HOVEDPROSJEKT:

Lasting av smartkort applikasjoner over internett

FORFATTERE:

Arve Bjørnerud
Martin Klaveness
Ola Østeng

Dato: 23 mai 2001

Sammendrag av hovedprosjekt

Tittel:	Lasting av applikasjoner over internett	Nr. :
		Dato : 23/05-01
Deltakere:	Arve Bjørnerud Martin Klaveness Ola Østeng	
Veileder:	Frode Haug	
Oppdragsgiver:	ErgoGroup	
Kontaktperson:	Morten Johansen	
Stikkord (4 stk)	Smartkort, java, ocf og internett	

Antall sider:	Antall bilag:	Tilgjengelighet (åpen/konfidensiell):

Kort beskrivelse av hovedprosjektet:

Smartkort teknologien har vært i stor vekst de senere år, denne oppgaven fokuserer på et nytt område innen denne teknologien. Et smartkort kan inneholde forskjellig informasjon, hvor informasjonen kan skreddersys hver bruker. Kortet kan for eksempel inneholde identiteten til brukeren, slik at man på en enkel og trygg måte kan handle over internett eller bruke bank terminaler. Smartkortet kan også inneholde penger for bruk på betalingsautomater, som for eksempel parkeringsautomater, telefonkiosker o.l.

Vår oppgave har vært å utvikle en prototyp som gjør det mulig for at en bruker, ved hjelp av internett, selv kan bestemme innholdet på sitt smartkort uten å måtte oppsøke kortutsteder.

FORORD

Denne oppgaven en del av den treårige ingeniørutdannelsen ved Høgskolen i Gjøvik. Den gjennomføres over store deler av andre semester det tredje året ved skolen. Oppgaven skal ta utgangspunkt i en realistisk og faglig relevant problemstilling, og legges opp slik at kunnskap og ferdigheter fra flere fagområder i studiet benyttes.

Oppgaven går ut på å lage ett system som gjorde det mulig å laste applikasjoner til smartkort over internett.

Vi vil rette en spesiell takk til Frode Haug ved HIG for hans gode råd og veiledning under fremdriften av prosjektet. Uten hans hjelp så ville prosjektet blitt atskillig vanskeligere å gjennomføre.

Vi ønsker også å rette en stor takk til Morten Johansen og Henrik Hartz ved ErgoGroup for deres verdifulle hjelp.

Gjøvik 23 mai 2001

Arve Bjørnerud	*Martin Klaveness*	*Ola Østeng*

Innholdsfortegnelse

Definisjoner

ADC	Application Delete Certificate. Sertifikat som trengs for å slette applikasjoner fra smartkortet.
Administrator	Person som har ansvaret for driften av systemet.
ALC	Application Load Certificate. Sertifikat som trengs for å laste applikasjoner til smartkortet.
ALU	Application Load Unit. Programmet som lastes/slettes fra smartkortet.
Applet	Java program som kan kjøres fra en nettleser
Applikasjon	Et program som er skrevet for bruk på smartkort. *Se ALU*.
Byte	Målenhet for størrelse av filer på datamaskin
Cipher/ decipher	Kryptere / dekryptere.
CMS	Card Managment System database. Database som inneholder de aktuelle applikasjoner som kan lastes til smartkortet.
Com port	PCens serielle komunikasjonsport
CPU	Central Processing Unit. PCens hjerne.
Dekryptering	En datastrøm som er kryptert gjøres lesbar igjen. *Se kryptering.*
EAL Standarden	Mal for å utføre sikkerhetsanalyse.
EEPROM	Electrically Eraseable Programable Read Only Memory. En lagringsbrikke som kan endres vha strøm.
EiD	Elektronisk Identifikasjon. Smartkort applikasjon for identifisering av bruker.
Ergo Group (EG)	Tidligere Posten SDS. Oppdragsgiver.
Flash	Programmeringsspråk for utvikling av program som brukes på internett.
GUI	Brukergrensesnittet
Harddisk	PCens lagringsplass.
HIG	Høgskolen i Gjøvik
I/O	Input/Output. Kommunikasjon med eksterne enheter.
Instruksjoner	En instruksjon er noe som sendes til et smartkort for å utføre en ønsket operasjon.
Internet explorer	Nettleser som vi har valgt å jobbe med.
IP-adresse	Internett Protokoll. Unik adresse for hver enkelt bruker tilkoplet internett
ISO	International Standard Organization.
Java virtual machine	Plattform som alle java program startes fra og kjøres i.
kB	kiloByte. 1024 Byte.
Klient	Delen av systemet brukeren vil jobbe mot.
Linux	Operativsystem
MAC	Operativsystem
MB	MegaByte, 1024 kb eller 1048576 byte.
Modus	Tilstanden til ....
MULTOS	Operativsystemet på smartkortet
MySql	Database basert på SQL standarden.
OCF	Open Card Framework. Ett rammeverk for Java. Gir mulighet for å sende serier av kommandoer, svar sekvenser.
Parallell port	PCens
PC	Personal Computer. Datamaskin.
Pentium	Betegnelse på en PC sin kapasitet.
Posten SDS	Tidligere Statens Data Sentral. Nå en del av Posten under navnet Ergo Group.
Protokoll	En bestemmelse på hvordan data behandles og overføres.
RAM	Random Access Memory. PCens hurtiglager.
ROM	Read Only Memory. Informasjon blir brent inn i en brikke, kan ikke slettes.
RSA	Krypteringsalgoritme utviklet av R. Rivets, A. Shamir og L. Adleman.
Server	En PC med høykapasitet, som serverer data til flere klienter.

Servlet	Delen av systemet som ligger på serveren
Smartkort	Kort med egen CPU, RAM, ROM, I/O, EEPROM og co-prosessor
SQL	Structured Query Language. Ett enkelt instruksjonsett for å snakke med databaser.
SSL	Secure Socket Layer. En krypteringsmetode for dataoverføring.
Systemet	Består av server og klient.
TCP/IP	Transmission Control Protocol/Internet Protocol. Språket som benyttes til kommunikajone mellom flere PCer over internett.
USB port	Universal Synchronus Bus. En av PCens kommunikasjonsporter.
Windows 2000, Windows 95/98 og Windows NT	Operativsystem som vi har valgt som plattform.

1 INNLEDNING

1.1 Bakgrunn

1.1.1 ErgoGroup

Posten SDS ble etablert i 1972 gjennom ett stortingsvedtak. I 1986 ble statens datasentral til aksjeselskap, posten overtok disse aksjene i 1995. I 1998 inngikk selskapet som ett av fire forettningsområder i Posten Norge. Fra 1 januar 2001 skiftet Posten SDS navn til ErgoGroup (EG). Firmaet har de siste årene vokst jevnt og har i dag ca. 1200 ansatte. Og hadde en årsomsettning i 1999 på 1.4 milliarder kroner. ErgoGroup har kontorer over hele landet med hovedkontor i Oslo og med regionskontorer i Gjøvik, Trondheim og Mo i Rana. Ved avdeling Gjøvik jobber det i dag 8 personer ved utviklingsavdelingen.
EG sine satsningsområder er infrastruktur tjenester, elektroniske tjenester og administrative støttefunksjoner. Smartkort hører til elektronisk tjenester, der ErgoGroup utvikler og tilbyr et spekter av tjenester innen elektronisk meldingsutveksling, salg og distribusjon av informasjon fra databaser samt smartkort og elektronisk ID.

Dette er tredje året EG samarbeider med HIG angående smartkort hovedprosjekt. Fra tidligere år så har HIG samarbeidet med Posten SDS på flere prosjekter. De begynte i 1999 med oppgavene:

- Campus Card 1
- Campus Card 2
- Helsekort
- Cunning card

Og i 2000 med prosjektene:

- Lånekassen
- Raufoss badeland
- Vinmonopolet AS

Beskrivelse av oppgavene finnes på: http://higweb.hig.no/at/data/hprog.php3

1.1.2 Teknologi

Smartkortteknologien begynte så tidlig som 1970 og de første kortene var nokså simple teknisk sett. De inneholdt kun en minnebrikke som det kun var mulig å lagre data på. Prosessering av data i kortet var ikke mulig. Andre generasjon smartkort er CPU basert. Her kunne det i tillegg også utføres databehandling, som for eksempel kryptering. Tredje generasjons kort er såkalte contactless card, dette innebærer at kortet ikke trenger å være i fysisk kontakt med en kort terminal. Det vi skal jobbe med er andre generasjons kort.
Disse består av en CPU, en co-prosessor som utfører kryptering samt minne og I/O enheter. Det har en lagringskapasitet på 16kB, og krypteringsprosessoren opererer på en 1100 bits krypteringsnøkkel.

1.1.3 Prosjektgruppen

Vi valgte oppgaven fra EG om lasting av smartkort applikasjoner over internett fordi smartkortteknologien er forholdsvis ny. Vi har hørt en del om tidligere hovedprosjekter ved HIG, om smartkort i masemedier og av tidligere studenter, og syntes dette hørtes ut som en interessant og utfordrende oppgave. Vi ser på dette som en utmerket mulighet til å bli en del av denne utviklingen.

Fra før av har vi ingen kunnskaper direkte rettet mot smartkort så dette vil for oss bli spennende og lærerik prosess.
Den bakgrunnen vi har er fra skolegang ved HIG og andre skoler. Vi innehar en bred kunnskap innen programmering med erfaringer fra bl.a. C++, Delphi/pascal,

Assembler, Java, HTML, CSS, VO, Perl og PHP for å nevne de viktigste. Vi har også en god forståelse av nettverk og kommunikasjons flyt mellom maskiner.

1.2 Prosjektmål

Hovedprosjektet vårt består av fire hovedmål med hovedvekt på de to første punktene.

- Oversetting av eksisterende C kode til Java kode.
  Det eksisterende programmet som er skrevet i C er beregnet på å laste applikasjoner til ett smartkort fra en stasjonær PC hos utvikler. Oppgaven vår går ut på å lage en klient/server løsning, som gjør det mulig å laste ned applikasjoner fra internett til smartkortet.
- Sikker kommunikasjon mellom smartkort og server.
  For å sikre dataflyten fra server til kort må det benyttes kryptering. Oppdragsgiver vil at et ferdig javabibliotek benyttes for krypteringen.(SSL) krypteringen skal benyttes.
- Utføre risikoanalyse.
  Se på sårbarheten til systemet ved å definere mulige angrep på dataflyten mellom server og kort. Disse angrepene skal så testes ut i praksis. Alt dette etter EAL standarden.
- Brukergrensesnitt.
  Sette opp et web grensesnitt ved hjelp av flash og javascript

1.3 Oppgavebeskrivelse

Oppgaven består i å lage en ny løsning for lasting av smartkortapplikasjoner over internett. Eksisterende loader for smartkortapplikasjoner skal oversettes fra C++ til en Java-applet som integreres med en brukervennlig web-side. EG ser for seg at en server skal kunne tilby en smartkortbruker å laste ned en smartkortapplikasjon til smartkortet sitt. På serversiden skal det ligge smartkort-applikasjoner og sertifikater for hhv sletting og lasting. Disse skal overføres til klienten på en sikker måte. Brukerens PC vil ha installert en smartkortleser og eventuelt Java Plugin for kjøring av applet. Dette er en programvare som brukeren typisk vil få i en installasjonspakke sammen med smartkortet. Brukeren skal kunne gå til siden på internett og velge den applikasjonen han ønsker. EG ser for seg at når lastingen av applikasjonen skal starte, lastes det ned en applet til klienten som gjør det mulig å sende kommandoer til brukerens smartkort lokalt. På serveren kunne de tenke seg en servlet som styrer selve laste-prosessen ved å sende kommandoer til appleten på klient-siden. Selve lastingen skjer vha en rekke smartkortkommandoer.

Oppgaven går også ut på å foreta en trusselanalyse. Trusselanalysen er todelt. Den første delen går ut på å sette opp mulig angrep på systemet, dette er en prosess som vil foregå under hele utviklings perioden, fra kravspesifikasjonen til ferdig kode. Den andre delen går ut på å gjennomføre de oppsatte angrepene på systemet.

Oppdragsgiver ønsker i tillegg et kreativt brukergrensesnitt ved hjelp av FLASH og javascript, dette er ikke et prioritert mål. Det vil si at hvis vi får nok tid skal det gjennomføres. Det viktigst er å få til selve mekanismen bak, med andre ord overføring mellom servlet og smartkort til å virke på en tilfredstillende måte.

Vi skal ikke:

- lage en sikker kommunikasjon mellom kort og server, det brukes ett ferdig bibliotek.
- fokusere på sikkerhet mellom bruker og klient. Siden de applikasjoner som skal lastes på kortet ikke krever høy sikkerhet.
- sette oss inn i den fysiske kommunikasjonen mellom kort, kortleser og maskin.

1.4 Rammer

Hos EG har vi fått tildelt to grupperom som vi kan benytte til prosjektformål, disse grupperommene skal fordeles på tre grupper. Av utstyr har vi tilgang til to PC-er med mulighet til å ta med egen. Disse maskinene er tilkoplet internett. Av litteratur har EG en del bøker tilgjenglig. Hvis det skulle bli behov for andre bøker, har de sagt seg villig til å kjøpe inn dette. Det er store krav til sikkerhet rundt utviklingen og distribusjon av smartkort programmer/applikasjoner.

1.5 Avgrensninger

Et smartkort kan inneholde forskjellige applikasjoner. I fremtiden er det tenkt at eieren av et smartkort selv kan velge hvilke applikasjoner som skal ligge på kortet.
Slik som situasjonen er i dag må en kunde oppsøke en kortutsteder for å legge inn nye applikasjoner, selv om kunden har den nødvendig maskin og programvare.
For å laste applikasjoner på smartkort må du ha egne load sertifikater og tilsvarende delete sertifikater for å slette, noe som bare kortutsteder har.

1.6 Målgruppe for rapporten og oppgaven

Denne oppgaven og rapporten er i første hånd beregnet for bruk av ErgoGroup i deres arbeid videre med smartkortteknologi og internett.

1.7 Arbeidsformer

Under arbeidet med prosjektet har vi i perioden januar til april hatt gruppemøter hver torsdag og fredag hvor vi som gruppe har jobbet med oppgaven. I tilegg til gruppemøtene har vi hatt møter med veileder nesten hver onsdag. Ved behov for mer klarhet i oppgaven har vi hatt møter med oppdragsgiver. Dette har foregått på den måten at vi har foreslått en møtedag og fått tilbakemelding på dette. Etter påsken har vi satt av mer tid til prosjektet og har jobbet 3-4 dager i uken i tilegg til møter med veileder.

1.8 Organisering av prosjektrapporten

- Del 1: Innledning.
  Forord, og en enkel beskrivelse av hva oppgaven går ut på. Samt litt bakgrunnsinformasjon om ErgoGroup og studentene som har jobbet med oppgaven.
- Del 2: Kravspesifikasjon.
  Beskrivelse av hva systemet skal gjøre og inneholde.
- Del 3: Designdokumentet.
  Beskriver hvordan vi har tenkt å løse oppgaven.
- Del 4: Implementasjon.
  Kode eksempler, hvordan vi har utført kodingen og hvilke verktøy som er benyttet.
- Del 5: Testing.
  Hvilke tester vi har gjennomført, samt sikkerhetsanalyse av systemet.
- Del 6: Konklusjon og drøfting av resultatet.
  Hva de faglige resultatene kan rukes til og hva gruppen føler de sitter igjen med. Hva som kunne vært gjort annerledes ved en gjentakelse av utviklingsarbeidet. Egenevaluering av gruppearbeidet.
  Evaluerer hvorvidt det er samsvar mellom kravspesifikasjonen og den løsningen som ble valgt. Avvik drøftes.
- Litteraturliste, weblinker, verktøy.
  Henvisning til litteratur som er benyttet.
- Del 7: Vedlegg.

1.9 Organisering av kvalitetssikring

Dokumentasjon

- Kode Syntaks , se *vedlegg C*.
- Møtelogg/ Aktivitetslogg , se *vedlegg B*
- Rapport
  Vi har utarbeidet vår egen mal som vi har brukt til all rapportskriving. Malen er definert på følgende måte:
  - Vi benytter oss av 3 overskriftstyper.
    - Nivå 1: begynner inntil venstre kant og skrives med Arial størrelse 12.
    - Nivå 2: tabuleres ett hopp til høyre og skrives med Arial, størrelse 12.
    - Nivå 3: tabuleres to hopp fra venstre kant, og skrives med Arial, størrelse 8 fet.
  - For normal tekst bruker vi Arial størrelse 10, og tabulerer ett hopp til høyre i forhold til overskriften type 1 og 2 , mens for type 3 tabuleres det ikke.
  - Topptekst bestående av logoen til prosjekt gruppen, samt kapittelnavn.
  - Bunntekst bestående av sidetall.

Verktøy

- JBuilder 4.0
  Benyttes under kodingen
- Milestones Simplisity
  Lager Gantt skjema.
- Microsoft Word
  Benyttes til å skrive rapporten.
- Together v4.2
  Verktøy for å lage design diagrammer.

Konfigurasjonsstyring

- En prosjekt perm som behandles av loggfører, loggfører har som oppgave å samle inn møtelogg og aktivitetslogg.
- JBuilder inneholder et ferdig konfigurasjons system (CVS). Dette vil holde orden på kode skrevet av forskjellige brukere, samt automatisk dokumentere klasser og funksjoner.
- Loggfører skal skrive ut og sette inn i prosjekt permen all kode, en gang i uken.

Backup:

- Kopiere kode til server/annen maskin på slutten av hver dag.
- En katalog pr. person med katalog for hver dato hvor kildekode lagres.

Problemrapportering og tiltak:

- Fravær
  - Rapportering: ingen
  - Meldeplikt.
  - Tiltak: andre gruppemedlemmer tar over oppgaver til fraværende.
- Ikke oppnådd kontakt med veileder/oppdragsgiver
  - Rapportering: møtelogg
  - Tiltak: purring via email eller telefon
- Ikke overholdt tidsfrist i forhold til milepæl.
  - Rapportering: avvik/status -rapport til veileder og oppdragsgiver
  - Tiltak: Jobbe mer så en tar igjen fremdriftsplan, hvis dette ikke er mulig må oppgaven revalueres.

DEL : 2

Kravspesifikasjon

2 Kravspesifikasjon

Denne kravspesifikasjonen bygger på den generelle kravspesifikasjon for teknisk datasystem. Den er basert på «The STARTS Purchasers' Handbook» kapitel 4 og appendix B, oversatt til norsk av Frode Haug.

Ut fra denne malen har vi valgt å se bort fra del 1 som er for beslutningstagerne/sjefene siden vi ikke trenger å overtale noen til å satse på prosjektet, da dette allerede er bestemt gjennomført. Vi har også valgt å droppe noen av punktene fra malen som ikke var relevante for vår oppgave.

Vi har måtte forandre opprinnelige nummereringen for å tilpasse denne til resten av rapporten. Del 2 i den opprinnelige malen har blitt til 2.1 i denne rapporten, tilsvarende har del 3 blitt 2.2 osv.

