn forrige iterasjon. Grunnen til dette er i hovedsak valget av systemutviklingsmodell; det vi lærer i en iterasjon drar vi med oss i neste. Vi sparer utrolig mye tid på at i denne iterasjonen er verktøyene vi bruker i systemutviklingen kjent og utviklingsmiljøet er allerede på plass.

Det vi har brukt tid på er implementeringen av å oppdatere databasen og testingen av dette. Testingen av ”ferdigproduktet” kunne ikke starte før de forskjellige funksjonaliteter var på plass. Dette tok mer tid enn beregnet fordi vi ville dra med oss minst mulig mangler inn i neste iterasjon

Iterasjon 2 planla vi å bruke 288 timer. Det reelle er ca 259. Grunnen til at iterasjonen har færre timer over lengre periode er at Krasniqi og Harbosen har arbeidet en uke med andre fag.

Oversikt over tidsbruk

	Timer
Planlegging	6,5
Kravspesifisering	31,5
Design	29,5
Implementasjon	74,5
Sluttesting	28
Rapportskriving	35,5
Møter	13,5
Div	40,5
Sum	**259,5**

2.5.2. EVALUERING AV ITERASJONEN

PRODUKT

Systemet har etter andre iterasjon blitt noe mer komplekst. Vi ser også at noen avgjørelser vi gjorde i første iterasjon med fordel kunne vært annerledes. Samtidig mener vi at vi har vært nødt til å gjøre noen feil for å lære underveis.

Begrensningene i brukergrensesnitt og ressurser på klientapplikasjonen er en kilde til frustrasjon. Vi mener allikevel at vi har klart å utnytte mulighetene som ligger i Java2ME på en effektiv måte. Vi ser at vi må ta mer hensyn til hastighetsbegrensningen i systemet for å oppnå en brukervennlig applikasjon.

Vi har også ved denne avslutningen en del feil og mangler som må tas hensyn til i tredje iterasjon. Se vedlegg F.

PROSESSEN

Den inkrementelle utviklingsmodellen viser seg fortsatt som godt egnet for vårt prosjekt. Vi føler at det er motiverende med synlige resultater på et tidlig tidspunkt, og at vi har "noe å vise frem".

Vi har nok satt av litt for snaut med tid til testing av systemet og ferdiggjøring av prosjektrapporten. Det viser seg stadig at ting tar lengre tid enn antatt. Vi vil forsøke å ta med dette i planleggingen av tredje iterasjon.

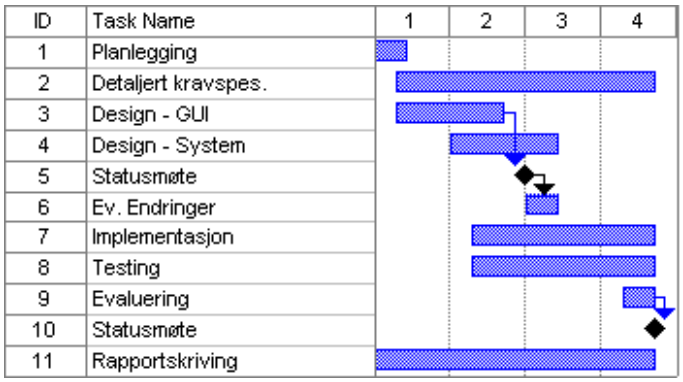
Det har vært lettere å følge fremdriftsplanen i denne runden. Selv om vi har bommet litt på budsjettert tidsforbruk, har vi bedre kontroll på fremdriften av prosjektet. Avviket skyldes at vi enten har beregnet for stor arbeidsmengde eller ikke har vært så effektive som vi ønsket.

2.5.3. HOVEDKONKLUSJON

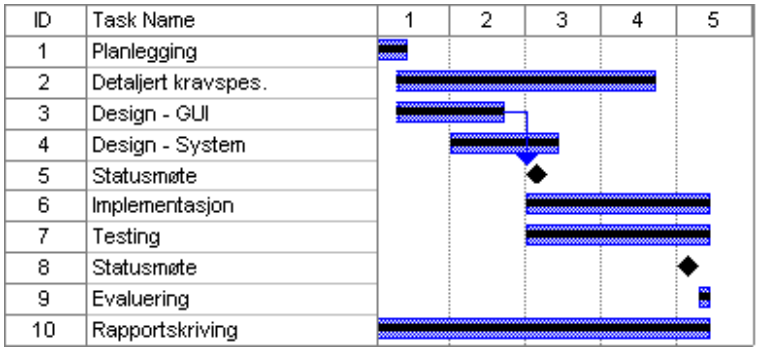
Det er fortsatt god stemning internt i gruppa (til veileders frustrasjon), og vi føler at vi har kontroll på prosjektet. Vi ser at planlegging og oppfølging av fremdriftsplan blir enda viktigere når avslutningen på prosjektet nærmer seg.

2.6. VEDLEGG

A. GANTT-SKJEMA



Figur 6-2-A-1 Planlagt framdriftsplan



Figur 6-2-A-2 Reell framdriftsplan

B. HEVN-M GUI

I dette vedlegget viser vi de fleste av skjermbildene som inngår i denne iterasjonen. Vi lar illustrasjonene tale for seg selv. Det er viktig å være klar over at det faktiske utseendet og navigasjonsmulighetene kan variere mye på forskjellige plattformer/telefoner.

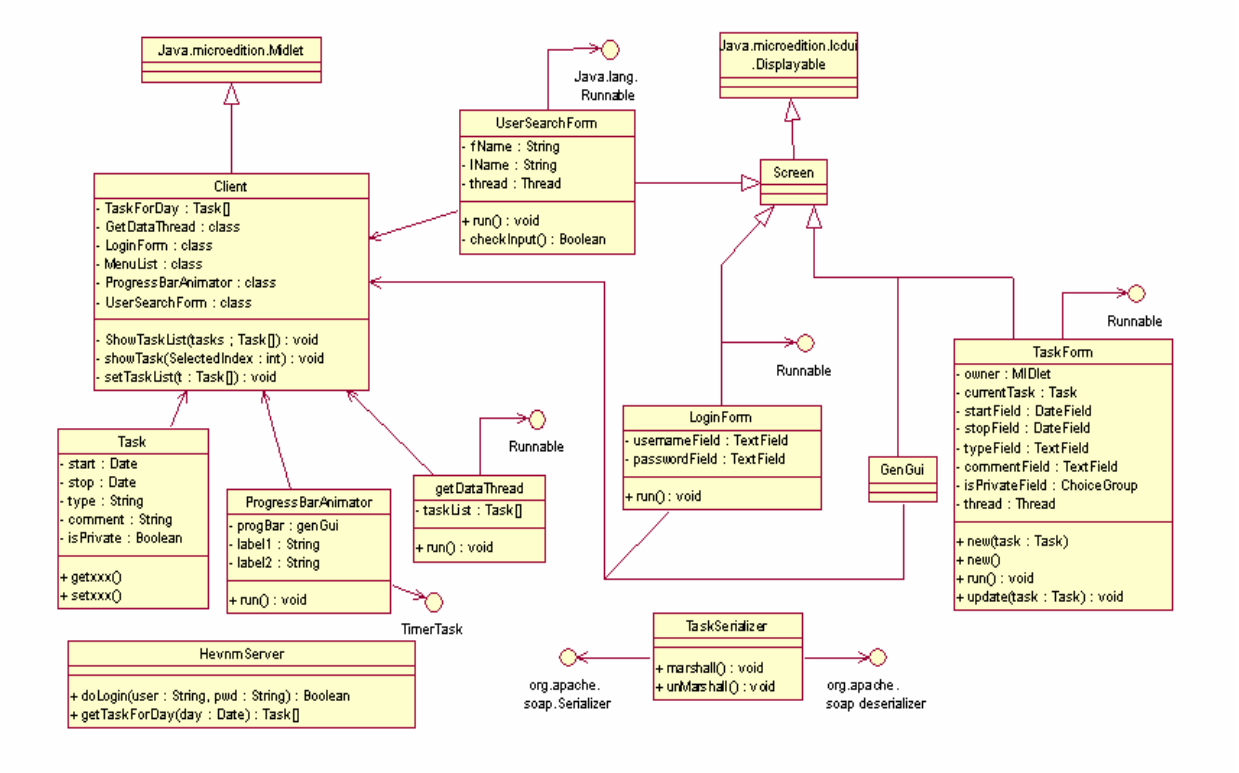

Fant disse navnene:
Rosa Elefant
Tjena Grabben
Kaj Jack
Kosovare Krasniqi
Nytt søk
Ok
Velg Dato:
Mon, 31 Mar 2003
Avslutt
OK
Henter data
Henter data


Melding:
Fant ingen registreringer for valgt dato
Done
Velg Dato:
Wed, 26 Feb 2003
Avslutt
OK
Henter data
Et øyeblikk

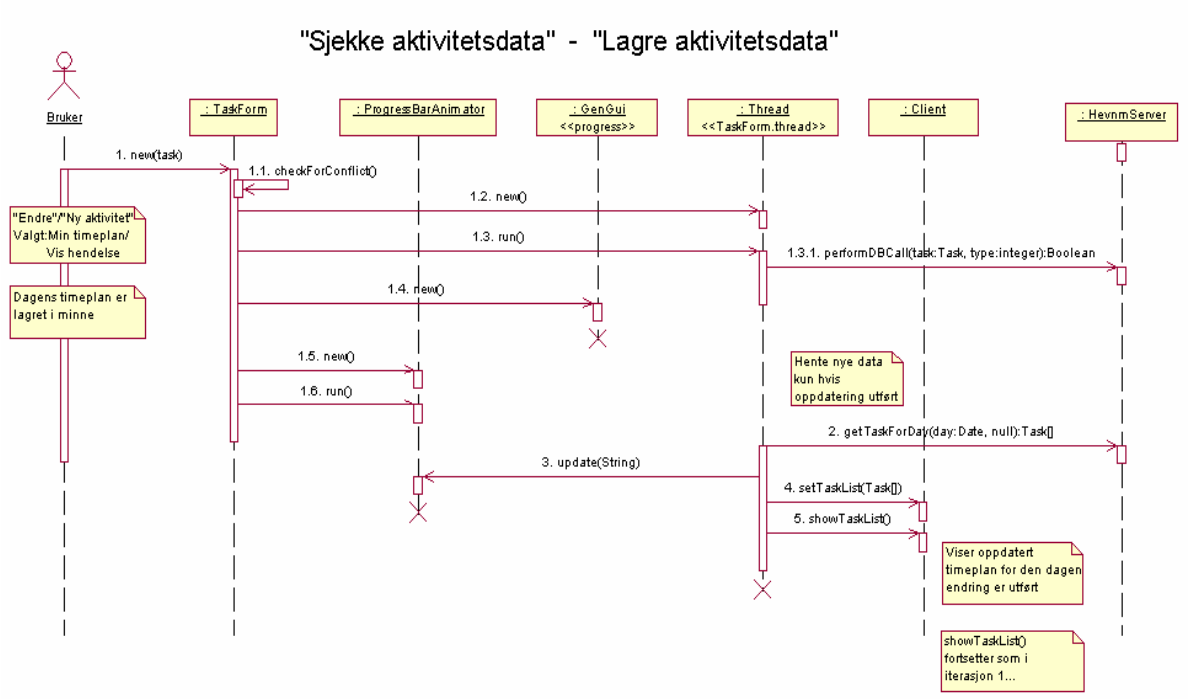

26.1.2003
11:30 - - 12:15
Undervisning
Tilbake
Menu
Menu
1 Avslutt
2 Detaljer
3 Endre
4 Ny
5 Slett
Tilbake
Menu
11:30 - - 12:15
Undervisning
K-102
Tilbake
Avslutt

