HOVEDPROSJEKT:

Online rapportering av fangst om bord på Autolinefartøy

Online report of catch on Autoline vessels

FORFATTERE:

OLE PETTER JAHREN	99HINDS
STEIN ERIK NARVESEN	99HINEA
STIG VOLD SAGSTUEN	99HINDP

Dato:

23.05.02

Sammendrag av hovedprosjekt

Tittel:	Onlinerapportering av fangst om bord på autolinefartøy	Nr. : 7 (Data nr)
		Dato : 23.05.02
	Online reporting of catch on Autoline vessels	
Deltakere:	Ole Petter Jahren	
	Stein Erik Narvesen	
	Stig Vold Sagstuen	
Veileder:	Arne Wold	
Oppdragsgiver:	Mustad &Søn A.S	
Kontaktperson:	Geir Liaklev	
Stikkord (4 stk)		

Antall sider: 112	Antall bilag:1 stk CD	Tilgjengelighet (åpen/konfidensiell):åpen

Kort beskrivelse av hovedprosjektet:

Fangstregistrering er et område med globalt sett høy grad av fokus. Mustad har ved hjelp av et produkt de kaller Mustad LineController, fått frem et verktøy som i nær fremtid kan tas i bruk til både å telle fisken og samtidig veie den fortløpende.
Spesielt med tanke på kontrollvirksomhet vil online rapportering av fangst være interessant. Kontrollvirksomhet diskuteres i Norge, og Mustad vil gjerne prøve å være litt i forkant av en slik utvikling.
De ønsker derfor å videreutvikle sin LineContoller slik at de kan få videresendt sine fangstdata via satellitt til en database, hos for eksempel fiskerimyndighetene.
Vår oppgave vil da her bli å finne frem til et egnet sendesystem som kan hente ut fangstdata fra databasen på skipet, samt et mottakersystem som legger de aktuelle dataene inn i databasen på land. Deretter skal da de aktuelle brukerne av systemet ha adgang til sine fangstopplysninger gjennom en internettløsning.

Når vi nå ser tilbake på oppgavedefinisjonen og hva målet var, mener vi at vi har løst oppgaven bra på de fleste områder. Vi mener nå selv at vi har kommet frem til en forholdsvis god prototyp som oppdragsgiveren kan dra nytte av.

Forord

Denne rapporten beskriver arbeid, fremgangsmåter og resultater i forbindelse med et hovedprosjekt som har blitt utført av tre studenter ved høgskolen på Gjøvik. Prosjektet har strukket seg over en tidsperiode på fem måneder fra perioden januar til mai 2002.

Bakgrunnen for dette prosjektet er en henvendelse fra oppdragsgivere Geir Liaklev og Carl Aarnes, fra firmaet O.Mustad & Søn A.S, til høgskolen i Gjøvik. Der gjorde de det klart at de ønsket å lage et online rapporteringssystem av fangst om bord på autoline fartøy. Oppgaven vår er en videre utvidelse av et tidligere prosjekt ved Hig, gjennomført år 2000.

Under gjennomføringen av prosjektet har vi fått hjelp av personer med forskjellig bakgrunn både ved Hig og fra utenforstående. Vi vil rette en stor takk til vår veileder, Arne Wold, Hig, for nyttige råd og støtte. Takker også spesielt firmaet Skantiradio for utlån og veiledning angående bruk av satellitt-telefon av typen mini-M og vår oppdragsgiver for økonomisk støtte, lån av bil, tips og godt samarbeid underveis.

Vi vil i tillegg takke:

Harald Liodden, lærer Hig, (databaser)
Øyvind Kolloen, lærer Hig, (dynamiske websider)
Erlend Efskind, advokatfullmektig, (foreslo kontaktpersoner)
Ellen E. Fasmer, fiskerimyndigheter, (satrap 2.0 til fiskerimyndighetene)
Thomas Hegge, student Grafisklinje ved Hig, (grafisk brukergrensesnitt)

for deres bistand til svar på spørsmål og hjelp med problemer som dukket opp under utviklingen av systemet.

Det finnes en egen hjemmeside for prosjektet og den kan man finne under Høgskolen i Gjøvik sine internettsider. Adressen til prosjektet er:

http://studweb.hig.no/hovedprosjekt/v2002/elektro/online-rapportering/

Gjøvik den 23/05-02

Ole Petter Jahren	Stein Erik Narvesen	Stig Vold Sagstuen

Innholdsfortegnelse

Forord ... 3
Innholdsfortegnelse ... 4
1 Innledning ... 6
1.1 Organisering av rapporten/oppgaven ... 6
1.2 Definisjon av oppgaven ... 7
1.3 Formål ... 7
1.4 Problemområde og avgrensning ... 8
1.5 Målgruppe ... 8
1.6 Egen bakgrunn og kompetanse ... 9
1.7 Arbeidsformer ... 10
1.8 Terminologi ... 10
2 Kravspesifikasjon ... 11
2.1 Visjonsdokument (versjon 1.3) ... 12
Introduksjon ... 12
Posisjonering ... 12
Marked (Business Opportunity) ... 12
Problemer med dagens fangstkontroll ... 12
Produktfortrinn ... 12
Brukermål ... 13
Produktoversikt ... 13
2.2 Use case diagram ... 14
2.3 Use-case-beskrivelser ... 15
USE CASE: LOGGFØRING AV DATA ... 15
USE CASE: UTFØR FEILKONTROLL ... 17
USE CASE: MOTTA FANGSTDATA ... 18
USE CASE: GENERER/SENDER KVITTERING ... 19
USE CASE: FORETA BACKUP ... 19
USE CASE: SJEKKE INNHOLD (FANGST) PÅ WEBSIDE ... 19
USE CASE: KOMMUNISERE MED DATABASESERVER ... 20
USE CASE: SJEKKE BRUKERNAVN OG PASSORD ... 20
USE CASE: LEGG INN BRUKERE/REDERI/SKIP ... 20
USE CASE: OPPDATER BRUKERE/SKIP/REDERI ... 22
USE CASE: SØK ... 23
USE CASE-PRIORITERING ... 24
2.4 Redegjøre for metodebruk ... 25
3 Design ... 26
3.1 Systemarkitektur ... 27
3.2 Databasestruktur ... 28
3.3 Filstruktur ... 30
3.4 Layout ... 31
3.5 Navigasjonsbeskrivelse ... 32
4 Implementering ... 33
4.1 Valg av komponenter til sending ... 34
4.2 Valg av programmeringsspråk og software til rapporteringssystemet ... 35
4.3 Sende/mottakersystemet ... 36
4.4 Web- løsning ... 39
5 Testing ... 41
5.1 Testing av mini-M ... 42

5.2 Testing av dataoverføring med 28.8 modem 43
5.3 Testing av internett sidene 43
6 Diskusjon av resultater 44
6.1 Prosjektperioden for gruppa 45
6.2 Sluttproduktet 45
6.3 Svakheter og forslag til forbedringer ved sende- og mottakersystemet 46
6.4 Svakheter og forslag til forbedringer ved internettløsningen 47
6.5 Oppdragsgiver 48
6.6 Veileder 48
7 Konklusjon 49
8 Referanser 51
9 Vedlegg 52

Vedlegg A: LineController på web 53
Vedlegg B: Installasjonsguide 75
Vedlegg C: Kontakt med myndighetene 99
Vedlegg D: Brev til myndighetene 100
Vedlegg E: Kostnader for å sette prosjektet ut i live 102
Vedlegg F: Plansje over systemet 104
Vedlegg G: Dataoverføringen 105
Vedlegg H: Eksempel på møtereferat fra hovedprosjektmøter 108
Vedlegg I: Metadata 110
Vedlegg J: CRC forklaring 113
Vedlegg K: Eksempel på loggføring av timebruk 114
Vedlegg L: Datablad for mini-M TT3064A 115
Vedlegg M: Kildekode på CD

1 Innledning

1.1 Organisering av rapporten/oppgaven

Kapittel 1 – Innledning
Gir en helt kort introduksjon til prosjektet, og inneholder beskrivelsen av sammenhengen mellom kapitlene, definering av oppgaven, målgruppen og definisjoner av uttrykk brukt senere i rapporten.

Kapittel 2 – Kravspesifikasjon
Her beskrives hva som ble forventet løst i prosjektet ut fra hvilke krav oppdragsgiveren hadde. Denne delen ble utarbeidet etter samråd med Mustad og utviklet gjennom prosjektperioden.

Kapittel 3 – Design
I dette kapitlet så fremhever vi hovedtrekkene av systemet.

Kapittel 4 - Implementering
Begrunner valg av hardwarekomponenter til dataoverføring, programmeringsspråk på websystemet og sendeprogrammet, database og systemutviklingsverktøy. Beskriver også utviklingsmiljøet og filstrukturen.

Kapittel 5 - Testing
Beskriver prosessen hvordan vi utførte tester på både sende/mottakersystemet og på websystemet. Dessuten omhandler det også feil som ble oppdaget og rettet på.

Kapittel 6 – Diskusjon av resultater
Inneholder diskusjon av gruppen, hvordan forholdet til veileder og oppdragsgiver var, og hvordan resultatet av systemutviklingen ble. I tillegg har vi tatt med eventuelle endringer som kan gjøres.

Kapittel 7 – Konklusjon
I fellesskap har vi kommet frem til en konklusjon av prosjektet. Her har vi også belyst temaer som samarbeid og faglig utvikling

Kapittel 8 - Referanser
Inneholder referanseliste (henvisninger) til litteratur, tidskrifter og internettsider som vi har brukt til prosjektet.

Del 9 - Vedlegg
Oversikt over alle vedleggene til rapporten

1.2 Definisjon av oppgaven

Mustad videreutvikler i dag et system som brukes til å lagre fangstdata om bord på Autolinefartøy. Dette systemet heter LineController og er allerede utplassert på en rekke båter. Vår oppgave er å finne frem til et egnet sendesystem som kan hente ut fangstdata fra databasen på en slik farkost, samt et mottakersystem som legger de aktuelle dataene inn i databasen på land. Dette for å ha en kontrollvirksomhet på fangst om bord. Deretter skal da de aktuelle dataene kunne hentes ut på Internett av de aktuelle brukerne av systemet. Disse brukerne kan for eksempel være fiskerimyndighetene, diverse rederier eller skipperen selv. De skal derfor kun ha adgang til sine egne fangstopplysninger gjennom en brukerdefinert inngangsportal. Et brukervennlig websystem vil da være en stor del av oppgaven.
Se også blokkskjema for systemet i vedlegg F, side 102.

1.3 Formål

Visjonen til oppdragsgiveren er at dette systemet skal fungere som et helautomatisk registreringssystem, hvor alt fra vekt og lengde på hver enkelt fisk til, totale fangstmengde og sortering av fiskeslag fortløpende skal bli registrert i databasen om bord på skipet. Fangstfusk har i den senere tid blitt et stort problem innenfor fiskeindustrien [1] og Mustad ønsker derfor å være i forkant av utviklingen av et kontrollsystem. I andre land er det påbudt med en kontrollør om bord, men oppdragsgiveren mener at et slikt system vil kunne erstatte denne.

Emnet er valgt fordi det virket spennende og som en god utfordring. Ingen av oss hadde drevet med noe lignende tidligere og dette gav oss muligheten til å lære noe nytt og utvikle oss selv i en ny retning. Vi skulle få muligheten til å lage noe ingen hadde gjort tidligere.

Det å sende data fra en båt via satellitt til en server på land trodde vi kom til å bli vanskelig og det var flere som stilte seg spørrende til om vi kom til å klare oppgaven. Dette var nok også litt av grunnen til valgt oppgave, siden man skulle få lov til å bevise noe både for seg selv og andre.