Innholdsfortegnelse

2.1 : Bruker beskrivelse

2.1.1 Omgivelser:

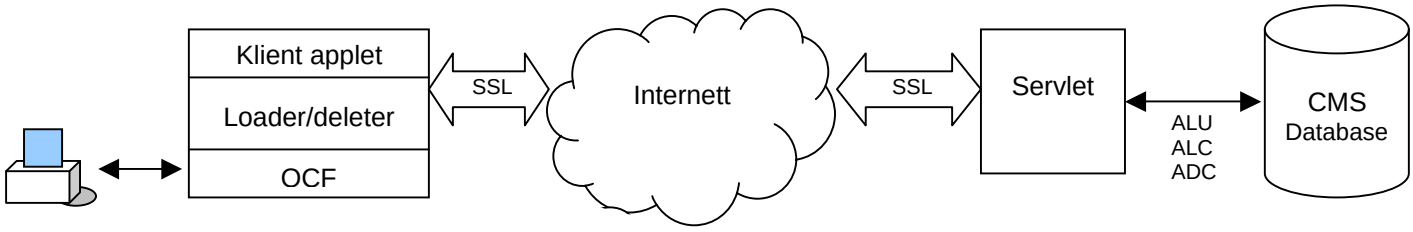
Systemet vil bestå av to deler. En klient og en server del. Klientprogrammet skal fungere på datamaskiner som kan kjøre en Java plattform (Java virtual machine). For eksempel IBM kompatible og Mac. Klienten må også være utstyrt med en smartkort leser og være tilkoblet internett.
Serverprogrammet skal gå som en servlet på en Windows NT maskin. For å få fysisk tilgang til denne maskinen må vedkommende ha høy sikkerhetsklarering hos ErgoGroup.

2.1.2 Systemets brukere:

De finnes to forskjellige brukergrupper. Disse er sluttbrukeren og administrator. Sluttbrukeren må kunne sette opp smartkort, kortleser og drivere på sin lokale maskin. Dette vil bli forklart gjennom en bruksanvisning som følger med kortleseren og er ikke noe vi skal utarbeide. Sluttbrukeren trenger ingen forkunnskaper for å benytte klientprogrammet. Programmet skal være selvforklarende.

Administrator må ha god kunnskap om smartkortteknologien og om hvordan applikasjoner generelt lastes. Han må ha en god forståelse av hvordan systemet fungerer (servlet og klient). Det er ikke en del av vår oppgave å ha opplæring i bruken av systemet. Dette er noe ErgoGroup selv må ta initiativet til.

2.1.3 Funksjon:



figur 2.1.3

Klienten vil snakke med smartkortet ved hjelp av Open Card Framework (OCF). OCF er ett rammeverk for Java som gir mulighet for å sende komandoer til kortet.
Loader/deleter modulen vil generere enkeltkomandoer utifra pakker den mottar fra servleten som sendes videre til OCF modulen.
SSL krypteringen vil bli tatt hånd om av egne moduler.
Servleten er delt i fire hovedmoduler. Den ene er kommunikasjon med klienten, hvor den krypterer og sender sett med komandoer. Den andre delen vil hente appliksjoner/sertifikater fra en database. Den tredje delen vil konvertere det som hentes fra databasen til kommandosett. I tillegg vil servleten bestå av en kontroller som virker som et bindeledd mellom de andre modulene.

2.1.4 Operasjon:

Systemet er ikke beregnet for kommersielt bruk, men mer som en prototype. Derfor er kravet til oppetid ikke en prioritert faktor. Ved normal drift vil programmet sende mange datapakker med relativt lite størrelse (< 5kB). Det er forventet at lasting av en komplett applikasjon skal ta mindre en 30 sekunder på en normal forbindelse.(> 33 600 bps).

Ved en feilsituasjon på servlet eller klienten vil brukeren informeres om feilen.

2.1.5 Aspekter omkring livssyklus:

Systemet er hovedsakelig ment som en prototype, men skal enkelt kunne videreutvikles til kommersielt bruk. Dette er et aspekt vi må ta hensyn til under utviklingen. For å hjelpe senere utvikling, må vi holde en god dialog med EG, samt utarbeide en detaljert og lettfattelig dokumentasjon.

2.1.6 Ytelse:

Oppgaven går ikke ut på å lage ett system som støtter flere brukere. Det er nok at det fungerer på en og en bruker av gangen. Ved videre utvikling vil flerbrukerstøtte bli ett meget viktig punkt.

2.1.7 Begrensninger:

Software:

- Klient og server skal fungere på Windows 95/98/2000/NT
- Klienten skal fungere under Internet Explorer 4.0 eller høyere. Dette betyr at den ikke trenger å være kompatibel med andre nettlesere.
- For å kunne kjøre klient appleten må maskinen ha installert java 2 v1.3.

Hardware:

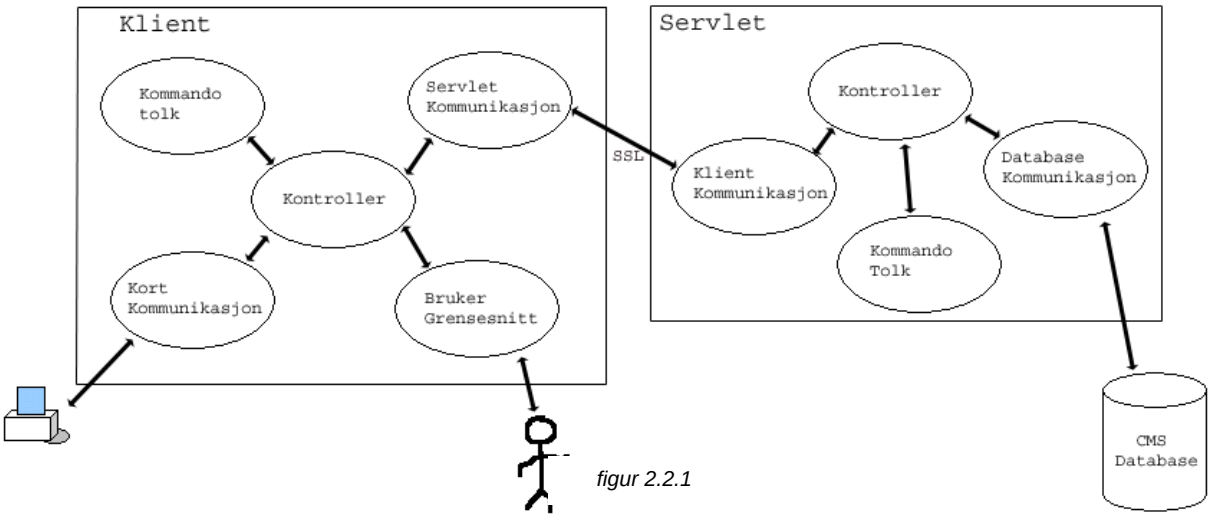
- Klient og server skal fungere på en IBM kompatibel datamaskin. Pentium eller høyere.

2.1.8 Antagelser:

SSL krypteringen som benyttes mellom klient og server er fortsatt under utvikling. Det er utgitt prøveversjoner, men disse kan inneholde småfeil. Det antas at SSL krypteringen vil være sikker nok til det formålet det skal benyttes.
Oppdragsgiver ønsker at servleten skal hente applikasjoner/sertifikater fra ett annet system, card managment system (CMS). Dette systemet er ennå ikke fullt utviklet. Derfor må det utvikles en modul som enkelt kan endres til å støtte CMS.

2.2 : Detaljert kravspesifikasjon

2.2.1 Funksjonell struktur og tverr-relasjoner



figur 2.2.1

Systemet er delt i to hoveddeler, en klient og en servlet del. Klienten mottar en forespørsel fra brukeren, dette fører til at klienten sender en last/slett kommando til servleten. Dette fører til at servleten leter etter rett sertifikat/applikasjon i databasen. Finnes dette, sendes en pakke med kommandoer tilbake til klienten. Denne pakken vil inneholde flere smartkort instrukser, som klienten må pakke ut for så å sende til kortet. Klienten vi ha ansvaret for at instruksjonene ble akseptert. Kontrollerene er de delene som har mest ansvar. Kontrollerenes oppgave er å samordne dataflyt mellom metoder og klasser innad i klienten og servleten.

2.2.2 Data spesifikasjon og dataordliste

2.2.2.1 Data rammeverk

- Servleten skal jobbe mot en database, datastrømmen mellom disse vil være SQL kommandoer.
- Kommunikasjonen mellom servlet og klient skal være kryptert ved hjelp av SSL. Instruksjoner som hentes fra databasen vil bli pakket til en datablokk, som så sendes til klienten. Antall instruksjoner pr pakke skal være enkelt å justere, dette for å finne den mest optimale løsningen i forhold til hastighet.
- Kommunikasjonen mellom klient og kort vil være MULTOS kommandoer. Her vil en og en instruksjon sendes til kortet. Kortet vil så kvittere for hver mottatte instruksjon.

2.2.2.2/2.2.2.3 Data input og output

Servlet vil ha to I/O kanaler:

- Kommunikasjon med database.
  Her sendes det SQL setninger til databasen. Database motoren vil håndtere SQL setningen og ut fra dette returnere et resultat sett til servleten.
- Kommunikasjon med klient.
  Dette er en kommunikasjon som skal gå over internett, derfor må denne datastrømmen krypteres (SSL). Til klienten vil det sendes krypterte datapakker, mens fra klienten mottas det krypterte datapakker.

Klienten vil ha tre I/O kanaler:

- Kommunikasjon med servlet.
  Samme som kommunikasjon med klient på servlet.

- Kommunikasjon med smartkort.
  Her vil det gå en datastrøm med MULTOS instruksjoner til kortet. Kortet vil behandle hver instruksjon og sende svar tilbake på om instruksjonen ble godtatt. Det er viktig å merke seg at kortet ikke tar initiativet til å sende noe til klienten, det er alltid klienten som må starte data utvekslingen.
- Bruker grensesnitt.
  Kontrolleren vil sende en liste til brukeren over applikasjoner som finnes på kortet som kan slettes og applikasjoner i databasen som kan lastes inn på smartkortet. Brukeren vil velge fra listene hvilken applikasjon som skal lastes eller slettes. Det vil da sendes en melding til kontrolleren om at applikasjonen skal lastes/slettes.

2.2.2.4 Tverr-funksjonale data definisjoner

- Kort-id (Kort – Database)
  Ved oppstart eller innsetting av smartkort, vil kort-id hentes fra kortet. Denne identifikatoren sendes til servleten. Servleten vil spørre databasen om hvilke applikasjoner som finnes på kortet og hva som kan laster på det.
- Liste med applikasjoner som kan lastes/slettes. (Database – bruker)
  Listene som servleten hentet fra databasen sendes tilbake til klienten. Klienten vil vise denne listen til brukeren.
- Forespørsel om lasting/sletting (Bruker – Database)
  Brukeren velger ut fra listen hvilken applikasjon som skal slettes/lastes. Denne forespørselen sendes til servleten sammen med kort-id. Servlet vil da hente ut applikasjonen og sertifikater fra databasen.
- Smartkort instruksjoner (Database – Kort)
  Applikasjonen og sertifikatene som ble hentet ut omformes til instruksjoner før de sendes til klienten. Klienten sender så en-og-en instruksjon til smartkortet.

Her er det viktig å merke seg at dataflyten vil forandre seg fra modul til modul. Det kan foretaes endringer av de data underveis, som for eksempel at servlet pakker inn flere instruksjoner før de sendes til klienten. Klienten pakker opp igjen pakkene, før hver enkelt instruksjon sendes til kortet.

2.2.3 Overordnede operasjonelle systemkrav

2.2.3.1 Normal Operasjon / feilsituasjon

2.2.3.1.1 Modus og kontroll

Det finnes fire modi for servlet:

- Oppstart.
  Initialisering av moduler, oppretter tilkopling til database.
- Operativ.
  Kommuniserer med klient, pakker data, henter i DB.
- Feil.
  Logger feil og avslutter.
- Avslutting.
  Fjerner instanser av objekter som er generert, stenger DB forbindelse.

Det finnes fem modi for klienten :

- Oppstart.
  Generere instanser av objekter.
- Operativ.
  Behandle input fra bruker, utføre forespørsel.
- Vente tilstand.
  Venter på forespørsel fra klient, bruker tilnærmet ingen ressurser
- Feil.
  Rapportere til bruker.
- Død.
  Fjerne instanser av objekter som er generert.

2.2.3.1.2 Ytelse

Systemet skal kunne være lett å bruke. Brukeren vil ikke få muligheten til å gjøre feil. Dette vil robustheten i klienten og serveren forhindre. Brukeren vil kun få tilgang til applikasjoner som det er plass til og som kan lastes på kortet.
Systemet trenger ikke å støtte flere enn en bruker om gangen. Men en utvidelse av dette vil bli aktuelt når systemet kommer i kommersiell drift.
Ett krav til ytelsen er at en operasjon mot kortet (lasting/sletting) skal ikke ta mer en 30 sekunder.

2.2.3.1.3 Sikkerhet

Serveren vil befinne seg i sikre omgivelser hos ErgoGroup og vil derfor ikke være spesielt utsatt for fysisk fare. Ett problem som kan skade systemet er at en uvedkommende /uerfaren person legger feilaktige data inn i databasen. Ett annet faremoment er at uvedkommende prøver å lese av/endre datastrømmen mellom en klient og servlet. Dette kan forhindres ved hjelp av kryptering.

2.2.3.1.4 Innebygde tester

Klienten vil rapportere feil ved:

- Ugyldig kort
- Ingen kommunikasjon med kortet
- Mislykket lasting/sletting operasjon
- Ingen kontakt med server
- Plassmangel på kortet.

Servlet vil rapportere feil ved:

- Feil i forbindelsen med databasen
- Ukjent forespørsel fra klient

2.2.4 Funksjonelle krav

2.2.4.1 Funksjonelle krav til bruker grensesnitt (klient)

2.2.4.1.1 Input fra kontroller

Mottar status meldinger om hvilke operasjoner kontroller modulen utfører. Når kontroller laster instruksjoner til smartkortet, vil en progressbar gradvis øke, dette for at brukeren skal se progresjonen.
I tillegg skal brukeren informeres om hvilke applikasjoner som kan lastes på kortet og hvilke som kan slettes.

2.2.4.1.2 Prosessering

Omformer mottatte statusmeldinger til nyttig brukerinformasjon som vises til brukeren. Dette innebærer oppdatering av progressbar og tekst til brukeren.

Genererer meldinger ut ifra brukerens ønsker, og disse sendes til kontroller modulen. Brukeres ønsker vil være å laste eller slette en applikasjon.

2.2.4.1.3 Output til kontroller

Sender meldinger om brukerens ønsker til kontroller om å laste/slette en applikasjon.

2.2.4.2 Funksjonelle krav til kort kommunikasjon (klient)

Denne modulen vil benytte et ferdig OCF bibliotek.

2.2.4.2.1 Input fra kontroller

Modulen vil mottar en og en smartkort instruksjon.

2.2.4.2.2 Prosessering

Omforme og videresende instruksjonene til driveren til smartkortleser.
Disse instruksjonene vil bli behandlet av smartkortet. Gyldigheten på instruksjonen sendes som svar tilbake til modulen.

2.2.4.2.3 Output til kontroller

Statusmelding fra kortet og eventuelle feil som kan ha oppstått med kommunikasjon med kortet.

2.2.4.2.4 Kontroll

- Kommunikasjonen med kortleser kontrolleres, er ikke kortet i kortleses vil bruker informeres.
- Vellykket overføring av data. Når en instruksjon sendes til kortet forventes det at den ble godtatt slik at neste instruksjon kan lastes. Vis det under lastingen skulle oppstå en feil, vil lastingen avbrytes og kontroller informeres.

2.2.4.3 Funksjonelle krav til servlet kommunikasjon (klient)

2.2.4.3.1 Input fra kontroller

- Meldinger om hvordan operasjoner har gått.
- Ønsker fra brukeren om å laste/slette en applikasjon. Dette sendes sammen med kort-id.
- Ved oppstart sendes en melding om kort-id. Dette så servlet kan respondere med en last/slett liste.

2.2.4.3.2 Prosessering

- Kryptere mottatt informasjon fra kontroller og sende dette videre til servlet.
- Dekryptere mottatte data fra servlet og videresende til kontroller.

2.2.4.3.3 Output til kontroller

- Feilrapportering ved dekryptering.
- Ferdig dekrypterte data fra servlet.

2.2.4.3.4 Kontroll

- Kontroll på dekryptering

2.2.4.4 Funksjonelle krav til Kommando tolk (klient)

2.2.4.4.1 Input fra kontroller

- Mottar pakker med instruksjoner.

2.2.4.4.2 Prosessering

- Deler opp pakken i enkelt instruksjoner.

2.2.4.4.3 Output til kontroller

- Vektor med instruksjoner.
- Eventuelle feil ved opp-pakking.

2.2.4.4.4 Kontroll

- Sjekker at pakker er riktig lengde og format.

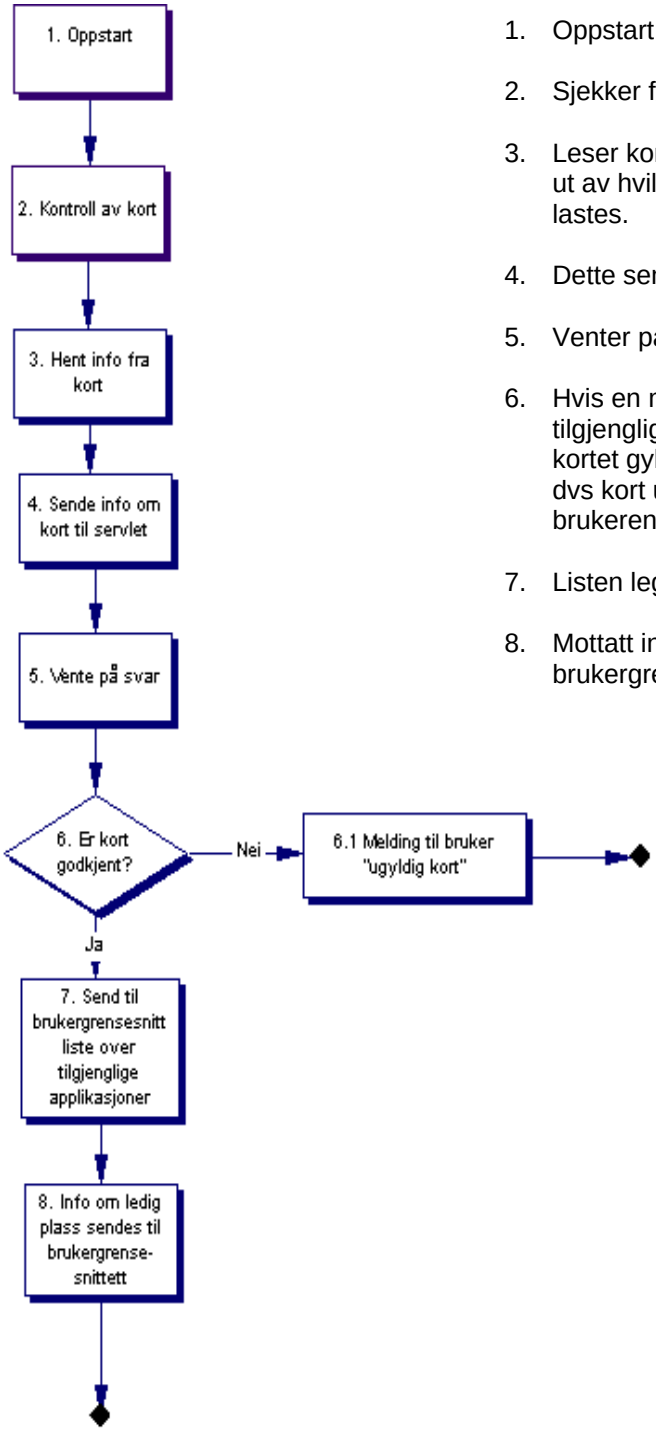
2.2.4.5 Funksjonelle krav til kontroller (klient)

2.2.4.5.1 Input

- Fra kort kommunikasjon vil det bli mottatt statusmeldinger, som forteller hvordan den siste instruksjonen mot kortet gikk. I tillegg vil det ved innsetting av nytt kort i kortleser bli mottatt en melding, som forteller at kortet er klar til bruk.
- Fra kommando tolk vil det mottas en vektor med MULTOS instruksjoner
- Fra servlet kommunikasjon blir det mottatt pakker med instruksjoner, og informasjon til brukeren.
- Fra brukergrensesnittet mottas det informasjon om brukerens ønsker. Dette vil være å laste/slette en applikasjon.