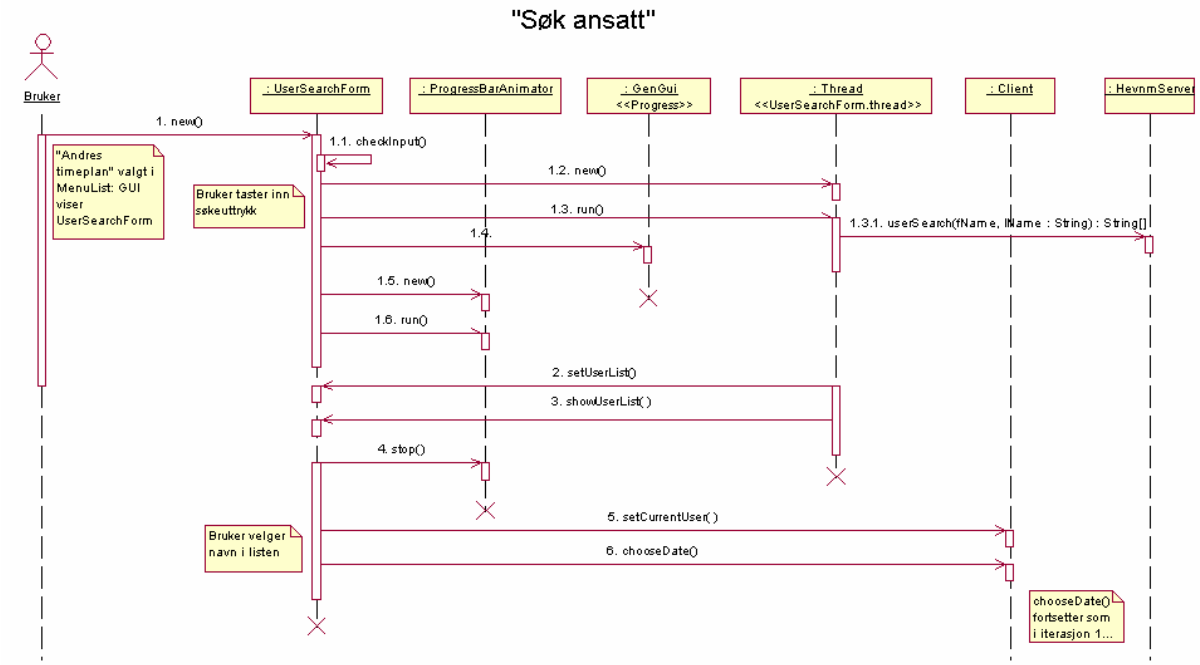

Endre Aktivitet
Dato:
Mon, 10 Mar 2003
Start:
08:00:00 AM
Stopp:
Avbryt
Lagre
Endre Aktivitet
Undervisning
Kommentar:
Frokost
Privat
Avbryt
Lagre
Ny Aktivitet:
Dato:
Mon, 07 Apr 2003
Start:
09:25:00 AM
Stopp:
Avbryt
Lagre

C. KLASSEDIAGRAM



D. SEKVENSDIAGRAM





E. TESTPLAN OG TESTRESULTAT

Min timeplan, Andres timeplan

Funksjon	Valg	Forventet resultat	Resultat
Meny	Avslutt	Avslutter HEVN-M	Avslutter HEVN-M
	Min timeplan	Velg dato (viser dagens dato)	Velg dato
	Andres timeplan	Søk ansatt	Søk ansatt
Velg Dato	Velger datofelt	Viser kalender	Viser kalender
	Velg angitt dato	Ingen aktiviteter: Melding om at ingen aktiviteter er registrert	Melding om at ingen aktiviteter er registrert
		Registrerte aktiviteter vises	Registrerte aktiviteter vises
	Tilbake	Meny	Meny
	Ny	Ny aktivitet	Ny aktivitet
Min timeplan	Avslutt	Avslutter HEVN-M	Avslutter HEVN-M
	Detaljer	Vise hendelse	Vise hendelse
	Tilbake	Velg Dato	Velg Dato
	Ny	Ny aktivitet	Ny aktivitet
	Endre	Endre aktivitet	Endre aktivitet
	Slett	Slett aktivitet	Slett aktivitet
Ny aktivitet	Angitt starttid som er lik en tidligere reg starttid og stopptid før tidligere reg stopptid	Registrerer ny aktivitet	Operasjon ble ikke gjennomført
	Angitt starttid som er lik en tidligere reg starttid og stopptid etter tidligere reg stopptid	Registrerer ny aktivitet	Registrerer ny aktivitet
	Angitt starttid før tidligere reg starttid og stopptid r lik tidligere reg stopptid.	Får melding om at det allerede er registrert aktivitet på gitt tidspunkt. Får spørsmål om det skal overskrives. Ja: Ny aktivitet overskriver Nei: Ny aktivitet	Vil ikke overskrive. Tidligere registrerte opplysninger blir stående
	Angitt starttid etter tidligere reg starttid og stopptid er lik tidligere reg stopptid	Får melding om at det allerede er registrert aktivitet på gitt tidspunkt. Får spørsmål om det skal overskrives. Ja: Ny aktivitet registreres og tidligere	Får melding om at det allerede er registrert aktivitet på gitt tidspunkt. Får spørsmål om det skal overskrives. Ja: Ny

		registrert aktivitet forskyver stopptidspunktet tilsvarende Nei: Ny aktivitet	aktivitet registreres og tidligere registrert aktivitet forskyver stopptidspunktet tilsvarende Nei: Ny aktivitet
	Angitt starttid etter stopptid	Operasjon blir ikke gjennomført	Får melding om at det allerede er registrert aktivitet på gitt tidspunkt. Får spørsmål om det skal overskrives. Ja: Operasjon ble ikke gjennomført. Nei: Ny aktivitet
	Angitt stopptid før starttid		Får melding om at det allerede er registrert aktivitet på gitt tidspunkt. Får spørsmål om det skal overskrives. Ja: Får melding om at operasjonen var vellykket, men aktivite vises ikke i timeplanen Nei: Ny aktivitet
	Angitt starttid og stopptid innenfor tidsrommet til annen aktivitet	Får melding om at det allerede er registrert aktivitet på gitt tidspunkt. Får spørsmål om det skal overskrives. Ja: overskrives Nei: Vis timeplan	Får melding om at det allerede er registrert aktivitet på gitt tidspunkt. Får spørsmål om det skal overskrives. Ja: overskrives Nei: Vis timeplan
	Angitt starttid og stopptid som ikke har aktivitet registrert	Timeplan vises med ny aktivitet registrert	Viser timeplan for den dagen som ble valgt til å begynne med i ”Min timeplan”. Ny aktivitet blir registrert på riktig dato med riktige opplysninger
	Utelatt dato	Default dato registreres	Default dato registreres
	Utelatt starttid	Default starttid registreres	Default dato registreres
	Utelatt stopptid	Default stopptid registreres	Default stopptid registreres. Godtar starttid etter stopptid

	Utelatt type	Default type (møte) registreres	Default type (møte) registreres
	Utelatt kommentar	Timeplan vises uten kommentar oppgitt	Timeplan vises uten kommentar oppgitt
	Privat ikke merket	Aktuell aktivitet vises for andre brukere	Aktuell aktivitet vises for andre brukere
	Privat er merket Avbryt	Aktuell aktivitet vises ikke for andre brukere Vise timeplant	Vises ikke for noen Vise timeplant
Endre aktivitet	Angitt opplysninger på dato som ikke har noe registrert	Timeplan vises med nye opplysninger	Ingenting skjedde. Ingenting ble lagret
	Angitt starttid som allerede har aktivitet registrert	Systemet ber om bekreftelse på at registrert aktivitet skal overskrives innenfor angitt tidsrom Hvis overskrive: Starttid legges inn og tidligere registrerte tider forblir. Hvis ikke overskrive: Registrering avbrytes og "Endre aktivitet" vises	Systemet ber om bekreftelse på at registrert aktivitet skal overskrives innenfor angitt tidsrom Hvis overskrive: Starttid legges inn og tidligere registrerte tider forblir. Hvis ikke overskrive: Registrering avbrytes og "Endre aktivitet" vises
	Angitt stopptid som allerede har aktivitet registrert	Systemet ber om bekreftelse på at registrert aktivitet skal overskrives innenfor angitt tidsrom Hvis overskrive: Starttid legges inn og tidligere registrerte tider forblir. Hvis ikke overskrive: Registrering avbrytes og "Endre aktivitet" vises	Lagring ble ikke gjennomført
	Angitt starttid og stopptid innenfor tidsrommet til annen aktivitet	Systemet ber om bekreftelse på at registrert aktivitet skal overskrives innenfor angitt tidsrom Hvis overskrive: Starttid legges inn og tidligere registrerte tider forblir. Hvis ikke overskrive: Registrering avbrytes og "Endre aktivitet" vises	Systemet ber om bekreftelse på at registrert aktivitet skal overskrives innenfor angitt tidsrom Hvis overskrive: Starttid legges inn og tidligere registrerte tider forblir. Hvis ikke overskrive:

			Registrering avbrytes og ”Endre aktivitet” vises
	Angitt starttid, stopptid og type innenfor tidsrommet til annen aktivitet	Systemet ber om bekreftelse på at registrert aktivitet skal overskrives innenfor angitt tidsrom Hvis overskrive: Starttid. Stopptid og type legges inn og tidligere registrerte tider og type forblir. Hvis ikke overskrive: Registrering avbrytes og ”Endre aktivitet” vises	Systemet ber om bekreftelse på at registrert aktivitet skal overskrives innenfor angitt tidsrom Hvis overskrive: Starttid. Stopptid og type legges inn og tidligere registrerte tider og type forblir. Hvis ikke overskrive: Registrering avbrytes og ”Endre aktivitet” vises
	Angitt identiske opplysninger	Systemet ber om bekreftelse på at registrert aktivitet skal overskrives innenfor angitt tidsrom Hvis overskriv: Opplysningene legges inn på ny. Hvis ikke overskrive: Registrering avbrytes og ”Endre aktivitet” vises	Systemet ber om bekreftelse på at registrert aktivitet skal overskrives innenfor angitt tidsrom Hvis overskriv: Opplysningene legges inn på ny. Hvis ikke overskrive: Registrering avbrytes og ”Endre aktivitet” vises
	Utelatt type	Type utelatt	Lagring ble ikke gjennomført
	Utelatt kommentar	Kommentar utelatt	Lagring ble ikke gjennomført
	Privat ikke merket Privat er merket	Aktuell aktivitet vises for andre brukere Aktuell aktivitet vises ikke for andre brukere	Aktuell aktivitet vises for andre brukere Aktuell aktivitet vises ikke for andre brukere
	Avslutt	Avslutte HEVN-M	Avslutte HEVN-M
	Avbryt	”Vis timeplan”	Vis timeplan
Slett aktivitet	OK	Vil du virkelig slette: Ja: Sletter alle opplysninger på angitt dato. Nei. Vis timeplan	Sletter alle opplysninger på angitt dato. Vis timeplan
Andres timeplan	OK	Søk ansatt	Søk ansatt
	Avslutt	Avslutter HEVN-M	Avslutter HEVN-M

Søk ansatt	Kun fornavn: Angir en bokstav som finnes i et fornavn.	Navn funnet: Lister alle som har angitt bokstav i fornavnet sitt	Navn funnet: Lister alle som har angitt bokstav i fornavnet sitt
	Kun fornavn: Angir bokstav som ikke finnes i et fornavn.	Får melding om at ingen oppfyller søkekriteriet	Får melding om at ingen oppfyller søkekriteriet
	Kun etternavn: Angir en bokstav som finnes i et etternavn Kun etternavn: Angir bokstav som ikke finnes i et etternavn.	Navn funnet: Lister alle som har angitt bokstav i etternavnet sitt Får melding om at ingen oppfyller søkekriteriet	Navn funnet: Lister alle som har angitt bokstav i etternavnet sitt Får melding om at ingen oppfyller søkekriteriet
	Kombinasjon fornavn/etternavn: Angir bokstaver som finnes både i etternavn og fornavn	Navn funnet: Lister alle som har angitte bokstaver i fornavns- og etternavnsfeltet	Navn funnet: Lister alle som har angitte bokstaver i fornavns- og etternavnsfeltet
	Kombinasjon fornavn/etternavn: Angir bokstaver hvor det er match kun i fornavn.	Får melding om at ingen oppfyller søkekriteriet	Får melding om at ingen oppfyller søkekriteriet
	Kombinasjon fornavn/etternavn: Angir bokstaver hvor det er match kun i etternavn	Får melding om at ingen oppfyller søkekriteriet	Får melding om at ingen oppfyller søkekriteriet
	Angir ingen bokstaver i verken etternavns- eller fornavnsfeltet	Får melding om at minst ett av feltene må fylles inn.	Får melding om at minst ett av feltene må fylles inn.
	Tilbake	Meny	Meny
Progressbar – søk ansatt	Avbryt	Søk ny	Søk ny
Navn funnet:	Navn merket ? OK	Velg Dato	Velg Dato
	Nytt søk	Søk ansatt	Søk ansatt
Velg Dato	Velger datofelt	Viser kalender	Viser kalender
	Velg angitt dato hvor aktiviteter er registrert	Viser angitt dags hendelser	Viser angitt dags hendelser
	Velg dato hvor aktiviteter ikke er registrert	Viser melding om at ingenting er registrert	Viser melding om at ingenting er registrert
	Tilbake	Meny	Meny
	Avslutt	Avslutter HEVN-M	Avslutter HEVN-M

Progressbar – Velg Dato	Avbryt	Velg Dato	
Vis hendelse	Tilbake	Velg Dato	Velg Dato
	Avslutt	Avslutter HEVN-M	Avslutter HEVN-M
	Avbryt	Meny	
	Nytt søk	Søk ny	

Andres timeplan testet 03.04. Min timeplan testet 08.04 og 09.04

F. ULØSTE OPPGAVER

Uavklarte oppgaver og kjente mangler ved avslutning av 2. iterasjon:

Brukernavnet lagres ikke på klienten.