1.4 Problemområde og avgrensning

For å kunne løse oppgaven må vi finne ut hvilken sende- og mottakermodul vi skal bruke for overføring av data mellom båtene via satellitt og til en databaseserver. Her finnes det flere typer å velge mellom og vi må gjøre en vurdering ut fra pris, drift, vedlikehold, pc-tilkopling, krypteringsmuligheter, datarate, type signaler, feilprosent og om den lar seg integrere i det eksisterende systemet på Autolinefartøyene. Dette medfører at ulike leverandører må kontaktes og en må sammenligne de forskjellige leverandørenes tilbud.

Vi ser for oss bruk av telenettet (satellittelefon) med eller uten bredbånd for overføring av data, og vår oppgave blir da å finne en sammensetting av de rette komponentene ut fra oppdragsgivers ønsker. Vi ser for oss toveiskommunikasjon fra båt til database på land slik at vi kan få bekreftelse av mottatt sending og feilkontroll. Hyppigheten på transaksjonen blir sannsynligvis en gang om dagen, men med mulighet for flere overføringer dersom det skulle vise seg nødvendig.

Databasen på serveren må lages og denne skal være tilgjengelig både for rederiet og eksterne fiskerimyndigheter. Dataene vi skal sende ligger ferdig lagret i databasen på båten. De data som skal sendes er turnummer, stubbnummer, settetid, haletid, antall kroker (stk), egne%, fangst kg, fangstrate, kg/krok og dagbok. Disse dataene vil være interessante for rederiet, men myndighetene vil trolig kun være interessert i de fangstrelaterte dataene. For å løse dette ser vi for oss en presentasjon av databasen på en webside med ulike grensesnitt.

Vi må sjekke om alle faktiske data kommer fram med muligheter for feilkontrollering. For at dette skal fungere må vi ha en form for loggføring av databasen. Tanken er at dataene blir loggført før transaksjonen gjennomføres. Vi må undersøke hvilket nivå av sikkerhet som kreves rundt en slik transaksjon av data, og sørge for at sikkerhetsnivået er tilfredsstillende slik at uvedkommende ikke får tilgang til databasen.

Vi må undersøke om det er dekning for telenettet overalt der båtene ferdes, eller om data kun kan sendes i visse områder.

Myndigheter må kontaktes for å undersøke interessen for et slik system og om eventuelle krav som må oppfylles. Eksempelvis om databaseserveren må ligge sentralt hos fiskerimyndighetene. Spørsmålet er om systemet kan erstatte den nåværende kontrolløren ombord på båtene.

1.5 Målgruppe

Rapporten er beregnet på de som skal bruke systemet , det vil si O.Mustad & Søn og deres utviklere, og noen av sidene forutsetter et visst datateknisk kunnskapsnivå og kjennskap til det eksisterende LineControllerprogrammet. Likevel mener vi at den vil gi utenforstående et innblikk i prosjektets gang, samt hvordan vi har resonnert og kommet fram til vårt resultat.

1.6 Egen bakgrunn og kompetanse

Gruppesammensetningen er litt spesiell siden vi var sammensatt fra tre forskjellige studieretninger ved Hig:

Ole Petter Jahren: dataingeniør, retning systemutvikling.
Relevant erfaring til prosjektet:

Databaser1
Databaser2
Litt erfaring med html som var en liten del av faget programmering mot www
Systemutvikling1
Systemutvikling2
Datakommunikasjon

Stig Vold Sagstuen: dataingeniør, retning programutvikling.
Relevant erfaring til prosjektet:

Grafiske brukergrensesnitt (Delphi)
Databaser1
Objekt-orientert programmering
Systemutvikling1

Stein Erik Narvesen: elektroingeniør, retning telematikk.
Relevant erfaring til prosjektet:

Telekommunikasjon
Diverse programmeringsfag som er standard på ingeniørutdanningen

Hva måtte læres:

- Php (hypertext preprocessor) programmeringsspråk for å lage dynamiske websider, var en del av valgfaget klient og serversiderprogrammering som begynte samtidig som hovedprosjektet
- Valg av servere, hvordan sette dem opp mot valgt database
- Eksisterende produkt og kode
- Delphi komponenter til bruk for sending
- Mysql, og Interbase databaser
- Javascriptfunksjoner
- Kryptering
- CRC- sjekk
- Hardwarekomponenter til bruk for sending (mini M satellittelefon) og mottak (modem)
- Verktøy for å lage websider med php (Macro media homesite 5)
- Style sheet
- Hayes AT- kommandoer
- Bruke bilderedigeringsprogrammet Photoshop6.0

1.7 Arbeidsformer

Det meste av arbeidet har foregått som gruppearbeid på elektrolabben i L-bygget på Hig, der vi tidlig satte opp to eksterne pc'er til bruk for sending og mottak av data. Arbeidsfordeling ble fordelt slik at en hadde ansvar for å finne frem til passende hardwarekomponenter, ta kontakt med leverandører og stå for den tekniske delen ved oversendingen. Samme person har også hatt oppgaven med å ta kontakt med myndighetene og undersøkt interessen for et slikt kontrollsystem.

Gruppemedlem nummer to hadde ansvaret for programmeringsdelen i delphi og den tredje skulle ta seg av de dynamiske websidene. Arbeidsoppgavene var likevel så nært knyttet at en måtte hele tiden holde seg oppdatert på hva de andre gjorde, for å få tilpasset de ulike delene i systemet til hverandre.

Til å begynne med fulgte vi utviklingsmodellen RUP (Rational Unified Process). Denne modellen var fundamentet i faget systemutvikling2 som en av gruppemedlemmene hadde tatt, og hadde god erfaring med i fra tidligere prosjekter. Etter hvert gikk vi over til evolusjonær utvikling og grunnen skal forklares senere.

I starten hadde vi ukentlige møter med oppdragsgiver og vår veileder, både i Mustad sine lokaler og på Hig. Etter hvert som vi fikk klarlagt store deler av systemet tok vi kontakt når vi hadde spørsmål eller ville få tilbakemelding på det vi hadde gjort.

1.8 Terminologi

- Alle betegnelser om data er de fangstdataene som skal overføres, sånn som turnummer, stubbnummer, settetid, haletid, antall kroker (stk), egne%, fangst kg, fangstrate, kg/krok og notat.

- Ved betegnelser som oversending og transaksjoner menes det sending av disse dataene.

- Brukerdefinert tilgang på websidene vil si at en administrator setter rettighetene til de forskjellige brukerne av systemet.

- Ord som sender, klient og avsender refererer til LineControllerprogrammet om bord på båten. På samme måte er server, mottaker og serverdatabase det samme som mottakerprogrammet.

- Når vi beskriver systemets administrator, så er det etter vår mening fiskerimyndighetene

De forskjellige synonymene er brukt for å få litt variasjon i språket.

2 Kravspesifikasjon

Her beskrives hva som ble forventet løst i prosjektet ut fra hvilke krav oppdragsgiveren hadde. Vi har da gått litt i dybden og tatt for oss alle tenkelige områder angående systemet. Dette kan være alt fra marked til brukermål. Vi har også gjort en analyse av hendelser rundt systemet (use case).

2.1 Visjonsdokument (versjon 1.3)

2.2 Use case diagram

2.3 Use-case-beskrivelser

2.4 Redegjøre for metodebruk

2.1 Visjonsdokument (versjon 1.3)

Introduksjon

Vi har som mål å finne ut hvordan satellittoverføring av ferdiglagrede data fra en database på båt til en databaseserver på land skal forgå. En del av oppgaven blir å finne komponenter som kan integreres mot det eksisterende produktet Mustad LineController som finnes i båtene. Dette er et produkt som registrerer fangstdata automatisk i en database og det er disse dataene vi skal sende. Dette medfører at det må opprettes en databaseserver for mottak av oversendte data. Her må det foregå en feilkontroll slik at man kan forvisse seg om at de faktiske data har kommet fram og er konsistente. Det blir også viktig med sikkerhetshensyn slik at u-vedkomne ikke får tilgang til eller kan manipulere databasen. Hvert rederi vil få tildelt rettigheter på en webside til fangstdata på egne skip, og et skip vil få tilgang til sine data. Selve databasen blir (sannsynligvis) liggende hos fiskerimyndighetene og de vil ha tilgang til alle skipsrapporter.

Posisjonering

Marked (Business Opportunity)

Markedet antas å være betydelig sett i et globalt perspektiv. I dag er det blant annet påbudt med fiskekontrollør på fangstbåtene i USA og det er snakk om at andre land kommer til å få påbudet. Man bør heller ikke se bort i fra muligheten om at et slikt påbud også kan komme til Norge. Hensikten med systemet er å virke som et kontrollorgan isteden for kontrolløren som blant annet er påbudt om bord på fangstbåter i USA.

Etter det vi har fått opplyst finnes det i dag et lignende system for en slik kontroll, men dette systemet er ikke helautomatisk. Det er skipperen som taster inn fangstdataene og selv velger tidspunkt for transaksjonen.

Problemer med dagens fangstkontroll

I de land der det er påbudt med fangstkontrollør koster dette en del. Kontrolløren må i tillegg til lønn få kost og losji. Båtene er gjerne ute flere uker av gangen og dette medfører en relativt treg rapportering.

Produktfortrinn

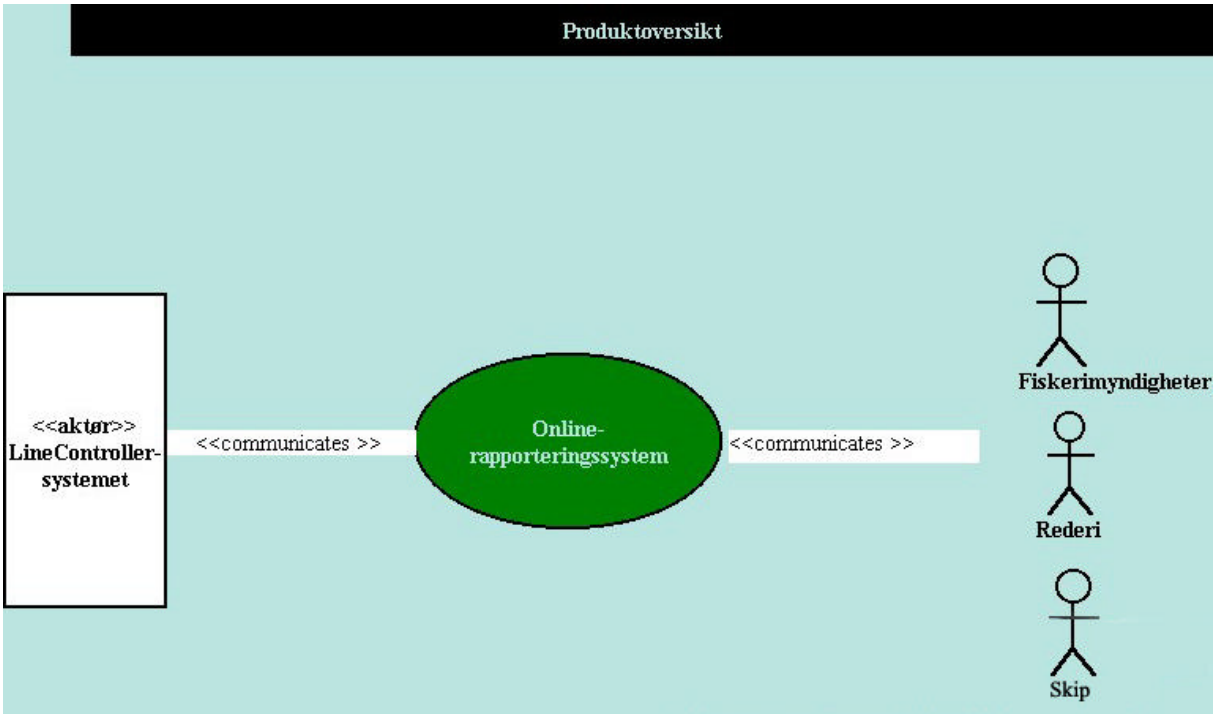
Utarbeidelsen av et sendesystem kombinert med det allerede eksisterende Mustad LineControllersystemet og en serverdatabase på land, vil kanskje kunne erstatte fangstkontrolløren og være kostnadsbesparende. Det vil samt gi en mer effektiv rapporteringsmulighet.

