**NTNU**
**Norges teknisk-naturvitenskapelige universitet**

**Fakultet for fysikk, informatikk og matematikk**

**Institutt for datateknikk og informasjonsvitenskap**

**BOKMÅL**

Sensurfrist: 22. juni

# Eksamen i fag
# TDT4140 Systemutvikling

## Tirsdag 27. mai 2004
## kl 0900 - 1400

**Hjelpemidler A1:**
Kalkulator tillatt
Alle trykte og håndskrevne hjelpemidler tillatt:

**Faglig kontakt under eksamen:**
Professor Tor Stålhane, tlf. 94484

Poengene viser hvor mange poeng det er mulig å få på hver oppgave. Innen en oppgave teller deloppgaver likt, med mindre annet er angitt.

# Lykke til!

.

# Innledning

Over alt i oppgaven der vi bruker ordet ”systemet” mener vi Mugin som er beskrevet i vedlegg A.

Dersom du trenger informasjon som ikke står i oppgaveteksten må du:

- Kort forklare hvorfor du trenger denne informasjonen
- Gjøre de nødvendige antagelsene. Disse antagelsene må beskrives i besvarelsen.

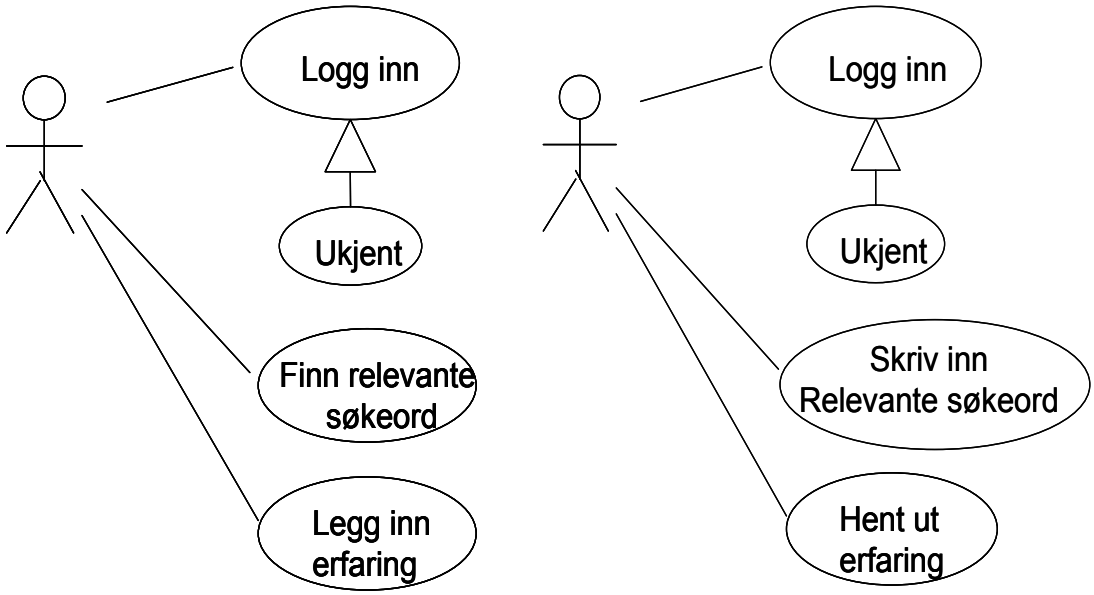
# Oppgave 1 – Use case, 25 poeng

1. Forklar hensikten med use case og hvordan vi bør gå fram for å lage de.

***Use case er en metode for å illustrere brukernes måte å bruke systemet på. Hvert use case diagram beskriver vanligvis et scenario – funksjoner og roller. Noen gode regler for å lage use case:***

- ***Involver brukerne i hele prosessen. Pass p på å bruke brukernes begreper og modeller av virkeligheten.***
- ***Spesifiser et scenario***
- ***Fokus på hva som skal gjøres, ikke på hvordan det skal gjøres***
- ***Identifiser alle roller som deltar***
- ***Hold det enkelt. Store, kompliserte scenarier fører til å brukeren mister oversikten***
- ***Vær sparsom med bruk av <<include>> stereotyper.***
- ***Pass på nivået – det skal ikke være så høynivå at det blir intetsigende, men heller ikke så lavnivå at vi mister oversikten.***

2. Lag use case diagram for følgende funksjoner
   - En person legger inn en erfaring i Mugin
   - En person leter i Mugin etter erfaring på et bestemt område, f.eks. testing.