2.2.4.5.2 Prosessering

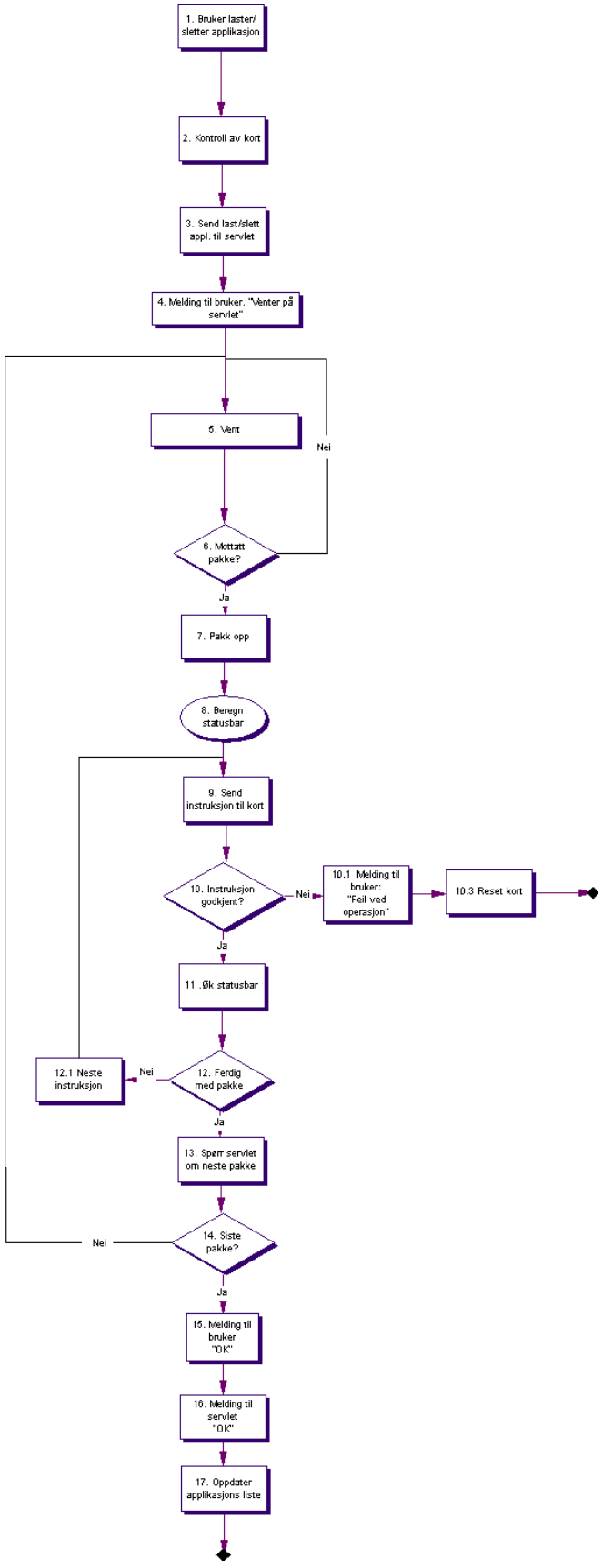
Oppstart



1. Oppstart av applet(init();).
2. Sjekker først at kort er i leser.
3. Leser kort-id fra kortet. Dette for å finne ut av hvilke applikasjoner som kan lastes.
4. Dette sendes til servleten.
5. Venter på svar fra servlet.
6. Hvis en mottar en liste med tilgjenglige/lastede applikasjoner, så er kortet gyldig. Ellers så er listen tom, dvs kort ugyldig. Dette meldes til brukeren.
7. Listen legges ut i brukergrensesnitt.
8. Mottatt info om ledig plass sendes til brukergrensesnitt.

figur 2.2.4.5.2a

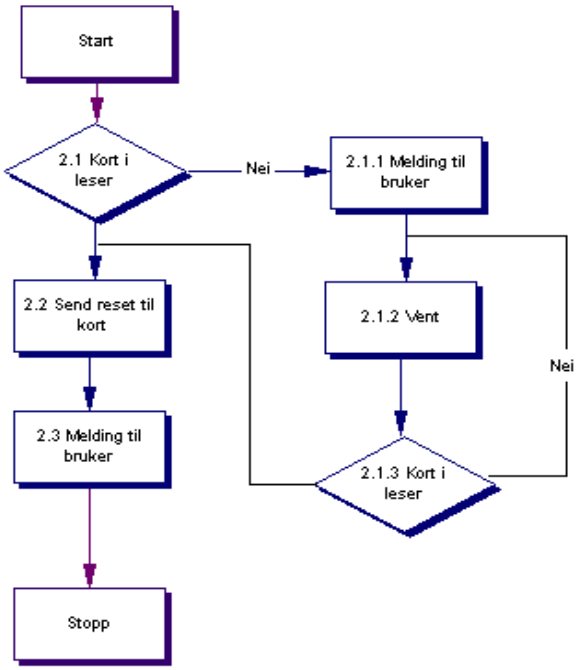
Lasting/sletting av en applikasjon



1. Dette startes av brukeren ved at han velger en applikasjon og trykker last eller slett.
2. Kortkontroll modul kalles.
3. Sender til servlet at kort med kort-id x ønsker å laste/slette applikasjon. Spør om første pakke.
4. Bruker informeres
5. Venter på pakke.
6. Venter på pakke.
7. Pakker opp den mottatte datapakken. Denne deles opp i en vektor.
8. Beregner statusbar.
9. Sender instruksjon til kort.
10. Hvis instruksjon ikke ble godtatt, informeres bruker og servlet. Kortet restartes og operasjon avsluttes.
11. Øker statusbar.
12. Hvis det er flere instruksjoner i pakken vil neste instruksjon lastes.
13. Hvis det ikke er flere instruksjoner i pakken spørres servlet om neste pakke.
14. Får man en ny pakke starter prosessen på ny i punkt 5.
15. Var det siste pakke så får brukeren melding om at ”lasting/sletting OK”.
16. Servlet informeres om at lasting/sletting på kort-id x gikk ”OK”.
17. Applikasjonesliste oppdateres.

figur 2.2.4.5.2b

Kontroll av kort modul



figur 2.2.4.5.2c

2.1 Sjekker mot kort kommunikasjon om kort er i kortleser. Hvis ja 2.2, nei 2.1.1.

2.1.1 Sender melding til bruker om å sette inn kort i kortleser

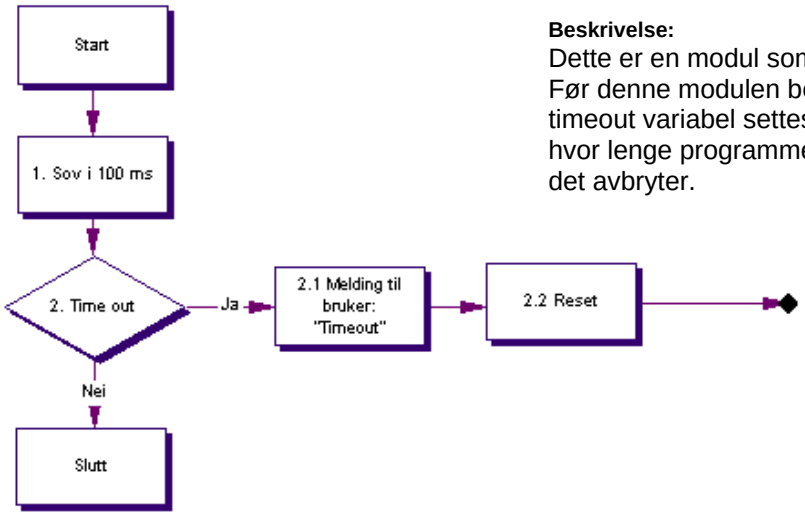
2.1.2 Kjører vent modul

2.1.3 Samme som 2.1

2.2 Sender Reset instruksjon til kortet. Det fører til at minne nullstilles og andre operasjoner som kan være startet mot kortet blir avbrutt.

2.3 Sender melding til bruker at kortet er i leser og klar til bruk.

Ventemodul



Beskrivelse:
Dette er en modul som utfører venting. Før denne modulen benyttes må en timeout variabel settes. Denne forteller hvor lenge programmet skal vente før det avbryter.

figur 2.2.4.5.2d

2.2.4.5.3 Output

- Til kort kommunikasjon sendes en og en MULTOS instruksjon.
- Til kommando tolk sendes det pakker med instruksjoner.
- Til servlet kommunikasjon blir det sendt last/slett forespørsler av en applikasjon. Det vil ved oppstart også bli sendt informasjon om kortet.
- Til brukergrensesnitt sendes det en vektor med lastbare/eksisterende applikasjoner på kortet. Mens applikasjoner lastes/slettes vil det i tillegg bli sendt progresjon og statusmeldinger.

2.2.4.5.4 Kontroll

- Sjekker at kort er i kortleser før kommandoer sendes til kort.
- Sjekker at en og en operasjoner mot kortet blir godkjent (ingen feil)

2.2.4.5.7 Feilrapportering

Dette gjøres mot brukeren.

2.2.4.5.8 Gjenervervelse etter feil

Ved eventuelle feil skal systemet ikke prøve på nytt, men avslutte operasjonen og gi melding til brukeren. Dersom kontrolleren, etter å ha mislykket, fortsetter å sende kommandoer til kortet, vil kortet etter seks forsøk låses og bli ubrukelig.

2.2.4.6 Funksjonelle krav til Database kommunikasjon (servlet)

2.2.4.6.1 Input fra kontroller

Mottar meldinger for å spørre databasen.

2.2.4.6.2 Prosessering

Omforme melding til SQL format.

2.2.4.6.3 Output til kontroller

Datablokker som inneholder applikasjoner og sertifikater. I tillegg vil det sendes en melding om oppgitt kort-id er gyldig.

2.2.4.6.4 Kontroll

Oppnådd kommunikasjon med database.

2.2.4.6.7 Feilrapportering

Bruker må informeres om feil på serveren.

2.2.4.7 Funksjonelle krav til kommando tolk (servlet)

2.2.4.7.1 Input fra kontroller

Mottar datablokker med applikasjon og sertifikat.

2.2.4.7.2 Prosessering

Omforme mottatte datablokker til instruksjoner, som legges i datapakker. En datapakke består av et bestemt antall instruksjoner. Antall instruksjoner i en datapakke skal enkelt kunne varieres, dette for å finne ut hva som er det optimale.

2.2.4.7.3 Output til kontroller

Ferdig pakket instruksjoner.

2.2.4.8 Funksjonelle krav til klient kommunikasjon (servlet)

2.2.4.8.1 Input fra kontroller

- Pakker med instruksjoner.
- Liste over applikasjoner som kan lastes/slettes, og ledig plass på kort.

2.2.4.8.2 Prosessering

- Kryptere mottatt informasjon fra kontroller og sende dette videre til klient.
- Dekryptere mottatte data fra klient og videresender til kontroller.

2.2.4.8.3 Output til kontroller

- Feilrapportering ved dekryptering.
- Ferdig dekrypterte data fra klient.

2.2.4.8.4 Kontroll

- Kontroll på dekryptering.

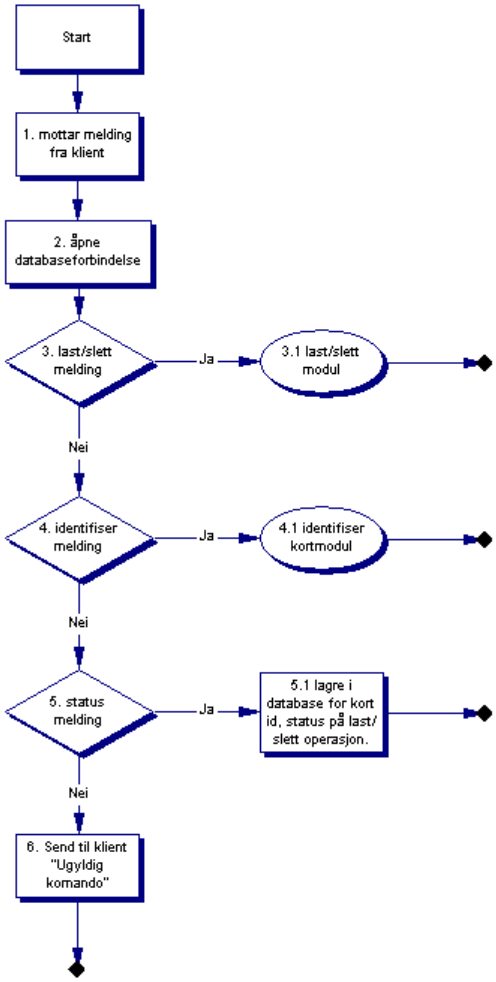
2.2.4.9 Funksjonelle krav til kontroller (servlet)

2.2.4.9.1 Input

- Fra klient kommunikasjon vil det mottas meldinger om hva brukeren ønsker å laste/slette. Det vil også bli mottatt forespørsler om å identifisere kort-id. I tillegg vil det bli mottas melding om å sende neste datapakke.
- Fra kommando tolk mottas det datapakker
- Fra database kommunikasjon kommer det datablokker, og melding om kortet er gyldig.

2.2.4.9.2 Prosessering

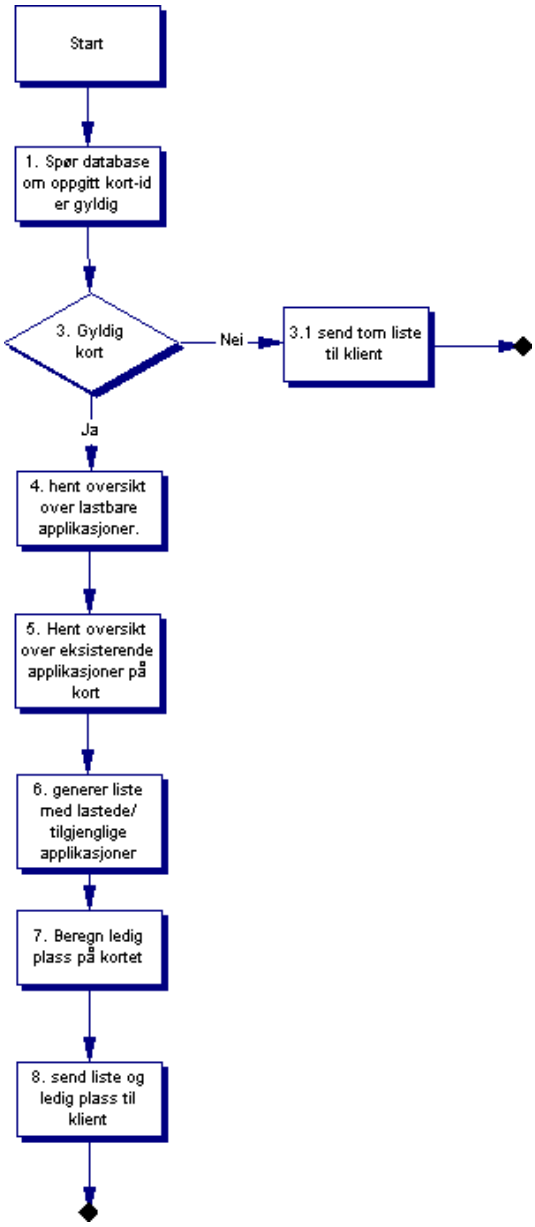
Oppstart



figur 2.2.4.9.2a

1. Servlet starter opp ved at den mottar en forespørsel fra klienten.
2. Forbindelsen med databasen åpnes.
3. Hvis brukeren ønsker å laste/slette en applikasjon så startes last/slett-modulen.
4. Hvis brukeren ønsker å identifisere sitt kort, startes identifiser modulen.
5. Hvis klienten raporterer status på last/slett opperasjoner så legges dette inn i databasen med tilhørende kort id, status og applikasjon. (5.1)
6. Ellers er det mottatt en ukjent kommando, dette rapporteres tilbake til klient.

Identifiser kort modul



1. I meldingen om å identifisere kortet finnes det også en verdi som forteller kortets identifikasjonsnummer. Denne brukes under en spørring mot databasen om kortet er gyldig.

3. Hvis kortet ikke er gyldig informeres klienten om dette (3.1), og servleten avsluttes.

4. Ellers så hentes det ut fra databasen hvilke applikasjoner som kan lastes på den oppgitte kort id.

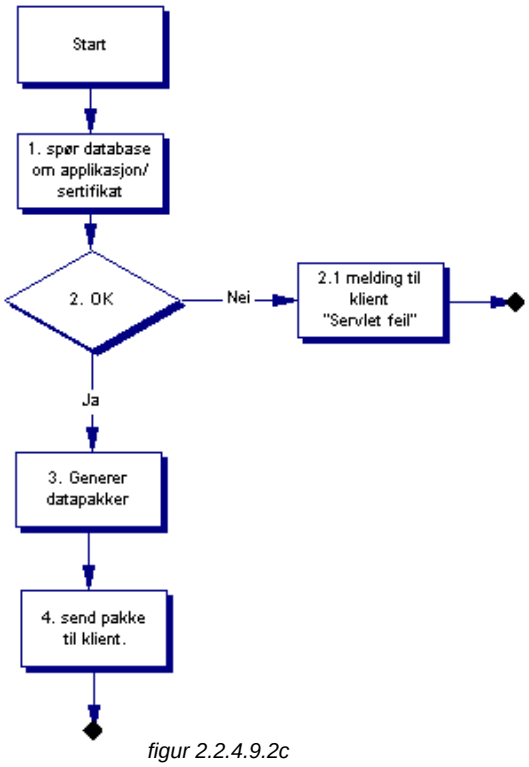
5. I tillegg hentes det fra databasen hvilke applikasjoner som allerede finnes i kortet.

6. Ut fra det som ble hentet fra punkt 4 og 5, genereres det en liste med eksisterende/tilgjenglige applikasjoner.

7. Det beregnes også hvor mye ledig plass det er på kortet, ut fra minnestørrelse og applikasjoner lastet på kortet.

8. Listen, og hvor mye ledig plass som finnes på kortet sendes til klienten.

Last/slett modul



figur 2.2.4.9.2c

0. Når klienten spør servleten om å slette/laste en applikasjon, blir det i tillegg sendt med en kort id og hvilke pakke klienten ønsker.

1. Ut fra hvilken applikasjon det gjelder, spørres databasen om å hente opp selve applikasjonen samt last/slett sertifikat.

2. Hvis dette ikke gikk så får klienten melding om dette og servleten avsluttes.

3. Ellers så vil datapakkene genereres. Hvis det er en last forespørsel pakkes applikasjonen og last sertifikatet. Hvis det var en slett forespørsel pakkes kun slett sertifikatet.