Man vet også at det kan foregå en del snusk og bestikkelser mellom farkoster og fangstkontrollører, slik at fangsten ikke blir riktig bokført. Dette vil kunne bli unngått ved bruk av det nye systemet.

Brukermål

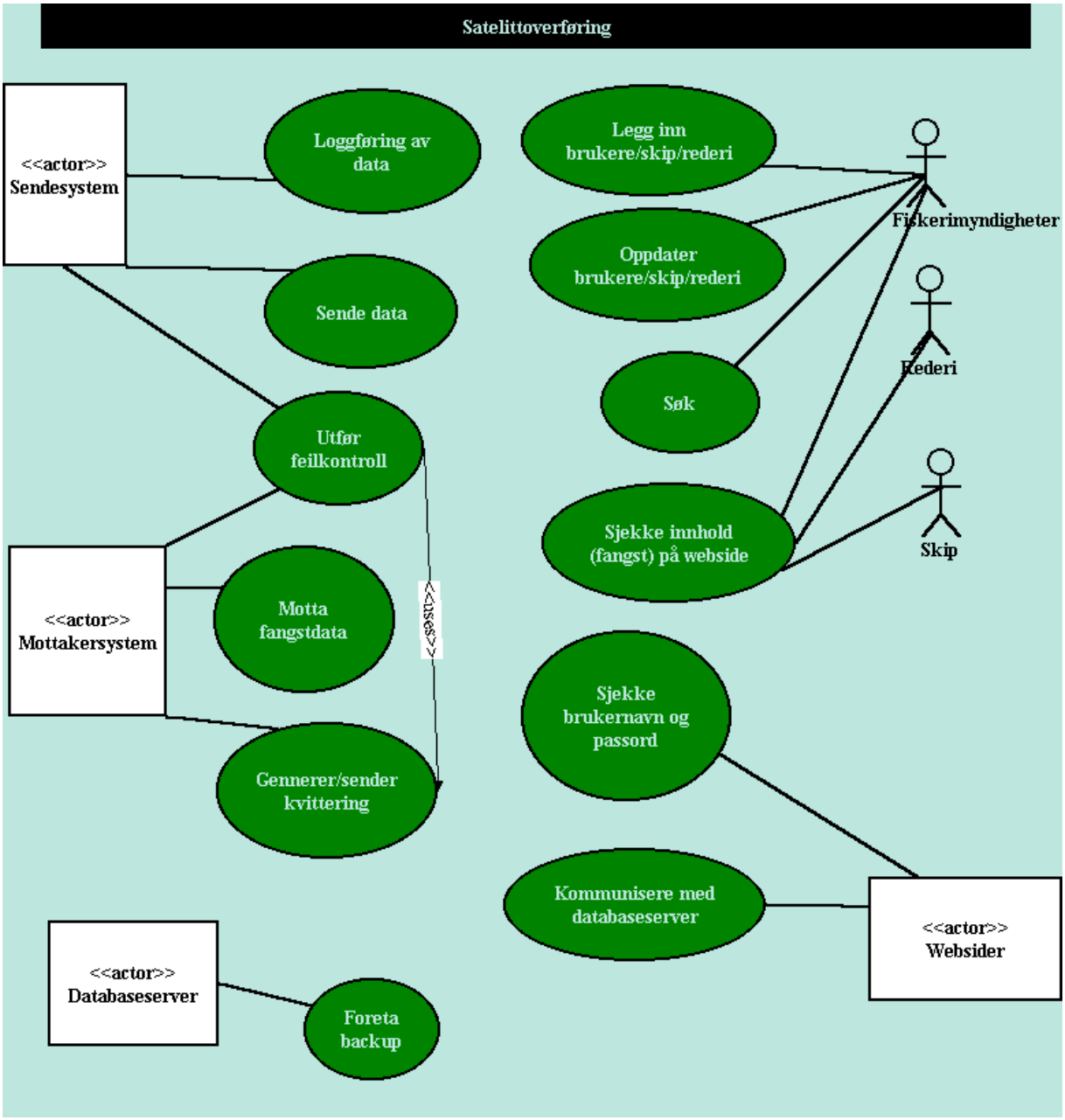
Sendesystem:	få kontakt med satellitt, sende relevante fangstdata til database på land med kryptering og crc- sjekk, motta kvittering for vellykket transaksjon, eller beskjed om omsending
Databaseserver på land:	decryptere, utføre feilsjekk på mottatte data fra databaser i forskjellige skip, generere/sende kvittering, organisere fangstdata
Fiskerimyndigheter:	kontrollere fangstdata i database til forskjellige fartøyer, administrative oppgaver som å legge inn nye brukere, skip, rederi og foreta oppdateringer/endringer
Rederi:	kontrollere egne skips fangstdata
Webside:	generere automatisk oppdateringer på fangstsidene

Produktoversikt



Figur 2.1: Produktoversikt

2.2 Use case diagram



Figur 2.2: Use Case diagram

2.3 Use-case-beskrivelser

USE CASE: LOGGFØRING AV DATA

High-level	
Aktør:	Sendesystem
Type:	Primær
Beskrivelse:	Hver gang data skal sendes fra båt til serverdatabase loggføres transaksjonen i en tekstfil på pc'ens harddisk før sending.
Low-level	
Aktør:	Sendesystem
Hensikt:	Ved å lage en logg kan man gå tilbake og finne ut hvilke data som skulle vært sendt dersom systemet krasjer eller annen feil skulle inntreffe.
Oversikt:	Ved hver sending skrives dato og klokkeslett og alle dataene være seg eksempelvis fangstnr, line, GPS koordinater, fangst i antall kg osv, til en tekstfil på harddisken. Dersom det skulle skje en feil under sendingen blir det skrevet en feilmelding i loggen.
Prebetingelse:	Det er blitt trykket SAVE i LineControllerprogrammet og satt timer har endt sin nedtelling.
Postbetingelse:	Funksjonen for å skrive til fil er blitt tilkalt og virker.

Aktørhendelser	*Systemhendelser*
1. En stubb blir halt (skipper trykker save)	
	2. Data om stubb blir lagret
	3. Forbindelse blir opprettet
	4. Data hentes ut fra database
	5. Data skrives til logg

Tabell 2.1: Happy day scenario for use case loggføring av data

USE CASE: SENDE DATA

High-level	
Aktør:	Sendesystem
Type:	Primær
Beskrivelse:	Data sendes automatisk når en stubb er halt.
Low-level	
Aktør:	Sendesystem
Hensikt:	Skal kunne oversende fangstdata for hver stubb båten har halt opp.
Oversikt:	Lagringen av data om en stubb skjer i 3 etapper i LineController. Skipper trykker på ”start logging” knappen, deretter på ”stop logging” knappen og når han til slutt trykker på Lagre/Save så er alle dataene om en bestemt stubb lagret. Feltet sendt_sjekk blir da satt til 1 og det settes en timer på 5 sekunder før dataene sendes til databaseserveren på land. Dersom sendingen ikke går i orden forblir sendt_sjekk lik 1.Dataene blir forsøkt sendt på nytt når kvittering om at CRC-sjekken ikke stemmer har ankommet. Ved vellykket sending blir sendt_sjekk satt til 2. CRC-rutinen er nærmere beskrevet i vedlegg J ,side 113. Gangen i datasendingen er beskrevet nærmere i vedlegg G, side 105.
Prebetingelse:	Det er blitt trykket SAVE i LineControllerprogrammet og timeren har telt ned.
Postbetingelse:	Får kontakt med satellitt og mottaker.

Aktørhendelser	*Systemhendelser*
	1. Data om stubb blir lagret 2. Timer blir satt 3. Timer telles ned 4. Kontakt med mottaker via satellitt opprettes 5. Data blir oversendt

Tabell 2.2: Happy day scenario for use case å sende data

Extensions

4
a) Får ikke kontakt med satellittelefon
Det gjøres forsøk på å initialisere startvariable for satellitten for å gjøre den klar for sending. Dersom dette ikke lar seg gjøre så får skipperen en beskjed om å sjekke at satellittelefonen er skrudd på. Deretter prøves det igjen.

b) Får kontakt med satellittelefon, men ikke satellitt .
Systemet håndterer dette som om det skulle vært ett opptattsignal eller andre årsaker til at forbindelsen ikke kunne opprettes. Det prøves igjen til forbindelse er opprettet.

c) Får ikke opprettet forbindelse med mottakersystemet (opptattsignal).
Systemet får melding om at det er opptatt og prøver igjen med jevne intervaller.

USE CASE: UTFØR FEILKONTROLL

High-level	
Aktør:	Mottakersystem, Sendesystem
Type:	Primær
Beskrivelse:	Man gjør en sammenligning av sendte og mottatte data, for å kunne forvisse seg om at det ikke har oppstått noen feil under transaksjonen.
Low-level	
Aktør:	Mottakersystem, Sendesystem
Hensikt:	Unngå feilaktig rapportering.
Oversikt:	Før data blir sendt til mottaker, kjøres det en CRC-sjekk på alle dataene. CRC-sjekken sendes over som en bokstav helt til slutt i strengen som skal overføres. Når dataene kommer frem blir det kjørt en ny CRC-sjekk på dataene. Denne sjekken må testes mot den verdien som ble medsendt. Hvis verdiene stemmer overens vil dataene bli lagret i databasen, kvittering blir generert og sendes til avsender. Dersom sammenligningen slår feil, blir det bedt om omsending.
Prebetingelse:	Man har fått kontakt med mottakersystemet.
Postbetingelse:	Har fått opprettet forbindelse fra båt til mottakersystemet.

Aktørhendelser	*Systemhendelser*
	1. Data mottas av mottaker 2. CRC-sjekk kjøres på dataene 3. CRC-verdiene blir sammenlignet 4. Kvittering blir generert 5. Kvittering sendes til avsender

Tabell 2.3: Happy day scenario for use case utfør feilkontroll

USE CASE: MOTTA FANGSTDATA

High-level

Aktør:	Mottakersystem
Type:	Primær
Beskrivelse:	Mottakersystem tar i mot sending av data fra de forskjellige båtene.

Low-level

Aktør:	Mottakersystem
Hensikt:	Mottar fangstdata via satellitt fra forskjellige båter for å kunne lagre dataene i databasen. Ved å sjekke på innkommende skipsnummer blir fangsten lagt under tilhørende skip. En vil senere kunne logge seg på websiden som administrator, tilhører av gjeldene rederi eller skip for å sjekke fangsten. En bør merke seg at må det ligge et unikt nr i hver LineController som knytter fangsten til båten og båten knyttes igjen til rederi.
Oversikt:	Programmet i mottakersystemet lytter på meldinger fra COM- porten og når ett ringesignal oppdages så opprettes det en forbindelse ved bruk av Hayes At- kommandoer. Dataene blir motta og verifisert med CRC-sjekk og lagres hvis de ikke inneholder feil. Forbindelsen blir deretter brutt.
Prebetingelse:	Det har blitt trykket SAVE i LineControllerprogrammet og timeren har blitt telt ned
Postbetingelse:	Sendingen blir ikke avbrutt og CRC-sjekken stemmer overens.