***Det er en smak sak om man vil ha med et eget "logg ut" use case. Egentlig øker det kompleksiteten, uten å legge til noe ny info – noe som vi ikke allerede visste.***

3. Lag tekstlige use case for følgende funksjon:
   - Ny bruker blir registrert i systemet.

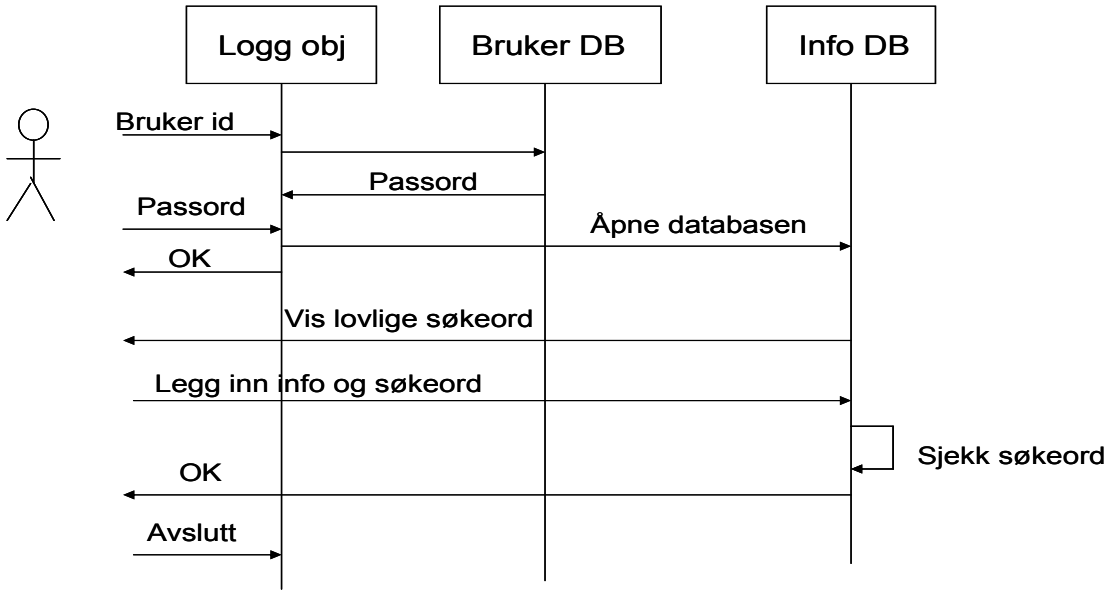
| ***Ny bruker blir registrert i systemet*** |
|---|
| ***1. Administrator logger seg på Hugin***<br>***2. Administrator legger inn brukernavn og passord***<br>***3. Administrator logger seg av*** |
| ***1.1 Feil brukernavn eller passord. Gi feilmelding***<br>***1.2 Inkrementer antall forsøk***<br>***1.3 For mange forsøk – terminer sesjonen***<br>***1.4 Ellers, gå tilbake til 1***<br><br>***2.1 Brukernavn eller passord er ikke unikt***<br>***2.2 Velg nytt brukernavn eller passord***<br>***2.3 Gå tilbake til 2*** |