4. Den pakken som klienten forespurte om, sendes tilbake før servleten avsluttes.

2.2.4.9.3 Output

- Til klient kommunikasjon sendes det datapakker og en liste over applikasjoner som kan lastes/slettes, samt hvor mye ledig plass som finnes på kortet.
- Til kommando tolk sendes det datablokker.
- Til database kommunikasjon sendes det meldinger om å spørre/legge til i databasen.

2.3 : Begrensninger

2.3.1 Software design begrensninger

2.3.1.1 Software standarder og språk

Applikasjoner som skal benyttes under utviklingen er :

- Borland JBuilder versjon 4
  Skal benyttes til all Java koding i forbindelse med klient og servlet.
- Borland JBuilder CVS
  Dette er en revisjonsmodul som er innebygd i JBuilder som skal benyttes til versjonskontroll av kildekode.
- Together 4.2
  Moduleringsverktøy.
- MS Word
  Dokumentasjon.

2.3.1.2 Software grensesnitt

- Kort grensesnitt
  ISO7816-4 for kommunikasjonsstandarder mot smartkort
- Database grensesnitt
  ISO9075 Standard series for structured query language

  (Internett link til hvor ISO standarder finnes, ligger i litteratur listen)

2.3.1.3 Software pakker/verktøy

For å benytte systemet vil klienten måtte ha et windows operativsystem installert på sin maskin samt en versjon av nettleseren Internet Explorer 4.0 eller høyere.

2.3.1.4 Software kommunikasjonsstandarder og grensesnitt.

Kommunikasjonen i systemet vil foregå over internett og vil derfor benytte seg av TCP/IP protokollen. Dataene vil bli sendt til en IP-adresse på port 80. Alle dataene som blir sendt via internett vil også bli kryptert ved bruk av SSL kryptering.

2.3.1.5 Database

Databasemodulen skal være enkelt utskiftbar. I pilotsystemet skal det utvikles en MySql testdatabase. Ved en eventuell kommersiell drift så skal denne modulen byttes ut så den støtter card managment system (CMS).

2.3.1.6 Operativsystem

Systemet, når det er ferdig utviklet skal gå på en Java virtual machine. Det betyr at det i praksis kan benyttes på de fleste operativsystemer, Windows9x/2000/nt, mac, Linux. Det som trengs for å benytte systemet er en nettleser som støtter Java applets.

Men pilot systemet er i første hånd beregnet til å kjøre på et Windows system med nettleseren Internet Explorer versjon 4 eller høyere.

2.3.1.7 Toleranser, marginer og muligheter/tilfeller

Kortene som skal jobbes med har en lagringskapasitet på 16Kb. Av denne plassen vil ca. 3 Kb benyttes av operativsystemet MULTOS. Den resterende plassen kan da benyttes til applikasjoner som bruker ønsker på kortet. Ved lasting av applikasjoner til kortet sendes en forespørsel til servleten. Servleten prosesserer forespørselen og lager en pakke med alle kortinstruksjoner og applikasjonen som den sender til klienten. Klienten pakker ut dataene og sender disse til kortet. Denne operasjonen skal ikke overskride 30 sekunder.
Dersom applikasjonen som ønskes lastes er større en ledig plass på kortet vil lastingen automatisk bli avbrutt av MULTOS.

2.3.2 Hardware design begrensninger

2.3.2.1 Hardware krav og omgivelse

Det som kreves av hardware på klienten er en kortleser.
refererer til modul 3.4.2 kort kommunikasjon.

2.3.2.2 Hardware standarder

Kortleseren og smartkort må følge ISO7816 standarden.

- 7816-1: Fysisk karakteristika
- 7816-2: Dimensjoner og plassering av kontaktene
- 7816-3: Elektroniske signaler og transmisjons protokoller

(Link til ISO 7816 finnes i litteratur listen.)

2.3.2.3 Hardware grensesnitt

Grensesnittet mellom smartkort og PC følger ISO7816-3 standarden.

2.4: Aspekter omkring livssyklus

2.4.1 Dokumentasjon

refererer til vedlegg C for beskrivelse av kildekodemal.

2.4.2 Modul- og integrasjonstesting

Ved testing av modulene og samspillet mellom de vil det være naturlig og begynne med GUI klassen. Først få til grensesnittet og få det til å kalle de resterende klassene på klientdelen. Deretter lage en kommunikasjon mellom klienten og servleten der vi oppretter en forbindelse og sender/mottar data.
Da kommunikasjonen mellom klient og servlet er fullført vil neste naturlig steg være å hente ut og legge inn data i databasen, og sende dette mellom klient og servlet. Det siste som vil bli utviklet er det å sende data til og fra kortet.

2.4.3 Konfigurasjons og versjonsstyring

- En prosjekt perm som behandles av loggfører, loggfører har som oppgave å samle inn møtelogg og aktivitetslogg.
- JBuilder inneholder et ferdig konfigurasjons system (CVS). Dette vil holde orden på kode skrevet av forskjellige brukere, samt automatisk dokumentere klasser og funksjoner.
- Loggfører skal skrive ut og sette inn i prosjekt permen all kode, en gang i uken.

2.4.4 Krav til support, service og vedlikehold

Systemet stiller små krav til service og vedlikehold. Ved eventuell videreutvikling er det ErgoGroup som vil ha ansvaret for service.

2.4.5 Krav til utvidelser

En utvidelse av systemet som allerede er klart at vil komme, er å bytte ut MySql database modulen med en CMS modul. Dette er en utvidelse som det er lagt stor vekt på i systemet at skal kunne gjennomføres på en enkel måte.

2.5 : Aspekter omkring installasjon

2.5.1 Hardware installasjon

Hardware installasjons aspektet ved systemet er kun et klientside problem. Klienten må installere en kortleser til sin hjemme-pc enten tilkoblet kommunikasjonsporten (com1, com2 eller parallellport) eller til en USB port. I tillegg må drivere for kortleseren installeres.

Når det gjelder hardware installasjon på server delen så vil dette begrense seg til en PC som er tilkoblet internett, som kan kjøre servleten og med tilstrekklig harddiskplass.

2.5.2 Overgang / omlegging

Skal ikke erstatte ett eksisterende system.

2.5.3 Opplæring

Behovet for opplæring ved bruk av klienten og kortleseren skal være fraværende. Klienten skal lages på en slik måte at alle som skal benytte den bare kan sette seg ned og bruke systemet, men brukeren bør ha en viss kjennskap til bruk av grafisk grensesnitt.
Sammen med kortleseren vil det følge med en enkel brukermanual der det beskrives hvordan kortleseren skal tilkobles PC-en og hvordan tilhørende driverdiskett skal installeres.

Det vil ikke bli gjennomført opplæring på bruk av servlet. Ut fra oppgitt dokumentasjon og tilstedeværelse under utvikling, skal EG selv kunne videreutvikle og vedlikeholde.

2.6 : Akseptanse krav

Det viktigste er at selve lastingen av en applikasjon over internett skal fungere. Dernest at overføring er trygg, dvs implementasjon av SSL. Underordnede krav er å lage et grafisk brukergrensesnitt vha Flash.

2.7 : Prosjektstyring, inkludert kvalitetsikring

2.7.1 Hovedinndeling av prosjektet.

- Forprosjekt
- Kravspek/analyse
- Koding/design
- Testing
- Flash / hjemmeside

2.7.2 Krav til statusmøter og beslutningspunkter.

- Avtalt 3 statusrapport, se *vedlegg B*
  ca. 20 februar, ca. 30 mars og ca. 1 mai
- Beslutningspunkter/milepæler
  - Ferdig forprosjekt
  - Ferdig kravspek
  - Ferdig design
  - Ferdig koding
  - Ferdig testing
  - Ferdig sikkerhetsanalyse
  - Foreta fremføring
  - Foreta webdesign, siste tidspunkt for start.
  - Ferdig rapport.

2.7.3 Rutiner for organisering av kvalitetssikring

2.7.3.1 Dokumentasjon:

- Kode Syntaks, se *Vedlegg C.*
- Møtelogg/ Aktivitetslogg, se *Vedlegg B.*
- Rapport, bruke samme layout som forprosjekt.

2.7.3.2 Verktøy

- Together 4.2
  Brukes for å sette opp en konseptuel modell under design perioden
- JBuilder 4.0
  Benyttes under kodingen
- Milestones Simplisity
  Gantt Skjema.
- Microsoft Word
- Inspiration 6.0
  Utarbeiding av flytdiagram

2.7.3.3 Backup

- Kopiere kode til server/annen maskin på slutten av hver dag.
- En katalog pr. person med katalog for hver dato hvor kildekode lagres.

2.7.3.4 Problemrapportering og tiltak

- Fravær
  - Rapportering: ingen
  - Meldeplikt.
  - Tiltak: Andre gruppemedlemmer tar over oppgaver til fraværende.
- Ikke oppnådd kontakt med veileder/oppdragsgiver
  - Rapportering: møtelogg
  - Tiltak: Purring via email eller telefon
- Ikke overholdt tidsfrist i forhold til milepæl.
  - Rapportering: avvik/status -rapport til veileder og oppdragsgiver
  - Tiltak: Jobbe mer så en tar igjen fremdriftsplan, hvis dette ikke er mulig må oppgaven revalueres.

DEL : 3

Design

3 Design

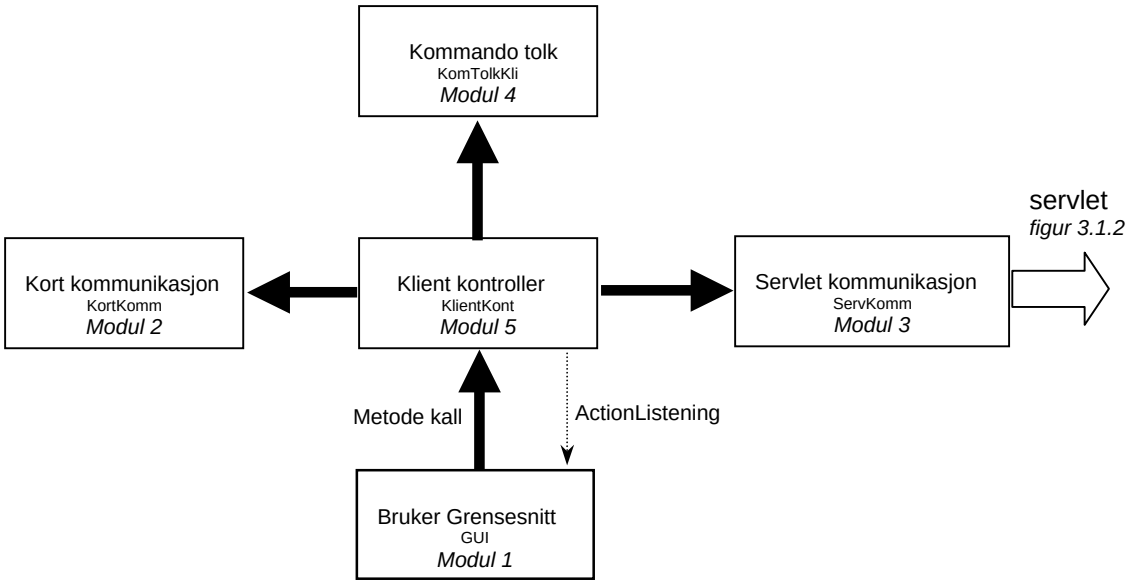
Kravspesifikasjonen beskriver hva systemet skal gjøre, men ikke hvordan en skal gjøre det. Hensikten med dette dokumentet er at det skal gi tilstrekklig informasjon om systemet slik at den/de som skal programmere kan utføre dette uten å vite noe om den organisasjonen de lager systemet for.

Under utarbeiding av designdokumentet valgte vi å lage vår egen mal for hvordan dokumentet skulle bygges opp. I punkt 3.1 har vi valgt å gi en enkel oversikt over hvordan klient og server initialiseres. Deretter gikk vi over til å benytte objektorientert design metoder for å forklare hvordan systemet skulle se ut, og fungere. I punkt 3.2 har vi forklart i detalj vha. kode/pseudokode hvordan klassene er bygd opp og fungerer.

Innholdsfortegnelse

3.1 Overordnet samspill mellom moduler

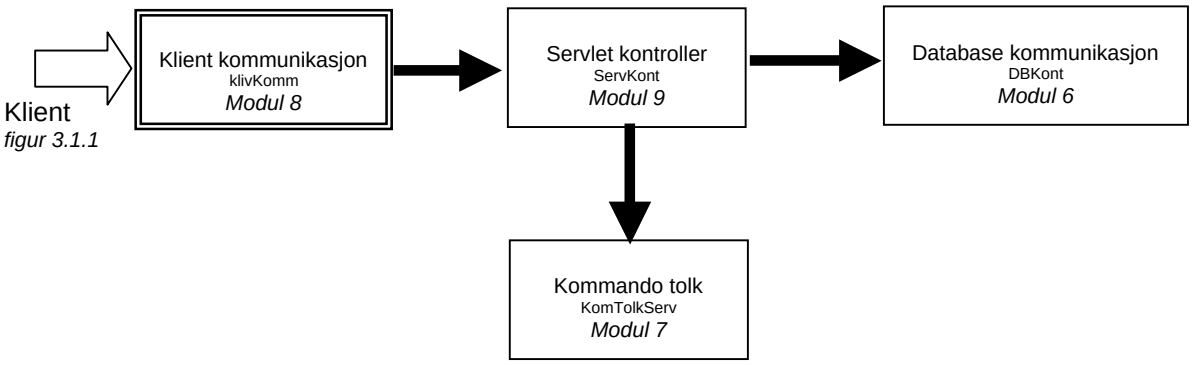
3.1.1 Klient



figur 3.1.1

Det som starter hele systemet, er at en bruker åpner nettsiden som inneholder klient appletten. Den første klassen som da lastes er bruker grensesnittet (modul 1). Denne klassen setter opp det grafiske grensesnittet og instansierer klient kontroller objektet. Klient kontrolleren (modul 5) vil videre instansierer de tre resterende klassene kort kommunikasjon (modul 2), kommando tolk (modul 4) og server kommunikasjon (modul 3). Dette fører til at bruker grensesnittet kan kalle metoder i klient kontrolleren. Klient kontrolleren kan kalle metodene i kort kommunikasjon, tolk og servlet kommunikasjon objektene.
For at et objekt skal kunne sende meldinger tilbake til det objektet som det ble instansiert av det må det brukes en annen metode. I Java heter dette ActionListening. Dette vil bli brukt for å sende meldinger fra klient kontrolleren til bruker grensesnittet. For de tre andre objektene (kort kommunikasjon, kommando tolk og server kommunikasjon) vil retur verdier på metoder kalt av klient kontrolleren være tilstrekkelig.

3.1.2 Servlet



figur 3.1.1

figur 3.1.2

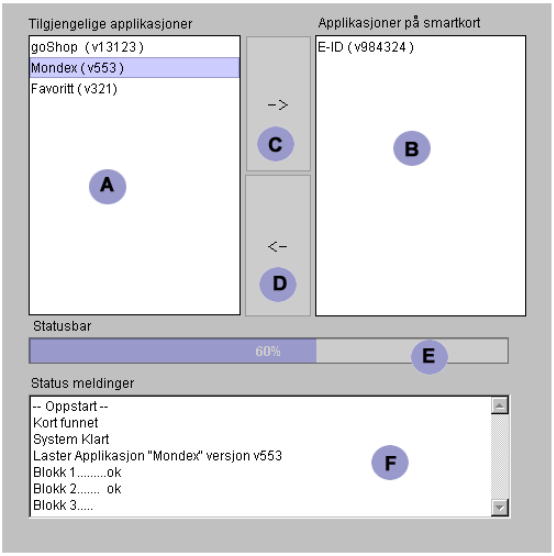
Det som vil starte opp servleten er et kall fra klienten (modul 3 i figur 3.1.1). Klienten vil sende en forespørsel om å laste/slette en applikasjon eller å identifisere et kort. Dette vil bli mottatt av servleten i en vanlig doGet()/doPost() metode. Disse metodene er standard metode som finnes i alle typer servlet program og tilsvarer init() metoden i andre typer program. Klient kommunikasjon (modul 8) vil prøve å dekryptere den mottatte meldingen, lykkes dette vil servlet kontrolleren instansieres (modul 9). Konstruktoren til servleten vil ta den dekrypterte meldingen inn som parameter og retur vil være en pakke med informasjon til klienten. Klient kontrolleren vil instansiere kommando tolk (modul 7) og database kommunikasjonen (modul 6) objektene.

Servleten vil ha et mye enklere hendelsesforløp enn klienten. Servleten vil kun eksistere i perioden fra den har fått en forespørsel fra klienten, til et svar er returnert. I tillegg kan flere servlet objekter kjøres samtidig, hvor hver servlet objekt snakker med sin egen klient applet.

3.1.3 Real Use-Case

Real Use-Case viser et konkret design av hvordan et Use-Case vil bli realisert. Ved hjelp av real Use-Case kan vi forklare hvordan bruker grensesnittet virker. Det vil si i detalj beskrive hva brukeren kan gjøre og hva systemet svarer.

Use-Case:	Lasting/sletting av applikasjoner.
Aktører:	Bruker.
Mening:	Gi bruker mulighet for å laste eller slette applikasjoner på sitt smartkort.
Beskrivelse:	Brukeren setter i smartkortet i kortleser og blir presentert med en liste over tilgjenglige applikasjoner, samt en liste over applikasjoner som allerede finnes på kortet. Brukeren kan så velge hvilke applikasjoner han ønsker å slette fra kortet eller laste på kortet. Statusbaren vil oppdateres ved hver transaksjon og hele tiden vise hvor mye ledig plass det finnes på kortet. Ved lasting/sletting vil statusbaren vise progresjonen ved operasjonen. I tillegg vil brukeren bli presentert med en tekstlig beskrivelse av hva som skjer.



figur 3.1.3

Typisk hendelsesforløp:
Viser til figur 3.1.3.

Aktør operasjoner:	**System svar:**
1. Use-Caset begynner når en bruker setter kortet inn i kortleseren.	
	2. Sjekker om kortet er gyldig.
	3. Fyller inn listene (A) og (B), setter i tillegg statusbaren, (E), til å vise hvor mye ledig plass det er igjen på kortet.
4. Bruker velger fra listene: a.) Hvis lasting (C), se seksjon lasting av applikasjon b.) Hvis sletting (D), se seksjon sletting av applikasjon	
	5. Statusbaren (E) oppdateres til å vise progresjonen over valgt operajon. I tillegg vil meldingsfeltet (F) oppdateres med operasjonsmeldinger.

Alternativt hendelsesforløp:
- Linje 2: Kort ugyldig, brukergrensesnittet vil forbli inaktivt. Viser feilmelding (*figur 3.2.1.1c*).
- Linje 5*:* Feil ved lasting/sletting, bruker får en feilmelding (*figur 3.2.1.1f*).

Seksjon: Lasting av applikasjon

Aktør operasjoner:	**System svar:**
1. Har valgt en applikasjon fra liste (A), og trykket på knapp (C).	
	2. Begynner å laste valgt applikasjon til kortet.