Aktørhendelser	*Systemhendelser*
1. Sendesystemet ringer opp mottaker	
	2. Mottakersystemet svarer
3. Sender over data	
	4. Mottar fangstdata

Tabell 2.4: Happy day scenario for use case motta fangstdata

Extensions

1
a) Mottakersystemet er opptatt
Sendesystemet prøver å opprette forbindelse igjen med ett gitt tidsintervall, ca 30 sekunder, helt til en forbindelse er opprettet

2
a) Brudd i sending
Use caset feilkontroll tar over.

USE CASE: GENERER/SENDER KVITTERING

High-level	
Aktør:	Mottakersystemet
Type:	Primær
Beskrivelse:	På grunnlag av "USE CASET UTFØR FEILKONTROLL" lager mottakersystemet en kvittering for mottatte data. Den sender enten CRC_OK eller CRC_FAILED.

USE CASE: FORETA BACKUP

High-level	
Aktør:	Databaseserver
Type:	Primær
Beskrivelse:	Det blir tatt backup av databasen ved at det skrives til en eksakt lik database på en annen disk hver gang det gjøres endringer.

USE CASE: SJEKKE INNHOLD (FANGST) PÅ WEBSIDE

High-level	
Aktør:	Administrator, Rederi, Skip
Type:	Primær
Beskrivelse:	Fiskerimyndigheter skal ha tilgang til all fangstdata, et rederi til sine egne skip og et skip til egen fangst.
Low-level	
Aktør:	Administrator, Rederi, Skip
Hensikt:	Ved å logge seg inn på websidene som administrator vil en kunne sjekke fangstdata til ethvert skip. Systemet virker da som et kontrollorgan og de kan ta utskrifter av fangstdataene. Et rederi er interessert i sine egne skips fangstdata og ta uskrift av disse. Et skip vil kontrollere sine fangstresultater.
Oversikt:	Fangstdataene er organisert etter når det ble lagt inn i databasen, og brukerne av systemet kan velge mellom fire alternativer for å hente ut fangsten: Alle, dagens, 7 dager tilbake og 14 dager tilbake.
Prebetingelse:	Logget inn i systemet.
Postbetingelse:	Det må ligge fangstdata inne.

USE CASE: KOMMUNISERE MED DATABASESERVER

High-level	
Aktør:	Webside
Type:	Primær
Beskrivelse:	Websiden kommuniserer med databaseserveren med støtten som ligger innebygd i php En kan da kjøre sql- spørringer direkte mot databasen og på denne måten leses og skrives det.

USE CASE: SJEKKE BRUKERNAVN OG PASSORD

High-level	
Aktør:	Webside
Type:	Primær
Beskrivelse:	Websiden skal ha en innloggingsmeny der det må oppgis brukernavn og passord. Brukernavnet er unikt og ved inntasting av rett passord blir brukernavnet sjekket opp mot hvilken tilgang brukeren har. Brukeren blir da sendt til den siden tilgangen gjelder for. Ved inntastning av feil passord får man beskjed at brukernavn eller passord er feil.

USE CASE: LEGG INN BRUKERE/REDERI/SKIP

High-level	
Aktør:	Fiskerimyndigheter
Type:	Primær
Beskrivelse:	Administratorer (ansatt i/av fiskerimyndigheter) vil kunne logge seg på websiden og legge inn nye brukere, rederi og skip.

Low-level	
Aktør:	Fiskerimyndigheter
Hensikt:	Kunne legge inn brukere, rederi og skip i et GUI bedre tilpasset enn Interbase sin IBConsole.
Oversikt:	For å kunne legge inn noe nytt må administratoren fylle ut en html-form og velge alternativ der det er rullgardiner/”dropdownboxes”. Det blir da gjort forskjellige feilsjekker om inntastede verdier er gyldige, avhengig om en skal legge inn ny bruker, rederi eller skip.
Prebetingelse:	Logget seg på systemet med et administrator brukernavn og passord.
Postbetingelse:	Fylt inn alle felter korrekt og trykket lagre knappen (valgt et alternativ i rullgardinen for tilgang ”bruker”).

Aktørhendelser	*Systemhendelser*
1. Klikker på linken *ny* for skip 2. Velger det rederi skipet tilhører (må ha blitt lagt inn først) 3. Skriver inn navnet på skipet 4. Fyller ut byggeår 5. Skriver navnet på kontaktperson 6. Fyller inn telefonnummer med retningsnummer først 7. Trykker lagreknappen	
	8. Sjekker om det er valgt et rederi 9. Sjekker om feltene navn, byggeår, kontaktperson og telefonnr er fylt ut. 10. Tester om telefonnummeret består av 11 numeriske tegn 11. Undersøker om byggeåret består av 4 tall, er nyere enn fra 1930 og ikke nyere enn dagsdato. 12. Dersom programmet ikke finner noen feil hentes siste skipsnr i databasen og det plusses på 1 13. Gjør om input fra navn og kontaktperson slik at stavelsen BjArNe mOrSoM blir lik Bjarne Morsom 14. Legger inn dataene under det nye skipsnummeret. 15. Gir beskjed om at skipet ”navn” er lagt inn.

Tabell 2.5: Happy day scenario for use case legge inn skip

Extensions
2
a) Finner ikke riktig rederi.
Finner man ikke rederiet betyr dette at rederi ikke er lagt inn i databasen. Dette må da legges inn først.

8 – 11
a) Finner feil
Skriver ut feilmeldingen(e) i rødt på skjermen uten å lagre noe i databasen.

USE CASE: OPPDATER BRUKERE/SKIP/REDERI

High-level	
Aktør:	Fiskerimyndigheter
Type:	Primær
Beskrivelse:	Administrator kan gå inn å gjøre oppdateringer på brukere, rederi og skip
Low-level	
Aktør:	Fiskerimyndigheter
Hensikt:	Det kan oppstå forandringer hos et skip som for eksempel at det skifter navn, selges til et annet rederi, bytter kontaktperson eller telefonnummer. Et rederi kan skifte lokaler og i noen tilfeller kan det være nødvendig å gjøre om på et brukernavn. Det kan da være lurt å ha en man side man kan gjøre slike oppdateringer.
Oversikt:	Administrator velger hvilket skip, rederi eller hvem brukeren som skal oppdateres og fyller inn de feltene en måtte ønske å endre på og trykker oppdater.
Prebetingelse:	Logget seg på systemet med et administrator brukernavn og passord, databasen er ikke tom.
Postbetingelse:	Fylt inn alle felter korrekt og trykket oppdater knappen

Aktørhendelser	*Systemhendelser*
1. Klikker på linken *oppdater* for rederi	
	2. Henter ut liste over alle rederier og legger den i rullgardinen
3. Velger i rullgardinen hvilket rederi man skal oppdatere	
4. Skriver inn endringer i de feltene som skal oppdateres	
5. Trykker *oppdater*	
	6. Sjekker om det har blitt skrevet inn noen ugyldige verdier i formen.
	7. Lagrer endringer av valgt rederi

Tabell 2.6: Happy day scenario for use case oppdater skip/rederi

Extensions

6
a) Finner feil
Feilmeldingen(e) blir skrevet med rødt i toppen av siden, blir ikke lagret noen data i databasen før feilene er rettet opp.

USE CASE: SØK

High-level	
Aktør:	Fiskerimyndigheter
Type:	Primær
Beskrivelse:	Administrator kan gjøre et søk i databasen på skip, rederi og bruker, der søket baserer seg på navn.
Low-level	
Aktør:	Fiskerimyndigheter
Hensikt:	Ved å taste inn en forbokstav eller deler av starten på et navn, kan man fort finne ut om skipet, rederiet eller brukeren er registrert i systemet.
Oversikt:	Ved å klikke på sjekkboksen foran et skip, rederi eller bruker (eller alle tre på en gang), velger man hvilken tabell man vil søke i. Ved søk i skip og rederi blir forbokstaven i søket gjort om til stor bokstav. Dette er fordi både skips- og rederinavn blir lagret med stor forbokstav og Interbase skiller mellom store og små bokstaver. Skal man søke etter en bruker derimot, blir ikke forbokstaven gjort om siden brukernavnet er unikt og blir ikke lagret på samme format som navn på skip og rederi. Brukernavnet kan ha enten stor eller liten forbokstav.
Prebetingelse:	Logget på systemet med administrator rettigheter.
Postbetingelse:	Ligger data i databasen.

Aktørhendelser	*Systemhendelser*
1. Klikker på linken søk	
	2. Henter fram søkealternativer
3. Velger hva man vil søke etter ved å klikke på en eller flere av sjekkboksene.	
4. Skriver inn søkeordet og trykker på *søk* knappen	
	5. Sjekker hvilke sjekkbokser som har blitt huket av
	6. Gjør om søkeordet slik at forbokstav blir gjort til stor bokstav i søket for skip og rederi, før sql- setning kjøres.
	7. Henter resultat

Tabell 2.7: Happy day scenario for use case søk

Extensions

5
a) Finner ingen markert sjekkboks
Gir feilmelding om at man må velge hva man vil søke etter.

7
a) Finner ikke noe resultat
Det blir skrevet ut hva man søkte på og antall treff : 0 på skjermen.

USE CASE-PRIORITERING

A) Risk – teknisk kompleksitet
B) Usability – betydningen av brukervennlighet
C) Coverage – alle systemets deler skal berøres i tidlige iterasjoner,
bred implementasjon over flere komponenter (informasjon og innsikt i design).UC som bør berøres tidlig/grunnlaget for løsningen!
D) Criticality – funksjoner med høy forretningsmessig verdi
E) Betydning for systemets arkitektur

Vi bruker en skala fra 1 til 3, hvor 3 er viktigst/mest kritisk

Kategori	Vekt	Skala
A	3	1-3
B	2	1-3
C	1	1-3
D	3	1-3
E	2	1-3

Use Case	A	B	C	D	E	Total
Loggføring av data	1	1	1	2	1	14
Sende data	3	1	3	3	2	25
Utfør feilkontroll	2	1	1	3	2	22
Motta fangstdata	3	1	3	3	2	25
Generer/sender kvittering	2	1	1	3	2	16
Foreta backup	1	1	1	2	1	14
Legg inn brukere/skip/rederi	2	3	2	2	2	24
Oppdater brukere/skip/rederi	2	3	2	2	2	24
Søk	1	2	1	1	1	13
Kommunisere med databaseserver	1	3	3	3	2	25
Sjekke innhold (fangst) på webside	1	3	3	3	2	25
Sjekke brukernavn og passord	1	2	1	2	1	16

Tabell 2.8: Use case prioritering

1. Sende data	7. Utfør feilkontroll
2. Motta fangstdata	8. Generer/sender kvittering
3. Kommunisere med databaseserver	9. Sjekke brukernavn og passord
4. Sjekke innhold (fangst) på webside	10. Foreta backup
5. Legg inn brukere/skip/rederi	11. Loggføring av data
6. Oppdater brukere/skip/rederi	12. Søk

Tabell 2.9: Prioritert use case liste

2.4 Redegjøre for metodebruk

Som nevnt startet vi med RUP (Rational Unified Process) som vår utviklingsmodell og bruk av UML (Unified Modelling Language). Gantt- skjemaet ble planlagt etter modellen og i startfasen så det ut til å virke bra. Vi lagde tidlig et visjonsdokument og lagde en del use case og use case beskrivelser. (Det må nevnes at use case ikke er særegent for RUP, men er noe Ivar Jacobsen tidligere hadde utviklet og ble senere tatt med som en nyttig del av utviklingsmodellen og utviklingsverktøyet UML.

Det skulle vise seg at det var alt for mye vi ikke hadde kjennskap til, når det gjaldt sammensetningen av komponenter i delphi eller bruk av php som programmeringsspråk. Vi måtte først finne ut hvordan komponentene virket og hva både språk og komponenter hadde å tilby av innebygd funksjonalitet.