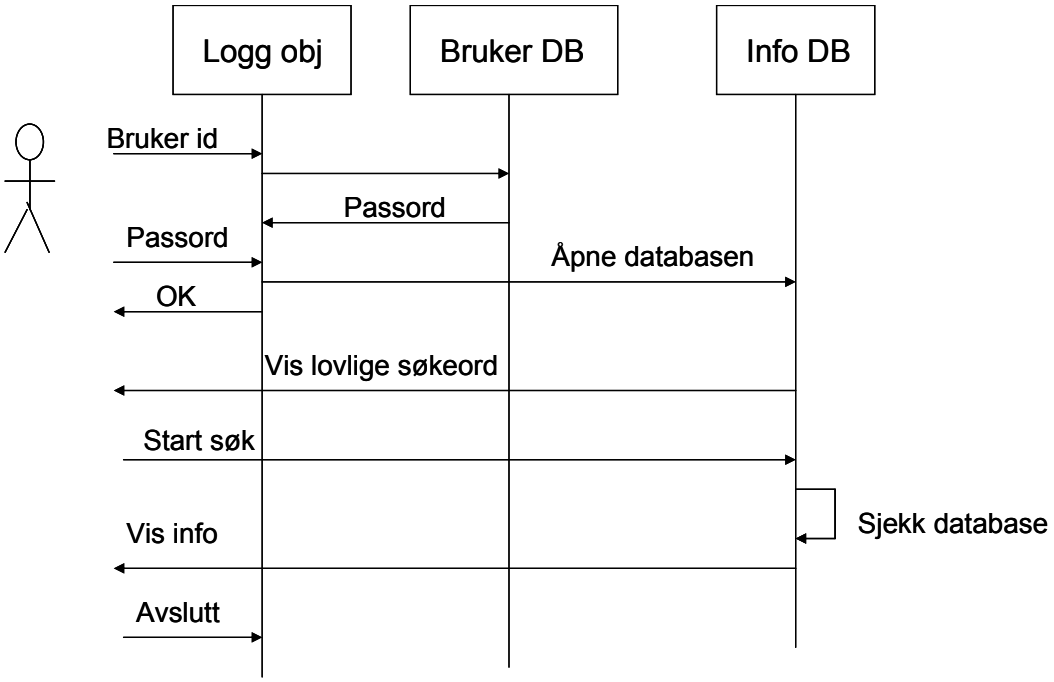
***Punktene 1.2 og 1.3 er ikke absolutt nødvendige, men vil definitivt trekke besvarelsen oppover.***

4. Når egner diagramformen for use case seg best og når egner tekstformen av use case seg best? Begrunn svaret.

***Diagramformen egner seg best ved den første kommunikasjonen / diskusjonen med brukerne – særlig viss brukerne mangler erfaring med datasystemer.***
***Tekstformen egner seg best når vi kommuniserer med kunder med erfaring i å bruke datasystemer og for kommunikasjon med utviklere, designere, testere, osv***.

5. Lag sekvensdiagrammer for de situasjonene som er beskrevet i oppgave 1, spørsmål 2.






***Det er egentlig en smak sak om man vil ta med returene i diagrammet. Egentlig er det opplagt og gir ikke noe ny info, bare et mer komplekst diagram. I begge diagrammene er det viktig at man viser handtering av både søkeord og info. Alternativt kunne man skrive en kommentar at man alltid får et vindu med alle lovlige søkeord når man har logget seg inn som bruker på systemet.***

# Oppgave 2 – Planlegging, 30 poeng

En god prosjektplan består av en tidsplan – Ganttdiagram – og en risikoanalyse. Ganttdiagrammet beskriver hvem som skal gjøre hva når i prosjektet.