Fullstendig logging fra serverklassen, med riktig nivå

Feilhåndtering er fortsatt ikke fulltestet.

Uavklart hvordan håndtere hendelser der stopp kommer etter midnatt, dvs.
stopptid før starttid.

Konfliktsjekk der en ny hendelse ønskes lagret på en dag forskjellig fra den som er lagret i klientminnet.

Sjekke om bruker har gitt student-tilgang til sin timeplan.

En del "ulogisk oppførsel" i navigering og oppdateringer, en del småfeil i brukermeldinger bør endres.

Feil koding av norske teksttegn i kobling mellom database og server.

3. ITERASJON NR.3

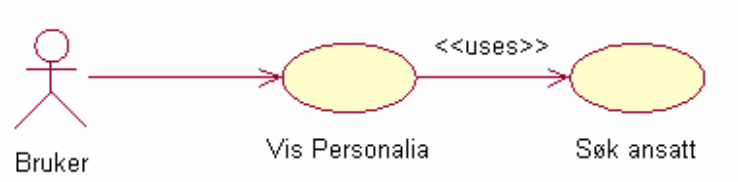
I denne iterasjonen utvikler vi funksjonaliteten for å lese HEVN-M brukernes personalia. I tillegg vil de fleste feil og mangler fra tidligere iterasjoner bli rettet opp.

3.1. KRAVSPESIFIKASJON

3.1.1. FUNKSJONELL SPESIFIKASJON

Dette er en detaljert kravspesifisering for de funksjonaliteter som tilhører denne iterasjonen. Dette er uttrykt ved UseCase-beskrivelsene vis personalia og søk ansatt.

USECASE DIAGRAM



Figur 6-3-1 Viser sammenhengen mellom UseCasene

USECASE BESKRIVELSER

VIS PERSONALIA

Mål: Vise personalopplysninger om ansatt

Aktør: Bruker

Beskrivelse: Vise personalopplysninger for en gitt ansatt. Med personopplysninger menes fornavn, etternavn, telefon jobb, telefon hjem, mobiltelefon, email, adresse.

Prebetingelser:

- UseCase ”Søk ansatt” utført og en brukerid er lagret i minne. Bruker har valgt ”Personalia” i GUI for annsattliste etter ”Søk ansatt” eller ”Vise timeplan”

Postbetingelser:

- GUI viser valgt ansatt sine personalia eller melding om ”Ingen tilgang”.
- Bruker kan velge ”Avslutt”? avslutter HEVN-M.
- Bruker kan velge ”Tilbake” ? forrige skjerm (”Søk ansatt” eller ”Vis timeplan”)

Scenarier:

Viser progressbar mens data hentes.

1. Starter separat tråd som utfører SOAP-kall til server og ber om personalia for valgt ansatt.
2. Server lager SQL forespørsel og utfører søk i databasen.
3. Resultatet returneres fra databasen til serverklasse.
4. Server returnerer personalia
5. Tråden avslutter. GUI viser personalia**.**

Alternative scenarier:

5.1 Personalia ikke tilgjengelig. Databasen returnerer ”tomt resultat”.
5.2 Server returnerer melding.
5.3 Tråden avslutter. GUI viser melding ”Ingen tilgang”.

3.1.2. TESTING AV IMPLEMENTASJON

Testing av modulen ”Vis personalia” vil skje undervis i kodingen og implementasjonen. Dette utføres av gruppemedlemmene. Det vil ved avslutning av iterasjonen bli utført en test etter en testplan som er satt opp av gruppemedlemmene. Det vil testes på at ”Vis personalia” kan velges fra resultatlista til ”Søk ansatt” og ”Vis timeplan” for andre ansatte.

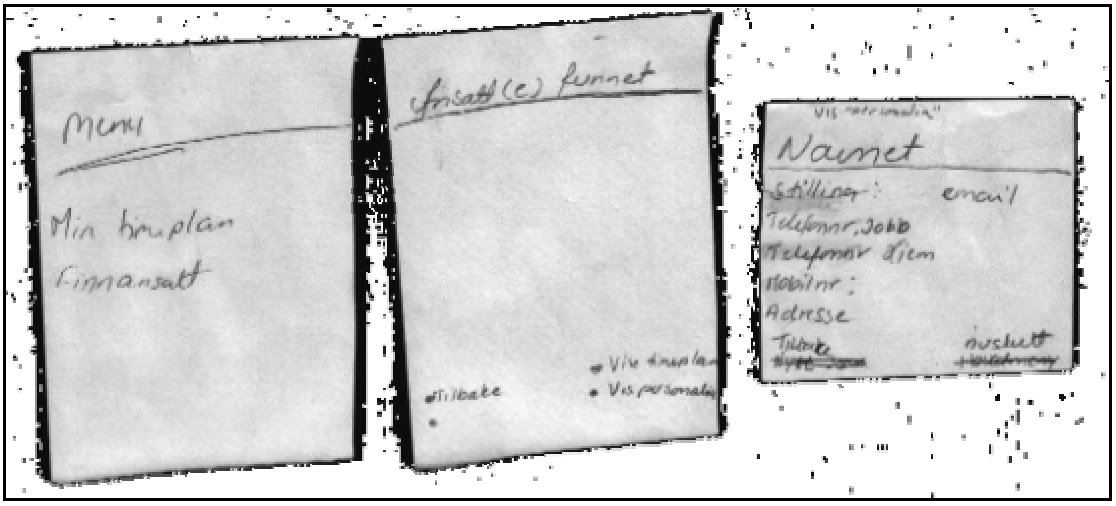
AKSEPTANSE KRAV

Henviser til postbetingelser i UseCase ”Vis personalia”.

3.2. ANALYSE OG DESIGN

3.2.1. HVA SKAL SYSTEMET GJØRE

I tredje og siste iterasjon har vi valgt å lage en funksjonalitet – ”Vis personalia”. Her skal brukere kunne få oppgitt personalopplysninger om en gitt ansatt. Opplysninger vi mener vil være aktuelle er telefonnumre (jobb, hjem, mobil), adresse, email. ”Vis personalia” skal kunne velges fra resultatlista ”Søk ansatt” og ”Vis timeplan” da ”Finn ansatt” er valgt i hovedmenyen.



Bilder 6-3-1 Papirlappmetoden – Første utkast av design, på papir.

3.2.2. HVORDAN LØSTE VI DET

PERSONALIA

Bruker vil når ”Vis personalia” velges fra ”Finn ansatt” eller ”Vis timeplan” for andre ansatte få oppgitt ansatts fornavn, etternavn, telefonnumre (jobb, hjem, mobil), emailadresse og hjemadresse. Vi mener at disse opplysningene er mest aktuelt å få på mobiltelefonen dersom du ikke har PC tilgjengelig. Se Vedlegg B for flere eksempler på brukergrensesnittet.

Bilde 6-3-1 Vis personalia

3.3. IMPLEMENTERING

3.3.1 GENERELT

Arbeidet med å implementere "Vise personalia" har ikke ført til store utvidelser av systemet. Det er derimot gjennomført ytterligere endringer og forbedringer av eksisterende kode. Dette for å innarbeide ny læring og rette opp feil og mindre gode løsninger fra tidligere.

Tilgang til personalia skulle i følge kravspesifikasjonen kunne begrenses av brukerne. Dette er implementert ved å benytte feltet som begrenser studentenes tilgang til informasjonen i HEVN-J(beregnet for begrenset tilgang via Web). Dette betyr at dersom en bruker angir "ingen studenttilgang til personalia" i HEVN-J, vil ikke personalia kunne hentes ut til mobile klienter i HEVN-M. Samme fremgangsmåte er benyttet for å begrense tilgangen til timeplandata. Da benyttes databasefeltet som begrenser tilgang til timeplandata.

3.3.2 ULØSTE OPPGAVER FRA TIDLIGERE ITERASJONER

Brukernavnet lagres.
Starttid etter stopptid godtas ikke(uavhengig av dato).
Konfliktsjekking er bedret.
Student-tilgang til timeplan og personalia respekteres.
Navigering og oppdateringer endret.
Feil koding av norske teksttegn er endret ved å gjøre forandringer i klassen som håndterer XML returen fra server(HttpSessionTransport).

Feil og mangler som fremkommer under testing og evt. optimalisering etter 3. iterasjon vil beskrives i overordnet del av rapporten.

3.3.3 KLIENT

Klientapplikasjonen er i 3. iterasjon utvidet med en klasse for grafisk fremstilling av personalia. Denne klassen sørger for å initiere metodekall til serveren for å hente informasjonen. Metodekallet utføres i en separat tråd mens brukergrensesnittet viser en "progress bar", og gir mulighet for å avbryte handlingen.

Det er implementert en løsning for å "huske" brukernavnet for innlogging. Når en godkjent innlogging finner sted lagres det benyttede brukernavnet i ikke-flyktig minne på klienten. Dette brukernavnet kommer da opp som standardverdi i tekstfeltet ved neste pålogging. Dette fører til mindre inntasting, og enklere bruk. Litteratur om emnet anbefaler å bruke samme framgangsmåte for passord[1]. Vi har derimot valgt å ikke lagre brukerens passord på grunn av sikkerhetsmessige årsaker.

Navigasjonen er endret noe slik at valgene i hovedmenyen er “Min timeplan”, der lese, endre og slette timeplanoppføringer er tilgjengelig, og “Søk ansatt”. Når en ansatt er funnet, kan klientbrukeren velge å se timeplan eller personalia. ”Vise personalia” bygger derfor videre på funksjonaliteten for å søke og velge en bruker blant de ansatte.

Implementasjonen av å lagre endringer er forbedret fra iterasjon 2 (se avsnittet for server).

Beskrivelse av ev. optimalisering av kode for hastighet, minnebruk og applikasjonsstørrelse vil beskrives i overordnet del av rapporten.

Kodeeksempel fra klientklassen InfoForm:

```
public void run(){
  if(! keepOn){return;}
  String[] strings = null;
  try{
     // do the call, get the results
     Vector data = null;
      Object obj = owner.doSoapCall("getPersonalInfo", null);
      if(obj instanceof Vector){
         data = (Vector)obj;
      }
      if(data == null){
         throw new Exception("Resultatvektor = ittno!");
      }
      // get results into array
      Enumeration resultObjects  = data.elements();
      int noOfStrings = data.size();
      // if match found, one or more...
       strings = new String[noOfStrings];
       int count = 0;
       while(resultObjects.hasMoreElements()){
strings[count]=(String)resultObjects.nextElement();
         count++;
       }
       showInfo(strings);
       data = null;
       obj = null;
       resultObjects = null;
       strings = null;
       System.gc();
  }catch(Exception e){
     System.out.println("Feil: " + e.toString());
     e.printStackTrace();
     Alert al = new Alert("Feil!",
                          "En feil oppsto under operasjonen",
                          null, AlertType.ERROR);
      al.setTimeout(Alert.FOREVER);
      owner.getDisplay().setCurrent(al, this);
  }
```

```
}
public void showInfo(String[] infoStr){
   StringBuffer sb = new StringBuffer();
   sb.append(infoStr[0]);
   sb.append(" ");
   sb.append(infoStr[1]);
   name = sb.toString();
   if(infoStr.length < 3){
      info = "Ingen tilgang til personalia!";
   }
   else{
      sb.delete(0, sb.length());
      for(int i=2;i<infoStr.length;i++){
         sb.append(infoStr[i]);
         sb.append(":\n");
         if(++i < infoStr.length){
           sb.append(infoStr[i]);
           sb.append("\n");
         }
      }
      info = sb.toString();
   }
   sb = null;
   this.append(info);
   this.setTitle(name);
   owner.getDisplay().setCurrent(this);
}
```

3.3.4 SERVER

På serversiden er funksjonalitet mot databasen skilt ut i en egen klasse. Denne endringen er gjennomført i koden for tidligere iterasjoner også. En ny metode for å besvare forespørsler om personalia via SOAP er implementert.

