Prosjekt i TTK4235 Tilpassede datasystemer

Øving i systemutvikling – «Heisprosjektet»

Styresystem for heis

Sist revidert: 2017 (RRH/BOHE)

Institutt for teknisk kybernetikk
NTNU

Innhold

1 Introduksjon

1.1 Motivasjon

I industrisammenheng benyttes programmeringsspråket C i en rekke systemer, særlig i sanntidsapplikasjoner og maskinvarenær programvare.

I denne oppgaven skal vi lage et styresystem for en heis. Systemet skal beskrives i UML og deretter implementeres i C.

Læringsmålet for prosjektet er å gi praktisk erfaring med systemutvikling med utgangspunkt i den pragmatiske V-modellen som metodikk, UML som verktøy for systemkonstruksjon, og programmeringsspråket C som implementeringsspråk.

1.2 Vurdering

Vurdering av prosjektet gjøres på grunnlag av

- UML-basert designdokumentasjon (muntlig presentert)
- Kodekvalitet
- Fullførelsesgrad (Factory Acceptance Test, FAT)

For detaljer om format og gjennomføring av vurderingen, se avsnitt 6 samt oppdatert informasjon på it's learning.

2 Utstyrsbeskrivelse

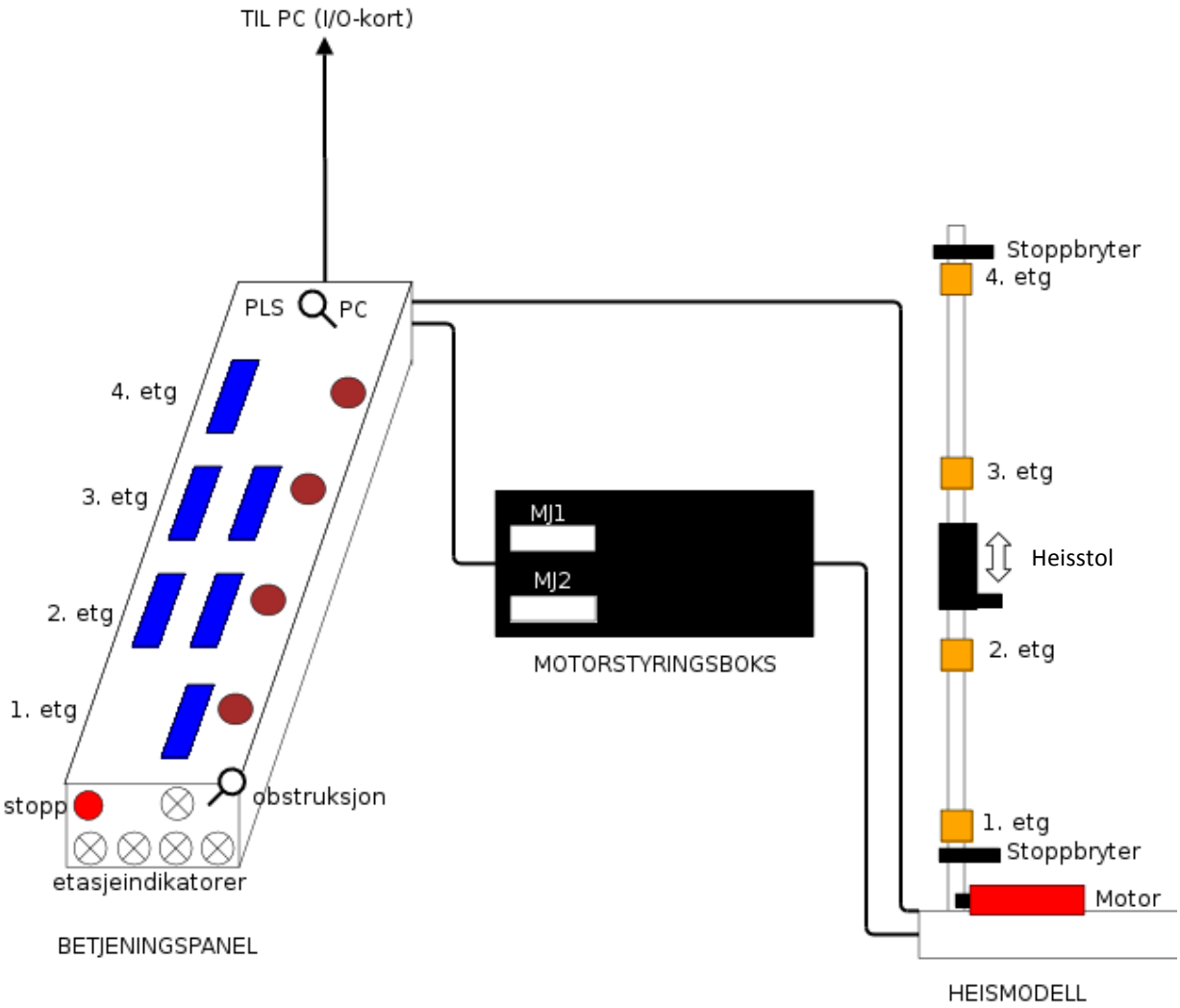
Vi skal bruke en fysisk modell av en heis. Modellen består av tre hoveddeler:

1. selve heismodellen,
2. et betjeningspanel og
3. en motorstyringsboks.

Det finnes en heismodell på hver arbeidsplass på Sanntidssalen.

2.1 Heismodell

Modellen er en miniatyrheis med 4 etasjer (Figur 1). Den har en likestrømsmotor som kan brukes for å kjøre en *heisstol* opp og ned langs en *bane* (tilsvarer *heissjakta* i en virkelig heis). Heisstolen er forbundet med motoren via et gir og en snor. I toppen og bunnen av banen er det *endestoppbrytere* som beskytter modellen mot skade ved feil i styresystemet. Dersom heisen beveger seg utenfor definert område (dvs. under første etasje eller over fjerde etasje), vil tilsvarende endestoppbryter bli aktivert, og pådraget til motoren brytes. Heisstolen må da *manuelt skyves bort fra bryterne* for at en skal kunne sette på et nytt pådrag. Ved hver etasje sitter det en *etasjeindikator* (en nærhetsdetektor) som gir et signal når heisstolen er i den aktuelle etasjen.



Figur 1 Skisse av heismodellen

2.2 Betjeningspanel

Betjeningspanelet består av en rekke knapper og brytere. Øverst på panelet finnes det en bryter merket «PC-PLS». Denne indikerer om heisen skal styres fra PC eller PLS. I denne oppgaven skal vi styre heisen fra en PC, så bryteren må stå innstilt på det.

Betjeningspanelet er delt i to, det har et *etasjepanel* og et *heispanel*.

2.2.1 Etasjepanel

På oversiden av boksen finnes etasjepanelet. Her finnes betjeningsorganer som en typisk vil finne utenfor heisdørene på en virkelig heis. Det er bestillingsknapper *(OPP/NED)* for samtlige etasjer, med lys som kan vise at en bestilling er foretatt. Videre er det ett lys for hver etasje (etasjeindikatorer) som kan skrus på for å indikere hvilken etasje heisen befinner seg i.

2.2.2 Heispanel

På kortsiden av boksen finner vi betjeningspanelet. Her er det betjeningsorganer en kan finne inne i heisstolen i en virkelig heis. Det er bestillingsknapper for hver etasje, en stoppknapp og en obstruksjonbryter. Alle knappene har lys som kan settes. Stoppknappen representerer nødstoppknappen som finnes inne i en heis.

Obstruksjonsbryteren er tenkt å simulere f.eks. obstruksjon i døråpningen. Denne skal vi *ikke* bruke i dette prosjektet.

På heispanelet er det også et lys merket DØR ÅPEN, som skal indikere at heisdøra er åpen.