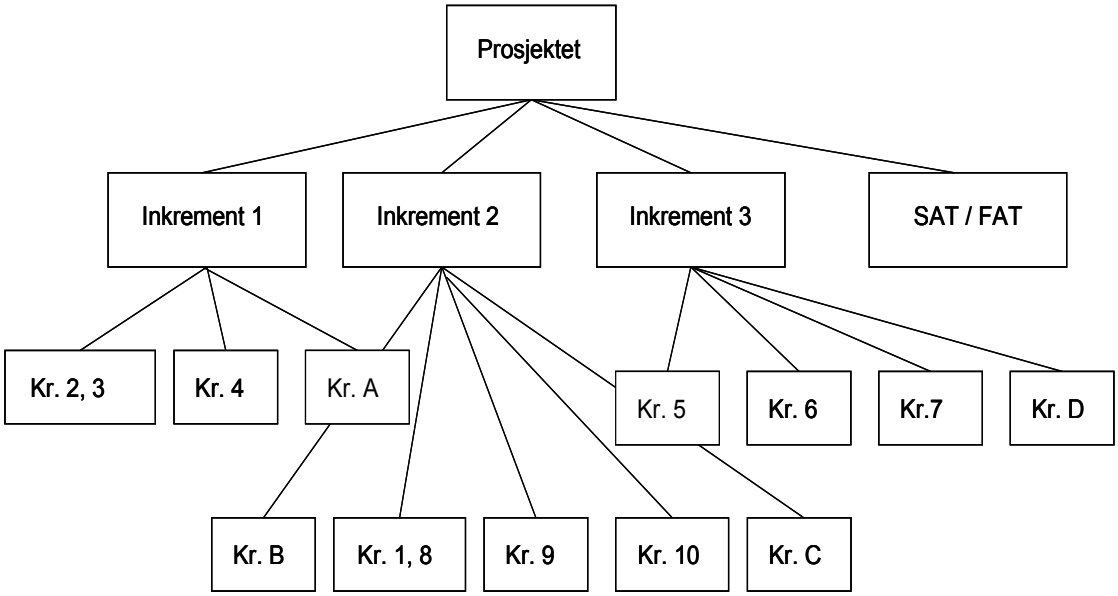
For å se hvordan systemet blir brukt så tidlig som mulig, krever SoT at systemet skal levers, testes og godkjennes i tre inkrementer.

1. Lag en WBS – Work Breakdown Structure – for prosjektet. Bryt prosjektet ned til aktiviteter som etter ***din mening*** tar maks ett ukeverk.

***I den WBSen som er vist under er det lagt til fire aktiviteter – A, B, C og D. B er den jobben som er nødvendig for å slitte opp systemet slik at det kan kjøres fra andre steder. Resten er testaktiviteter. SAT og FAT – slutt-testene – er ikke brutt ned videre. Vi setter av ett ukeverk til dette. Noen småaktiviteter – e.g. 2 og 3 og 1 og 8 - er slått samme til ei pakke ut fra forutsetningen om at begge kan gjøres på ett ukeverk. Det er imidlertid ikke noen feil å sette av ei uke pr. krav. Dette er helt og fullt opp til hver enkelt student.***

***Det er gunstig å starte med en overordna pakke for hvert inkrement selv om dette ikke er noe krav. Det er også eksplisitt bedt om at det skal inkluderes testaktiviteter for hvert inkrement. Disse aktivitetene må derfor være med.***

***Den rekkefølgen som er brukt er den jeg synes ser greiest ut. Det finnes imidlertid andre rekkefølger på aktivitetene som er like bra.***




2. Del systemkravene opp i tre inkrementer som tilfredsstiller følgende krav:
   - Første inkrement skal gjøre det mulig å samle inn og gjenbruke erfaringer.
   - Andre inkrement skal gjøre det mulig bruke systemet både hjemme og ute.
   - Tredje inkrement skal realisere resten av kravene.