For å begrense datamengden, og for å unngå å sende brukernavnet (i klartekst) som identifikasjon på hvilken bruker det ønskes informasjon om, lagres en tabell over brukernavnene som tilfredsstiller søkekriteriene i "søk ansatt" på serveren. Personnavnene som sendes til klienten ligger i en tabell med tilsvarende indekser, slik at videre referanser til en bruker kun blir denne indeksen. Dermed ivaretaes sikkerheten bedre, og datamengden reduseres til et enkelt heltall. Tabellen på serveren lagres i et objekt som kun den påloggede klientbrukeren har tilgang til(sesjon). Dette gjelder all funksjonalitet som innebærer identifikasjon av en gitt bruker.

Det viste seg at kodingen av funksjonaliteten for lagring av nye og endrede hendelser på timeplanen i iterasjon 2 var noe for enkel. Dette er blitt rettet opp i denne runden. Løsningen innebærer at ved et forsøk på lagring sjekkes for konflikter på klienten(hvis timeplandata for denne dagen er i minnet). Ved konflikt får brukeren mulighet til å overskrive. Da sendes forespørsel til serveren, med beskjed om å lagre uansett. Hvis ingen konflikter avdekkes på klienten, sendes forespørselen til serveren og blir sjekket der. Konflikter meldes til brukeren,

som kan velge å overskrive. Ved vellykket operasjon sendes oppdaterte data tilbake til klientapplikasjonen. Dette er dermed implementert slik at funksjonaliteten tilsvarer mulighetene(og begrensningene) i HEVN-J. Brukeren kan velge å overskrive eksisterende aktiviteter, men da fjernes den overlappende delen av tidligere lagret aktivitet.

Alle databaseoperasjoner som innebærer flere enkeltendringer, utføres som en transaksjon der endringene ikke gjøres permanente før hele transaksjonen er gjennomført. Dette gjelder blant annet oppdateringer, dette gjøres som en rekke handlinger mot databasen; slette den opprinnelige aktiviteten, om nødvendig slette deler av overlappende aktiviteter og lagre oppdatert aktivitet.

Kodeeksempel fra ny serverklasse DbConnection:

```
  /**Update data in a task object already in the database
   *  assumes that no conflict occurs.
   *  @param t Task object with the data to store
   *  @param con A open connection object to the database
   */
   public void update(Task t) throws Exception {
      StringBuffer sqlBuffer = new StringBuffer();
      sqlBuffer.append("UPDATE hendelse SET sluttid=");
      sqlBuffer.append(t.getStop().getTime());
      sqlBuffer.append(", hendelsestype='");
      if(t.getType() == null){
         t.setType("");
      }
      sqlBuffer.append(t.getType());
      sqlBuffer.append("', kommentar='");
      if(t.getComment() == null){
         t.setComment("");
      }
      sqlBuffer.append(t.getComment());
      sqlBuffer.append("',oppsett =0 ,erpersonlig=");
      if(t.getIsPrivate().booleanValue()){
         sqlBuffer.append("1 ");
      }
      else{
         sqlBuffer.append("0 ");
      }
      sqlBuffer.append("WHERE login='");
      sqlBuffer.append(userID);
      sqlBuffer.append("' AND starttid=");
      sqlBuffer.append(t.getStart().getTime());
      //logging.fine("Executing SQL statement:\n" +
                sqlBuffer.toString());//todo
      Statement stat = con.createStatement();
      int changedRows = stat.executeUpdate(sqlBuffer.toString());
      if(changedRows < 1 && con.getAutoCommit()){
         throw new Exception("No change made to database");
      }
   }// end dbUpdate(...)
```

3.4. TESTING

3.4.1 GENERELT

For generell beskrivelse av testing, se 3.1.2, i kravspesifikasjon for denne iterasjonen.

All testing er foretatt av gruppemedlemmene på egne arbeidsstasjoner. Den nye funksjonaliteten "Vis personalia" og oppretting av tidligere feil og mangler har også blitt testet som en integrert del av kodingen. Ved avslutningen av iterasjon 3 ble det laget en testplan kun over den nye funksjonaliteten. Vi har valgt å teste oppretting av feil og mangler ved avslutningen av prosjektet. Da blir hele systemet testet inngående.

Det er ikke benyttet særskilte testklasser under avsluttende testing.
Avsluttende tester er utført etter testspesifikasjonene nedenfor.

3.4.2 TESTSPESIFIKASJON

Testing av systemet foregår i testomgivelser på våre lokale arbeidsstasjoner med telefonemulator.

HVA SKAL TESTES

- "Finn ansatt":
  - Presentere oppgitt brukes personalia

- "Vis timeplan" for andre
  - Presentere oppgitt brukes personalia

3.4.3 TESTRESULTAT

Testingen avdekket ingen store feil eller mangler ved implementasjonen. De punktene som ble testet, viste at funksjonaliteten var i overensstemmelse med kravspesifikasjonen.

Detaljerte testresultater finnes i *vedlegg E* for denne iterasjonen.

3.5. KONKLUSJON

3.5.1. DRØFTING AV FRAMDRIFTSPLAN FOR 3. ITERASJON

Oversikt over tidsforbruket.

Vi planla, som i de andre iterasjonene, å bruke 288 timer til denne iterasjonen. Etter vårt grovoverslag har vi brukt 210 timer hvorav 82 er gått med til å skrive hovedprosjektrapport. Dette er arbeid som er beregnet til å gjøres når iterasjon 3 er ferdig, så her ligger vi på forskudd ifølge framdriftsplanen.

En av grunnene til at vi ikke har brukt så mye tid, er at i samråd med oppdragsgiver bestemte vi å lage kun en funksjonalitet og rette opp feil og mangler fra forrige iterasjoner. Det ble bestemt at vi skulle bruke tid etter iterasjonen til profilering, testing og rapportskriving. Vi mener at vi kan forsvare at forbruket av tid i denne iterasjonen er lavere enn planlagt. I 1. iterasjon brukte vi mye "overtid", slik at det samlede resultatet bør bli som forventet.

Siden vi har vært i gjennom de forskjellige fasene i iterasjonen to ganger tidligere hadde vi fått god erfaring i hvordan planlegging, kravspesifisering, design og implementasjon skulle foregå. Vi brukte derfor relativt lite tid til dette.

I følge gantt-skjema for iterasjon 3 er det satt opp to statusmøter. Det første skulle foregå underveis i iterasjonen for å motta råd fra veileder**.** Dette ble det ikke noe av da veileder var noe travel og vi fikk redegjort hva iterasjonen skulle inneholde før iterasjonen startet. Siden vi kun skulle lage en ny funksjonalitet og rydde opp i "mangellista" så ble siste statusmøte utsatt til uka etter iterasjonens avslutning da dette passet best for veileder

	Timer
Planlegging	16,5
Kravspesifisering	10
Design	11,5
Implementasjon	55
Sluttesting	15
Rapportskriving, iterasjon 3	21,5
Hovedprosjektrapport	82
Sum	**210**

Pausene er ikke silt ut.

Se vedlegg A - Gantt-skjema for å sammenligne den planlagte planen med den endelige planen.

3.5.2. EVALUERING AV ITERASJONEN

PRODUKT

Da dette er siste iterasjon i dette prosjektet, fremstår resultatet av arbeidet som et mer fullstendig produkt enn tidligere. Vi ser derfor tydeligere de sterke og svake sidene ved arkitektur, implementasjon og brukervennlighet for systemet.

Den største svakheten i forhold til brukervennlighet synes å være begrensningene i kontroll over layout, navigasjon og utseende på klientapplikasjonen. Konseptet med J2ME og MIDP 1.0 bygger i stor grad på fleksibilitet når det gjelder plattformens (telefonens) implementasjon av forskjellige funksjoner. Dette medfører at når vi f. eks trenger å gi brukeren mulighet til å "si ok", kan vi i koden kun be om dette. Vi har ingen kontroll over hvordan telefonen velger å presentere dette for brukeren. En del telefoner har en fysisk "ok - knapp", og legger denne funksjonen dit. I andre tilfeller havner dette valget til sist i en menyliste over tilgjengelige valg(options). Den samme problemstillingen gjelder alle kommandoer. Resultatet er at det er umulig å forutsi brukergrensesnittets faktiske design.

Styrken i dette er at det løser kompatibilitetsproblemer mellom ulike plattformer, slik at utviklere ikke trenger å ta hensyn til hvilken telefon el. applikasjonen skal kjøres på. Det ser imidlertid ut til at MIDP 1.0 ikke er klar for denne oppgaven ennå. Derfor utgir de største produsentene egne javabiblioteker for utbygging av J2ME til merke- og modellavhengig applikasjonsutvikling.

Et annet grunnleggende trekk ved HEVN-M er bruken av SOAP for kommunikasjon mellom klient og server. I motsetning til J2ME og MIDP 1.0 virker denne arkitekturen svært moden og "praktisk brukbar". En av de størst fordelene vi ser i bruken av SOAP i vårt prosjekt, er at mye av logikken og håndteringen av kommunikasjon utføres ved hjelp av standardløsninger. Http som transportprotokoll og XML som dataformat gjør løsningen plattformuavhengig og anvendelig også når det er nødvendig å passere brannmurer. Serverutviklingen forenkles betydelig da Jakarta-Tomcat (i vårt tilfelle) tar seg av sesjonshåndtering, lytting og oppkobling mot klient etc.

Arbeidet med å konvertere dataobjekter fra Java til XML er på den andre side en svakhet med hensyn på resursbruk. Dette merkes selvfølgelig tydeligst på klientsiden i vårt system, da det er her minne og prosessorhastighet er mest begrenset. Vi viser til overordnet del av rapporten for en nærmere analyse av dette, og av generell resursbruk.
Denne konverteringen(serialisering – marshalling) er det som har vært vanskeligst å implementere. Kodingen av dette har vært en lærerik utfordring.

Bortsett fra at det kan oppstå uønskede resultater i brukergrensesnittet på enkelte plattformer, ser det ut til at HEVN-M er blitt et system som vi trygt kan vedkjenne oss å være opphav til.

Denne rapporten, som også er et produkt av dette prosjektarbeidet, begynner nå ved avslutningen av tredje iterasjon å ta form. Det har frem til nå vært litt vanskelig å finne en

god og oversiktelig struktur, men vi ser at resultatet vil kunne bli bra. Mye på grunn av at rapportskriving og redigering har vært utført løpende fra starten av prosjektet.

PROSESSEN

Som i de tidligere iterasjonene vil vi prise vår valgte systemutviklingsmodell – inkrementell modell. Vi er enige i at den er som sydd for prosjektet vårt. Vi har i denne iterasjonen gått sikkert til verks i forhold til første iterasjon da det ble mye famling. Kontroll over utviklingsprosessen har vært mye enklere nå i 3. iterasjon, vi har lært mye av de to foregående gjennomløp.

Det har heller ikke i denne fasen oppstått noen "indre uroligheter" i gruppen. Vi har en åpen og uformell dialog der konstruktiv kritikk og gjensidig oppmuntring bidrar til motivasjon og godt arbeidsmiljø. Som vår veileder har påpekt fører dette til at vi ikke får erfaring med konfliktløsning. Da vi som gruppe har relativt høy gjennomsnittsalder og bakgrunn fra forskjellige konfliktarenaer, ser vi ikke dette som et stort savn.