2.3 Motorstyringsboks

Denne boksen sørger for effektforsyning til motoren i tillegg til en forsterkning av pådraget som en setter ut fra PC-en (IO-kortet). Motoren kan få mellom 0 V (i ro) og 5 V (fullt pådrag). Dette kan settes ved et funksjonskall i styringsprogrammet. En må også sette et retningsbit for motoren for å få heisstolen til å kjøre i ønsket retning (0 tilsvarer *opp* og 1 tilsvarer *ned*).

Ut fra motorstyringsboksen kan en også hente en hastighetsmåling i form av et analogt tachosignal (et signal som har spenning proporsjonal med motorhastigheten), og en digital posisjonsmåling. Disse to målesignalene er det ikke nødvendig å bruke i dette prosjektet.

3 Virkemåte og oppkobling

Heismodellen er laget for å likne på en virkelig heis. For å få den til å fungere som ønsket, er det en del punkter en bør merke seg:

- Alle lys må settes eksplisitt. Lyset merket *3 ETASJE* settes for eksempel *ikke* automatisk selv om heisen befinner seg i 3. etasje, dette må gjøres av programvaren dere skal skrive.
- Heisen stopper dersom en av endestoppbryterne aktiveres. Dette skal normalt ikke skje og indikerer en feil i programvaren. Hvis det skjer, må heisen resettes manuelt ved å skyve heisstolen bort fra bryteren.
- Når motorpådraget settes til null vil friksjonen i giret sørge for at motoren taper energi og stanser. Det kan imidlertid ta litt tid å bremse på denne måten. Et triks kan være å skifte retning på motoren et lite øyeblikk rett før pådraget settes til null.
- Rød og blå ledning er spenning til motoren. Disse kobles til motorstyringsboksen på henholdsvis *M+* og *M-*.

4 Funksjonsspesifikasjon

Dette avsnittet tar for seg funksjonsspesifikasjonen for styringssystemet. Dette er utgangpunktet for arbeidet iht. V-modellen, og vil danne grunnlaget for FAT-kriteriene (*Factory Acceptance Test*).

Det er et mål å gi heisen en oppførsel som likner en virkelig heis. Styringssystemet som dere skal lage skal tilfredsstille følgende funksjonelle krav.

4.1 Oppstart

Ved oppstart kan heisstolen være i en ikke-definert tilstand mellom to etasjer. Ikke-definert tilstand betyr at det ikke er mulig for styringssystemet å avgjøre hvor heisstolen befinner seg. Heisstolen skal kjøres til en definert tilstand før heisen skal reagere på knappetrykk. La heisstolen kjøre opp eller ned til den kommer til en etasje, og stopp der.

Dersom heisen står i en etasje ved oppstart, er heisen allerede i en definert tilstand.

Hvis heisen befinner over 4. eller under 1. etasje, må den manuelt flyttes før programmet startes.

4.2 Håndtering av bestillinger

Det er ikke ønskelig å komme i en situasjon hvor noen som står i endeetasjene (første eller fjerde) må vente på en heis som aldri kommer. Heisen skal heller ikke stoppe flere ganger enn nødvendig, så hvis heisen på vei opp passerer en etasje der det kun er bestilt transport nedover (eller motsatt), skal den ikke stoppe. Vi antar at alle passasjerer som venter på heisen i en etasje, går på når heisen stopper der.

Eksempel på bruk av betjeningsknappene: Hvis en står i 2. etasje og ønsker å komme til 4. etasje, vil en først trykke på knappen *OPP* i 2. etasje på etasjepanelet (dette tilsvarer at en står i 2. etasje og signaliserer at en ønsker å kjøre oppover). Når heisstolen kommer og døren åpnes, vil en så trykke på bestillingsknappen merket 4 på heispanelet (dette tilsvarer at en går inn i heisen og trykker på knappen for 4. etasje).

4.3 Bestillingslys og etasjelys

Når en for eksempel står utenfor heisen i en etasje og trykker *OPP* eller *NED*, skal lyset i den aktuelle knappen lyse inntil bestillingen er ekspedert, dvs. heisen er ankommet etasjen og døren er åpnet. Det samme gjelder lyset i bestillingsknappene inne i heisen. Når heisen beveger seg, skal etasjeindikatorene vise den siste etasjen heisen var i. Er heisen f.eks. mellom 2. og 3. etasje og på vei oppover, skal altså lampen som indikerer 2. etasje være tent.