   Bruk numrene på krava i vedlegg A til å vise hvilke krav hvert inkrement skal tilfredsstille. Pass på at hvert inkrement skal være et ferdig testet, kjørbart delsystem

- ***Inkrement 1: Krav 1, 2, 3, 4, 8, 9. Dette gjør at man kan registrere seg som bruker og legge in og hente ut erfaringer basert på et sett av predefinerte søkeord. Det er også mulig å droppe krav 1, 8 og 9 i første inkrement og ha et felles passord for alle i den første testfasen. I dette tilfelle ville det være naturlig å legge krav 1, 8 og 9 til Inkrement 2. personlig vil jeg foretrekke dette for å gjøre forskjellene mellom arbeidsinnsatsen i inkrementene mer lik.***
- ***Inkrement 2.: Viss ikke krav 1, 8 og 9 er tatt med før må de inn her. I tillegg må man her dele opp systemet i et tynt presentasjonslag og et lag som gjør beregninger og data lagring. Det er ikke nødvendig at studenten diskuterer trelagsmodellen, tykke og tynne klienter osv, selv om det er flott viss de har det med.***
- ***Inkrement 3: Krav 5, 6 og 7.***

3. Lag et Ganttdiagram for prosjektet.

***Hver rute i diagrammet angir en halv kalenderuke. Jeg har forutsatt at det er to personer som jobber på prosjektet. Dette gir to ukeverk pr uke. Det er OK å gjøre andre antagelser, men de må uansett dokumenteres. I tillegg må det være en fornuftig sammenhang mellom størrelsen på en jobb, antall personer som jobber i prosjektet og den tiden som er satt av til hver jobb i Ganttdiagrammet.***

***Dersom studenten har valgt å sette mer enn to personer på prosjektet bør det være aktiviteter som går i parallell. Det er urealistisk å anta at tre personer eller mer kan jobbe effektivt sammen på samme arbeidspakke.***

| Pakke | Uke | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 |
| 2, 3 | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | |
| A | | | | | | | | | | | | | |
| B | | | | | | | | | | | | | |
| 1, 8 | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | |
| 10 | | | | | | | | | | | | | |
| C | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | |
| D | | | | | | | | | | | | | |
| S / F | | | | | | | | | | | | | |

4. Lag en risikoanalyse for prosjektet og dokumenter den i en tabell. Bruk det dere har opplevd i "Fellesprosjektet" og eventuelle andre utviklingsprosjektet til å identifisere mulige risikofaktorer.

***Det viktigste med risikoanalysen er at den er skikklig dokumentert og at de momentene som er vist i tabellen under er tatt med. Det er ikke et krav at man skal bruke en tabell – det kan godt være ei punktliste isteden.***

| ***Hendelse*** | ***Konsekvens*** | ***Sannsynelighet*** | ***Risiko*** | ***Reaktive tiltak*** | ***Proaktive tiltak*** | ***Ansvarlig*** |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

***Risiko er produktet av konsekvens og sannsynelighet. Konsekvens og sannsynelighet kan angis som tall – f.eks. fra 1 til 10 – eller som lav, middels og høy. I det siste tilfell må man finne en eller annen måte å "multiplisere" de kvalitative verdiene på for å få til en gradering av risikoen. Studentene står relativt fritt i hvordan de vil gjøre dette, men det må være fornuft i det – f.eks. må det gi en brukbar rangering av risikofaktorene.***

***Det er ikke viktig hvilke risikofaktorer som er tatt med i tabellen. Sensor må bruke sin egen erfaring til å se om det som er med er fornuftig eller ikke. De faktorene som er tatt med – minimum 3 – må være relevante for et prosjekt av denne størrelsen.***

5. Hvorfor er det viktig å oppdatere risikotabellen i løpet av prosjektet?

***Risikotabellen viser de risikofaktorene vi ser nå som angår prosjektet. Over tiden forandrer prosjekttilstanden seg. Noen risikofaktorer forsvinner – er ikke relevante lenger – og nye dukker opp, f.eks. fordi vi etter hvert får en bedre forståelse av de oppgavene vi skal utføre***.

# Oppgave 3 – Klassediagram, 25 poeng

1. Forklar hva et klassediagram er og hva det bør inneholde. Forklar forskjellen på innhold i klassediagrammene som trengs tidlig i utviklingen - før vi begynner å tenke systemdesign, men bare trenger en oversikt over de viktigste klassene i designfasen – i systemdesign og i implementasjonsfasen. Hvorfor er det forskjellige krav til klassediagram i disse tre fasene?