Et viktig moment i vårt prosjekt er at vi har god tilgjengelighet til en teknisk kompetent oppdragsgiver (HiG). Dersom vi skulle gjennomført et lignende prosjekt for en oppdragsgiver uten egen kompetanse på de tekniske løsningene, ville det krevd mer ressurser å klarlegge krav og ønsker. Vi har også hatt muligheten til uformelle småsamtaler med oppdragsgiver, når avklaringer har vært påkrevd.

For fremtidige, lignende prosjekter kan vi absolutt anbefale inkrementell utvikling. Også når det gjelder prosessen med å produsere dokumentasjon og rapport, har vi erfart at det er lettere å gjøre denne jobben løpende, i iterasjonene, i stedet for å få dette arbeidet til slutt.

3.5.3. HOVEDKONKLUSJON

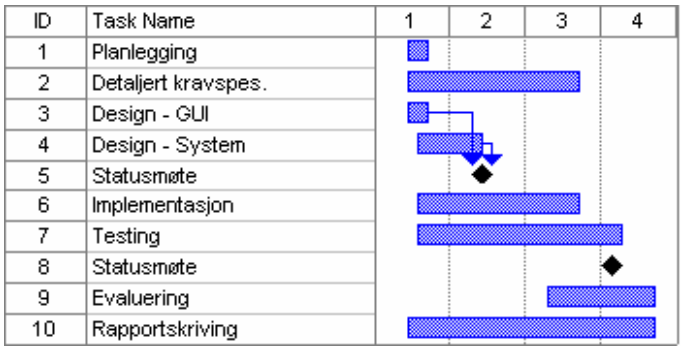
Vi har nå endelig klart å avslutte en iterasjon innenfor tidsfristen. Dette viser at vi har lært mye siden første iterasjon der vi måtte utsette fristen i over to uker. Dette er vi stolte av.

Vi som gruppe har klart å komme igjennom tre iterasjoner uten å ødelegge den gode stemningen innad i gruppen. Gruppen samarbeider godt og er som skapt for hverandre.

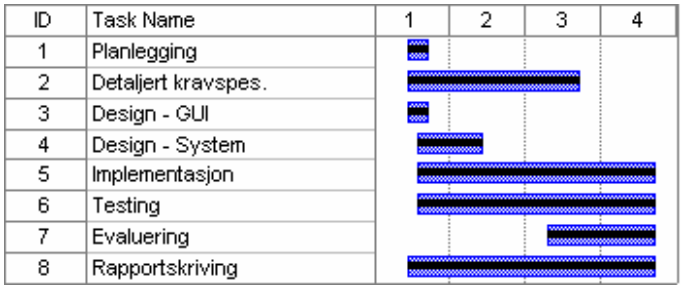
I den siste delen av prosjektet skal vi utføre testing, profilering og skrive/redigere de overordna kapitlene i hovedprosjektrapporten før vi avslutter den 19. mai. Da skal denne rapporten leveres inn.

3.6. VEDLEGG

A. GANTT-SKJEMA



Figur 6-3-A-1 Planlagt framdriftsplan



Figur 6-3-A-2 Reell framdriftsplan

B. HEVN-M GUI

I dette vedlegget viser vi de fleste av skjermbildene som inngår i denne iterasjonen. Vi henviser til tilhørende UseCase, og lar stort sett illustrasjonene tale for seg selv. Det er viktig å være klar over at det faktiske utseendet og navigasjonsmulighetene kan variere mye på forskjellige plattformer/telefoner.

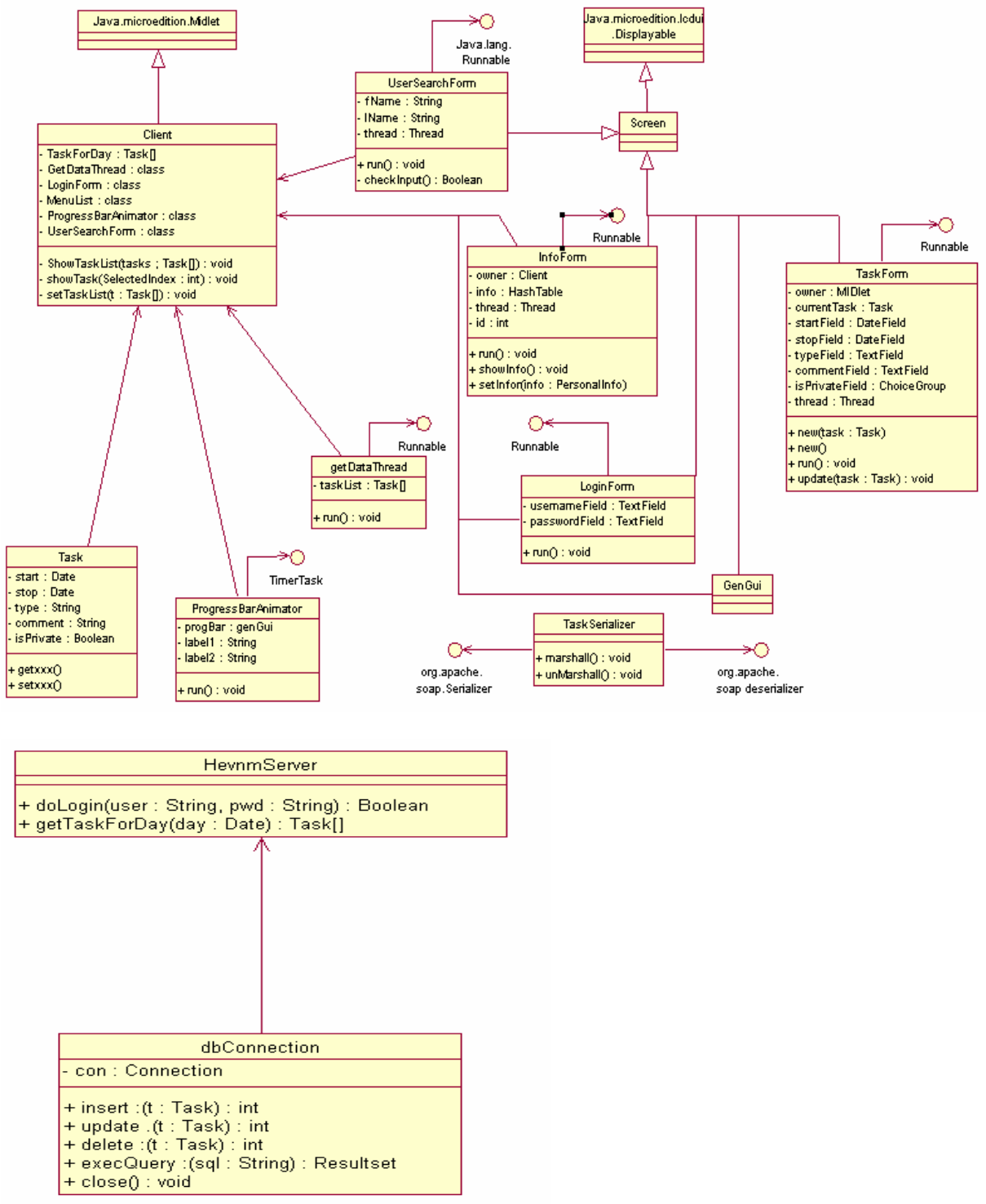
”Søk ansatt”

”Vis personalia”

Dette eksemplet viser noe av variasjonen i skjermbilder på forskjellige telefonemulatorer.

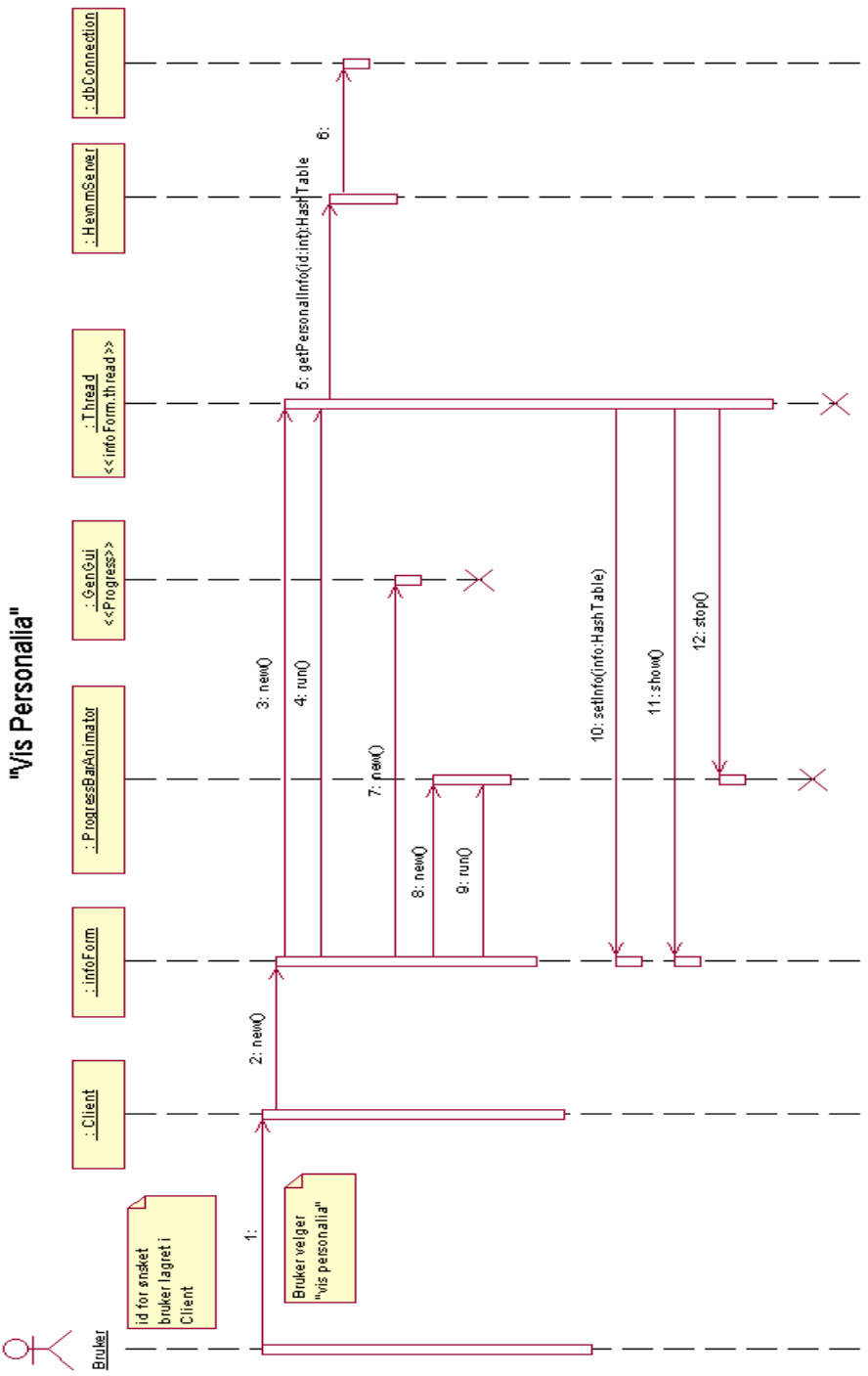
C. KLASSEDIAGRAM

Diagrammet viser klassene i systemet, og sammenhengen mellom disse.



D. SEKVENSDIAGRAM

Dette diagrammet viser samhandlingen mellom objekter ved gjennomføring av denne iterasjonens funksjonalitet.



E. TESTPLAN OG TESTRESULTAT

Vis personalia

Funksjon	Valg	Forventet resultat	Resultat
Ansatte funnet	Vis personalia	Personalopplysninger om gitt ansatt	Personalopplysninger om gitt ansatt
Vis timeplan	Vis personalia	Personalopplysninger om gitt ansatt	Personalopplysninger om gitt ansatt
Viser progress bar	Avbryt	Tilbake til liste over søkeresultat	Tilbake til liste over søkeresultat
Vis personalia	Tilbake	Tilbake til liste over søkeresultat	Tilbake til liste over søkeresultat
	Avslutt	Avslutter HEVN-M	Avslutter HEVN-M
Nettverksfeil	Velg ”vis personalia” for en valgt ansatt i liste	Feilmelding	Ingen ting
Server stanset	Velg ”vis personalia” for en valgt ansatt i liste	Feilmelding	Feilmelding

F. ULØSTE OPPGAVER

For kjente feil og mangler ved systemet ved avslutning av 3. iterasjon henvises til overordnet del av rapporten. De fleste feil og mangler fra 1. og 2. iterasjon er rettet opp. (*Se pkt 3.3 implementasjon.*)

Kapittel 7 ANALYSE AV SYSTEMET

7.1 GENERELT

Dette kapitlet inneholder informasjon om forskjellige karakteristika ved systemet. Dette er ikke en formelt grundig systemanalyse, men en gjennomgang av de viktigste operasjonelle resultatene. Vi vil forsøke å belyse forhold av særlig betydning for fremtidig videreutvikling, optimalisering og integrering av løsningen. Vi har her valgt å legge ekstra vekt på ressursbruken for klientapplikasjonen, da dette er et kritisk punkt for hvor praktisk brukbart HEVN-M kan bli dersom det settes i drift.