Vårt arbeid ble mer og mer likt evolusjonær utvikling eller ”code and fix”, mye grunnet den dårlige hjelpen det var å finne på delphi- komponentene. En måtte kode og så teste for å se hva som ble utfallet. Samme var det med php- programmering, der en stadig fant ut nye ting og innbygde funksjoner som kunne erstatte gammel kode. Det var umulig å vite hvilke parametere som skulle bli sendt hvor, men slik er det nok når man starter med blanke ark.

Dersom vi skulle ha utviklet et lignende system i framtiden, hadde det nok vært lettere å følge RUP- modellen.

3 Design

I dette kapitlet så fremhever vi hovedtrekkene av systemet.

3.1 Systemarkitektur

3.2 Databasestruktur

3.3 Filstruktur

3.4 Layout

3.5 Navigasjonsstruktur

3.1 Systemarkitektur

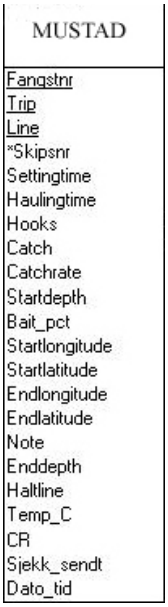
Vår oppgave gikk som tidligere nevnt ut på å få sendt data fra en database om bord på et LineController fartøy, og inn til en databaseserver på land. Disse dataene skulle da senere legges ut på internett med en brukerdefinert tilgang.
Ettersom det eksisterende systemet var bygget opp med Delphi komponenter, så var det naturlig å bygge videre på det når vi skulle programmere mot sende- og mottakerdelen. Når det gjelder hardwaren på sendedelen, så var vi nødt til å finne en komponent som håndterer satellittoverføring og som samtidig tillot oss å ha en åpen linje til mottakeren. Valget falt på å bruke en satellittelefon som kunne fungere som et modem. Dette vil være den beste måten å overføre data på, da vi lett kan foreta feilsjekking av overførte data og eventuelt sende datapakken på nytt. Standard AT- kommandoer har da blitt implementert i koden på LineControlleren og den skal også kunne fungere på andre modemer enn det vi har testet på og anbefalt.

Når det gjelder webløsningen så var vi også nødt til å ta utgangspunkt i den eksisterende databasen. Alle dataene til det eksisterende systemet ble lagret i en Interbase database. Vi måtte da finne et egnet språk som kunne kommunisere med denne databasen. Her har vi valgt å bruke programmeringsspråket Php.

Alt dette er nærmere beskrevet i kapittel 4.

3.2 Databasestruktur

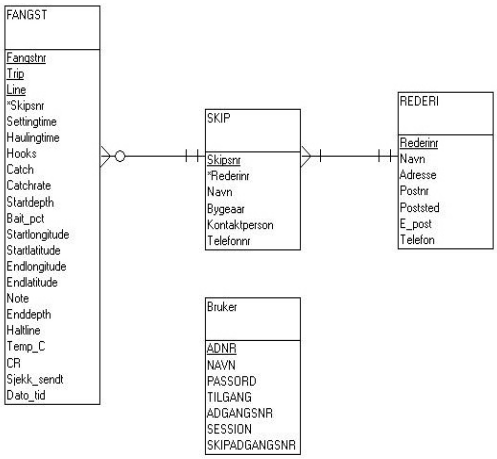
Vi har valgt å bygge videre på den eksisterende databasen til Mustad i vårt sendesystem se Figur 3.1.
Strukturen på denne databasen er jo meget enkel. En tabell som inneholder 3 primærnøkler. Disse er Fangstnr, Trip og Line.



Figur 3.1: Database om bord på båten

Den fullstendige databasen på mottakerdelen, det vil si serveren til relevante myndigheter er vist i Figur 3.2.

Denne inneholder 3 tabeller som er knyttet opp mot hverandre ved hjelp av fremmenøkler og èn tabell som ikke har noen direkte kobling mot de andre tabellene. Databaseintegriteten er vedlikeholdt på følgende måte.
Tabellen FANGST har 3 primærnøkler(Fangstnr, Trip og Line). Den har èn fremmednøkkel(Skipsnr). Skipsnr er primærnøkkel i tabellen SKIP. Dette vil si at en post ikke kan registreres i tabellen FANGST uten at fremmednøkkelen Skipsnr refererer til en verdi som faktisk finnes i tabellen SKIP.
Tabellen SKIP har fremmednøkkelen Rederinr som refererer til tabellen REDERI der dette feltet er primærnøkkel. Dette vil da si at en post ikke kan registreres i tabellen SKIP uten at fremmednøkkelen Rederinr refererer til en verdi som faktisk finnes i tabellen REDERI.
Dette gjør at databasen er korrekt rent logisk. Ett skip må jo nødvendigvis tilhøre et rederi og en fangst må jo tilhøre ett spesifikt skip.



Figur 3.2: Oppsett av database på serversiden ved hjelp av modulator

Mer utfyldig om databasene kan leses i kapittel 4.

3.3 Filstruktur

På sendedelen så er filene lagt inn i LineController systemet og vi følger den strukturen som er valgt fra før. Det er for så vidt ingen filstruktur å snakke om, fordi alle filene er samlet i en mappe. Alt vi har gjort av forandringer på det eksisterende systemet, er å legge til ekstra kodelinjer. Det vil si prosedyrer/funksjoner for å håndtere overføringen av data.

På webdelen ligger alle filene i rotkatalogen som var default for apacheserveren.
C:\Program Files\nusphere\apache\nsdocs Se vedlegg B installasjonsguide side 75.

Alle filene er merket med forstavelser avhengig av om de tilhører administrator- rederi- eller skipssider.

Eksempel på administratorfiler med forstavelsen *ad*:

-adhentefangst.php
-adslettbruker.php
-adsok.php

Eksempel på rederifiler med forstavelsen *red*:

-redskipoversikt.php
-redskipendrepassord.php
-redskiptop.php

Eksempel på skipsfiler merket forstavelsen *skip*:

-skiphentefangst.php
-skipendrepassord.php
-skiptop.php

For å se filene i sin helhet, så kan man åpne dem i den vedlagte CD'en. (Vedlegg M)
Vi har valgt å sende med kildekoden på CD fordi det rett og slett hadde vært for mye å skrive ut.

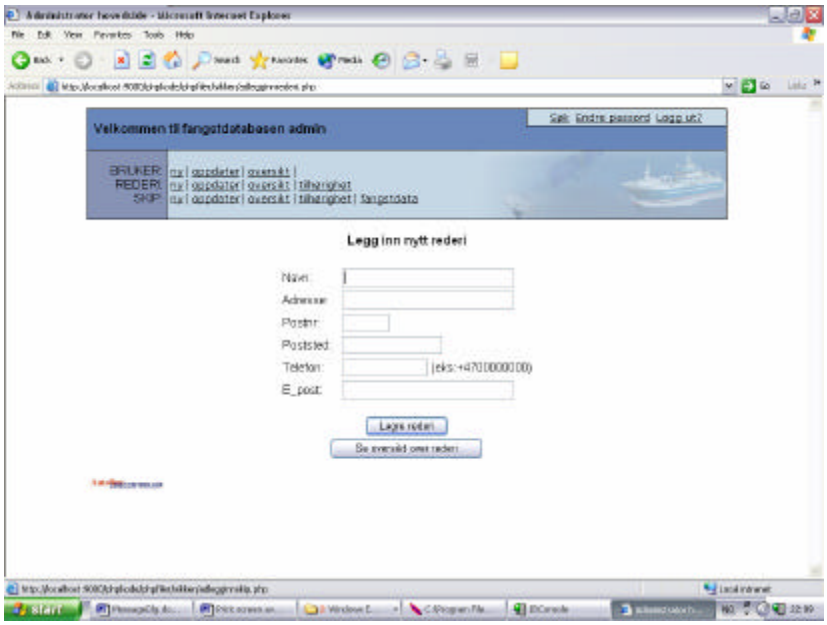
3.4 Layout

Det grafiske brukergrensesnittet ble ikke prioritert i starten av prosjektet. Da la vi heller hovedtyngden av arbeidet på å få et robust og brukervennlig system. Men etter hvert som vi nærmet oss slutten, så ville vi også prøve oss med layouten. Ettersom vi ikke har så stor erfaring med grafikk og bildebehandling, så fikk vi her litt råd og veiledning av en student på grafisk ingeniørutdanning.

Viktige punkter som vi har vurdert med tanke på det grafiske brukergrensesnittet :

- Lettforståelig
- Navigasjonsvennlig
- Ikke ”skrikende” farger
- Gjennomgående likhet på de forskjellige sidene

Eksempel på en side på webløsningen er vist i Figur 3.3

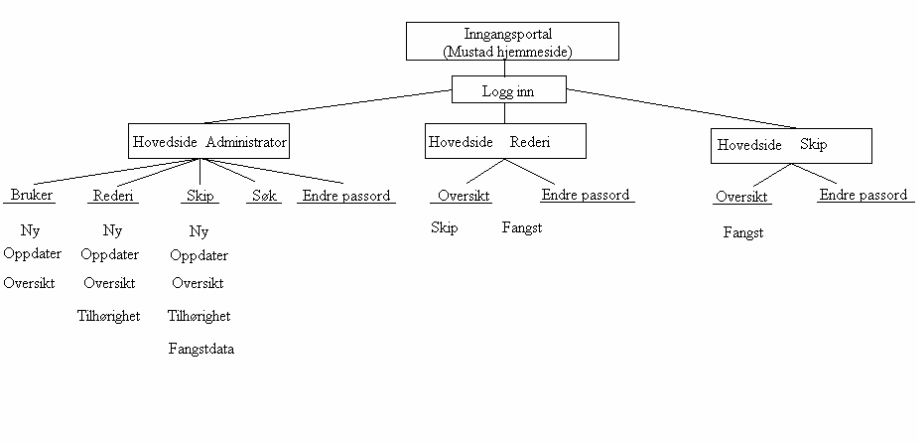


Figur 3.3: Bilde av en nettside på webløsningen

Vi har her satt de forskjellige sidene som man kan navigere mellom inn i en meny. Den har vi plassert på toppen og den er synlig på alle sidene i systemet. Dette gjør det veldig enkelt for brukeren å navigere mellom de forskjellige alternativene.

3.5 Navigasjonsbeskrivelse

Vi har laget en brukerveiledning og et navigasjonskart (Figur 3.4) som nye brukere av systemet kan lese gjennom før de tar systemet i bruk, eller som hjelp underveis. Det er dessuten en hjelpfunksjon på nesten hver side hvor man kan få råd og hint om funksjonaliteten på den aktuelle siden. Vedlegg A, side 53, beskriver nærmere alle de forskjellige funksjonene på hver side og inneholder også et navigasjonskart.



Figur 3.4: Navigasjonskart over webløsningen

Systemet er delt opp i tre forskjellige brukernivåer og hver bruker har kun adgang til et. De tre forskjellige nivåene er ”Administrator”, ”rederi” og ”skip”. Det er da her tenkt at en administrator skal ha oversikten over alle de andre brukerne og deres data. Et rederi har kun tilgangen til sine skip og sin fangst, mens et skip kun har adgang til seg selv.
Man får tildelt rettigheter, brukernavn og passord av en administrator, men passordet kan byttes ut senere, til det som brukeren selv synes er lettere (Må være over seks tegn).

4 Implementering

Begrunner valg av hardwarekomponenter til dataoverføring, programmeringsspråk på websystemet og sendeprogrammet, database og systemutviklingsverktøy. Beskriver også utviklingsmiljøet og filstrukturen.