4.4 Døren

Når heisen kommer til en etasje der noen skal inn eller ut, skal døren åpnes og være åpen i tre sekunder (dette indikeres ved at dørlyset på heispanelet er tent i tre sekunder).

Heisen skal alltid stå stille når døren er åpen.

Når tre sekunder har passert, skal døren lukkes (lyset slukkes), og heisen skal så håndtere andre ubetjente bestillinger i systemet. Hvis det ikke er noen ubetjente bestillinger, skal heisen stå i ro med døren lukket.

4.5 Obstruksjon

Obstruksjonsbryteren skal ikke ha noen innvirkning på systemet.

4.6 Stoppknappen

Når en person trykker på stoppknappen, skal heisen stoppe momentant, og alle bestillinger skal slettes. Nye bestillinger skal ikke registreres før stoppknappen er sluppet. Når stoppknappen er sluppet, skal heisen stå i ro inntil den har fått en ny bestilling.

Hvis heisen er i en etasje når stoppknappen trykkes, skal døren åpnes. Når stoppknappen slippes, skal døren forbli åpen i tre sekunder og deretter lukkes.

4.7 Generelt

Tenk over ulike situasjoner som kan oppstå og hvordan du ønsker at heisen skal oppføre seg dersom du selv sto inne i heisen. Systemet skal være robust mot uforutsette hendelser på den måten at passasjerenes sikkerhet ivaretas. Det må for eksempel alltid være mulig å komme inn og ut av heisen dersom dette er forsvarlig, altså når heisen står stille i en etasje. Systemet må også kunne håndtere en hvilken som helst sekvens av knappetrykk (f.eks. hvis noen uforvarende lener seg mot heispanelet og trykker inn alle etasjeknappene) uten at det systemet «krasjer», sikkerheten svekkes eller noen bestillinger aldri blir betjent.

5 Oppgaver

Det er anbefalt å benytte seg av den pragmatiske V-modellen for å strukturere arbeidet. UML benyttes som hjelpemiddel under systemkonstruksjonen, og det endelige systemet implementeres i C.

Funksjonsspesifikasjonen er allerede gitt i prosa-form i forrige avsnittet, og prosjektets oppgaver vil derfor være:

1. **Analyse** av funksjonsspesifikasjonen (use case-modell).
2. Design av en overordnet **systemarkitektur** (modularisering, modulgrensesnitt, kommunikasjonsmønster/kopling mellom modulene).
3. **Moduldesign** (oppførsel, indre struktur osv.).
4. **Implementasjon** av modulene i C på en Linux-maskin.
5. **Testing av modulene** hver for seg.
6. Testing av **det ferdige styresystemet**.

Resultatet av punktene ovenfor blir gjenstand for vurdering, se avsnitt 6. I det følgende beskrives deloppgavene litt nærmere.

> *Konkrete oppgaver som skal gjøres/besvares er markert med kursiv skrift (slik som dette).*

Teksten for øvrig gir utfyllende detaljer om oppgavene.

5.1 Analyse

Først skal produktspesifikasjonen analyseres. Dette er for at du som systemutvikler skal være sikker på at du og kunden er enige om hva som kreves av det ferdige systemet. Fremfor alt handler det om å systematisere informasjonen i kundens funksjonsspesifikasjon slik at du får oversikt. Hvis du er usikker på hvordan spesifikasjonen skal tolkes, kan du betrakte stud.ass. eller vit.ass. som kunde, og spørre for å oppklare usikkerheten. I tvilstilfeller er det faglærer som har det siste ordet.

> *Analysen skal resultere i et use case-diagram som viser alle aktører og use cases, og dere skal kunne redegjøre muntlig for use casenes scenarier, preconditions, triggere etc..*

Det er bare diagrammet og en muntlig redegjørelse som vil bli evaluert, men det anbefales at dere beskriver use casene i henhold til *Fowler: UML Distilled*, kap. 9 (se eksempel i figur 9.1 s. 101 og tilhørende tekst, samt relevante lysark fra forelesningene) som et grunnlag for den muntlige redegjørelsen.