***Et klassediagram er en beskrivelse av en klasse. En klasse består et sett av attributter som naturlig hører sammen, samt et sett av metoder som skal gjøre det mulig for andre klasser å gjøre beregninger og hente ut resultater fra klassens. I tillegg kan en klasse ha et sett av intern metoder som brukes til å administrere innholdet i klassen. Et komplett klassediagram skal inneholde navn på klassene, relasjoner mellom klassene – hvem er ansvarlig for hva – metoder og attributter. Vi kan med fordel benytte CRC metoden for de to første fasene.***

***Tidlig i utviklingsprosessen trenger vi bare å identifisere de viktigste klassene – bare med navn eller med navn pluss de viktigste attributtene og metodene. I tillegg må dette klassediagrammet inneholde de viktigste relasjoner mellom klasser – relasjoner som er viktige for å forstå hvordan vi skal realisere systemet.***

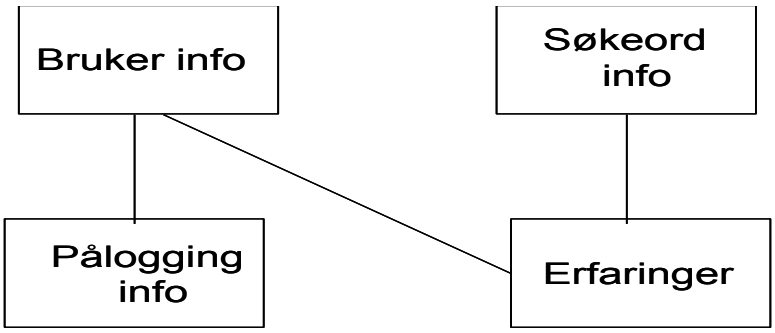
***I systemdesign må alle klassene være på plass. I tillegg må vi ha definert de attributtene og metodene som skal være tilgjengelig eksternt – utenfor klassen og relasjonene mellom kassene.***

***I implementasjonsfasen må vi ha på plass alle klasser med alle sine attributter og metoder. Vi må bestemme pakkestruktur og synelighet for alle attributter og metoder.***

***Vi trenger forskjellig mengde info i de tre fasene som er angitt fordi det er forskjellige beslutninger som skal taes i de tre fasene, fordi klassediagrammene skal oppfylle forskjellige behov og fordi de skal brukes til forskjellige aktiviteter i prosjektet.***

2. Lag samtlige klassediagram for hele systemet på et nivå som passer til bruk tidlig i utviklingen.

***Det er viktig at man finner igjen de klassene som er brukt til å lage objekter i sekvensdiagrammet – konsistens mellom klassediagram og sekvensdiagram.***