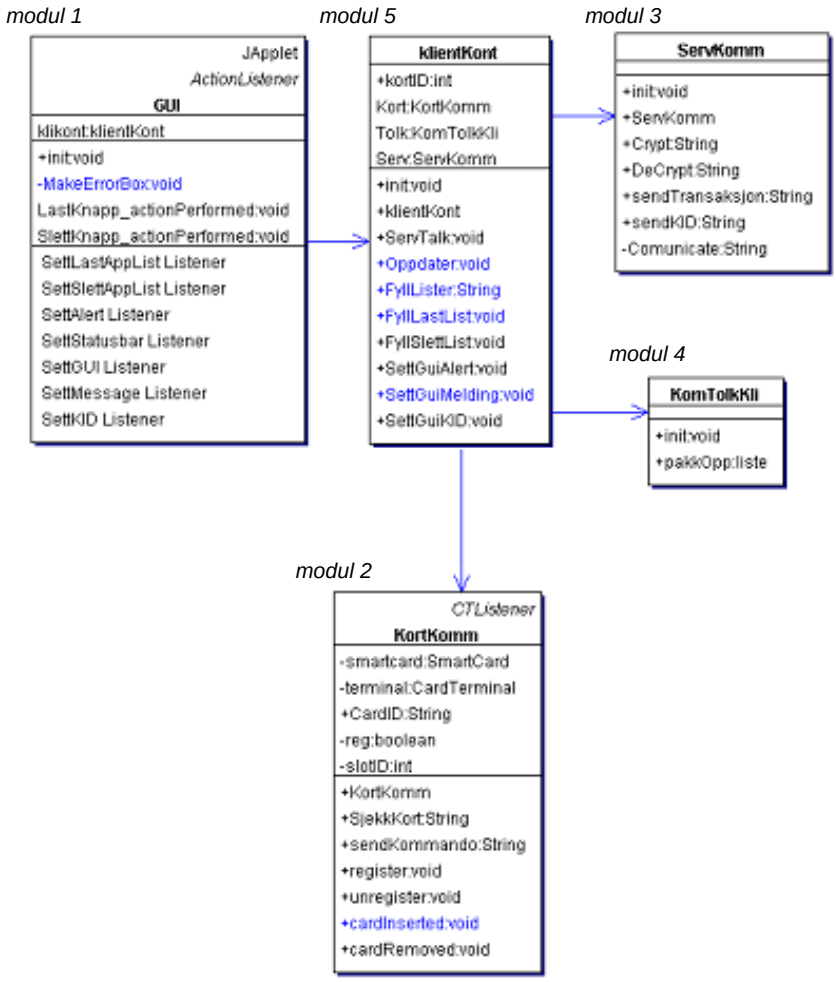
Seksjon: Sletting av applikasjon

Aktør operasjoner:	System svar:
1. Har valgt en applikasjon fra liste B, og trykket på knapp (D).	
	2. Begynner å slette valgt applikasjon fra kortet.

3.1.4 Design diagrammer klient.

Design klasse diagram oppsummerer definisjonene av de forskjellige klassene og grensesnitt mellom de. Viser de viktigste attributter, metoder og klasser som de ulike klassene inneholder. Disse objektene er en videreføring fra figur 3.1.1. Basert på design dokumentet har vi i detaljert design (avsnitt 3.2) forklart mer i detalj hva de ulike metodene i klassene gjør.

3.1.4.1 Design klassediagram



figur 3.1.4.1

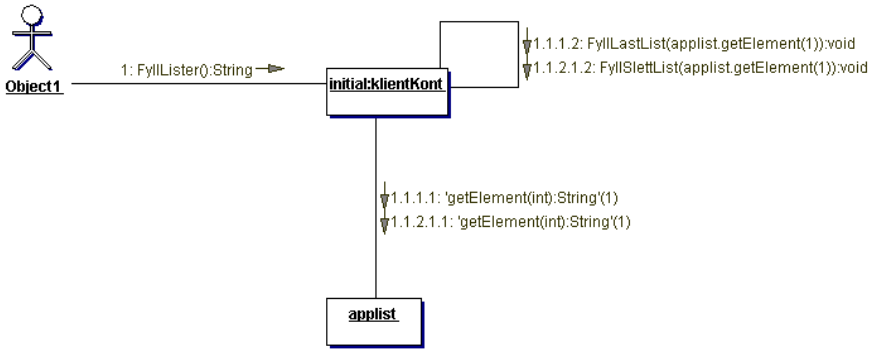
3.1.4.2 Kollaborasjonsdiagrammer.

Illustrerer hvordan objektene kommuniserer ved hjelp av meldinger for å fullføre oppgaver. For hver system operasjon lages det et kollaborasjonsdiagram, hvor system

operasjonen er start meldingen. Deretter beskrives det hva som skal skje i operasjonen ved sekvensiell nummerering av hendelsesforløpet.

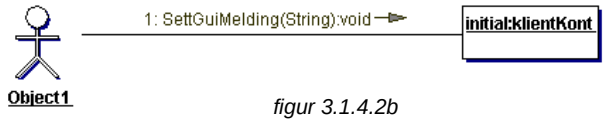
klientKont(modul 5):
Denne klassen er den mest avanserte av klient modulene og eneste av de vi viser med kollaborasjonsdiagram. Grunnen til dette er at denne klassen snakker med alle de andre, og vi kan derfor lage litt større diagrammer. De andre klassene inneholder kun metoder som jobber mot seg selv, og det vil i kollaborasjons diagram kun medføre en start melding(navn på metoden med eventuelle parametere), som er lite informative(eks *figur 3.1.4.2b*).

FyllLister():



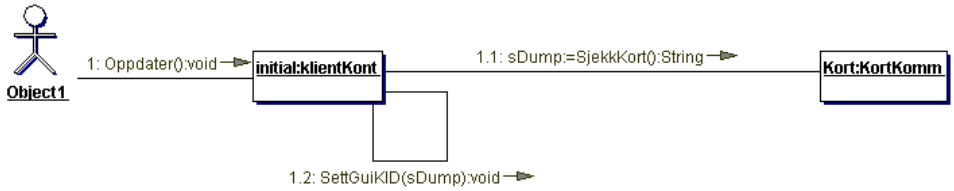
figur 3.1.4.2a

SettGuiMelding(string):



figur 3.1.4.2b

Oppdater():

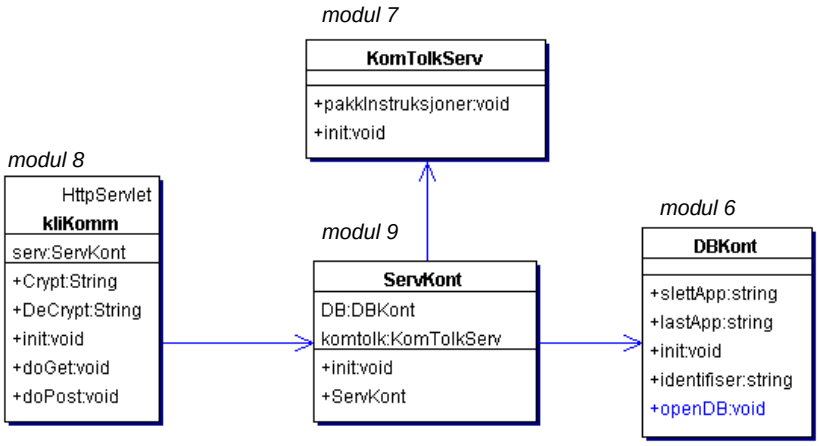


figur 3.1.4.2c

3.1.5 Design diagrammer servlet.

Viser til forklaring gitt i punkt 3.1.4

3.1.5.1 Design klassediagram



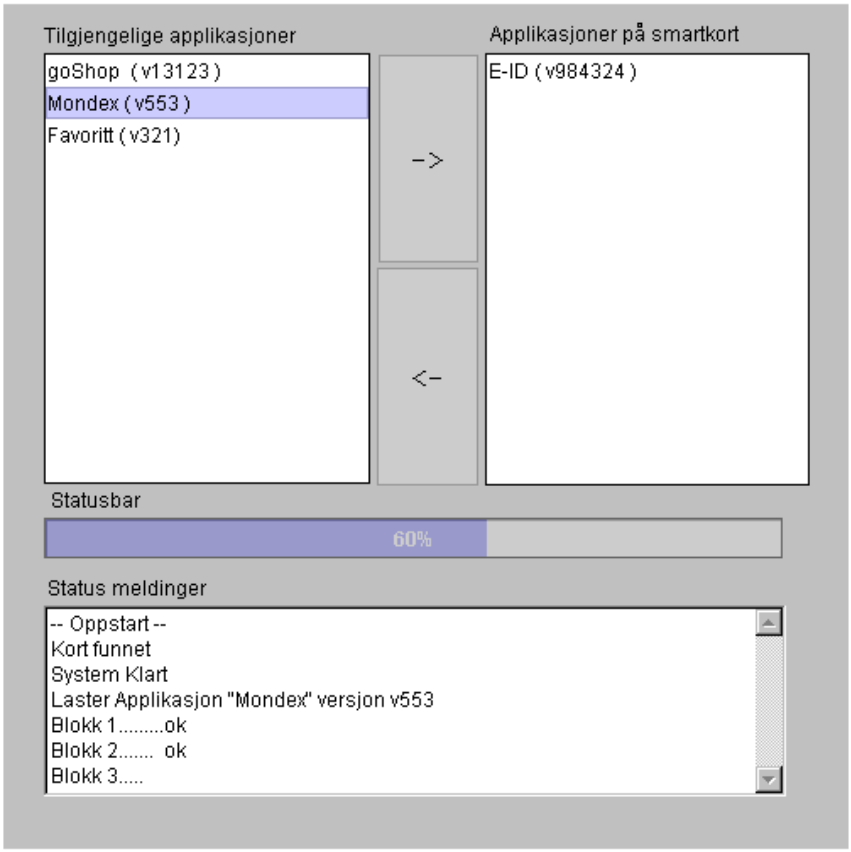
figur 3.1.5.1

3.2 Detaljert design

Ut fra design klasse diagrammene (figur 3.1.4.1 og figur 3.1.5.1), som lister opp hvilke metoder, attributter og klasser de forskjellige modulene har, kan vi nå gi en mer detaljert beskrivelse av hvordan vi har tenkt å gjennomføre de forskjellige metodene i klassene.

3.2.1 Detaljert design for bruker grensesnitt (klient modul 1)

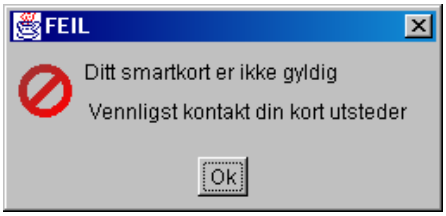
3.2.1.1 Skjermbilder



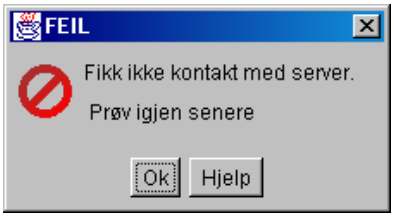
figur 3.2.1.1a

Figur 3.2.1.1a viser bruker grensesnittet. Dette vil være inaktivt til brukeren setter inn et gyldig kort. Ved inaktivt bruker grensesnitt vises figur 3.2.1.1b, og ved ugyldig kort vises figur 3.2.1.1c.

figur 3.2.1.1b



figur 3.2.1.1c



figur 3.2.1.1d

Ved server feil blir brukeren informert om dette ved feilmeldinger (*figur 3.2.1.1d* og *figur 3.2.1.1e*).

figur 3.2.1.1e

Ved feil ved lasting eller sletting av applikasjoner vil brukeren få feilmelding (*figur 3.2.1.1f*)

figur 3.2.1.1f

3.2.1.2 Klasse import

Denne modulen bruker følgende klasser :

Klientkont , *Klient Kontroller* , modul nr 5.
MessageBox , *Meldingsboks* , modul nr 23.

3.2.1.3 Klasse og metode oversikt

```
class GUI  {
        public void init() {
                // Lager utseende , skjermbilde 3.2.1.1a
                // Setter alle input knapper til inaktive.
                // Instansierer klient kontroller ( Klientkont )
                KlientKont kont = new KlientKont();
                // Kaller klient kontrollers oppstart rutine, denne vil
                // laste applikasjons listene.
                        Kont.Oppstart();
        }
        public void LastKnapp_actionPerformed(ActionEvent) {
                // Sender til kontroller at applikasjon skal lastes.
                KlientKont.transaksjon("L",appid);
        }
        public void SlettKnapp_actionPerformed(ActionEvent) {
                // Sender til kontroller at applikasjon skal slettes.
                KlientKont.transaksjon("S",appid);
        }
        class MessageListener implements ActionListener {
                public void actionPerformed (ActionEvent ae) {
                // Meldinger som sendes til GUI klassen legges i
                // statusvinduet.
                }
        }
        class StatusBar_Listener implements ActionListener {
                public void actionPerformed (ActionEvent ae) {
                // Setter statusbar verdi til mottatte verdi fra
                // ActionEvent.
                }
        }
        class SettLastAppList_Listener implements ActionListener {
                public void actionPerformed (ActionEvent ae) {
                // Setter liste med lastbare applikasjoner, fra
                // ae.toString().
                }
        }
        class SettSlettAppList_Listener implements ActionListener {
                public void actionPerformed (ActionEvent ae) {
                // Setter liste med slettbare applikasjoner, fra
                // ae.toString().
                }
        }
        class SettGUI_Listener implements ActionListener {
                public void actionPerformed (ActionEvent ae) {
                // Slår av og på bruker interface, slik at en bruker ikke
                // kan trykke på en knapp mens lasting sletting pågår.
                }
        }
```

```
        class SettKID_Listener implements ActionListener {
                public void actionPerformed (ActionEvent ae) {
                        //Skriver ut kort-id til GUI.
                }
        }

        class Alert_Listener implements ActionListener {
                public void actionPerformed (ActionEvent ae) {
                        // Lager popup bokser med feilmeldinger/informasjon.
                        // Mottar to parametere, alvorsgrad og beskrivende
                        // tekst. Skjermbilde 3.2.1.1b – 3.2.1.1f
                }
        }
}
```

3.2.2 Detaljert design for kortkommunikasjon (klient modul 2)

Viser til figur 3.1.4.1

3.2.2.1 Klasse import

Denne modulen bruker følgende klasser :

OcfLib, OCF bibliotek, modul nr. 20

3.2.2.2 Klasse og metode oversikt

```
class kortKommServ {
        public void init() {
                        //Kjøre sjekkKort() funksjonen.
        }
        public string sjekkKort() {
                        // Sjekk om kort i leser
                        // Henter kort ID, returnerer kortID, eller 0.
        }
        public string sendKommandoer (string kommando) {
                        //Mottar kommando fra kommando tolk og sender
                        //denne til kortet.
                        //Får tilbake svar om overføring.
        }
        public void registrer() {
                        //Registrerer eventHandles, legger til //eventListener.
        }
        public void unregistrer(){
                        //Avregistrerer eventHandles, fjerner //eventListener
        }
        public void cardInserted(CardTerminalEvent ctEvent){
                        //Venter på en cardTerminalEvent, sjekker om //kort
                        blir satt inn i leser.
        }
        public void cardRemoved(CardTerminalEvent ctEvent){
                        //Venter på en cardTerminalEvent, sjekker om //kort
                        blir fjernet fra leser.
        }
}
```

3.2.3 Detaljert design for servlet kommunikasjon (klient modul 3)

Viser til *figur 3.1.4.1*

3.2.3.1 Klasse import

Denne modulen bruker følgende klasser :

SSLLib, SSL bibliotek, modul nr. 21

3.2.3.2 Klasse og metode oversikt

```
class servkom {
        public string crypt(string streng, string key) {
                        //Krypterer streng vha. SSL.
        }
        public string Decrypt(string cipher, string key) {
                        //Dekrypterer streng vha. SSL
        }
        public pakke sendTransaksjon(string type, int appID, int kortID){
                        //Sender til servlet en forespørsel om å //laste/slette
                        en applikasjon. Meldingen vil //først krypteres før den
                        sendes.
        }
        public string sendKID(int kortID){
                        //Sender kort id til servlet, får liste over applikasjoner
                        //tilbake.
        }
        private string Comunicate(String s){
                        //Åpner kommunikasjon mot servlet, returnerer
                        //mottatt melding.
        }
}
```

3.2.4 Detaljert design for kommando tolk (klient modul 4)

Viser til figur 3.1.4.1

3.2.4.1 Klasse import

Ingen

3.2.4.2 Klasse og metode oversikt

```
class kommTolkKli {
        public liste pakkOpp(string streng){
                        //Mottar pakke med kort kommandoer fra
                        //kontroller.
                        //Pakker ut til enkeltinstrukser og sender //tilbake til
                        klient kontroller (modul nr. 5).
        }
}
```

3.2.5 Detaljert design for kontroller (klient modul 5)

Viser til figur 3.1.4.1

3.2.5.1 Klasse import

Denne modulen bruker følgende klasser :

KortKomm, Kort kommunikasjon , modul nr. 2

ServKomm, Servlet kommunikasjon, modul nr 3

KomTolkKli, Kommando tolk, modul nr 4

3.2.5.2 Klasse og metode oversikt

```
class klientKont {
        int kortID;
        public void init() {
                        //Instansierer kortkommunikasjon, //kommandotolk og
                        servlet kommunikasjon.
                }
        public void oppdater() {
                if (kid = kortkomm.sjekkKort() ){
                        liste applist = new liste();
                        applist.settliste(servkom.sendkid(kid));
                        if (applist.valid()){
                                settGuiListe(applist);
                                settGuiAktiv(true);
                        }else{
                                settGuiAlert(applist.errorlevel, applist.error);
                        }
                }else{
                        settGuiAlert(1, ”Kort ikke i leser”);
                }
        }
        public void FyllLastList( String txt ){
                        //Fyller inn tilgjenglige applikasjoner i listen for
                        //lastbare applikasjoner.
        }
        public void FyllSlettList(String txt){
                        //Fyller inn applikasjoner i listen for slettbare
                        //applikasjoner.
        }
        public String FyllLister() {
                        //Skriver ut last og slett listene til GUI.
        }
        private void settGuiAlert(int level,  string error){
                        //Sender feilmelding til GUI klasse.
        }
        public void SettGuiMelding(String Melding ) {
                        //Skriver til statusfeltet hva som skjer
        }
        public void SettGuiKID(String KID ) {
                        //Skriver ut KID (kort-id) til GUI.
        }
}
```

3.2.6 Detaljert design for database kommunikasjon (servlet modul 6)

Viser til figur 3.1.5.1

3.2.6.1 Klasse import

Denne modulen bruker følgende klasser :

SQLLib, Bibliotek for å snakke med SQL database*,* modul nr. 22

3.2.6.2 Klasse og metode oversikt

```
class DBkont {
        public void openDB() {
                        // åpner kobling mot database.
        }
        public string lastApp(int kortID, int appID){
                        // openDB()
        }
        public string slettApp(int kortID, int appID){
```

```
                                        // openDB()
        }

        public string identifiser(int kortID){
                                        //openDB();
        }
}
```

3.2.7 Detaljert design for kommando tolk (servlet modul 7)

Viser til figur 3.1.5.1

3.2.7.1 Klasse import

3.2.7.2 Klasse og metode oversikt

```
class kommTolkServ {
        public void pakkInstruksjon() {

        }
}
```

3.2.8 Detaljert design for klient kommunikasjon (servlet modul 8)

Viser til *figur 3.1.5.1*

3.2.8.1 Klasse import

Denne modulen bruker følgende klasser :

SSLLib, Bibliotek for SSL kryptering, modul nr. 21

3.2.8.2 Klasse og metode oversikt

```
class kliKomm {
        public void cryptDecrypt(string, key) {
                                        //Mottatte data vil dekrypteres, sendte data vil
                                        //krypteres.
                                        //Sender ferdig kryptert streng til servlet //(modul nr.
                                        8). Sender ukrypterte data til klient //kont. (modul nr.
                                        5).
                }
                public void doGet(HttpServletRequest request, HttpServletResponse
                response) {
                                        //Håndterer HTTP get requests
                }
                public void doPost(HttpServletRequest req, HttpServletResponse res){
                                        //Håndterer HTTP post requests
                }
        }
```

3.2.9 Detaljert design for kontroller (servlet modul 9)

Viser til *figur 3.1.5.1*

3.2.9.1 Klasse import

Denne modulen bruker følgende klasser :

DBKont, Database kontroller, modul nr. 6

KomTolkServ, Komando tolk på servlet, modul nr. 7

3.2.9.2 Klasse og metode oversikt

```
class ServKont {
        public string dbforespoersel(string forespoersel){
                        //Får en forespørsel fra klientkommunikasjon, og
                        //sender denne videre til DBkont.
        }
        public string packInstructions(string dbkontData){
                        //Sender den strengen med data som den får fra
                        //databasen til komTolkServ og får igjen en pakket
                        //datapakke. Sender pakken til klikomm hvis godkjent.
                        //Hvis ikke godkjent, sender feilmelding til klikomm.
        }
}
```

DEL : 4

Implementasjon

4 Implementasjon

4.1 Verktøy som er benyttet

Under utviklingen av prosjektet er det gjort en del bevisste valg av utviklingsverktøy.