Det lønner seg *ikke* å ta for lett på denne prosjektfasen – det er her du skaper deg den forståelsen og oversikten som du skal jobbe ut fra i resten av prosjektet. Hvis du er helt i villrede om hvordan du skal gå fram, bør du gå gjennom pensumlitteraturen/lysarkene og deretter evt. søke veiledning. Use case-modellen bør være mest mulig komplett, og fremfor alt: den bør være 100% meningsfull for deg og din labpartner!

5.2 Systemarkitektur

Når du har god oversikt over den ønskede funksjonaliteten, skal du etablere en overordnet arkitektur for systemet. Det kan du gjøre ved å dele opp i moduler med forskjellige oppgaver.

Det er viktig å lage moduler der alt som gjøres i en og samme modul henger naturlig sammen (dette kalles «cohesion» på fagspråket) og der modulene primært kommuniserer via funksjonskall og parameteroverføringer, ikke globale variabler o.l. (lite «coupling» mellom moduler).

> *Arkitekturmodellen skal presenteres som et overordnet kommunikasjonsdiagram, der alle hovedmodulene og deres innbyrdes dataflyt (dvs. hvilke moduler som kommuniserer med hverandre og hvilken vei dataflyten går) fremgår.*

For å få klarhet i hvilke egenskaper (attributter og operasjoner) hver modul må tilby, kan en effektiv metode være å benytte sekvensdiagrammer.

> *Dere skal som et minimum tegne et sekvensdiagram som viser hvordan modulene i den valgte arkitekturen samarbeider for å betjene følgende scenario:*
>
> - *Heisen står tom og stille i 2. etasje med døra åpen*
> - *En person signaliserer fra 1. etasje at hun ønsker transport oppover*
> - *Når heisen kommer, bestiller hun transport til 3. etasje*
> - *Scenariet avsluttes når heisen befinner seg i 3. etasje og heisdøra åpnes*
> - *For enkelhets skyld antar vi at ingen andre personer interagerer med systemet i løpet av dette scenariet.*

(Dere står selvsagt fritt til å lage flere sekvensdiagrammer etter behov.)

På dette stadiet har dere ikke tenkt ut «innmaten» i alle modulene, men dere skal ha en formening om hvilken oppgave hver modul har. Ved å lage et sekvensdiagram som bestilt her, stimuleres dere til å tenke gjennom hvilke funksjoner som må implementeres for å realisere det aktuelle scenariet.

5.3 Moduldesign

Noen av modulene som inngår i arkitekturmodellen er kanskje ferdige moduler (programbiblioteker etc.). Andre moduler må dere imidlertid designe og implementere selv.

> *Enkelte av disse modulene er det naturlig å beskrive med egnede UML-diagrammer før de implementeres. Bruk klasse- og tilstandsdiagram(-mer) til dette, evt. supplert med andre diagrammer ved behov.*

Motivasjonen for å gjøre dette er at det er lettere å vurdere og eksperimentere med designet på diagramnivå enn med (halv-) ferdig programkode, for i sistnevnte tilfelle er det veldig fort gjort å miste oversikten.

Eksempelvis vil de fleste heisstyringer inneholde en eller annen form for tilstandsmaskinoppførsel. Det er da naturlig å skille ut denne funksjonaliteten i en egen modul, og beskrive modulens oppførsel i form av en tilstandsmaskin. Gi moduler, tilstander og aksjoner (funksjoner) meningsfulle navn som ikke trenger ytterligere forklaring. Sørg for å tilfredsstille alle de funksjonelle kravene som er nevnt tidligere, og som er systematisert i use case-modellen.