Når denne rapporten skrives er omgivelsene og kravene til applikasjoner for mobil bruk i stadig utvikling. Med CLDC 1.0 og MIDP 1.0 har man forsøkt å fastsette en standard for kravene til maskinvareplattformen, men det er fortsatt stor variasjon mellom forskjellige produsenters tolkning og implementasjon av disse kravene. Etter informasjonssøking på Internet har vi kommet frem til at de praktiske kravene til klientapplikasjonen ikke overensstemmer med MIDP 1.0 på disse punktene: (MIDP – krav i parentes)
Ikke-flyktig minne for lagring av midlet: 64kB (128kB).
Arbeidsminne tilgjengelig for Java-heap: 140-700kB (32kB).
Dette er basert på spesifikasjoner av typiske mobiltelefoner for Java-bruk som f.eks. Nokia 3410, 3510i, 6310i, 7210, Siemens S55, Sony Erickson P-800, m. flere.

Under utviklingsarbeidet har vi fulgt den kjente programmeringsguruen Donald Knuths utsagn: “Premature optimization is the root of all evil”, og vår veileders Amerikansk influerte “If it isn't broken, don't fix it”. Det har altså ikke blitt tatt mye hensyn til optimalisering av minnebruk eller hastighet, men vi vil med dette kapitlet beskrive systemets potensial for forbedringer.

Noen mindre forbedringer og endringer i koden er blitt utført som følge av analysen. Dette er da nevnt i dette kapitlet.

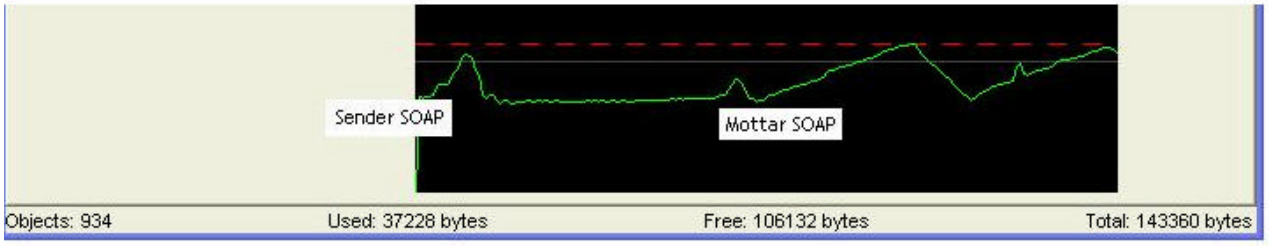
Profilering av klientapplikasjonen er i hovedsak utført med verktøy innebygd i Sun Wireless Toolkit.

7.2 MINNEBRUK

Vi har målt bruken av arbeidsminne(heap) under noen av de vanligste operasjonene for klienten. Ved målingene ble minne tilgjengelig for applikasjonen satt til 140kB. Figurene under viser en grafisk fremstilling av total minnebruk.

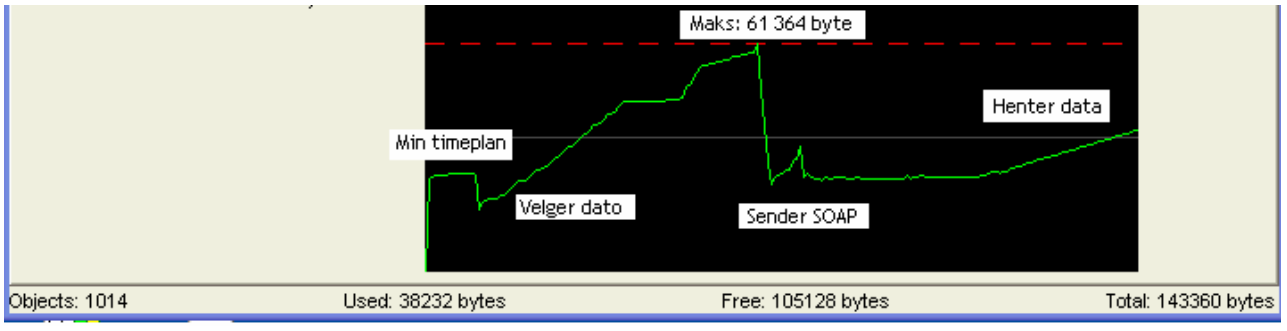
Oppstart og innlogging:

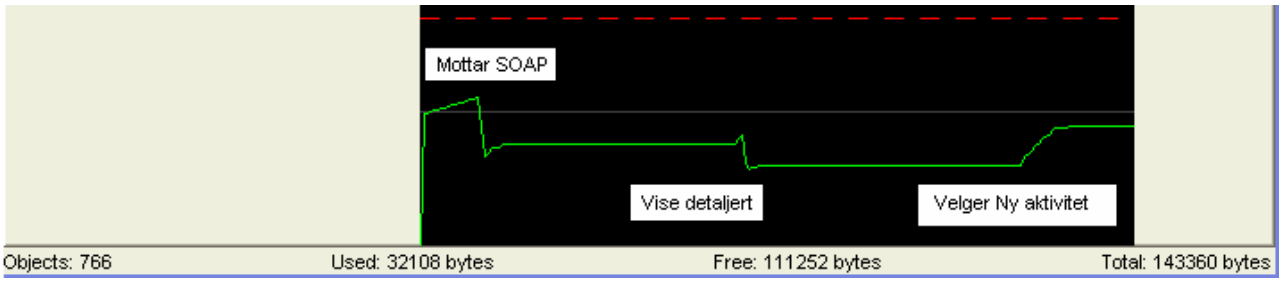
Maksimalt minnebruk ca 40kB under parsing av retur fra server.

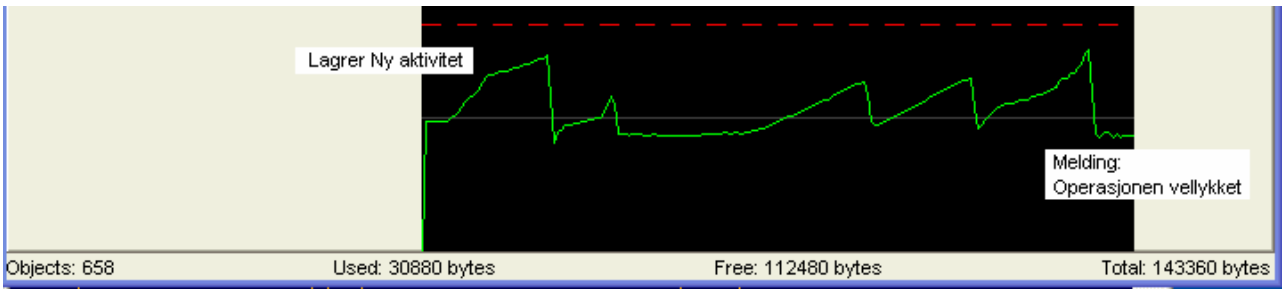


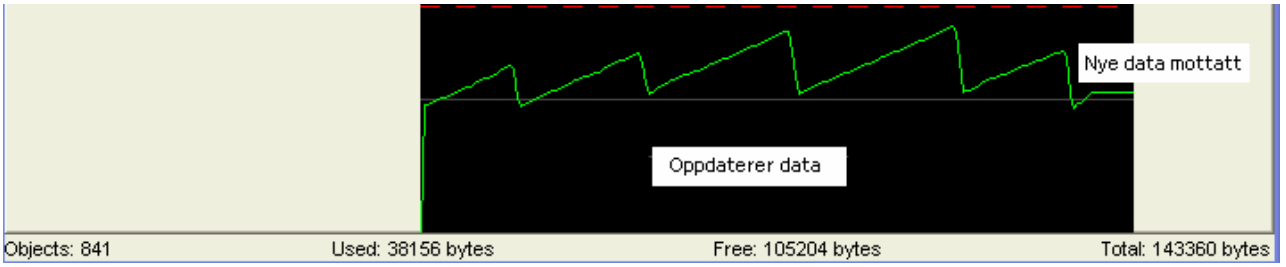
Hente og vise timeplandata:

Under denne operasjonen nås det høyeste minnebruket når GUI viser kalender for valg av dato. Gjennom forløpet nås topper 40 – 60kB før minne frigjøres.



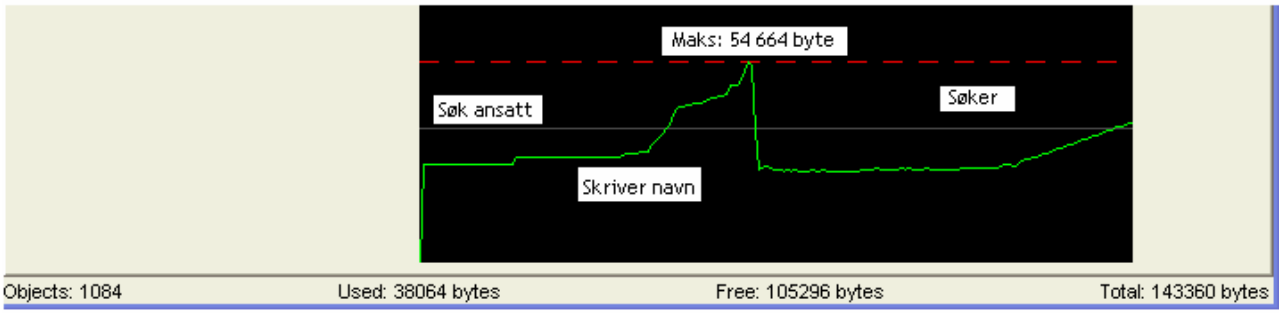


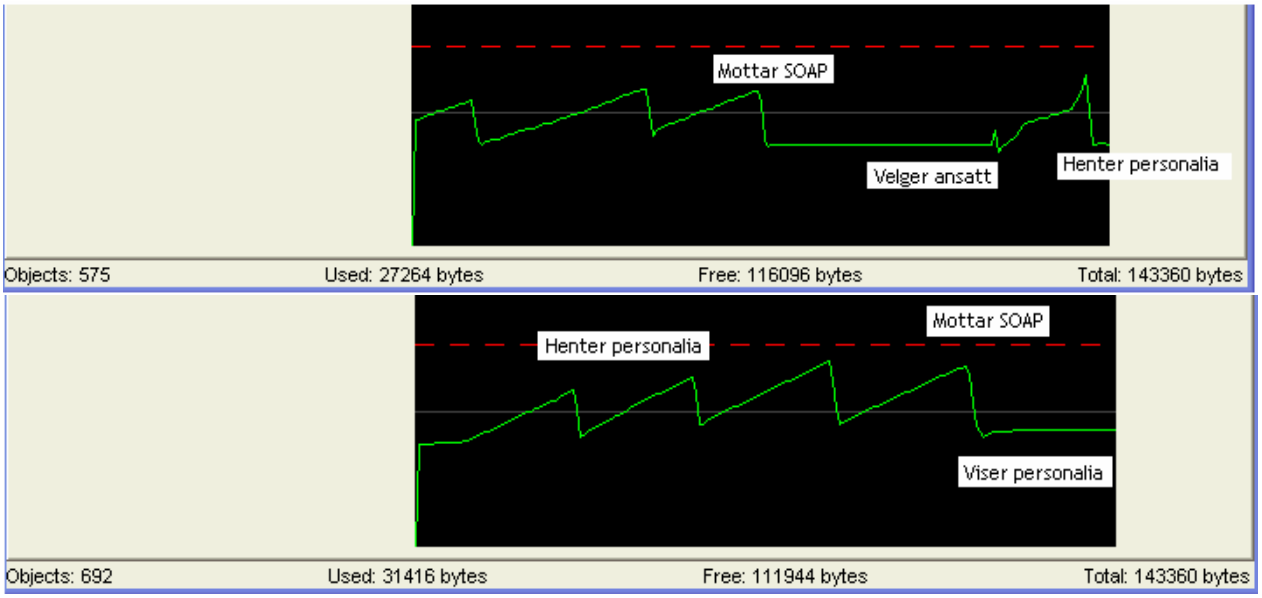




Søke etter bruker og vise personalia:

Tilsvarende som over nås en topp ved oppbygging av metodekall og parsing av retur. Maks ca 54kB.