Testing:

- Apache webserver 1.3.19 for windows
  Setter opp webserver lokalt på egen maskin for testing av websider med applet. Denne webserveren ble valgt siden vi hadde litt kjennskap til denne, og fordi at den er enkel å sette opp.
- Mysql database v3.23 for windows
  Brukes for å sette opp en midlertidig database for applikasjoner og sertifikater. Vi valgte å benytt MySql fordi vi har erfaring med den fra tidligere og er rimlig godt kjent med bruken av den.
- Apache Jserv v1.1.2
  Tilleggsprogram til Apache webserver som gir mulighet til bruk av java applets.
- Java servlet development kit v2.0
  Tilleggsprogram til Apache webserver for å kunne benytte java servlets.
- MyOdbc v3.50
  Windows databasedriver.
- Php v4.0 for windows
  Service som kjøres og gir mulighet til å kjøre php dokumenter lokalt.

Programmering:

- Borland Jbuilder v4.0
  Java utviklingsverktøy som benyttes til all Java koding. Vi valgte å benytte Borland Jbuilder i samråd med ErgoGroup, siden de benytter seg av det i sine prosjekter og har gode erfaringer.
- Homesite v4.5
  Program for å skrive og designe HTML dokumenter. Flere av gruppemedlemmene
  har god erfaring fra tidligere med dette programmet.
- OCF 1.2 All-in-One package.
  Bibliotek med alle OCF kommandoene for java. Inneholder også dokumentasjonen for bruk av OCF biblioteket

Dokumentering:

- Together v4.2
  Lager objekt orientert design. Etter å ha testet flere forskjellige objekt orientert design verktøy kom vi frem til at dette var det beste verktøyet.
- Microsoft Word 2000
  Tekstbehandler. Det mest vanlige tekstbehandlings programmet for windows maskiner.
- Milestones Simplicity
  Shareware program for å designe Gantt skjemaer. Dette ble valg pga at det var enkelt å forstå og enkelt å benytte.
- Adobe Photoshop v6.0
  Tegneprogram som vi benyttet for å redigere/tegne figurer.
- Inspiration v6.0
  Utviklingsverktøy for å lage flytdiagrammer. Ett av de få flytdiagram verktøyene som vi fant til fri nedlastning over internett.

4.2 Utviklingsmiljøet

Under arbeidet med oppgaven har vi jobbet sammen alle tre på en hybel tilkoblet høgskolen i Gjøvik sitt nett. Her har vi hatt to maskiner vi har jobbet på. Vi fikk låne smartkort og kortleser fra Ergo Group. All nødvendig software har vi fått tak i enten som freeware eller shareware versjoner over internett. Når det gjelder applikasjoner og sertifikater fikk vi ikke tilgang til disse utenfor EG lokaler. Derfor hindret dette oss litt i testingen.

4.3 Standarder

Under skrivingen av koden har vi utviklet en egen standard. Vi har prøvd å gjøre koden så enkel og lettlest som mulig for at andre etter oss skal kunne enkelt sette seg ned og forstå grovtrekkene i koden. Vi har prøvd å gi variabler og klasser så enkle og forståelsesfulle navn som mulig slik at de skal være lette å forstå.

4.3.1 Biblioteker

Hensikten med bruken av biblioteker er å få tilgang til metoder og funksjoner som er ferdig laget til bruk. På denne måten slipper vi å kode hvert enkelt element selv. Implementasjon av bibliotekene er det som kommer først i kildekoden til en klasse. Her følger noen eksempler på biblioteker vi har benyttet og formatet på hvordan de er implementert. Følgende eksempler er hentet fra kort kommunikasjons modulen (modul 3). Vi importerer kun de metodene fra hvert enkelt bibliotek som vi benytter. Dette gjør vi for å begrense størrelsen på appleten. Dersom vi skulle ha importert alle metodene i de bibliotekene vi benytter ville appleten bli for stor til at det ville være hensiktsmessig å benytte den over internett.

Ved første forsøk hadde vi importerte alle bibliotekene. Applet endte vi opp med en fil på 13.7 MB. Et eksempel på hvordan import rutinene ligger under.

```
import java.awt.event.*;
import opencard.core.service.*;
import opencard.core.util.*;
import opencard.core.terminal.*;
import opencard.core.event.*;
```

Når vi reduserte til kun de bibliotekene som ble benyttet endte vi opp med en applet på 733 kB. Eksempel på dette ligger under.

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import opencard.core.service.SmartCard;
import opencard.core.util.HexString;
import opencard.core.terminal.CardTerminal;
import opencard.core.terminal.SlotChannel;
import opencard.core.event.CTListener;
import opencard.core.event.CardTerminalEvent;
import opencard.core.event.EventGenerator;
```

4.3.2 Definering av variable

Ved definisjon av variable har vi prøvd å gi dem et navn som gjenspeiler hva de skal benyttes til. Fra koden vår har vi hentet følgende eksempler:

Fra GUI (modul 1):

- `int AntallLastApp, AntallSlettApp;`
  Variablene AntallLastApp og AntallSlettApp er definert som datatypen integer, og inneholder antall lastbare/slettbare applikasjoner.
- `boolean ErLast;`
  ErLast er av datatypen boolean. Beskriver tilstanden som sann/usann

Fra Klientkont(modul 5):

- KortKomm Kort = null;*
  Lager et nytt peker av klassen KortKomm, som vi kaller Kort.
  Ved å skrive "Kort = new KortKomm();" Dette gir oss tilgang til metoder og variable i klassen.
- private ActionListener SettLastAppList_Listener = null;
  Definerer SettLastAppList til å være ActionListener og setter verdien til å være null.

Fra DBkont(modul 6):

- String dbURL , dbBruker , dbPassord;
  Variablene er definert som datatypen string. Disse inneholder urlen, brukernavn og passord som trengs for å kunne logge seg på databasen.

4.3.3 Definering av metoder

Eksempel på hvordan vi har valgt å skrive metoder. Det vi har lagt vekt på er å prøve å få det så oversiktlig som mulig med tanke på innrykk og plasseringer av krøllparenteser. Dette eksempelet er hentet fra GUI (modul 1).

```
farge
void Knapp_actionPerformed(ActionEvent e) {
  final int Index;
    Debug.setText("");
    if (ErLast) {
        Index = LastList.getSelectedIndex();
    } else {
        Index = SlettList.getSelectedIndex();
    }
    if ( Index >= 0 && Index <=50 ) {
      Thread runner = new Thread() {
        public void run() {
          ViewAppInfo( ErLast , Index );
        }
      };
      runner.start();
    }
  }
```

4.3.4 Definering av kommentarer

Ved kommentering av koden har vi lagt vekt på å kommentere metoder og klasser i stedet for å kommentere hver enkel linje av koden. Men vi har i tillegg valgt å kommentere enkeltlinjer av koden der vi følte at det var behov for det. Dette kan være linjer med spesiell syntax, eller med viktig data/informasjon (formatet på datapakker vi har laget o.l). *Henviser til kode eksempel 4.4*

4.4 Kodeeksempel

Her viser vi et eksempel på en hel klasse for å vise en mer helhetlig oversikt på hvordan selve kodingen har foregått. Vi har valgt ut en klasse fra servleten, klikomm (modul 8). For flere kodeeksempler viser vi til CD-rommen, der ligger all kildekode og i tillegg har vi valgt å legge med en html dokumentasjon fra Together.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
/**
 *  Klasse som starter opp kontrolleren nå klienten sender en melding til
 *  servleten.
 *  Returnerer et svar til klienten på den mottatte meldingen.
 * @author Arve Bjørnerud
 * @version 1.0
 */
public class kliKomm extends HttpServlet {
  private static final String CONTENT_TYPE = "text/html";
  ServKont kont = null;

  /**Initialize global variables. Autogenerated by JBuilder*/
  public void init(ServletConfig config) throws ServletException {
    super.init(config);
  }

  /**
   * Metode som startes av klienten over internett. Sender med melding til
   * servlet
   */
  public void doPost(HttpServletRequest req, HttpServletResponse res)
                      throws ServletException, IOException {
    int ComPos;
    String Command, Argument;
    ServletInputStream sis = req.getInputStream ();   // Åpener en inputstream
    BufferedInputStream bis = new BufferedInputStream (sis);// inne i en
                                                          // bufferInputstream
    ObjectInputStream ois = new ObjectInputStream (bis);    // inne i en
                                                           //ObjectInputStream.
    String beskjed = "";
    try {                                              // Henter ut meldingen
      beskjed = (String)ois.readObject();              // sendt av appleten
    } catch (ClassNotFoundException cnfe) {
      throw new ServletException ("Fant ikke klassen " + cnfe );
    }
    res.setStatus (HttpServletResponse.SC_OK);         // Oppretter forbindelse
    ServletOutputStream sos = res.getOutputStream ();  // med applet for
    BufferedOutputStream bos = new BufferedOutputStream (sos);  // å sende svar
    ObjectOutputStream oos = new ObjectOutputStream (bos);      // tilbake.
    kont = new ServKont();
    beskjed = kont.Command(beskjed);
    oos.writeObject ( beskjed );                          // svarer til applet.
    oos.flush ();
  }

  /**Clean up resources. Autogenerated by JBuilder*/
  public void destroy() {
  }
}
```

4.4 GUI eksempel

Det endelige brukergrensesnittet ble litt forandret i forhold til design dokumentet. Her er det kun en knapp brukeren kan trykke på, for å laste/slette en applikasjon.
Det er lagt mindre vekt på bruk av popup vinduer siden dette kun er en prototyp og utviklere som skal videreutvikle systemet kan, ut fra egnen erfaring, finne disse irriterende.

figur 4.4

4.5 Kodestruktur

Her vil vi beskrive de ulike klassene og deres funksjonalitet i systemet. Dette vil ikke være en dypgående beskrivelse, men mer en oversikt over hver enkelt klasse som inngår i systemet. For å få en oversikt til sammenhengen mellom klassene, slik det ble i resultatet, finnes på CD-rommen.

4.5.1 Klient

Klienten har fått noen flere klasser enn det som ble satt opp i design dokumentet. Vi har valgt å ikke nummerere disse siden de ikke er med på noen figur og at det kunne forvirre mer enn å opplyse.

gui.java. Denne klassen setter opp det grafiske brukergrensesnittet for brukeren av systemet. Den inneholder alle metoder som oppdaterer brukermeldinger, statusbars og lister. Denne klassen kaller klient kontrolleren (modul 5). Tilsvarer modul 1.

klientKont.java. Dette er den klassen som er den sentrale modulen i klienten. Det er denne klassen som fordeler informasjon mellom de andre modulene (GUI, kortKomm, komTolkKli, liste). Denne initialiserer klassene kortKomm, ServKomm, liste og komTolkKli. Tilsvarer modul 5.

liste.java. Klasse som fyller listene med tilgjenglige applikasjoner og applikasjoner som kan slettes fra kortet. Deretter returneres disse listene til det grafiske brukergrensesnittet (GUI). Ny modul.

komTolkKli.java. Denne klassen setter sammen pakker som kommer fra serveren. Samtidig sjekker CRC checksummen for å se om pakken ble overført korrekt. I tillegg til dette holder denne klassen oversikt over hvor mange datapakker som har blitt mottatt. Tilsvarer modul 4

ServKomm.java. Har som ansvar og opprette forbindelse med servleten. Tilsvarer modul 3.

kortKomm.java. Inneholder metodene for å sjekke om kortet er satt i kortleser, og dersom kort er i leser så vil det åpnes en forbindelse (kanal) til kortet. Denne forbindelsen vil sendes videre til SCInterface klassen. Når kortet blir fjernet fra kortleser så vil forbindelsen avsluttes.Tilsvarer modul nr 2.

SCInterface.java. I denne klassen skjer all formatering og jobbing med kortkommandoer mot kortet. Når dataene er formatert, sendes de til kortet via kanalen som kortKomm klassen åpnet. Ny modul.

ADCFile.java. Utfører lasting av mottatt slett sertifikat inn i en intern struktur så det blir enklere å generere kommandoer ut fra dataene. Ny modul.

ALUFile.java. Utfører lasting av mottatt applikasjon inn i en intern struktur så man enklere kan generere kommandoer ut fra dataene. Ny modul.

ALCFile.java. Utfører lasting av mottatt last sertifikat inn i en intern struktur så man enklere kan generere kommandoer ut fra dataene. Ny modul.

ImageCanvas.java. Ferdig klasse funnet på internett. Inneholder metoder for å vise bilder. Ny modul.

MessageBox.java. Ferdig klasse funnet på internett. Inneholder metoder for å generere info/error bokser. Ny modul.

4.5.2 Servlet

klikomm.java Initialiseres av servkomm. Sender data videre til servKont. Returner data til klienten når prosessering er ferdig på servleten. Tilsvarer modul 8.

ServKont.java. Binder servlet modulene sammen. Initialiserer DBKont og komTolkServ. Tilsvarer modul 9.

DBKont.java. Denne klassen kobler seg til databasen og utfører spørringer mot denne. Ved lastingen/sletting hentes applikasjonen- og sertifikatet- filnavnene og disse sendes videre til klikont(modul 9). Tilsvarer modul 6.

komTolkServ.java. Klassen leser data fra mottatte filnavn og pakker så disse. Det legges også til CRC i pakkene for å garantere at pakkene kommer riktig fram. Tilsvarer modul 7.

DEL : 5

Sikkerhetsanalyse og testing

5 Sikkerhetsanalyse og testing

5.1 Sikkerhetsanalyse

Hva som må beskyttes.

- Datastrømmen.
- Server.

Hvem har interesse/mulighet av å gjøre noe som kan skade systemet.

- Konkurrenter.
- Hackere.
- Svindel.
- Brukere med høy sikkerhetstilgang.
- Ansatte som slutter.

Hva som kan gjøres.

- Pakkesniffing.
  Noen som kobler seg på datastrømmen og prøver å tilegne seg data på uærlig vis.
- Uvedkommende som kommer seg inn på server.
  Dette kan kun skje fysisk ved at det skjer innbrudd på bygg der serveren står.
- Modifisering av datastrømmen.
  Endring av informasjonen i datastrømmen i den hensikt i å få programmet til å stoppe eller laste feilaktig informasjon til kortet.
- Bruker prøver å laste applikasjoner som han/hun ikke har tillatelse til.
- Kort stjålet
- Informasjon kan legges inn feilaktig i databasen av administrator.

Hva resultatet kan bli.

- Kort kan bli sperret
- Hvis uvedkommende får tilgang til ALU, ALC eller ADC kan de utvikle sine egne smartkort applikasjoner eller sertifikat. De kan da for eksempel oppgi seg som noen de ikke er ved å skrive sin egen EiD.

Hva vi kan gjøre for å hindre det

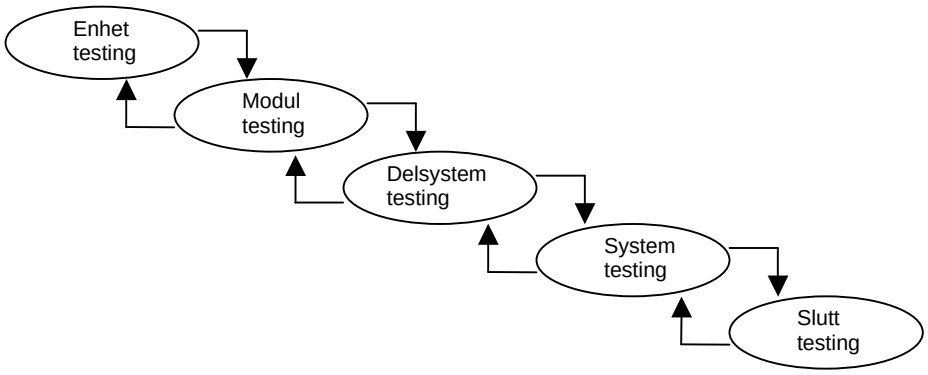
- Pakkesniffing og modifisering av datastrøm.
  Ved å kryptere datastrøm reduseres muligheten for at uvedkommende skal klare å tappe/endre dataene.
- Innbrudd
  Plassere serveren på et trygt sted.
- Lasting av applikasjoner man ikke har tilgang til
  Denne problemstillingen har vi lagt inn i programmet, ved at det for hvert smartkort kommer opp en liste over applikasjoner som kan lastes/slettes. Man vil ikke få muligheten til å laste applikasjoner som man ikke har tilgang til.
- Kort som er stjålet
  Kan gjøre om program slik at bruker må taste inn sin egen personlige PIN-kode før kortet kan benyttes. Dette er ikke gjort i programmet vårt siden vi ikke har operert med sensitive data.
- I forhold til at en administrator som legger inn feilaktig informasjon er det lite som kan gjøres. Det eneste tiltaket for å hindre at et smartkort sperres er å sjekke at smartkortet faktisk har nok plass til å laste applikasjonen.

5.2 Testing

Testingen ble utført likt i forhold til hvordan det ble skissert i kravspesifikasjonen *(se punkt 2.4.2)*

5.2.1 Planlegging teststrategi

Under testingen har vi valgt å benytte oss av en inkrementell strategi. Dette innebærer at vi hele tiden tester og bygger videre på den modulen vi har. Etter å ha lagd en modul/komponent tester vi at denne fungerer tilfredstillende før vi bygger videre på denne model/ lager ny modul.



figur 5.1.2

- Enhet testing:
  Hver enkelt komponent blir testet, for å forsikre oss om at de fungerer korrekt. De blir testet individuelt uten andre system komponenter. I vårt prosjekt er da de ulike metodene komponenter.
- Modul testing:
  En modul er en samling av forskjellige komponenter, som for eksempel et objekt. Modul testingen foregår uavhengig av resten av systemet.
- Delsystem testing:
  Vi ser på klienten og servleten som hvert sitt delsystem. Testingen av disse foregår uavhengig av hverandre.
- System testing:
  Hele systemet testes sammen. Funksjonaliteten til system testes. Hovedhensikten med denne testen er å finne feil som en bruker kan kan komme over under kjøringen av systemet.
- Slutt testing:
  Beta testing av programmet. Vi skal fra en brukers side prøve å forutse alt hva han/hun kan klare å finne på.