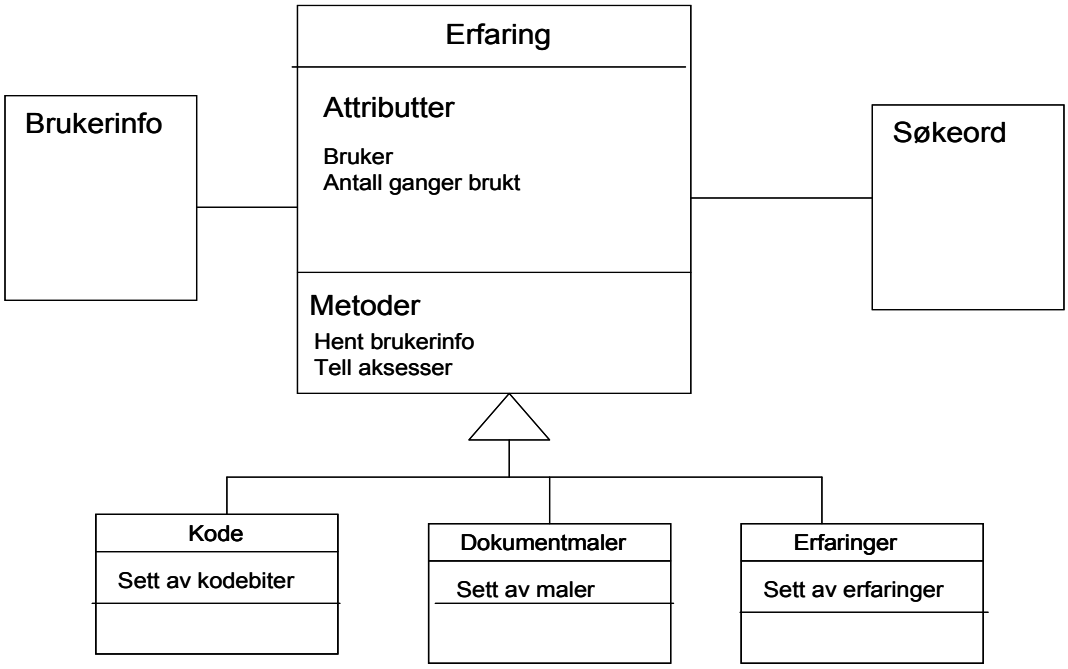

***Strengt tatt hadde vi klart oss med bare en av de to klassene "Pålogging" og "Bruker". Det må være en relasjon mellom "Erfaringer" og "Brukere" siden all erfaring er koblet til en person med navn, telefon og e-post adresse. Pålogging bruker bare brukernavn og passord. Klassen "Erfaring" må være med. Dette får studentene et tydelig hint om i spørsmål 4 i denne oppgaven.***

3. Forklar hvordan man kommer fra klassediagram i den tidlige fasen til klassediagram for designfasen

***To metoder er viktige:***

- ***CRC, som er en manuell eksekvering av use case der man identifiserer***
  - ***Hva hver klasse skal være ansvarlig for å administrere***
  - ***Hva den skal hente hos andre klasse***
  - ***Hva slags metoder klassen trenger for å oppfylle forpliktelsene sine.***
- ***En iterativ prosess der man bruker use case og enkle sekvensdiagrammer for å bestemme hvordan man skalfordele attributter og metoder.***

4. Lag klassediagram for klassen "Erfaring". Diagrammet skal også inneholde eventuelle spesialiseringer og relasjoner.




***Det er fullt mulig å legge på flere spesialiseringer – de som er brukt er de jeg kom på. Det kan for eksempel være aktuelt å inkludere sjekkliste som en egen spesialisering. På en eller annen måte må antall referanser til hver enkelt erfaring, slik at man kan lage statistikk over antall ganger hver enkelt erfaring er brukt. Dersom studenten ikke har gjort det klart at dette gjøres et annet sted, så må metoder og attributter for dette være med i klassen "Erfaringer".***

***Det er viktig at vi viser relasjonene til andre klasser. Disse klassene skal / behøver imidlertid ikke å beskrives nærmere.***

# Oppgave 4 – Testing, 20 poeng

Systemet kommer til å bli utvidet senere. Det er f.eks. allerede planlagt å utvide det med et planleggingsverktøy. Det er derfor viktig at systemet er lett å vedlikeholde. I tillegg må systemet være brukervennlig.