Merk også følgende: selv om C ikke har en direkte analog til klassebegrepet i UML, er klassebegrepet fortsatt svært nyttig og relevant. Generelt kan en si at en UML-klasse beskriver en modul, og moduler implementeres forskjellig i forskjellige programmeringsspråk. I språket C

kan en klasse f.eks. implementeres som en modul bestående av en headerfil (.h) med funksjonsdeklarasjoner for «public-metoder», og en .c-fil med implementasjon av metoder og attributter. En klasse kan også implementeres som en struct-datatype, der klassens attributter er feltene i struct'en. Avanserte programmerere kan endog bruke funksjonspekere til å knytte funksjoner («metoder») til struct'en. Det er derfor ikke unaturlig å bruke klassediagrammer og annet som har med klasser å gjøre i designdokumentasjonen deres.

Designdokumentasjonen skal evalueres ved enkel gjennomgang, og vil bestemme 1/3 av karakteren på prosjektet. I avsnitt 7 finner dere detaljerte evalueringskriterier for designdokumentasjonen.

5.4 Implementasjon

Kildekoden skal evalueres, og vil bestemme 1/3 av karakteren på prosjektet (utlevert kode kan utelates).

> *Koden skal skrives i programmeringsspråket C. Samtlige kodefiler leveres i en og samme zip-fil (andre formater godtas ikke).*

5.4.1 Generelt

PC-en har et IO-kort av typen *National Instruments PCI 6025E*. Dette kan en få tilgang til gjennom biblioteket *Comedi*. For å gjøre ting litt enklere for dere, er det laget en liten driver for heismodellen. Denne er lagt ut på fagets hjemmeside.

Driveren består av to lag:

- `io.c/io.h` inneholder enkle funksjoner for å kommunisere med *Comedi*-biblioteket
- `elev.c/elev.h/channels.h` inneholder funksjoner som er spesifikke for heisen (*elev*ator)

Funksjonene som dere trenger for å kjøre heisen finner dere beskrevet i `elev.h`. Det er laget et lite eksempel som viser hvordan dere bruker driveren. Bytt ut `main.c` med deres egen kode.

Dere kan velge å benytte dette biblioteket, eller bruke det som et utgangspunkt for deres egen heisdriver.

Strukturen kan beskrives som et sett med nivåer.

Egen kode (programvare)
`elev.c` (programvare)
`io.c` (programvare)
Comedi (grensesnitt, programvare)
NI PCI 6025E (IO-kort, maskinvare)

Dokumentasjon til Comedi-biblioteket finner du på `http://www.comedi.org/`.

5.4.2 Komme i gang

Start Linux: Hvis ikke Linux allerede kjører på maskinen, start maskinen på nytt og velg Ubuntu i oppstartsmenyen.

Last ned og pakk ut IO-driver og eksempelkode: Last ned og pakk ut filen `heisdriver.zip` fra it's learning.

Kompiler: Kompiler kildekoden med den medfølgende *Makefile*. Dere må kanskje gå inn i katalogen `heisdriver` (`cd heisdriver`) først. Bruk kommandoen `make`, og kjør programmet ved å skrive `./heis`.

Ta vare på arbeidet deres: Dere må finne en løsning for å ta vare på koden fra uke til uke, fordi det kan hende at maskinene på salen oppdateres (dette gjøres ved å klone en «master-pc», og vil derfor slette alt som er lagret lokalt). Dere kan bruke minnepenn (lite trygt, vanskelig å dele), eller lagre i «skyen» med for eksempel Dropbox eller Google Drive. Et annet alternativ er å bruke nettverksområdet deres på NTNUs servere.

5.4.3 Implementasjon

I oppgaven som denne, der noe (her: heisen) skal styres basert på hendelser utenfor styresystemet (her: knappetrykk, etasjesensor osv.), er det gjerne naturlig å implementere en del av styresystemet som en tilstandsmaskin.

Implementasjon av tilstandsmaskiner kan gjøres på mange ulike måter. Vi anbefaler å bruke implementasjonen som er forelest i forbindelse med eksempelet «Rundeteller».

5.5 Testing av moduler

Dette er noe dere gjør for deres egen skyld, for å sikre at modulene virker enkeltvis før systemintegrasjon, jfr. V-modellen.