4.1 Valg av komponenter til sending

4.2 Valg av programmeringsspråk og software til rapporteringssystemet

4.3 Sende/mottakersystemet

4.4 Web- løsning

4.1 Valg av komponenter til sending

Vi hadde flere muligheter å velge mellom når det gjaldt å få sendt data fra en database til en annen. En mulighet ville være å bruke inmarsat-C systemet med X.400 og X.25 protokoll for sending. Denne metoden ble av oss sett på som avleggs og umoderne.

En annen mulighet ville vært å sende dataene som attatchment (vedlegg) over Internett. Vi ville heller prøve å utnytte mulighetene som ligger i inmarsat-M systemet, som er en smalbånds digital teknologi for overføring av stemme med en hastighet på (4.8kbps) og for fax / data på (2.4kbps)

Etter å ha undersøkt de forskjellige tilbudene kom vi frem til at en oppkobling mot satellittelefon ville være den riktige løsningen på sendedelen. Her kunne vi nemlig implementere standard Hayes kommandoer (som er et internasjonalt standardisert sett med modem- kommandoer) inn i vår programkode.

Ved å velge en modemløsning fremfor en attatchmentløsning så vil det også bli lettere å få lagt inn de riktige dataene inn i databasen på land. Feilsjekking av data og eventuelt omsending av datapakker vil også kunne gå raskere når vi har opprettet en modemforbindelse.

Deretter undersøkte vi hvilke leverandører som fantes i Norge og hvilket tilbud de hadde av satellittskommunikasjonsutstyr. 1. prioriteten vår ble da et firma ved navn ”Skantiradio” som holder til i Oslo. Disse ble valgt både fordi de hadde lokaler ikke altfor langt vekk fra Gjøvik og fordi vår kontaktperson der var veldig hyggelig og imøtekommende. For å holde prisnivået på et moderat nivå og allikevel holde oss innenfor et komponentvalg som ikke gir oss for store begrensninger, fant vi ut at en maritim satellittelefon av typen ”mini-M, TT-3064A” ville være det beste valget vist i Figur 4.1.

Figur 4.1: Bilde av mini-M TT3064A (maritim versjon)

Denne telefonen har muligheter for å fungere som et modem, i tillegg til at den har et DCE interface med en 9-pin female kontakt . Vi vil da her ha muligheten til å koble denne til en com- port på pc-en for deretter å sende og motta data. Med tanke på dekningsområde så vil man med Inmarsat- systemet ha dekning omtrent overalt. De reklamerer med en global dekning på 98%. Det er kun Nord- og Sørpolen som ikke har dekning. Jan Erik Skjønsberg på ”Skantiradio” har vært behjelpelig og positivt innstillt til vårt prosjekt, så det var derfor ikke noe problem med å få lånt en slik mini-M for å teste ut om vårt system ville fungere med den. Men på grunn av at vi bare fikk låne den en uke, så måtte vi simulere sending med standard 28.8 modem over det interne telenettet her på høgskolen gjennom hele prosjektperioden, bortsett fra den uken hvor vi lånte mini-M’en.

I tillegg så hadde de bare en landversjon av mini-M'en til utlån, se Figur 4.2, men denne skal fungere på samme måte som den maritime versjonen.

Figur 4.2: Bilde av mini-M TT3060A (landmobil)

Den aktuelle satellitt-telefonen ble da lånt og utprøvd mot slutten av prosjektet (Uke 17). Den skulle også lånes ved fremføringen av vårt prosjekt.
Datablad til mini-M'en finner man i vedlegg L, side 115.

4.2 Valg av programmeringsspråk og software til rapporteringssystemet

Systemet vi skulle bygge videre var utviklet i delphi med BDE og de lagret sine data i en interbasedatabase. Det falt da naturlig å bruke samme programmeringsspråk og database, men vi prøvde også ut en MySqldatabase. Grunnen til dette var at skolens webserver var satt opp med MySql, og gjorde det mulig å legge ut websidene på nett slik at vår oppdragsgiver kunne følge med på utviklingen.

En del av oppgaven skulle være en dynamisk internettløsning og alle tre tok faget klient og serversideprogrammering som omhandler utvikling av slike løsninger. Her var tre programmeringsspråk pensum (Java Servlets, Asp og Php). Vi valgte programmeringsspråket Php (Hypertext Preprocessor) ettersom dette språket virket som det mest robuste, samtidig lettfattelig og med store likheter med tidligere programmeringsspråk vi har tatt ved høgskolen.

Det inneholder også mye innebyd funksjonalitet som gjør det enklere å kode. Skulle en gjort det samme i Java, ville en fått utallige flere kodelinjer.

I startfasen prøvde vi en del forskjellige webservere, men det endelige valget falt på apachewebserver. Noe av grunnen til valget var at denne ble brukt på skolen og fulgte med pakken til NuSphere, som vi fant da vi søkte på nett. Dette er en pakke som inneholder apachewebserver, MySql og php. Pakka hadde et installeringsprogram, slik at man slapp å sette opp alle delene manuelt, noe som vi erfarte kan føre til en del hodebry.

4.3 Sende/mottakersystemet

Anvendelse av verktøy og utviklingsmiljøer :

Koden på klient-siden er integrert i det eksisterende LineController- programmet som var utgangspunktet for oppgaven vår. All koden er implementert i Borland Delphi 5. På serversiden er det også utelukkende brukt Borland Delphi 5. Databasen som brukes av BDE-motoren i Delphi er lagd i Interbase.

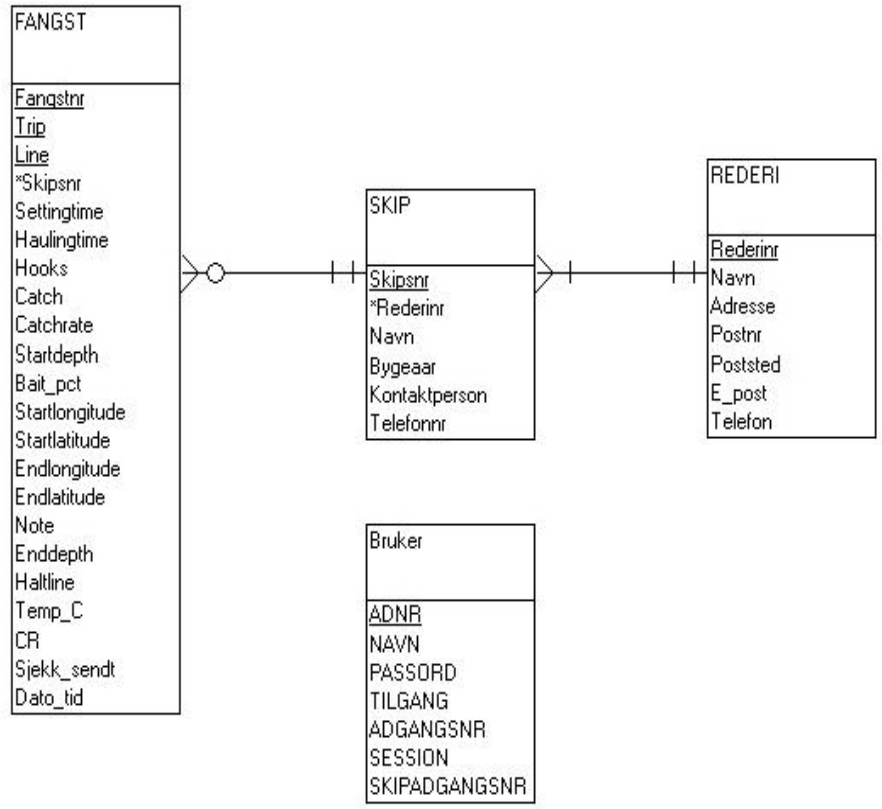
I tillegg til standardkomponentene i Delphi 5 måtte vi ha en komponent for modem til modem kommunikasjon. Etter å ha lastet ned en del komponenter fra delphisiden www.torry.net endte vi med å bruke komponenten XComDrv. Grunnen til valget av nettopp denne komponenten var at den var lett å installere og hadde bra med hjelpefunksjoner. Beskrivelse av denne komponenten [3] er tatt rett fra www.torry.net .

Helt til det nærmet seg slutten av kodingen har vi brukt en egen database for lagring av fangstdata i LineController. Denne har vært så å si identisk med den eksisterende databasen brukt av LineController, bortsett fra at vi la til noen felter som var nødvendige for å opprettholde databasestrukturen vår. Feltene som er lagt til er merket med gult på bildet under.

Figur 4.3 : Bilde av Mustad databasen i LineController programmet slik den blir illustrert i Modelator

Vi støtte imidlertid på flere integreringsproblemer med vår egen database og fant ut at det mest hensiktsmessige var å bruke den eksisterende databasen og utvide denne etter behov.

På serversiden derimot lagde vi vår egen database. Vi brukte programmet Modelator for å generere SQL-skriptet for databasen som senere ble kjørt i Interbase sin IB-Console. Figur 4.4 på neste side viser databasen på server-siden.



Figur 4.4 : Bildet viser databasen på serveren slik den blir illustrert i Modelator

Metadataene til figur 4.4 ligger i vedlegg I, side 110.

Når det gjelder koden og navnsetting av variable så har vi prøvd så langt det lar seg gjøre å holde oss til 3-bokstav's forkortelser foran variabelnavnene for lettere å skille strenger, tall og arrayer osv. fra hverandre. For å holde en viss orden i koden og lettere finne fram så har vi valgt å skille prosedyrene i programmet fra hverandre med en horisontal linje. Eksempler på det som er nevnt her er vist under.
En integer begynner med int- og deretter resten av variablenavnet,
En Streng begynner med str- og deretter resten av variabelnavnet,
En array begynner med arr- osv.

```
//*****************************************************************************
//Sender de 22 verdiene fortløpende
procedure TMustadFrm.sendStrenger;
var
intI : Integer;
begin
  for intI:=1 to 22 do
    begin
      Modem.SendString(strSendeTekst[intI]);
    end;
end;

//*****************************************************************************
```

```
procedure TMustadFrm.loggForing(arrSendeTekst : array of String);
var
  loggFil : TextFile;
  dtiDato : TDateTime;
  strDato : String;
  intTeller : Integer;
  strEttFelt : String;
  arrDbFelter : array[1..21] of String;
begin
  arrDbFelter[1] := 'Fangstnr';              arrDbFelter[2] := 'Trip';
  arrDbFelter[3] := 'Line';                  arrDbFelter[4] := 'Skipsnr';
  arrDbFelter[5] := 'Settingtime';           arrDbFelter[6] := 'Haulingtime';
  arrDbFelter[7] := 'Hooks';                 arrDbFelter[8] := 'Catch';
  arrDbFelter[9] := 'Catchrate';             arrDbFelter[10] := 'Startdepth';
  arrDbFelter[11] := 'Bait_pct';             arrDbFelter[12] := 'Startlongitude';
  arrDbFelter[13] := 'Startlatitude';        arrDbFelter[14] := 'Endlongitude';
  arrDbFelter[15] := 'Endlatitude';          arrDbFelter[16] := 'Note';
  arrDbFelter[17] := 'Enddepth';             arrDbFelter[18] := 'Haltline';
  arrDbFelter[19] := 'Temp_c';               arrDbFelter[20] := 'Cr';
  arrDbFelter[21] := 'Sendt_sjekk';
  dtiDato := now;  //Datoen akkurat nå...
  strDato := DateTimeToStr(dtiDato);
  AssignFile(loggFil, 'logg.txt');
  Append(loggFil);
  Write(loggFil, 'Dato :'+' ['+strDato+']');
  Writeln(loggFil, '');
      for intTeller:=0 to 20 do
       begin
         Write(loggFil, arrDbFelter[intTeller+1]);
         strEttFelt := ':['+arrSendeTekst[intTeller]+'] ';
         Write(loggFil, strEttFelt);
       end;
    //Legger på ett linjeskift
    Writeln(loggFil, '');

  CloseFile(loggFil);
end;
//*****************************************************************************
```

NB !
Testingen av dataoverføringer med LineController programmet kan ikke gjøres sånn helt uten videre. For å få kjørt programmet må man nemlig ha ett IO-kort som vi har fått låne av Mustad. Dette kreves siden programmet i virkeligheten leser mange verdier fra ulike maskiner og telle-enheter inn på IO-kortet. I tillegg brukes det veldig mange komponenter som ikke er standard i Borland Delphi 5 som vi også har fått av Mustad. Dette var grunnen til at vi brukte mye tid på å få installert komponentene og å kalibrere IO-kortet. Denne jobben ble gjort i samarbeid med Carl Aarnes og det var han som stod for det meste av dette arbeidet siden han har god kjennskap til systemet og har gjort denne installasjonsjobben før.