1. Forklar de to begrepene "vedlikeholdbarhet" og "brukervennlighet". Se appendix B.

***Vedlikeholdbarhet: hvor enkelt det er å finne og rette feil og hvor enkelt det er å gjøre endringer i systemet. I tillegg må det være lett å teste at endringa funger som planlagt (testbarhet) og at endringene ikke ødelegger allerede eksisterende funksjonalitet (stabilitet).***

***Brukervennliget: hvor enkelt er det å forstå og lære seg å bruke systemet.***

2. Lag en test for vedlikeholdbarhet – et sett av testtilfeller med input og godkjenningskriterier.

***En test for vedlikeholdbarhet må gå ut på å endre kode i systemet. Det etterfølgende er ment som et eksempel.***

***Test for vedlikeholdbarhet:***

- ***Kunden legger inn en feil i koden. For at feilen skal være realistisk, bør dette enten være en feil som man tidligere har fjernet under debugging eller en feil som man har opplevd i et tidligere, liknede system.***
- ***Feilen rapporteres gjennom de avtalte kanaler***
- ***Utvikleren skal rette feilen, teste at rettingen fungerer og at rettingen ikke har medført andre feil i systemet.***
- ***Hele jobben skal ta maksimalt 48 timer.***

***En liknende test kan gjøre for en endring, f.eks. en enkel utvidelse eller endring av en funksjon i systemet.***

3. Lag en test for brukervennlighet – et sett av testtilfeller med input og godkjenningskriterier.

***En test for brukervennlighet må involvere brukeren. En god test må bestå av opplæring etterfulgt av at brukerne skal bruke systemet til å utføre en eller flere oppgaver. Kravene til testen må gjelde feilrate og gjennomføringstid. Det etterfølgende er bare ment å være et eksempel.***

***Test for brukervennlighet:***

- ***Brukerne skal gjennomgå et tre dagers kurs. Kurset skal dekke bruken av systemet til å gjøre oppgavene G1, G2 og G3.***
- ***Brukene utfører deretter oppgavene.***
- ***Testen er godkjent viss ingen av brukerne***
    - ***bruker mer enn 20 minutter på hver oppgave***
    - ***ikke stiller mer enn to spørsmål som har med systemet å gjøre***
    - ***gir funksjonen en karakter på minst 3. Karakteren 0 betyr ubrukerlig og karakteren 5 betyr fantastisk.***

# Vedlegg A System for lagring av prosjekterfaring.

Vårt firma – Saker og Ting eller SoT – skal lage et system for å ta vare på erfaringer fra prosjekter. Systemet, som heter Mugin, skal være tilgjengelig både for de som sitter i bedriftens lokaler – ”hjemme” - og de som er utplassert hos kunder - ”ute”.

Systemet skal tilfredsstille følgende krav:

1. All tilgang til systemet skal være passordbeskyttet.
2. Alle med tilgang skal kunne lagre erfaringer.
3. En erfaring kan være en kodebit – f.eks. en komponent – ei sjekkliste, en dokumentmal, et pattern eller et eksempel.
4. Alle med tilgang til systemet kan søke etter og skrive ut eller kopiere erfaringer.
5. Hver enkelt erfaring skal bestå av
   - info om den som har lagt inn erfaringen – navn, telefon og e-post adresse
   - ett eller flere søkeord fra ei forhåndsdefinert liste av lovlige søkeord. Søkeordene definerer erfaringsområde – f.eks. ”systemtest” – og erfaringsinnhold – f.eks. ”testplan mal”.
   - selve erfaringa i form av fri tekst
   - dato da erfaringa ble lagt inn i systemet.
6. Det er mulig for brukerne å foreslå nye søkeord. De nye søkeordene registreres på et eget område i systemet. Systemadministrator vil med jamne mellomrom gå gjennom forslagene og oppdatere lista over lovlige søkeord.
7. Systemet skal holde oversikt over hvor ofte hver enkelt erfaring blir aksessert. Dette skal brukes til å lage en månedlig oversikt over de mest brukte erfaringene.
8. En person er systemadministrator. Han er den eneste som har tillatelse til å slette erfaringer, f.eks. viss de ikke er relevante lenger.
9. De som skal bruke systemet må registrere seg som brukere. Dette skjer ved personlig henvendelse til systemadministrator som tildeler brukernavn og passord.
10. Brukerne kan selv senere endre sitt passord, men passordet må være unikt i systemet – flere personer kan ikke ha samme passord.

# Vedlegg B – ISO 9126