5.2.2 Komponent testing klient

5.2.2.1 GUI

Det vi har testet her er om brukergrensesnittet ble som vi hadde forestilt oss, og det ble det siden vi har benyttet oss av JBuilder, som er kjent for WYSIWYG (What You See Is What You Get) prinsippet.

5.2.2.2 klientKont

Ble testet mot GUI. GUI starter metoder i klient kontrolleren, kontrolleren vil da svare med enten å returnere en verdi i metoden som ble startet eller bruke actionlistening.

5.2.2.3 liste
Ble testet mot kontrolleren. Sender en string som inneholdt en oppdiktet liste. Det var da liste sin oppgave å dele opp den mottatte strengen, sjekke for feil og returnere en array tilbake. Array skulle inneholde listen.

5.2.2.4 komTolkKli
Denne ble testet ved at servlet leste en fil, som ble delt opp i pakker. Det var da kommunikasjonstolkens oppgave å sette sammen pakkene så innholdet var likt filen på serveren. Får å sjekke samsvaret ble den sammensatte pakken sent til GUI hvor den ble sjekket manuelt. KomTolkKli ble testet sammen med KomTolkServ.

5.2.2.5 ServKomm
Vi testet denne ved å skrive ferdig kode for kommunikasjon på både klient og servlet. Servleten hadde som oppgave og returnere teksten den mottok i store bokstaver. Mens klienten bare sendte en oppdiktet tekst. Teksten som ble mottatt ble sendt tilbake til GUI for manuell sjekking.

5.2.2.6 kortKomm
Modulen ble sjekket med at vi først sjekket at GUI fikk melding om at kortet ble tatt inn og ut. Etter dette testet vi å sende kommandoer til kortet. Ved å gjøre enkle operasjoner mot det og sende svaret til GUI for manuell sjekking.

5.2.2.7 SCInterface
Dette var den siste modulen som ble sjekket, den var avhengig av at alle andre moduler fungerte 100%. Dette siden smartkortet vil bli sperret etter 3 mislykkede lastinger eller slettinger. Modulen ble først testet ved at alle kommandoer som skulle sendes til kortet ble sendt til GUI. Der ble den sjekket mot ”Guide to loading and deleting application v2.10” og en oversikt over kommandoer som sendes til kortet for en vellykket lasting/sletting.

5.2.2.8 ADCFile
Ble testet ved at sertifikatet først ble lastet inn, deretter ble hele strukturen skrevet ut til GUI for manuell sjekking.

5.2.2.9 ALUFile
Ble testet ved at applikasjonen først ble lastet inn, deretter ble hele strukturen skrevet ut til GUI for manuell sjekking.

5.2.2.10 ALCFile
Ble testet ved at sertifikatet først ble lastet inn, deretter ble hele strukturen skrevet ut til GUI for manuell sjekking.

5.2.2.11 ImageCanvas
Ble ikke testet siden den er en del av MessageBox som er en ferdig modul.

5.2.2.12 MessageBox
Ble testet ved å lage en ny MessageBox i GUI.

5.2.3 Komponent testing servlet
Testing av servleten ble gjort på to forskjellige måter, enten via kall fra klienter eller kjøre servleten direkte i en nettleser. Får å få til dette måtte en ekstra metode opprettes i klikomm (public void doGet(……)). Denne modulen finnes ikke i den endelige løsningen siden den var en sikkerhetsrisiko, en form for bakdør.

5.2.3.1 klikomm
se 5.2.2.5 ServKomm.

5.2.3.2 ServKont
Denne ble testet ved å sende kommandoer til servlet, får så å se om den returnerte rett svar i forhold til det den ble spurt om.

5.2.3.3 DBKont

Fra GUI og nettleseren ble det sendt kommandoer som ba databasen om å hente ut informasjon som last/slett liste. Denne informasjonen ble manuelt sjekket mot databasen for å se at spørringene fungerte rett.

5.2.3.4 komTolkServ

se 5.2.2.4 komTolkKli

5.2.4 Integrasjonstesting klient

Integrasjonstesten ble foretatt hos ErgoGroup. Grunnen til dette er at applikasjoner og sertifikater ikke bør forlate bygningen hos EG. Testen ble utført på hele systemet. Servleten fikk tilgang til to applikasjoner med sertifikater, som ble lagt inn i databasen. På klient siden hadde vi tre forskjellige smartkort. For å kunne jobbe mot de tre kortene måtte kort identifikasjonsnummeret for kortene ligge i databasen, disse ble lagt inn manuelt. Testingen gikk da ut på å laste/slette de to applikasjonene på de tre ”gyldige” kortene. Appleten og servleten ble kjørt på samme maskin, hvor appleten ble kjørt gjennom JBuilder.

MERK : Systemet må kun testes på gyldige kort, med tilhørende gyldige applikasjoner og sertifikater. Hvis ikke kan en risikere og sperre smartkortet man tester systemet mot. Under normal drift vil ikke dette være et problem siden databasen kun inneholder gyldige kort, og man vil da kun få en feilmelding hvis kortet ikke er gyldig.

5.2.5 Bruker testing

Denne testen gikk ut på å kjøre appleten på en annen maskin enn der hvor servleten står. Maskinen hadde ikke installert nødvendig programvare, kun driver til smartkortet. Appleten ble da kjørt gjennom en nettleser. Det viste seg da at brukeren må ha installert Java 2 v1.3 for å starte appleten. Denne kan hentes ned fra java.sun.com eller installeres fra CD-rommen (*j2sdk-1_3_0_02-win.exe*). I tillegg må brukeren ha installert PCSC (Personal Computer Smart Card) drivere for sin kortleser, disse følger normalt med kortleseren.

For at appleten skal kunne få tilgang til klientmaskinens ressurser, må den signeres. Får å signere en applet må en ha et signeringssertifikat, det hadde ikke vi under testingen. Så for at appleten skulle få kontakt med kort leseren og konfigurasjonsfeil, skrev vi om Java sin sikkerhetssperre (java.policy).
Disse tilleggs innstillingene ble lagt til :

```
permission java.util.PropertyPermission "OpenCard.loaderClassName", "read";
permission java.util.PropertyPermission "java.home", "read";
permission java.util.PropertyPermission "user.home", "read";
permission java.util.PropertyPermission "user.dir", "read";
permission java.io.FilePermission "<<ALL FILES>>", "read";
permission java.util.PropertyPermission "*", "read, write";
permission java.lang.RuntimePermission "loadLibrary.OCFPCSC1";
```

Appleten trengte da ikke å være signert lenger. Dette er bare aktuelt å gjøre under testing, hvis Ergo Group skal bruke appleten i kommersielt bruk må de selv signere appleten. Noe de hevder ikke er et problem.

DEL : 6

Konklusjon og drøfting av resultatet

6 Diskusjon av resultatet

Vi har lagd en prototyp for lasting av smartkort applikasjoner over internett med dette har vi lagt grunnlaget for å bruke smartkort med Java. Når systemet blir lagt ut for kommersielt bruk vil det bli enklere for bruker å laste ned/slette applikasjoner på smartkortet sitt, slik at man ikke trenger å oppsøke kortutsteder.

6.1 Evaluering i forhold til kravspesifikasjon

- SSL.
  Bruk av SSL som skulle være en del av den sikre kommunikasjon mellom server og smartkort ble også droppet i samråd med ErgoGroup. Vi skulle i stedet bruke en egen krypteringsmåte for å sikre sikker kommunikasjon. Se *6.2.4* for en utøypelse av problemet
- Flash.
  Et av de fire målene som ble satt, var at det skulle brukes Flash design av brukergrensesnittet. Dette har vi måttet kutte ut på grunn av tidstrøbbel. Vi testet ut Flash design på en prosjektoppgave i klient og serverside programmering. På denne oppgaven klarte vi å få det til, men det var en tidkrevende prosess. Den Flashen vi lagde der var også av et mye enklere format, så vi måtte ha brukt mye tid for å få det til å virke. Vi besluttet derfor på et statusmøte 30.april å ikke lage det til denne oppgaven. Dette kan være en mulig utvidelse av oppgaven vår i neste omgang.
- CMS.
  Ble ikke ferdig slik at vi kunne bruke det, men vi har prøvd å gjøre det så enkelt som mulig for framtidige prosjekt å bytte ut vår database med en CMS database. Bruk av CMS vil bli en naturlig utvidelse av systemet hvis det skal kommersialiseres.
- Arbeidsfordeling.
  Arbeidsfordelingen har ikke vært helt som planlagt i starten. Vi begynte med å ha en som skulle ha hovedansvaret for klienten, mens de to andre skulle ha ansvaret for servleten. Dette viste seg å bli feil inndelingen, siden klient delen var mye større en servlet delen.
  Underveis i kodingen viste det seg også at vi begynte å få litt dårlig tid, så for å effektivisere arbeidet fikk en person hovedansvaret for kodingen, mens de andre skulle komme med konstruktive innspill samtidig som de utarbeidet rapporten.
- Verktøy.
  ErgoGroup ville i utgangspunktet at vi skulle bruke Rational Rose som modelleringsverktøy men siden ingen av gruppe medlemmene hadde kjennskap til dette verktøyet valgte vi å bruke Together i stedet.

6.2 Evaluering og kritikk av kode

6.2.1 Meldinger mellom klasser

I programmeringsfasens etterkant oppdaget vi at formatet på meldinger mellom klasser var vidt forskjellig. Det ville vært en fordel om alle meldinger fulgte det samme malen. Dette ville minske behovet for individuell oppdeling og sjekking av meldinger i hver klasse. En melding kunne være formatert på denne måten :

Navn på parameter.	Parameter forklaring.
Til modul	Navnet på modulen meldingen skal til.
Fra modul	Navnet på modulen meldinger kommer fra.
Applikasjons ID	Dette sendes med, hvis det trengs.
Kort ID	Sendes alltid, for å sikre at operasjoner skjer mot rett kort.
Operasjon	Hva som skal gjøres med meldingen av mottageren.
Argument	Eventuell tilleggsopplysning for å kjøre operasjonen
Data lengde	Lengden på følgende data.
Data	Kan inneholde applikasjonsliste, instruksjoner, internemeldinge mellom moduler.
CRC	Sjekk av datastrømmen.

Et egenutviklet bibliotek ville da kunne tatt hånd om meldingen i de forskjellige klassene. Grunnen til at dette ikke ble gjort under utviklingen var at vi ikke så dette som et problem.

6.2.2 Applikasjons kontroll

Slik systemet er nå er det servleten sin oppgave å hente ut applikasjoner og sertifikater og sende disse til klienten. Klientens oppgave er å strukturere dataene fra applikasjon og sertifikat filene. Til slutt henter klienten informasjon ut fra strukturen og genererer instruksjoner fra den, som sendes til kortet. I test fasen hos Ergo Group kom det frem at dette var feil måte å gjøre operasjoner mot kortet på. Det som hadde vært en bedre og tryggere løsning var hvis servleten sto for innlasting, strukturering og instruksjonsoppbygning selv. Det eneste klienten da ville motta fra servleten var ferdige instruksjoner.
Grunnen til at dette er tryggere er todelt. For det første vil ikke hele sertifikater og applikasjoner overføres via nettet. Den andre grunnen er at de heller ikke vil 'oppholde' seg i appleten. Ut fra det vi kan nå angående hvordan en laster og sletter, ser vi at dette hadde vært en mye bedre løsning. Og det var dette EG ønsket helt fra begynnelsen av, grunnen til at dette ikke ble gjennomført var fordi vi ikke forsto problemet før det var for sent. Å endre nåværende kode slik at den laster/sletter fra servleten vil ikke by på store problemer. I praksis vil det bety å flytte klassene ALUFile.java, ALCFile.java og ADCFile.java fra klienten til servleten. I tillegg må noen metoder fra SCInterface.java flyttes.

6.2.3 Bruk av Popup bokser

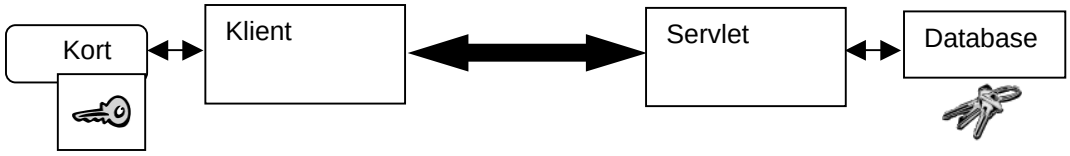
I design dokumentet *(se 3.2.1)* er det satt opp mange forkskjellige meldingsvinduer som skal informere brukeren om eventuelle feil eller utførte operasjoner. Under utvikling så vi fort at det ikke var noe behov for å generer meldingsbokser for hver melding som sendes til brukeren. Siden system tross alt er en prototyp, som skal videreutvikles, er det like greit å skrive ut meldingen i meldingsvinduet. Å implementere meldingsboks systemet i stedet for å skrive det direkte ut er trivielt. I praksis vil dette si å bytte ut alle "SettGuiMelding(Melding)" med "SettGuiAlert(Melding)".

6.2.4 Kryptering

Som en del av både oppgave beskrivelsen og kravspesifikasjonen, gikk kodingen ut på å lage en sikker overføring mellom klient og servlet. Her ble krypterings metoden SSL nevnt. Når vi nærmere begynte å se på SSL kryptering under java oppsto det problemer. Det viste seg at SSL kryptering benytter en spesiell form for webserver, dvs at i tillegg til å implementere SSL bibliotek i kildekoden, måtte vi i også sette opp en egen SSL server. Noe av problemet var at vi ikke fant programvare som satt opp denne form for tjeneste på serveren. Et annet problem var at serveren også måtte ha egne krypterings sertifikat, noe vi ikke fant ut hvem som utstedet. I samråd med EG bestemte vi oss for å droppe SSL kryptering siden ikke de heller hadde kompetanse på området.

To andre krypteringsalgoritme ble også diskutert :

Å skrive en egen kryptering algoritme ville i hovedtrekk ikke by på store problemer. Her så vi på flere algoritmer, men fant til slutt ut av RSA kryptering ville bli sikker nok. Det som imidlertid ble problemet var å finne en felles kryptering nøkkel for klient og servlet. På servleten ville nøklene lagres sammen med informasjonen om hvert kort. Klienten måtte hente nøkkelen fra kortet. Dette fører til at hvert kort må lastes med en unik 'krypteringsnøkkel' applikasjon.



figur 6.2.4

Ved oppstart må klienten hente denne nøkkelen fra kortet, før den kunne starte kommunikasjonen med servleten. Problemet med denne løsningen er at den er for komplisert, den ville ta mye lenger tid å utvikle enn det vi hadde til rådighet. Et annet problem ville også bli at applikasjonen ville forbruke verdifull plass på kortet.

Det ble også vurdert en annen metode å kryptere data på. Public og Private key kryptering. Her ville klient og servlet ha to sett med nøkler hver hhv en public og en private key. En melding som krypteres med public key kan kun dekrypteres av en tilsvarende private key. Fra klient til servlet ville krypteringen gå greit siden private key til kortet ligger i databasen. Men samme problemet oppstår igjen, med at klienten må ha en nøkkel for oppakking. Her måtte den lagres i kortet, med de ulemper som dette fører med seg.

6.2.5 Servlet som tråd

Når klienten henviser seg til servleten, vil servleten starte opp og utføre oppgaven den ble pålagt av klienten, før den returnerer svaret og dør.
Dette har vi innsett er en tungvint måte å kommunisere på. En bedre løsning hadde vært om servleten var aktiv for hele tiden fra klienten var aktiv. Kommunikasjonen mellom de ville ble mer dynamisk siden begge kunne ta initiativ til å snakke med den andre. Dette ville også være en fordel for bruk ved kryptering, siden klienten og servleten kun trenger å autorisere seg for hverandre en gang, i stedet for hver melding som sendes. I praksis ville dette bety at når klienten kalte servleten første gang, ville servleten startes i en egen tråd. Når da klienten henviste seg til servleten igjen trenger den ikke å opprette en ny kommunikasjonskanal, men bruke kanalen som allerede er opprettet. Når klienten avsluttes sendes det en kommando til servleten som forteller den om å drepe tråen den går i. Grunnen til at dette ikke ble brukt i denne løsningen er at vi ikke hadde nok kunnskaper i bruk av tråder, og derfor ikke forsto hvilke muligheter trådbruk gir.

6.3 Evaluering av prosjekt og oppdragsgiver.

6.3.1 Prosjektet

Under arbeidet med prosjektet valgte vi å følge en mal for kravspesifikasjonen som ikke var basert på Objekt Orientert Analyse (OOA), men på den tradisjonelle formen. Men siden vi på design begynte å ta i bruk elementer fra Objekt Orientert Design (OOD) så innser vi at det hadde vært hensiktsmessig også å bruke OOA. Grunnen til at vi ikke valgte å bruke OOA fra begynnelsen av var at vi ikke følte at vi behersket det nok til å skrive analyse der, men etter å ha gjennomført en obligatorisk øving i faget Systemutvikling II hadde vi mer kunnskap om det. Da var allerede kravspesifikasjonen ferdig og vi valgte derfor å ikke endre på den, men besluttet heller å ta i bruk elementer fra OOD til design dokumentet i stedet.

Ved senere prosjekter vil vi komme til å benytte OOA da vi føler at vi har fått en større forståelse for det og hva det kan benyttes til.

Vi føler at vi har kommet frem til en god rapport med tanke på at dette er første gang vi gjennomfører et så omfattende prosjekt, og at vi derfor ikke har noe sammenligningsgrunnlag. Den kanskje største utfordringen i utarbeidingen av prosjektet var å skrive design dokumentet, siden vi ikke hadde noen form for mal eller andre ting å sammenligne med.

I tillegg har det vist seg at Gantt skjemaet vi lagde i forbindelse med forprosjektet, ble vanskelig å følge. En grunn til dette kan nok være at vi har begrenset erfaring med å planlegge og gjennomføre dette på større prosjekter.

6.3.2 Oppdragsgiver

Samarbeidet med oppdragsgiver har gått bra. Vi fikk låne med oss utstyr slik at vi ikke hadde behov for å tilpasse jobbingen vår til EGs åpningstider. Dette har medført både fordeler og ulemper. En av de største ulempen må kunne sies å være at kontakten med EG har vært noe begrenset. Vi har hatt kontakt med dem når vi har følt behov for det og hatt konkrete spørsmål, men kunne muligens ha jobbet noe nærmere med dem slik at vi kunne ha fått mer tilbakemeldinger under arbeidet. Dette kan muligens være en av årsakene til misforståelser vedrørende oppgaven. Blant annet det med pakking av kommandoer som skulle sendes til kortet. *(som nevnt i punkt 6.2.2)*

En fordel med å jobbe som vi har gjort er at vi nå har hatt tilgang til 2-3 datamaskiner kun for dette prosjektet, mens hos EG var det satt opp to datamaskiner på deling for alle tre hovedprosjektsgruppene noe vi syntes var alt for lite.

6.3.3 Evaluering av gruppens arbeid

Når vi ser tilbake på gruppas arbeid gjennom prosjektperioden, må vi si oss fornøyd. Gruppa har jobbet målbevist og etter beste evne gjennom hele prosjektet. Motivasjonsnivået har vært litt varierende i perioder, spesielt før påske da vi hadde endel andre store oppgaver ved skolen samt eksamener. Når det gjelder fremmøte så må dette sies å være upåklagelig, alle har møtt opp og bidratt til det endelige produktet.