5.6 Testing av det ferdige systemet

Det anbefales sterkt at dere går gjennom testkriteriene (se vedlegg) på egenhånd for å sikre at systemet fungerer som forutsatt før systemet demonstreres for/testes av stud.ass. Dette blir som en generalprøve på FAT'en.

6 Evalueringskriterier

Evalueringen og karaktersetting av prosjektet gjøres på grunnlag av:

- Designdokumentasjon (UML-modeller etc.; deloppgavene 1-3 i avsnitt 5)
- Kodekvalitet (kildekode; deloppgavene 4 i avsnitt 5)
- Fullførelsesgrad (*Factory Acceptance Test,* FAT) som er ment å gi et bilde av hvor godt det ferdige styringssystemet oppfyller funksjonsspesifikasjonen.

Hvert av punktene bestemmer 1/3 av poengsummen på prosjektet. Detaljer om gjennomføringen av hvert punkt blir gitt via It's Learning.

6.1 Dokumentasjon

Dokumentasjonen skal beskrive analysen og systemdesignet slik det var på evalueringstidspunktet. Det er altså *ikke* et krav at denne dokumentasjonen (UML-modellen m.m.) skal oppdateres til å stemme 100 % med den endelige implementasjonen.

Vi kommer til å se på oversiktlighet, riktig bruk av UML (riktig diagram i forhold til hva som modelleres, abstraksjonsnivå og syntaks), om det er gitt gode og konsise beskrivelser under den muntlige presentasjonen der dette er nødvendig for å gi et komplett bilde av systemet. Vi ser også etter en naturlig modularisering (som vil fremgå av arkitekturmodellen) og samsvar mellom de ulike delene av modellen (konsistent navngiving på tvers av tilstandsdiagram og tilsvarende klasse i klassediagrammet, osv.). Det er ikke nødvendig å detaljmodellere funksjonalitet som er kjent stoff. (Eksempel: Hvis du trenger å sortere en liste, trenger du ikke å beskrive sorteringsalgoritmen i detalj.)

Dokumentasjonen skal evalueres muntlig og gruppevis etter kriteriene gitt i avsnitt 7. Dere vil da også få tilbakemelding på designvalgene dere har tatt, som dere kan ha nytte av i resten av prosjektet.

6.2 Kodekvalitet

Vurdering av kodekvaliteten er i stor grad en vurdering av hvor lett det er å lese og forstå koden. En kan se for seg at noen skal kunne fortsette utvikling og vedlikehold med så lite arbeid som mulig.

- Er det enkelt å få oversikt og finne frem i koden?
- Er kommentarene konsise og i overenstemmelse med den faktiske koden?
- Er koden modularisert med gode grensesnitt og god dekopling mellom modulene?
- Er navn på variable og funksjoner informative?

6.3 Fullførelsesgrad

Vi vil se på hva som virker og hva som ikke virker. Det er viktig at heisen fungerer stabilt og i henhold til produktspesifikasjonen. Testkriteriene er gitt i avsnitt 7.

Lykke til!

7 Vedlegg

Evalueringskriterier for designdokumentasjon (UML)