Observasjoner:

Testene viser at det gjennomgående oppstår en topp i brukt minne i forbindelse SOAP-kall. Dette fordi det opprettes mange nye objekter i forbindelse med å lese og skrive XML. Disse objektene har kort levetid, og fjernes derfor ganske raskt når minne frigjøres(garbage collection). Det høyeste minnebruket oppsto under henting av egne timeplandata. Her var brukt minne oppe i drøyt 60kB. Disse testene gir ikke et absolutt bilde av kravet til arbeidsminne, fordi minnehåndteringen i Java er en dynamisk prosess som styres av den virtuelle maskinen. Frigjøring av minne krever også resurser, derfor prioriteres dette etter behov slik at mye ledig minne fører til lav prioritet på tråden som utfører dette arbeidet. Klientapplikasjonen er med dette testet for kjøring med 140kB tilgjengelig arbeidsminne. Med denne mengden tilgjengelig minne bruker applikasjonen ca 60kB på det meste. Kravet til MIDP 1.0 på 32kB arbeidsminne er ikke oppnådd, men i de praktiske tilfellene vi har undersøkt er tilgjengelig minne høyere enn 140kB.

7.3 HASTIGHET

Den faktiske hastigheten på en mobiltelefon er vanskelig å forutsi. Vi har brukt antall instruksjoner (bytecodes) utført som mål på hvor hurtig koden kjøres. Prosessorkapasiteten på vanlige plattformer i dag ligger rundt 30 – 300Mhz (instruksjoner utført pr sek). De fleste telefoner har kapasitet på over 200Mhz.

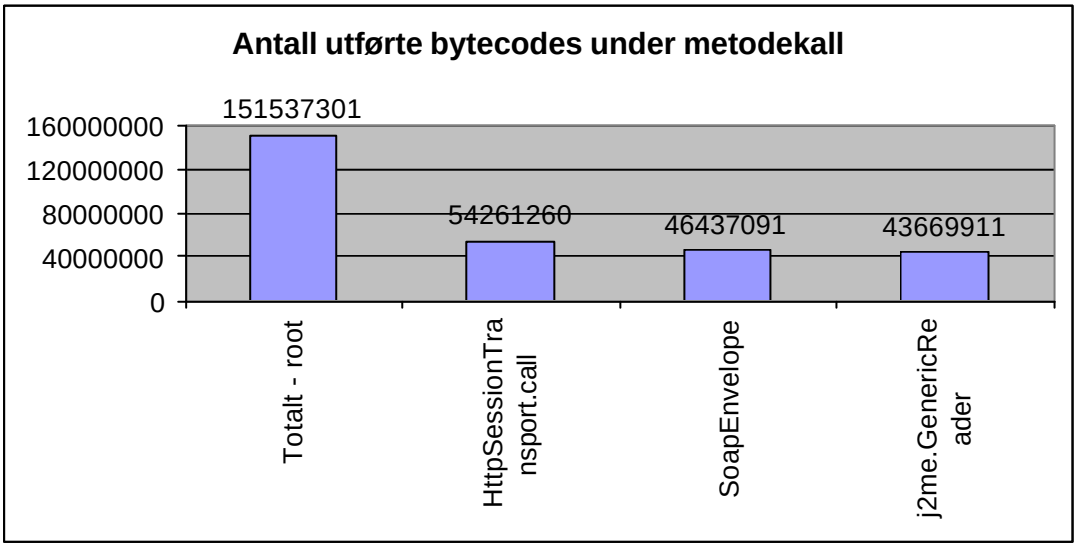
Dette punktet ble nedprioritert, og er ikke ment som en fullstendig profilering. Resultatene kan allikevel gi en pekepinn på hvor eventuell optimalisering vil ha størst effekt.

Vi har benyttet verktøy for profilering i Wireless Toolkit for å finne ut hvilke operasjoner som krever mye prosessering. Dette verktøyet presenterer resultatene i en trestruktur der nodene er metodekall. Antall instruksjoner som kreves utført som følge av hvert metodekall vises hierarkisk. Filer med resultatet av tester vi har utført ligger vedlagt på CD i katalogen "Testresultater". Filene kan åpnes i Wireless Toolkit via valgene <edit><utilities><open profiler session>, og velge ønsket fil.

Som et eksempel ble en av testene (HM-test) gjennomført slik:

- Starte applikasjon - Logge på.
- Velge "min timeplan" – velge dato.
- Vise timeplan – velge detaljer.
- Endre valgt hendelse – lagre.
- Tilbake til meny – velge "finn ansatt"
- Søke etter en ansatt og be om personalia.
- Lese personalia – avslutte applikasjonen.

Dette danner grunnlaget for figuren nedenfor (*figur 7-1*). Til venstre (på y-aksen) vises totalantallet instruksjoner utført, mens søylene viser antall instruksjoner som tilhører de nevnte metodene. Dette antallet inkluderer også alle metoder som kalles nedover i "treet" av metoder. Figuren påviser at over 30 % av instruksjonene har opphav i HttpSessionTransport.call – metoden, som utfører fjernmetodekallene.



Figur 7-1 Hastighetsmåling

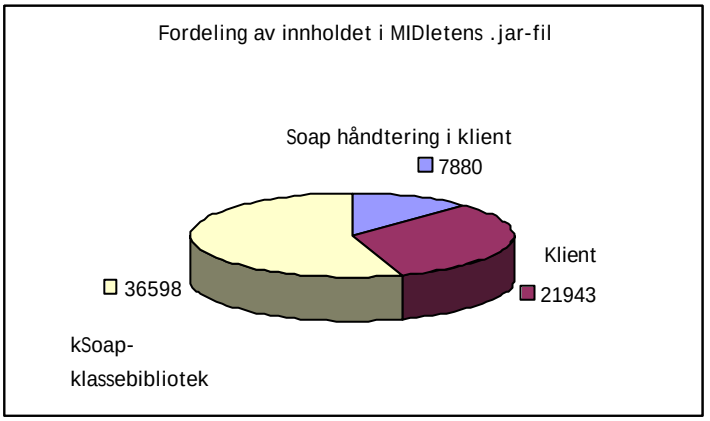
Observasjoner:
Analysen viser at det brukes mye ressurser på serialiseringen av dataobjekter mellom XML og Java. Objekter som representerer dato/klokkeslett er i koden representert med

java.util.Date -objekter. Disse konverteres til datatypen date/time i XML, som er et tekstlig format som f.eks.:`<start xsi:type="xsd:dateTime">2003-04-28T08:00:00.000Z</start>`. Ved å erstatte dette formatet med den samme heltallsrepresentasjonen som benyttes i databasen(*se Løsningsforslag/koding i statusrapport 24.2.02, vedlegg L*), kan denne konverteringen forenkles betydelig. En slik endring fører heller ikke til ulemper andre steder i systemet da et java.util.Date -objekt kun er en "wrapping" av et slikt heltall.

7.4 APPLIKASJONSSTØRRELSE

Klient

Som nevnt over, er en vanlig øvre grense for størrelse på applikasjonen(MIDlet Suite) 64kB. Mange plattformer har imidlertid mulighet for å lagre flere applikasjoner samtidig. Nokia apparater har typisk plass til 6 applikasjoner på 64kB hver. Det er størrelsen på den komprimerte .jar-fila som inneholder Java-klassene som brukes som mål på størrelsen. Selv om MIDP 1.0 spesifiserer lagringskravet til 128kB, har vi valg å "presse" størrelsen under 64kB. HEVN-M midleten hadde etter 3. iterasjon en ferdig størrelse på ca 77kB. Ved å fjerne alle unødige klasser fra kSoap biblioteket, og benytte en "code obfuscator" reduserte vi dette til 59kB. En "code obfuscator" er et verktøy som vanligvis benyttes for å beskytte kommersielle applikasjoner mot ulovlig kopiering(reversed engineering). Prosessen fjerner alle unødige referanser i klassefilene, forenkler klasse- og metodenavn m.m. En for oss ønsket bivirkning er at størrelsen på klassefilene reduseres. Figuren viser fordelingen av klassefilenes tilhørighet, særlig med hensyn på de klassene som kreves på grunn av bruken av SOAP.



Server

Størrelsen på serverklassene antas å ikke være noe kritisk punkt. Samlet størrelse på de klassene som utgjør HEVN-M Web Service er ikke mer enn 26kB. Det er oppsettet for å gjøre klassenes metoder tilgjengelige som krever plass(Tomcat og Apache-Soap).

Kapittel 8 DISKUSJON AV RESULTATER

Dette kapittelet tar for seg momenter angående resultatet av prosjektarbeidet. Vi forsøker å sammenholde resultatene med de opprinnelige kravene, og påpeker eventuelle avvik.

8.1 AVVIK I FORHOLD TIL KRAVSPESIFIKASJONEN

I planleggingsfasen var det usikkert om prosjektet skulle inkludere å sette HEVN-M i drift. Det har underveis blitt klarlagt at installasjon og drift av systemet ikke inngår i prosjektet. Likeledes er praktisk testing med telefon utelatt fra prosjektet. Det vil derfor være noe avvik i kravspesifikasjonen når det gjelder dette.

Mulighet for enkel oversettelse til andre språk i brukergrensesnittet er ikke implementert slik som beskrevet i kravspesifikasjonen*(se 2.1.4.4)*. Da J2ME ikke har støtte for internasjonalisering på samme måte som J2SE, ble det ikke utviklet en egen løsning for dette. Litteratur om J2ME[1] anbefaler heller ikke en slik fremgangsmåte, men at det i stedet distribueres egne applikasjoner ved behov for støtte av flere språk. Vi kunne imidlertid ha laget en egen klasse med de nødvendige konstante tekststrengene i tabeller. Dette ville forenklet eventuell oversettelse, men fortsatt krevd rekompilering.

Det er ikke benyttet kryptert nettverksforbindelse(https). Dette var ikke et absolutt krav, og det ble derfor benyttet vanlig http-forbindelse. I MIDP 2.0, den nyeste versjonen, er slik støtte implementert. Dette kan dermed være en svært aktuell forbedring av sikkerheten i systemet.

Det var i utgangspunktet planer om et samarbeid med NTNU/Telenor FoU i Trondheim(*se pkt 2.1.4*). Dette ble i samråd med veileder/oppdragsgiver ikke videreført. Et slikt samarbeid kan være aktuelt ved videreutvikling og praktisk testing av systemet. Arkitekturen beskrevet for utvikling er benyttet under prosjektet.(*fig. 2.1*)

Det er ikke implementert muligheter for ulike rettigheter hos forskjellige brukere. Databasen over passord og brukere inneholder en kolonne som kan benyttes for å gi lesetilgang og eventuelt skrivetilgang. Vår versjon av systemet gir alle registrerte brukere skrivetilgang på samme måte som i HEVN-J.

Passord krypteres før lagring i databasen, som beskrevet i pkt 2.1.4. Krypteringen foregår på serversiden. Passordene sendes derfor i klartekst over http. Dette er en svakhet med hensyn på sikkerheten, men er gjort slik i samråd med oppdragsgiver. Dersom HEVN-M settes i drift uten å benytte https, bør en være klar over dette forholdet.

Mengde funksjonalitet som er implementert i forhold til prioriteringslisten i pkt 2.1.4, er avgjort i samråd med oppdragsgiver. Det var viktig for oss å sette opp en prioriteringsliste, da det var usikkert i starten av prosjektet hvor mange funksjonaliteter vi ville få tid til å implementere.

8.2 DRØFTING AV FRAMDRIFTSPLAN

I begynnelsen av prosjektet laget vi en framdriftsplan(overordnet) som vi skulle arbeide etter for å kunne gjennomføre prosjektet til fastsatt tid. Den overordnede framdriftsplanen finnes som vedlegg[A].