4.4 Web- løsning

Kildekoden består av PHP kode, vanlig HTML og javascript. PHP er veldig mye brukt nå, og vi har valgt å bruke dette språket fordi det blant annet kan kjøres på Windowsbaserte servere. PHP sidene blir kjørt på serveren og sendt til klienten som vanlig html kode, det kreves derfor ikke noen ekstra ressurser av klienten. En ulempe her i motsetning til ASP er at det opprettes en ny prosess for hver forespørsel, ikke en tråd. PHP er opensource og det er derfor lett å finne relaterte kodesnutter til problemer. Vi har hatt stor nytteverdi av nettsiden www.php.net da denne siden inneholder en meget god og oversiktlig dokumentasjon på språket. Under er det vist ett par kodeeksempler fra websidene våre.

Eksempel på registrering av sessions og kobling mot database:

```
<?php
session_start();
session_register('loginnavn'); //registrerer variabelen loginnavn
session_register('sess');      //registrerer variabelen sess
require ("dbconnect.php");     //må finne og bruke fila dbconnect.php for at siden kan kjøres
include_once("adtop.php");     //legger til toppen på siden


$act  = ($action  ==  "check") ?  "check"  :  "display";

$sSQL = "select SESSION, TILGANG from BRUKER where NAVN='$loginnavn'";
// velger ut verdien session, tilgang
$stype = ibase_query($sSQL);
// fra databasen der navn er lik loginnavn
$row=ibase_fetch_row($stype);

if((strcmp($row[0], $sess) == 0) && (strlen($sess) > 0) && ($row[1]=='admin')){
// sammmenligner rad[0] "session" med variabelen sess som blir
// registrert på toppen av siden og tilgangen må være lik admin
```

Eksempel på vanlig HTML :

```
<center><table border="0">
<tr>
<td><input type="hidden" name="action" value="check"></td>
<td><input type="hidden" name="add" value="1"></td>
</tr>
<tr>
<td>Gammelt passord:  </td>
<td><input type="password" name="GAMMELTPASSORD" size="32" maxlength="32"
tabindex="1" value="<?php echo $HTTP_POST_VARS['GAMMELTPASSORD']
?>"></td>
</tr>
<!-- Beholder verdien når siden lastes på nytt -->
<tr>
<td>Nytt passord:  </td>
<td><input type="password" name="NYTTPASSORD" size="32" maxlength="33"
tabindex="2" value="<?php echo $HTTP_POST_VARS['NYTTPASSORD'] ?>"></td>
</tr>
<tr>
<td>Bekreft passord:  </td>
<td><input type="password" name="BEKREFTPASS" size="32" maxlength="32"
tabindex="3" value="<?php echo $HTTP_POST_VARS['BEKREFTPASS'] ?>"></td>
</tr>
<tr>
<td><input type="submit" name="lagre_knapp" value="Lagre" tabindex="4"></td>
</tr>
</form>
</table></center>
```

Fullstendig kildekoden kan finnes i vedlegg M på vedlagt cd.

5 Testing

Beskriver prosessen hvordan vi utførte tester på både sende/mottakersystemet og på websystemet. Dessuten omhandler det også feil som ble oppdaget og rettet på.

5.1 Testing av mini-M

5.2 Testing av dataoverføring med 28.8 modem

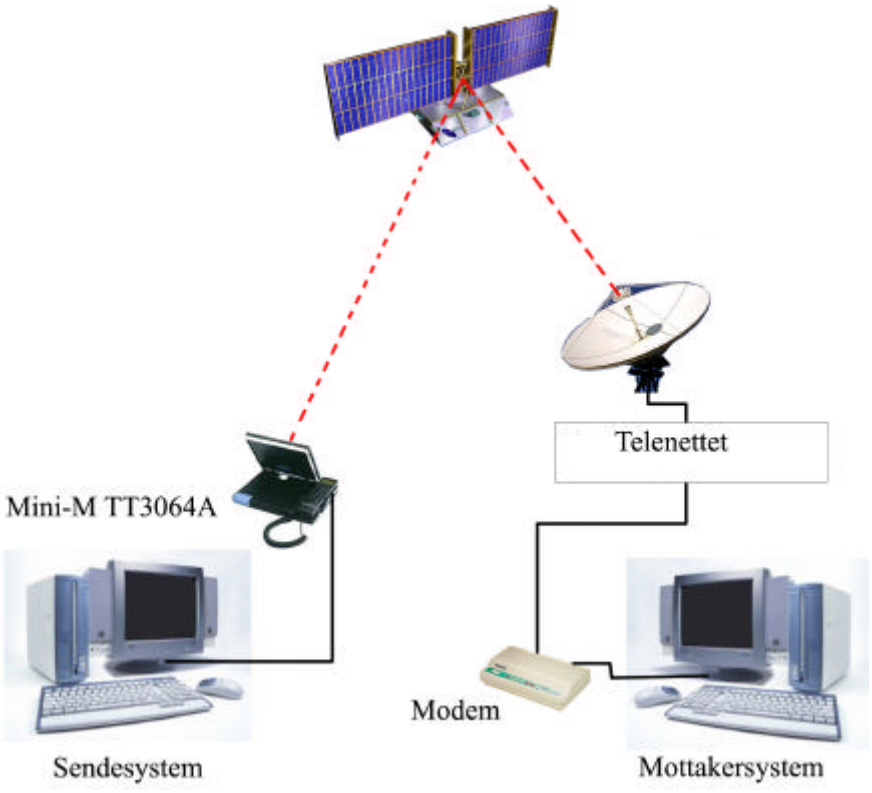
5.3 Testing av internett sidene

5.1 Testing av mini-M

Mini-m modell TT3060A (Landmobil) ble lånt i uke 17, 2002

Hadde noen problemer i startfasen med å få satt opp mini-M'en riktig, men etter litt feilsjekking og et par telefoner til Skantiradio, fant vi ut at batteriet på mini-M'en måtte være i ustand, i tillegg til at den strømforsyningen som fulgte med ikke leverte nok strøm til å drive telefonen uten fungerende batteri. Den var bare beregnet på å lade opp batteriet. Vi måtte derfor bruke en egen strømforsyning og vi valgte å kjøre på med 12V på inngangen.

Det viste seg nå at oppstarten av telefonen gikk greit, men vi hadde litt problemer med å lokalisere satellittene. Etter litt prøving og feiling så fikk vi kontakt med en satellitt og vi endte opp med et signal støyforhold på 47.5dBhz da vi rettet antennen mot "Atlantic Ocean Region East". Dette er meget bra da alt over 43dBhz ifølge manualen er gode sendeforhold. Vi koblet opp mot pc-en med Linecontroller-systemet Figur 5.1 for å teste ut sendingen, og alt fungerte som planlagt. I denne testen så hadde vi kun "happy day scenario". Det vil si at vi hadde optimale sendeforhold, ingen avbrytelser under sending og ellers fullført oppkobling. Dette for å holde kostnadsutgiftene på et minimum. Satellitt-telefonen skulle jo i teorien fungere som et vanlig modem, så alle de "worst case scenario" kunne heller simuleres med et 28.8 modem.



Figur 5.1: Oppkobling av systemet med mini-M

5.2 Testing av dataoverføring med 28.8 modem

Testingen av dataoverføringen med 28.8 modem har pågått jevnlig hele tiden. Helt fra starten av da vi fikk lånt og installert modemene og opprettet en internlinje mellom dem har det blitt testet ulike overføringer. I starten ble det brukt hyperterminal som er ett windows- program som muliggjør kommunikasjon med modem via COM-porten. Enkle AT kommandoer ble brukt for å koble opp modemene med hverandre. Videre sendte vi over enkle data sånn som tekststrenger og enkelte små filer med ferdigfunksjonene i hyperterminal. Vi skjønte ved hjelp av denne testingen at overføringen av data mellom to modemer var fullt mulig, og at overføring av data mellom en satellitt telefon og ett modem ikke nødvendigvis trengte å bli så mye vanskeligere. Dette ble senere bekreftet da vi tok kontakt med leverandøren av satellitt telefonen som vi valgte å bruke. Han sa at overføring mellom satellitt telefon og modem og overføring mellom to modemer er to sider av samme sak. Vi fortsatte videre med leting etter Delphi-komponenter som kunne styre denne overføringen rett fra LineController programmet. Da vi fant en passende komponent begynte testingen igjen med oppkobling, sending av data osv. fra testprogrammer som vi lagde i Delphi. Siden denne testingen begynte har det gått slag i slag. Sendesystem ble integrert i LineController på den ene maskinen vår og mottakersystem ble lagd på den andre.

5.3 Testing av internett sidene

Vi har prøvd ut websidene både internt i gruppa, og vi har fått utenforstående til å komme med innspill etter å ha testet funksjonaliteten. De har blitt prøvd ut av veileder, oppdragsgiver og andre medstudenter. Websidene har vært under kontinuerlig utvikling og alle råd fra testpersonellet har blitt vurdert. Det ble også oppdaget noen feil angående skriving til databasen, tidligere var det nemlig mulig å skrive inn flere tegn enn det variabelen kunne motta, men dette ble ordnet med en gang. Det grafiske brukergrensesnittet har også blitt påvirket av hva testepersonene har foreslått. Sluttresultatet er derfor et produkt av en løpende utvikling og vi mener at dette har vært til det beste for systemet. På grunn av knappheten av tid mot slutten av perioden, så har vi satt opp en del punkter som kunne vært forbedret. Disse kan det leses mer om i kapittel 6.

6 Diskusjon av resultater

Inneholder diskusjon av gruppen, hvordan forholdet til veileder og oppdragsgiver var, og hvordan resultatet av systemutviklingen ble. I tillegg har vi tatt med eventuelle endringer som kan gjøres.

6.1 Prosjektperioden for gruppa

6.2 Sluttproduktet

6.3 Svakheter og forslag til forbedringer ved sende- og mottakersystemet

6.4 Svakheter og forslag til forbedringer ved internettløsningen

6.5 Oppdragsgiver

6.6 Veileder

6.1 Prosjektperioden for gruppa

Vi visste at prosjektperioden ville føre til stor arbeidsmengde på gruppas medlemmer og vi var derfor nødt til å sette opp planer på kortsiktige mål og milepæler som skulle være klare. Disse ble så satt inn i et Gant- skjema og skulle senere være en retningslinje for hvilke prioriteter vi var nødt til å gjøre. Eksempler på loggførte timer er vist i vedlegg K, side 114. Eksempel på møtereferat kan man finne i vedlegg H, side 108.

Noen arbeidsoppgaver har vi brukt lenger tid på enn forventet, mens andre har gått raskere enn det som var planlagt. I startfasen av prosjektet ble det brukt urovekkende lang tid på å få satt opp våre pc-er med den rette softwaren.