Moment		0 Utilstrekkelig	1 Tilstrekkelig	1.5 Godt	2 Meget godt
1	Use case-diagram og -beskrivelse	Diagram eller beskrivelse **mangler eller er svært mangelfull**. Viser **liten eller ingen** forståelse for hensikten med modellen.	Diagram og beskrivelse har **vesenlige mangler** iht. spec. **Betydelige** syntaksfeil. Viser **en viss forståelse** for hensikten med modellen.	Diagram og beskrivelse **gode** iht. spec. Noen **mindre** syntaksfeil. Viser **god forståelse** for hensikten med modellen.	Diagram og beskrivelse **tilnærmet komplett** iht. spec. **Meget god** syntaks. Viser **meget god** forståelse for hensikten med modellen.
2	Overordnet kommunikasjons-diagram (systemarkitektur)	Diagram eller beskrivelse **mangler eller er svært mangelfull**. **Uegnet** design. Viser **liten eller ingen** forståelse for hensikten med modellen.	Diagram og beskrivelse har **vesenlige mangler** iht. spec. **Dårlig** egnet design. **Betydelige** syntaksfeil. Viser **en viss** forståelse for hensikten med modellen.	Diagram og beskrivelse **gode** iht. spec. **Egnet** design. Noen **mindre** syntaksfeil. Viser **god** forståelse for hensikten med modellen.	Diagram og beskrivelse **tilnærmet komplett** iht. spec. **Meget godt** egnet design. **Meget god** syntaks. Viser **meget god** forståelse for hensikten med modellen.
3	Klassediagram				
4	Tilstandsdiagram				
5	Sekvensdiagram				
6	Konsistens på tvers av systemet	**Ingen** åpenbar konsistens eller sammenheng mellom modelens deler.	**Dårlig** konsistens og sammenheng mellom modelens deler. **Inkonsekvent** navngiving. **Rotete** helhetsinntrykk.	**God** konsistens og sammenheng mellom modelens deler. **Delvis konsekvent** navngiving. **Godt** helhetsinntrykk.	**Åpenbar og gjennomført** konsistens og sammenheng mellom modelens deler. **Konsekvent** navngiving. **Meget godt** helhetsinntrykk.

Evalueringskriterier for fullførelsesgrad (FAT)

	Oppstart	
1.	Sørger systemet for at heisen kommer i en definert tilstand?	
2.	Ignoreres bestillinger før heisen har kommet i en definert tilstand?	
3.	Ignoreres stoppknappen under initialisering?	
	Håndtering av bestillinger	
4.	Går heisen til riktig etasje når en bestilling mottas fra etasjepanelet?	
5.	Går heisen til riktig etasje når en bestilling mottas fra heispanelet?	
6.	Hvis heisen er på vei fra 4. etg til 1. etg og noen har bestilt OPP i 2. etg: kjører heisen til 1. etg før den kjører til 2. etg?	
7.	Håndteres alle bestillingene hvis flere av bestillingsknappene trykkes samtidig?	
8.	Vil alle bestillinger bli ekspedert, selv med vedvarende trykking av andre knapper (unntatt stopp), dvs. blir heisen aldri "fastlåst" mellom noen av etasjene?	
	Bestillingslys og etasjelys	
9.	Blir riktig etasjelys tent når heisen ankommer en etasje?	
10.	Hvis heisen befinner seg mellom 2. og 3. etg og er på vei oppover, lyser etasjelyset i 2. etg?	
11.	Blir lyset tent i bestillingsknappene når de blir trykket?	
12.	Slukkes lyset i bestillingsknappene når bestillingen er ekspedert, dvs. når heisen ankommer etasjen?	
	Dør	
13.	Åpnes døren (lyser dørlyset) når heisen stopper i en etasje?	
14.	Er døren åpen i 3 sekunder?	
15.	Står heisen stille i de 3 sekundene døren er åpen?	
16.	Lukkes døren før heisen kjøres videre?	
17.	Lukkes døren og står heisen stille når det ikke er noen nye bestillinger?	
18.	Heisen står i ro med døren lukket etter å ha betjent alle bestillinger. Åpnes døren når en bestilling kommer fra etasjen heisen står i?	
	Stoppknapp	
19.	Stopper heisen når stoppknappen trykkes?	
20.	Blir bestillingene slettet (lysene på bestillingsknappene slukkes) når stoppknappen trykkes?	
21.	Er lyset i stoppknappen kun tent mens stoppknappen er trykket?	
22.	Ignoreres trykk på alle bestillingsknappene mens stoppknappen er trykket?	
23.	Blir heisen stående i ro etter at stoppknappen er sluppet?	
24.	Husker heisen hvor den er ved nødstopp mellom etasjer (dvs. kreves ikke ny initialisering)?	
25.	Åpnes døren hvis stoppknappen aktiveres i en etasje?	
	Generelt	
26.	Hvor stabilt er programmet? Må programmet startes på nytt under presentasjonen? · **Tre poeng** hvis programmet ikke må startes på nytt · **Ett poeng** hvis dette skjer kun én gang	