Under jobbingen ble det tidlig klart at noen gruppemedlemmer hadde ett større og bedre grunnlag med Java programmeringen enn andre. Det ble derfor en naturlig utvikling at disse tok ett større ansvar med programmeringen, og mindre arbeid med rapporten, spesielt mot slutten da tid ble en kritisk faktor.

6.4 Konklusjon

Vi syntes vi har fått gjennomført en bra oppgave selv om vi ikke fikk gjennomført alle de målene vi hadde satt oss. Det har vist seg i ettertid at ambisjonene vi hadde under utarbeidingen av kravspesifikasjonen var i høyeste laget, spesielt med tanke på at vi bare var tre stykker som arbeidet med oppgaven. Med det utgangspunktet vi hadde satt opp i kravspesifikasjonen viste det seg at tre stykker ble i minste laget for en så omfattende oppgave. For å få fullført alle de målene vi satte oss, burde vi hatt ett bedre grunnlag med Java programmering, eventuelt hatt med en person til på oppgaven.

Det er fortsatt en del koding som må gjennomføres før en kan kommersielt begynne med lasting av smartkort applikasjoner over internett. Den løsningen vi presenterer her er det første skrittet til en fullstendig løsning. Det har hele tiden vært hovedmålet med oppgaven, å legge grunnlaget for en kommeseriell levedyktig løsning, en pilot.

Hvis de punktene nevnt under avsnitt 6.1 og 6.2 utføres vil den neste versjonen kunne bli en meget bra løsning.

Arbeidet med oppgaven har gitt oss en verdifull plattform å bygge videre på, kanskje spesielt når det gjelder Java programmering og prosjektarbeid, men også lært oss mye om smartkort og bruk av disse.

I ettertid ser vi tilbake på prosjektet som en nyttig ressurs og en viktig erfaring som vi kan ta med oss inn i arbeidslivet, hvor prosjektarbeid har blitt en mer og mer viktig arbeidsform.

Litteraturliste

Tittel	Forfatter	Forlag	Referanse
Multos Developers Reference manual v1.3			www.multostechnet.com
Guide to loading and deleting application v2.10			www.multostechnet.com
OpenCard Framework v1.2			www.opencard.org
Java How To Program (third edition)	Deitel & Deitel	Prentice Hall	
Campuscard-II	Christian Bøe, Thomas Abrahamsen, Gaute Gillund og Guro Østbye		Hovedprosjekt 18.mai 1999
Applying UML and Patterns	Craig Larman	Prentice Hall	
Software Engineering	Ian Sommerville	Addison Wesly	
OCF 1.2 documentation			www.opencard.org

Internettlinker

The Source for Java Technology	java.sun.com
ISO 7816 - part 1-3	www.scia.org/Aboutsmartcards/iso7816_wimages.htm
ISO 7816 - part 4	www.cui.unige.ch/~zbinden6/smartcard/iso7816_4.html
ErgoGroup	www.ergogroup.no
MULTOS	www.multos.com

Verktøy

Program	Referanse
Milestone Simplicity	www.kidasa.com
Inspiration 6	www.inspiration.com
Together v4.2	www.togethersoft.com
Home Site v4.5	www.allaire.com

DEL : 7

Vedlegg

Vedlegg A.	Fremdriftsplan	2 sider.
Vedlegg B.	Prosjektdagbok	9 sider.
Vedlegg C.	Kildekodemal for kravspek.	1 side.
Vedlegg D.	Database struktur	1 side.

Vedlegg A

Fremdriftsplan

1 Ansvarsforhold

Prosjektleder :	Arve Bjørnerud
Loggfører :	Martin Klaveness
Web ansvarlig:	Ola Østeng
Dokumentansvarlig:	Ola Østeng

2 Øvrige roller og bemanning

Kontaktperson Ergo Group
Morten Johansen.

Veileder:
Frode Haug

Startaktiviteter: (egenstudie)

JBuilder 4.0 :	Arve Bjørnerud
MultOS :	Martin Klaveness
Sikkerhet, SSL:	Ola Østeng
Flash:	Ola Østeng

Hovedaktiviteter:

Programmering av servlet :	Arve Bjørnerud og Ola Østeng
Programmering av klient :	Martin Klaveness

3 Planlegging, oppfølging og rapportering

3.1 Hovedinndeling av prosjektet.

- Forprosjekt
- Kravspek/analyse
- Koding/design
- Testing
- Flash / hjemmeside

3.2 Krav til statusmøter og beslutningspunkter.

- Avtalt 3 statusrapport
  ca. 20 februar, ca. 30 mars og ca. 1 mai
- Beslutningspunkter/milepæler
  - Ferdig forprosjekt
  - Ferdig kravspesifikasjon
  - Ferdig design
  - Ferdig koding
  - Ferdig testing
  - Ferdig sikkerhetsanalyse
  - Foreta fremføring
  - Foreta webdesign, siste tidspunkt for start.
  - Ferdig rapport.

4 Gantt diagram

Viser til filen /pdf/gantdiagram.pdf på cd-rommen.

5 Evaluering av Gantt diagram

Det viste seg ganske raskt under arbeidet at Gantt diagrammet kom til å bli vanskelig å følge. Aktivitetene tok mye lengre tid enn det vi antok på forhånd, i tillegg var ambisjonene store i begynnelsen noe som gjorde at vi trodde vi skulle klare å få til mer enn vi klarte.

Utarbeidelse av kravspesifikasjonen tok en uke lengre tid en planlagt, og var ferdig 1.mars, mot planlagt 23.februar. Grunnen til at dette arbeidet ble forsinket skyldes endel usikkerhet knyttet til hvordan vi skulle utarbeide kravspesifikasjonen. Forsinkelsen førte derfor til at vi kom litt etter skjema i forhold til design dokumentet. Vi prøvde å arbeide mer for å ta igjen litt av det tapte, men det visste seg vanskelig på grunn av andre oppgaver i andre fag samtidig, i tillegg var det like før eksamen så designdokumentet ble ikke ferdig før 17.april, en måned etter planlagt.

Etter påsken var undervisningen slutt for 2 av gruppens medlemmer, dette gjorde at vi kunne bruke mer tid på hovedprosjektet. Men vi klarte ikke å ta igjen det vi lå etter, siden det ble mye å gjøre.

Under arbeidet med designdokumentet jobbet vi samtidig med koding. Vi hadde satt av altfor liten tid til kodingen så dette gjorde at kodingen ikke ble avslutte før 18.mai, noe som var nesten en måned etter den opprinnelige planen. Under kodingen, som nevnt i rapporten, begynte vi med klienten først før servleten ble påbegynt. I planleggingsfasen så vi for oss at servleten var den mest krevende så vi ville begynne tidlig på denne og planla å bruke to personer til denne oppgaven. Dette viste seg å bli galt siden klienten var størst.
Web design med Flash innså vi at vi ikke hadde tid til, så vi valgte å droppe dette på et statusmøte med gruppen 30.april.

Innleveringene av statusrapportene ble heller ikke levert etter den oppsatte planen, det skyldes blant annet at veileder ville ha den siste statusrapporten litt senere. Vi valgte derfor å levere statusrapportene 1.mars, 3.april og 7.mai.

Gjennomføringen av risikoanalysen fulgte den oppsatte planen, det samme gjorde innleveringen.

Vedlegg B

Prosjektdagbok

Prosjektdagbok

Vi har valgt å jobbe mest mulig samlet under hele oppgaven, og har under disse samlingen ført en aktivitetslogg over disse møtene. Vi har i tillegg ført logg over alle møter vi har hatt med veileder og med Ergo Group, Morten Johansen. Vi har også hatt et statusmøte og levert 3 statusrapporter til veileder. Vi har ikke levert møtereferater til de involverte parter etter hvert møte, siden de møtene vi har hatt med veileder og Ergo Group stort sett har vært å klare opp et par uklare punkter i kravspek/design.
Vi har valgt å legge med 2 eksempler på aktivitetslogg, to eksempler på møtelogg og de 3 statusrapportene

Møtelogg

Dato : 17/01-01
Log fører : Martin Klaveness
Status møte : Nei
Møte med : Frode Haug
Studenter tilstede : Ola Østeng, Arve Bjørnerud og Martin Klaveness
Antall sider : 1

Emner som ble tatt opp :
- Prosjektleder
- Statusrapporter
- Forprosjekt
- Forkortelser
- Kravspesifikasjon

Resulater / utfall av møte :
- Valg av prosjektleder gjøres på neste gruppemøte
- Veileder vil ha 3 statusrapporter underveis
- Arbeid med forprosjekt startes

Lasting av Smartkort
applikasjoner over internett

Møtelogg

Dato : 17/01-01
Log fører : Martin Klaveness
Status møte : Nei
Møte med : Morten Johansen og Asbjørn Hovstø
Studenter tilstede : Ola Østeng, Arve Bjørnerud og Martin Klaveness
Antall sider : 1

Emner som ble tatt opp :

- Kravspesifikasjon
- Bruke Rational Rose til modelering og klassediagram
- Signering av avtale
- Muligheter for utvidelser (pin kode +++)
- Definere angrep. (Noen på klient andre på server)
- Bøker

Resulater / utfall av møte :

- Fikk litteratur, begynne å sette oss inn i Jbuilder
- Arve skulle eksperimentere med Jbuilder

Aktivitetslogg

Dato : 18/01-01
Log fører : Martin Klaveness
Status møte : Nei
Møte med : Gruppen
Studenter tilstede : Ola Østeng, Arve Bjørnerud og Martin Klaveness
Antall sider : 1

Emner som ble tatt opp :
- Forprosjekt påbegynnes
- Begynner å samle definisjoner til fremmedordliste

Resulater / utfall av møte :
- Begynt på forprosjekt, og satt opp en grov skisse for denne

Aktivitetslogg

Dato : 20/01-01
Log fører : Martin Klaveness
Status møte : Nei
Møte med : Gruppen
Studenter tilstede : Ola Østeng, Arve Bjørnerud og Martin Klaveness
Antall sider : 1

Emner som ble tatt opp :
- Gantt skjema
- Forprosjektet

Resulater / utfall av møte :
- Ferdig med forprosjekt i grove trekk.
- Levere forprosjekt til veileder for synspunkter

Aktivitetslogg

Dato : 30/04-01
Log fører : Martin Klaveness
Status møte : Ja
Møte med : Gruppen
Studenter tilstede : Ola Østeng, Arve Bjørnerud og Martin Klaveness
Antall sider : 1

Emner som ble tatt opp :
- Flash

Resulater / utfall av møte :
- Vi besluttet å droppe Flash design på applikasjonen.

Statusrapport nummer 1

1. mars 2001

Lasting og sletting av applikasjoner over internett

Oppgaver fullført så langt

- Levert forprosjekt
- Ferdig med kravspesifikasjon
- Alle gruppens medlemmer har jobbet med hvert sitt kjerneområde
- Opprettet web side for gruppen og oppgaven

Oppgaver som vi jobber med nå/skal begynne med i nærmeste fremtid

- Utarbeide design for oppgaven

Planlegging

For øyeblikket ligger vi litt etter den opprinnelige planen for kravspesifikasjonen som var ment å være ferdig fredag den 23.februar. Kravspesifikasjonen ble ikke ferdig før den 1.mars. Grunnen til dette var nok hovedsaklig at vi feilberegnet arbeidsmengden til kravspesifikasjonen noe. Som en følge av dette valgte vi å utsette statusrapporten nesten en uke, til 1 mars, slik at vi leverte den samtidig som vi ble ferdige med kravspesifikasjonen.

Organisering

Organiseringen av gruppen ansvarsområder og ansvarsfordeling er blitt ganske jevnt fordelt. Alle har fått sine kjerneoppgaver som de skal sette seg inn i før vi tar fatt på selve oppgaven med å programmere systemet. Under jobbingen med kravspesifikasjonen så har fordelingen av arbeidsmengden blitt noe skjev, men dette er hovedsaklig pga. mangel på PC-er å jobbe på og at gruppemedlemmene har noe forskjellig forståelsesnivå på hvordan oppgaven skal løses så langt.

Gruppens samarbeid og motivasjon

Jobbingen innad i gruppen fungerer meget godt så langt. Alle bidrar så godt de kan og samarbeidet med oppgavene er bra. Motivasjonen har vært noe varierende fra tid til annen, men moralen er høy og vi greier å konsentrere oss om oppgaven.

Veilederkontakt

Kontakten og samarbeidet med veileder funger upåklagelig, vi har møter som regel en gang i uken og får mange nyttige innspill og tips i forbindelse med videre jobbing.
Samarbeidet med Ergo Group fungerer også bra, de stiller opp på møter når vi trenger det og svarer på spørsmål som vi måtte ha. Det eneste som kanskje kunne vært utført noe raskere er opprettelsen av grupperom for prosjektgruppene hos Ergo Group.

Totalstatus

Så langt har jobbingen med prosjektet gått rimelig bra, alle har bidratt med sin del.
Kravspesifikasjonen og første statusraport har blitt nesten en uke forsinket med tanke på tidsfristen på fremdriftsplanen, men dette har ikke hatt noe stor innvirkning på resten av oppgaven. Jobbingen med designfasen vil begynne snarlig og vi forventer at vi skal greie å ta inn igjen de dagene som vi ligger etter skjema i løpet av denne fasen da vi nå har fått litt mer erfaring med gruppejobbingen.

_____

prosjektleder
Arve Bjørnerud

Statusrapport nummer 2

3. april 2001

Lasting og sletting av applikasjoner over internett

Oppgaver fullført så langt

- Levert forprosjekt
- Ferdig med kravspesifikasjon
- Alle gruppens medlemmer har jobbet med hvert sitt kjerneområde
- Opprettet web side for gruppen og oppgaven http://higstud5.hig.no/smartkort/
- Opprettet web side med klient testing http://studenter.hig.no/arv_bjoe/gui/
- Levert statusrapport nr 1 og 2.

Oppgaver som vi jobber med nå/skal begynne med i nærmeste fremtid

- Utarbeide design for oppgaven
- Utarbeide program kode klient/servlet.

Planlegging

For øyeblikket ligger vi etter den opprinnelige planen for designdokument som var ment å være ferdig fredag den 14.mars. Grunnen til dette er hovedsaklig at vi feilberegnet arbeidsmengden til design dokumentet noe, og at vi hadde store start problemer. En annen grunn er at andre arbeidsoppgaver sånn som obligatoriske øvinger og eksamenslesning har tatt mye av tiden vår. Som en følge av dette ble statusrapporten også forsinket litt over en uke, til 3 april.

Organisering

Under jobbingen med design dokument og koding så har fordelingen av arbeidsmengden blitt noe skjev, dette er hovedsaklig pga. mangel på PC-er å jobbe på og at gruppemedlemmene har noe forskjellig forståelsesnivå på hvordan oppgaven skal løses så langt.

Gruppens samarbeid og motivasjon

Jobbingen innad i gruppen fungerer meget godt så langt. Alle bidrar så godt de kan og samarbeidet med oppgavene er bra. Motivasjonen har vært noe varierende fra tid til annen, men moralen er høy og vi greier å konsentrere oss om oppgaven.

Veilederkontakt

Vi har dessverre glemt og melde fra til veileder de dagene vi ikke trengte å ha møte , dette beklager vi på det sterkeste. Grunnen til at dette ble glemt er at gruppeleder er distre og har litt for mye å gjøre. Men de møtene vi har hatt har vært meget informative.
Ergo Group kommer fortsatt med mange gode innspill , og vi føler at vi begynner å bli litt husvarme. Vi har fått låne en smartkort leser av EG , denne har vi ennå ikke fått til å fungere sammen med javakoden , men et møte eller email med Morten vil løse det. Når dette er løst er klient delen som godt som ferdig.

Totalstatus

Jobbingen med designdokument vil får en høyere prioritet, siden vi nå er ferdig med en eksamen og to innleveringer, det gjenstår kun en eksamen fredag den 6.april . Etter den siste eksamen, vil vi prøve og jobbe nesten hver dag , da vil designdokumentet falle på plass rimelig fort. Utvikling av kode følger gantt-skjemaet bra og vil gå som normalt.

_____

prosjektleder
Arve Bjørnerud

Statusrapport nummer 3
7. mai 2001

Lasting og sletting av applikasjoner over internett

Oppgaver fullført så langt

- Levert forprosjekt
- Ferdig med kravspesifikasjon
- Alle gruppens medlemmer har jobbet med hvert sitt kjerneområde
- Opprettet web side for gruppen og oppgaven http://higstud5.hig.no/smartkort/
- Opprettet web side med klient testing http://studenter.hig.no/arv_bjoe/gui/
- Levert statusrapport nr 1, 2 og 3.
- Ferdig med design dokument

Oppgaver som vi jobber med nå/skal begynne med i nærmeste fremtid

- Snart fullført programmeringsdelen
- Jobbet en del med rapporten

Planlegging
Vi har etter å hatt ett gruppemøte blitt enige om å droppe Flash delen av oppgaven da den vil ta for lang tid å fullføre, og derfor ha en negativ innvirkning på de andre og mer vesentlige delene av oppgaven. Vi fant ut under arbeidet med prosjektet i klient server programmering at det innebar mye tidkrevende jobbing. Vi fokuserer derfor mer på de viktigste delene av prosjektet.

Organisering
Sånn som det har utviklet seg i arbeidet med prosjektet så har vi funnet ut at den beste måten for oss har vært å ha en som programmerer og de andre har ansvaret for å utarbeide prosjektrapportene og komme med innspill til programmeringen.

Gruppens samarbeid og motivasjon
Jobbingen innad i gruppen fungerer meget godt så langt. Alle bidrar så godt de kan og samarbeidet med oppgavene er bra. Motivasjonen har i den senere tid vært bedre enn tidligere og har derfor resultert i bedre resultater. En av hovedgrunnene til den bedrede motivasjonen er at vi etter påske kun har jobbet med hovedprosjektet siden vi ikke har andre fag og konsentrere oss om.

Veilederkontakt
Det har etter påske vært mindre behov for veilederkontakt enn tidligere. Dette fordi vi har kommet godt i gang med prosjektjobbingen. Vi har likevel hatt noen møter der vi har fått nyttige innspill om forandringer/forbedringer på oppgaven.

Totalstatus
Vi har fortsatt noe problemer med SSL delen av oppgaven, men etter at vi ble enige om å droppe Flashdelen ser det nå ut til at vi ligger greit an til å bli ferdige til den fastsatte datoen 23. mai forutsatt at SSL delen ikke bruker for lang tid.

_____
prosjektleder
Arve Bjørnerud

Vedlegg C

Kildekodemal for kravspek

```
/*
Sist endret : 25.01.2001
Endret av : Arve Bjørnerud

Endringer : 25012001  av AB
   La til funksjon ........
   Endret klasse .......

Endringer : 23012001 av MK
   Ny klasse

Opprettet : 20012001 av AB
   Lagde Grunnleggende design
*/


package test002;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Frame1 extends JFrame {                    // Alt innenfor { }
  public Frame1() {                                     // SKAL tabuleres inn et hopp
    if ( something ) {
      thendothis();
    } else {
      dontdoitatall();
    }
  }

  public thendothis() {                             // Variable skal begynne med
    int iTall;                                              // en forkortelse som forteller
    String sTekst;                                          // typen til variablen.
    TextField tfInput    = new TextField ();
  }
}
```

Vedlegg D

MySql database oppbygning

Database CardBase – tabell application

Database CardBase - tabell card

PRIMÆR Ja CardNr

Database CardBase - tabell cardapp

Database CardBase - tabell files

PRIMÆR Ja fid