Vi har også vært i kontakt med myndighetene i prosjektperioden. Dette ble gjort for å undersøke litt rundt kravene til et rapporteringssystem. Dette er nærmere dokumentert i vedlegg C, side 99 og vedlegg D, side 100.

Under prosjektperioden ble det ytret et ønske fra oppdragsgiver om å finne et egnet veiesystem som kunne integreres i LineControlleren. Vi undersøkte flere mulige leverandører og snakket med personer som hadde testet ut noen vektsystemer. Mot slutten av prosjektperioden tok vi en beslutning om å ikke lage et veiesystem. Dette ble gjort i samråd med oppdragsgiveren på grunn av at det ville ta for lang tid å sette seg inn i diverse utstyr og tilleggsprogrammering for å få dette til å fungere. Men det kan jo være et mulig prosjekt på Hig i fremtiden.

Gruppa som helhet har fungert bra og samarbeidet har vært upåklagelig.

6.2 Sluttproduktet

Vi mener selv at vi har fått til et relativt bra og brukervennlig produkt. Det er også de tilbakemeldingene vi har fått fra oppdragsgiver og veileder. Når det gjelder om prosjektet kunne vært løst annerledes, vil vi anta at det finnes mange ulike måter å løse det på. Det er nå engang slik at for å komme i mål må man ta en del valg underveis, om de er gode eller dårlige. Det finnes blant annet mange delphikomponenter en kunne valgt til bruk med satellittelefonen som kanskje er bedre, eller man kunne laget sine egne komponenter. Det ville også vært mulig å lage websidene med et annet programmeringsspråk og et annet oppsett av sidene. Likevel mener vi at vi gjorde en del gode valg i forhold til vårt kunnskapsnivå, erfaring og de retningslinjene vi hadde.

Når vi nå får prosjektet mer på avstand, så vil det sikkert dukke opp småting som med fordel kunne vært gjort annerledes. Det finnes en del svakheter ved systemet allerede og disse er belyst i punkt 6.3 og 6.4.

For å sette prosjektet ut i livet så har vi satt opp en liten oversikt over kostnader.
Disse er hentet fra internett og er gjenngitt i vedlegg E, side 102
Vi har også laget en installeringsguide til webdelen og denne finner man i vedlegg B, side 75.

Vi håper oppdragsgiver ser nytten av vårt arbeid og benytter vårt system i sin videre utvikling av LineControlleren.

6.3 Svakheter og forslag til forbedringer ved sende- og mottakersystemet

Grunnet tidspress så har vi ikke rukket å ta høyde for alt som kan gå galt ved systemet. Det må derfor justeres en del på sende- og mottakersystemet før det eventuelt kan tas i bruk ombord på AutoLinefartøyene.
Feilsjekking har jo blitt utført i den grad det er mulig å simulere satellittoverføring ved hjelp av to modemer. Dette har vi blitt fortalt at skal fungere på nøyaktig samme måte, men det er jo som kjent ofte forskjell på teori og praksis.

- Systemet/programmet tar det for gitt at en forbindelse kan opprettes mellom satellittelefonen på båten og modemet på serversiden. Uansett hva grunnen er til at forbindelsen ikke kan opprettes, så prøves det igjen etter et visst tidsintervall. Dette kan jo forårsake at programmet henger og må startes på nytt etter at man for eksempel har peilet inn satellittelefonen på en ny satellitt.

- Satellitten vi fikk låne for testing er en stasjonær satellittelefon for bruk på land og må peiles inn på den valgte satellitten før den kan tas i bruk. Når systemet skal tas i bruk til sjøs må det brukes en maritim satellittelefon som jo er i bevegelse hele tiden. Disse to satellittelefonene skal jo i teorien fungere på samme måten, men det er jo mulighet for at forbindelsen er vanskeligere å opprette til sjøs enn ved optimale forhold på land som har vært bakgrunn for vår testing.

- Hvis forbindelsen blir opprettet og deretter brutt akkurat når sendingen av data har startet eller er underveis så byr dette på problemer. Dette vil medføre at den siste verdien som sendes, nemlig CRC-sjekken ikke kommer fram. Det vil da være umulig å få verifisert dataene. Prosedyrene for å dele opp den mottatte strengen vil også få problemer siden de tar utgangspunkt i at 22 verdier alltid blir mottatt.

- En annen svakhet er at sånn programmet fungerer nå, så sendes dataene nærmest umiddelbart etter at de er registrert. I realiteten må skipperen ha mulighet til å oppdatere visse felter om en gitt stubb en periode etter at han har trykt på SAVE-knappen i programmet. Dette er ikke fullt så enkelt å justere som å øke antall sekunder, minutter eller timer som går før timeren(nedtelleren) for sending trigges. Man kan da risikere at oppdateringen av en stubb skjer samtidig med at den blir forsøkt sendt og dette er en lite heldig kombinasjon.

- Databasene tar ikke høyde for at f.eks. fangstnummeret eventuelt kan vokse forbi tillatt Integer-verdi. Mot slutten av prosjektet fant vi også ut at databasefeltet ”fangstnr” egentlig er overflødig når systemet skal tas i bruk. Grunnen til at det ikke er fjernet er at veldig mye kode måtte ha blitt gjort på nytt. I tillegg har dette feltet gjort testingen av overføringen mye enklere.

- Problemene som går på at sendingen ikke blir gjennomført riktig kan imidlertid løses manuelt. Det er lagd en knapp (”Send manuelt”) som henter ut en record i databasen der sendt_sjekk er satt til 1, og sender denne recorden til mottaker på samme måte som den blir sendt automatisk etter at det er trykt SAVE i LineController.

- Et annet scenario er at satellittelefonen er i bruk av personellet ombord på båten når en sending blir forsøkt startet. Dette har vi ikke testet konsekvensene av.

- Hvis forbindelse mellom satellittelefonen på båten og modemet på land av en eller annen grunn ikke kan opprettes etter en viss tid burde man hatt en mulighet for å avbryte oppringnings prosedyren. Eventuelt kunne man fått opp en pop-up dialog med diverse valg. Valgene kunne vært f.eks ”prøv igjen”, ”avbryt”, ”Prøv igjen etter inntastet antall minutter”.

- Loggføringen kunne kanskje vært delt inn i 2 forskjellige loggfiler, der den andre loggfila kunne vært en ”errorlog”. I denne errorloggen kunne man f.eks skrive mer detaljerte feilmeldinger om hva som er årsaken til feilsendinger.

6.4 Svakheter og forslag til forbedringer ved internettløsningen

- Det finnes ingen mulighet for å angre ulike valg som blir gjort på websiden når det gjelder f.eks registrering av en ny bruker. Hvis du taster inn feil data når du legger inn eksempelvis en ny bruker burde det kanskje ha vært mulighet for å angre dette valget. Det du imidlertid kan gjøre er å oppdatere informasjonen om brukeren senere.

- Ved innlegging av en ny bruker som skal ha tilgang til rederidata eller skipsdata, så taster man først inn generell info om brukeren som navn, adresse osv. Deretter blir man sendt videre til en ny side der man velger hvilken tilgang brukeren skal ha. Brukeren er dermed allerede lagt inn og hvis man trykker ”back” i nettleseren uten å velge tilgangsnivå så registreres brukeren inn uten tilgang til noe.

- Hvis du skal oppdatere en brukers navn , og navnet du vil endre til allerede finnes i databasen så hopper programmet tilbake og man må velge ”oppdater bruker” en gang til for endre navnet.

- Det legges inn skipsnr automatisk i databasen ved innlegging av nye skip. Disse nummerene må være de samme som ligger i AutolineControllerprogrammene på hvert enkelt skip. Altså skip nr.7 i tabellen skip på serveren MÅ referere til det Autolinfartøyet som har skipsid nr 7. I starten av utviklingen var programmet slik at man måtte taste inn disse nummerne manuelt og en bør nok gå tilbake til dette.

6.5 Oppdragsgiver

Oppdragsgiveren har som tidligere nevnt vært O.Mustad & Søn og i startfasen hadde vi ukentlige møter hos dem her på Gjøvik. Det at de har lokaler i kun kort avstand fra høgskolen har gjort at vi kunne holde et tett samarbeid i startfasen. Dette for å kartlegge krav og retningslinjer til prosjektet. Vi hadde allikevel ganske frie tøyler for valg av komponenter og programmeringsspråk, så lenge vi holdt oss til de linjer som var gitt i oppgaveformuleringen. Geir Liaklev og Carl Arnes som har vært våre kontaktpersoner har hele tiden vært aktivt interessert i fremdriften og de har kontinuerlig hjulpet oss med råd og tips under prosjektperioden.

6.6 Veileder

Vår veileder Arne Wold har vært til uvurderlig hjelp under fasene i prosjektet. Han har vært en viktig støttespiller spesielt når det gjelder programmeringen av oppringningsprogrammet. Dessuten har han også vært aktivt med i fremdriften av prosjektet ved at vi har hatt kontinuerlig kontakt, istedenfor statusrapporter så har vi heller valgt å holde fortløpende muntlig underretning med veileder. Dette har fungert veldig godt og vi har fått en god personlig kontakt med vår veileder.

7 Konklusjon

Konklusjon

Hovedprosjekt på høgskolen i Gjøvik er en stor milepæl for alle ingeniørstudentene. Her skal de få muligheten til å vise hva de har lært i løpet av sin studietid, for eksempel ved å utvikle et produkt eller system for et firma. Da vi skulle velge oppgave var det mange å velge mellom, men valget falt til slutt på å lage et online rapporteringssystem. Vi angrer ikke på valgt oppgave, fordi den har vært både lærerik og selvutviklende. Vi har fått erfare hvordan det er å jobbe tett sammen i et halvt år, og vi har lært oss hvordan vi skal henvende oss til bedrifter med diverse forespørsler. Disse erfaringene vil komme godt med i jobbsituasjoner og prosjektarbeid i arbeidslivet.

Ved utviklingen av rapporteringssystemet har vi måttet lære oss et nytt programmeringsspråk for å utvikle websidene, og vi har måttet sette oss inn i Photoshop 6.0 for å kunne jobbe med diverse layout og bilderedigering. Det at man måtte lære seg visse elementer fra starten av, har ført til at deler av prosjektet sikkert kunne vært utført med en mer robust og komprimert kode. Spesielt med tanke på satellittoverføringen var vi spente på hvordan det hele ville utvikle seg, siden ingen av oss hadde noen erfaring med lignende systemer fra før. Når vi nå ser tilbake på oppgavedefinisjonen og hva målet var, mener vi at vi har løst oppgaven bra på de fleste områder. Vi mener selv at vi har kommet frem til en forholdsvis god prototyp som oppdragsgiveren kan dra nytte av.

8 Referanser

[1] Erlend Efskind. Fisk og fusk. Moderne produksjon 22.01.02

[2] Fiskeridirektoratet. SatRap versjon 2.0. Brukerveiledning. 25.10.01

[3] Ondrej Ulrik. Project XComDrv. www.torry.net 20.03.02

[4] http://www.php.net

[5] http://www.interbase.com

[6] http://www.nusphere.com

9 Vedlegg

Vedlegg A: LineController på web.....53
Vedlegg B: Installasjonsguide.....75
Vedlegg C: Kontakt med myndighetene.....99
Vedlegg D: Brev til myndighetene.....100
Vedlegg E: Kostnader for å sette prosjektet ut i live.....102
Vedlegg F: Plansje over systemet..... 104
Vedlegg G: Dataoverføringen.....105
Vedlegg H: Eksempel på møtereferat fra hovedprosjektmøter.....108
Vedlegg I: Metadata.....110
Vedlegg J: CRC forklaring.....113
Vedlegg K: Eksempel på loggføring av timebruk.....114
Vedlegg L: Datablad for mini-M TT3064A.....115
Vedlegg M: Kildekode på CD