Med tanke på at vi har brukt en inkrementell modell så har vi måttet lage egne framdriftsplaner for hver enkelt iterasjon. Disse finnes som vedlegg i hver enkelt iterasjon i kapittel 6. Dette vil si at vi til sammen har brukt fire forskjellige framdriftsplaner i denne perioden.

Det har vært en del avvik i forhold til den opprinnelige planen. For å sammenligne den opprinnelige framdriftsplanen med den reelle framdriftsplanen, *se vedlegg A.* Årsaken til at den opprinnelige planen ikke holdt mål var særlig fordi den første iterasjonen tok lengre tid enn planlagt. Manglende erfaring gjorde det vanskelig å planlegge tid sforbruket, og at det var mye nytt stoff i forbindelse med XML-SOAP og Web Service arkitektur. Vi klarte allikevel å få kontroll over framdriften senere i prosjektet.

Bortsett fra avsluttingen av den første iterasjonen, har vi nådd de viktigste milepælene. Nå er vi i ferd med å avslutte prosjektet til planlagt tid.

Oversikt over tidsforbruket

Forbruket for de enkelte iterasjoner finnes i kapittel 6.

Tabellen under viser tidsforbruket for hele prosjektet:

	Timer
Forprosjekt & Iterasjon 1 Iterasjon 2	548 259,5
Iterasjon 3	210
Etterarbeid	160
Sum	**1177,5**

Tidsforbruk etter innlevering av rapporten er ikke tatt med i beregningen.

8.3 OM RAPPORTEN

Valget av inkrementell utviklingsmodell, og krav fra oppdragsgiver om at det skulle leveres fullstendig dokumentasjon for hver iterasjon, her påvirket strukturen på rapporten i stor grad. Etter vår oppfatning lider den ferdige rapporten av et noe rotete preg. Noe av innholdet i kapitlet (*kapittel 6*) som omhandler iterasjonene, blir nødvendigvis gjentatt for å beholde sammenhengen. Andre steder har vi flyttet deler av innholdet i kap. 6 ut til den overordnede

delen. Leseren av rapporten må dermed belage seg på litt hopping frem og tilbake for å få oversikt over innholdet.

På den andre side har denne fremgangsmåten resultert i at arbeidet med rapporten har vært en integrert del av prosjektet. Vi hadde allerede ved avslutning av 1. iterasjon en tilnærmet komplett rapport. Dette har også gjort det mulig for oppdragsgiver å komme med ønsker og kommentarer til rapporten på et tidlig tidspunkt.

Kapittel 9 KONKLUSJON

Dette kapittelet inneholder et sammendrag av prosjektet. Vi har gitt uttrykk for våre synspunkter om prosjektet og hva vi har lært av det.

9.1 KRITIKK AV OPPGAVEN

9.1.1 PRODUKT

Resultatet av prosjektet synes tydelig å bære preg av J2ME og MIDP sine begrensninger. Denne teknologien virker fortsatt noe umoden. Kommersielle leverandører av applikasjoner for mobiltelefoner løser dette ved å lage løsninger som er knyttet til bestemte modeller.

En av de største utfordringene var design av brukergrensesnittet. Det er stor forskjell på informasjonsmengden som kan presenteres på en PC i forhold til en liten mobiltelefon. Der HEVN-J kan vise frem timeplanen for en uke om gangen, har vi begrenset oss til å vise registrerte aktiviteter for en enkelt dag. Vi mener at vår løsning har utnyttet de tilgjengelige ressursene på en god måte.

Da vi ikke fikk mulighet til å teste systemet i praksis, men kun har benyttet emulator på klientsiden og LAN som nettverk, er vi noe usikre på hvilke problemer som kan oppstå dersom systemet skal settes i drift. En del usikkerhet er også knyttet til at vi har benyttet MySQL-database, med tilhørende JDBC-driver. Under de begrensninger som ble satt, er vår utviklingsdatabase allikevel det nærmeste vi kommer originalen.

Rapporten har, som tidligere nevnt, sine svakheter. Vi tror allikevel at vi har klart å formidle den nødvendige informasjonen for å gi et godt bilde av prosjektarbeidet. Bruken av javadoc for dokumentasjon av kildekoden, har gitt oversiktlig og brukervennlig veiledning for innsikt i koden.

Ved avslutning av dette prosjektarbeidet tør vi si at vi er stolte av produktet vi leverer. HEVN-M er blitt et system som inneholder mye fremtidsrettet arkitektur, og mange av problemstillingene som er blitt løst eller belyst, bør kunne være interessante for videre utvikling. Vi håper jo at en videreutviklet versjon av dette systemet vil settes i drift ved HiG, men utviklingsarbeidet i seg selv har hatt stor verdi for gruppedeltagerne, og forhåpentligvis for det teknologiske miljøet ved skolen.

9.1.2 PROSESSEN

Arbeidet med prosjektet har blitt styrt av den inkrementelle systemutviklingsmodellen. Som vi tidligere har sagt, ”Denne modellen er som sydd for prosjektet vårt”. Vi er veldig fornød med valget av denne modellen. Dette er en modell som gir synlige resultater tidlig i

prosjektfasen. Dette er motiverende og fører videre til bedre resultater. Et slikt hovedprosjekt kan være stressende, og til tider en psykisk påkjenning. Vi ser at vår mulighet til å levere "noe" tidlig, har virket beroligende på arbeidsmiljøet vårt.

Siden vi har brukt denne modellen (og er generelt ansvarsfulle mennesker), har vi hatt kontroll på prosjektet gjennom det meste av prosessen. Dette har også oppdragsgiver/veileder uttalt seg om. Vi har hatt de nødvendige verktøyene for å holde oversikt over fremdriften. Milepæler og planlegging av oppgaver har gitt oss tidlig varsel om problemer med prosessen, slik at vi har hatt mulighet til å ta fatt i dette på et tidlig stadium.

Det største problemet vi har hatt med tanke på denne utviklingsmodellen er rapportskrivingen nå i etterkant. Underveis var det veldig nyttig med en rapport etter endt iterasjon, men dette har blitt litt rotete i denne endelige rapporten. For mer informasjon om rapporten, *se pkt. 8.3*.

9.2 VIDERE ARBEID

Applikasjonen HEVN-M har blitt utviklet slik vi og oppdragsgiver ønsket. En applikasjon blir sjelden helt ferdigutviklet, noe som vi heller ikke kan si om HEVN-M. Det er derfor potensiale for forbedringer i systemet.

De viktigste punktene som kan forbedres er; kryptert nettverksforbindelse, hastighetsforbedringer, bedret kontroll over brukergrensesnittet og optimalisering av minnebruk og applikasjonsstørrelse. For nærmere detaljer om dette, *se kapittel 7 Analyse av systemet*. Vi antar at nyere versjoner av MIDP og CLDC vil være forbedret på flere av disse områdene. Dokumentasjon av systemet, og da særlig kildekoden, er gjort med tanke på muligheten for videre utvikling.

For kjente feil og mangler henvises til punkt 8.1.

9.3 EVALUERING AV GRUPPENS ARBEID

9.3.1 ORGANISERING

Organiseringen av gruppen har vært uformell og basert på daglig muntlig kommunikasjon. Gruppen har valgt en gruppeleder siden det er et krav, men de fleste avgjørelser er tatt i fellesskap. Gruppelederen er den som har hatt mest kontakt med oppdragsgiver/veileder. Den uformelle organiseringen har fungert godt siden gruppen er liten og arbeidsoppgavene oversiktlige.

9.3.2 ARBEIDSFORDELING

Vi har stort sett fulgt den planlagte fordelingen av arbeidet som står skrevet i Forprosjektrapporten, *vedlegg C, pkt 3.1*. De som har hatt ansvaret for de forskjellige oppgaver har sørget for å fordele oppgaven mellom gruppemedlemmene slik at alle har fått prøvd seg.

I starten av prosjektet forsøkte vi å samarbeide om koding for at alle skulle få litt erfaring med Java programmering. Det viste seg dessverre at dette tok mer tid enn vi hadde til rådighet. Dermed ble programmeringen hovedsakelig utført av Sveum som går programmeringslinja. Krasniqi og Harbosen har tatt seg mer av de andre oppgavene i prosjektet; som systemutvikling og rapportskriving. Denne fordelingen ble vi tidlig enig om var det beste for gruppen hvis vi skulle få et godt resultat.

9.1.3 SUBJEKTIV OPPLEVELSE AV PROSJEKTET

Det er første gang vi har arbeidet selvstendig med et så stort og krevende prosjekt i løpet av utdannelsen vår her ved HiG. Det har vært vårt eget ansvar å gjennomføre dette prosjektet på en verdig måte både for oss som studenter og for skolen som oppdragsgiver.

Vi er fornøyde med valget av prosjektoppgaven. Oppgaven har vært veldig interessant gjennom hele prosjektperioden. Det vi er litt skuffet over, er at vi ikke har fått muligheten til å sette applikasjonen i drift og teste det praktisk med en mobiltelefon.

Arbeidsforholdene har vært greie. Vi hadde en tøff periode i begynnelsen av prosjektet, men dette ordnet seg fort. Årsaken til dette kan være at vi manglet litt erfaring med systemutvikling. Vi har jobbet mye med rapporten gjennom hele prosjektet. Dette virket ikke så interessant og spennende i perioder, men vi ser nødvendigheten av dette nå.

Vi har gjennom hele prosjektet hatt ett godt forhold innad i gruppa. Dette har ført til at vi har jobbet ordentlig igjennom hele prosjektperioden og at vi i dag er stolte av det resultatet vi har oppnådd. I tilegg til faglig utvikling i programmeringsspråket Java, har to av oss lært bruddstykker av et annet fremmedspråk. Krasniqi har nemlig Albansk som morsmål! Irrelevant for prosjektarbeidet, men en viktig del av miljøet innad i gruppen.

9.4 HOVEDKONKLUSJON

Arbeidet med dette prosjektet har gått ut på å lage en applikasjon for å forenkle hverdagen til de ansatte ved HiG. Vi har arbeidet med å gjøre applikasjonen HEVN-J tilgjengelig via mobiltelefon. Resultatet av dette prosjektet er systemet HEVN-M.

Dersom systemet settes i drift vil det gi ansatte mulighet til å lese og oppdatere sin egen timeplan, og i tillegg få vist timeplan og personalia for en valgt ansatt.

Vi har som studenter utviklet oss på forskjellige områder, både faglig og personlig. I tillegg til at vi har fått gode kunnskaper inne den nyeste mobilteknologien, har vi videreutviklet våre programmerings- og systemutviklingskunnskaper.

Det har vært veldig interessant og lærerikt å jobbe sammen i en slik prosjektgruppe i flere måneder. Dette har ført til at vi har fått innsikt i hva som venter oss ute i arbeidslivet.

Kapittel 10 LITTERATUR- OG RESSURSLISTE

Her finnes henvisninger til litteratur og ressurser vi har brukt for å sette oss inn i vårt arbeidsområde. I tilegg til det som er listet opp her, har vi benyttet Internet som generell informasjonskilde utover de linkene som er nevnt her.

Litteraturliste:

1. Jonathan Knudsen, Wireless Java
   ISBN: 1-893115-50-x
2. Michael Morrison, Sams Teach Yourself Wireless Java with J2ME in 21 Days
   ISBN: 0-672-32142-4
3. Harvey M & P.J. Deitel, Java: how to program
   ISBN: 0-1-034151-7
4. Ian Sommerville, Software Engineering
   ISBN: 0-201-39815-X
5. Harald Westhagen, Prosjektarbeid
   ISBN: 82-00-22459-7
6. Bjerke, Brox, Lundbye og Rønning, HEVN (Hevn-Er-hVor-Når), 1998
7. Olsen, Lien, Espeland og Saghaug, HEVN-J (Hevn Er hVor Når Java), 2002

Ressursliste:

Web Services - Wireless Clients:
http://www.javaworld.com/javaworld/jw-08-2002/jw-0823-wireless_p.html

Serializing:
http://www-106.ibm.com/developerworks/webservices/library/ws-soapmap2/

UML:
http://www.modelingstyle.info/

kSOAP:
http://ksoap.enhydra.org

Midlet GUI:
http://www.onjava.com/pub/a/onjava/excerpt/wirelessjava_ch5/index1.html

J2ME:
http://wireless.java.sun.com/midp/ttips/