Heislab

Gruppe 13, Antonie og Sandra

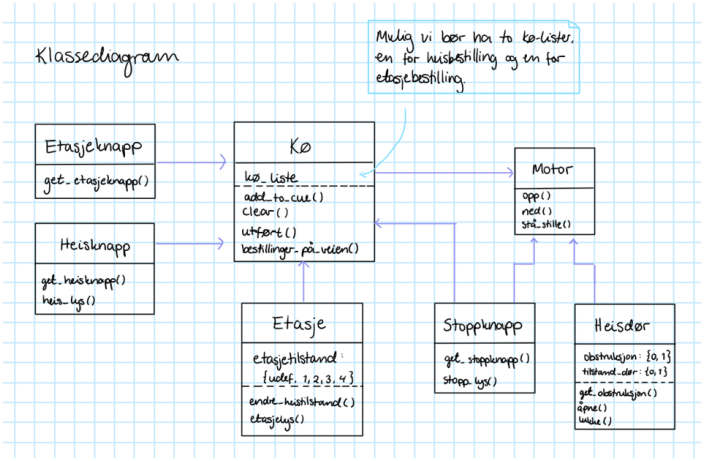
Våren 2024

1 Introduksjon

I dette prosjektet har vi utviklet programvare for logikkstyring av en heismodell med én heis. Dette har blitt gjort i programmeringsspråket C. Før implementasjon ble systemet beskrevet og dokumentert ved bruk av ulike UML-diagrammer. For å sikre et strukturert arbeid har også den pragmatiske V-modellen blitt forsøkt brukt. I denne rapporten vil vi vise til diagrammene som ble laget ved det opprinnelige designet, resultater ved FAT-testen, samt refleksjon rundt hele prosjektet.

2 UML

UML, eller Unified Modeling Language, er et 'språk' som brukes for å beskrive og analysere systemer. Ulike diagrammer beskriver ulike sider ved et system, som struktur og virkemåte. I dette prosjektet har vi benyttet oss av både klasse-, sekvens-, og tilstandsdiagram.



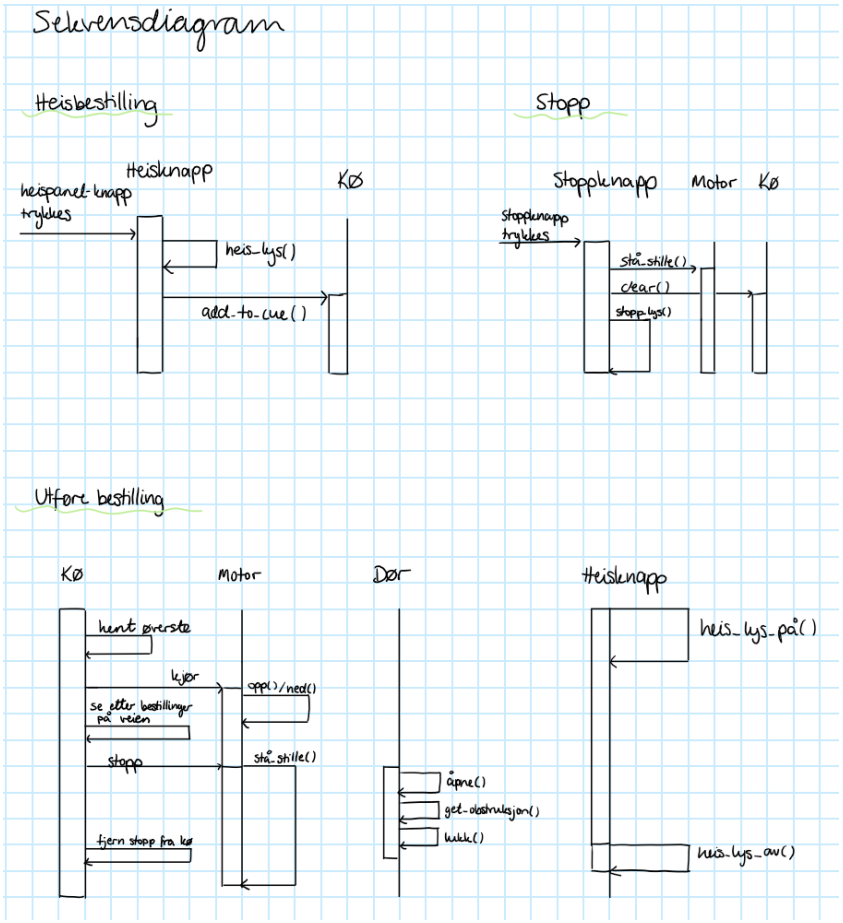
Figur 1: Klassediagram.

2.1 Klassediagram

Vi lagde i utgangspunktet et klassediagram bestående av 7 klasser: Etasjeknapp, Heisknapp, Kø, Etasje, Stoppknapp, Motor og Heisdør, se figur 1. Vi var usikre på hvordan vi skulle plassere I/O-elementer som knapper, derfor ble disse laget som egne klasser. Etter implementasjon endte vi opp med klassene 'door', 'floor', 'motor', 'que' og 'stop', i tillegg til 'elevio' fra den utgitte koden.

2.2 Sekvensdiagram

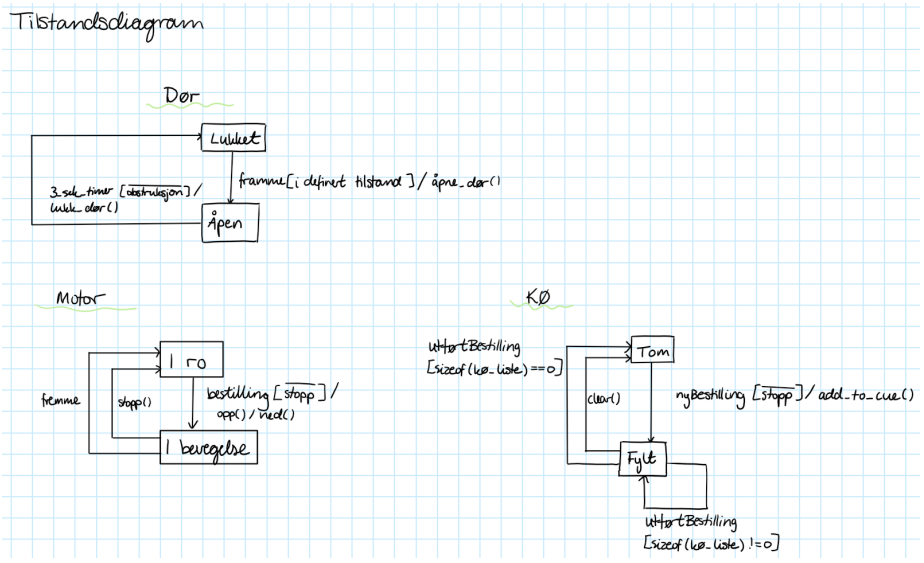
Hensikten med et sekvensdiagram er å vise hvordan en gruppe objekter samhandler for å utføre en oppgave. Vi valgte her å lage sekvensdiagram for å legge til en heisbestilling, utføre bestilling og bruk av stoppknapp, vist i figur 2.



Figur 2: Sekvensdiagram av hhv. ny heisbestilling, bruk av stoppknapp og utførelse av bestilling.

2.3 Tilstandsdiagram

Et tilstandsdiagram er nyttig for å vise hvordan tilstanden til et system eller delsystem skal endre seg som følge av ulike triggere. Vi har laget tilstandsdiagram over henholdsvis dør, kø og motor, se figur 3.



Figur 3: Tilstandsdiagram for hhv. dør, kø og motor.

3 FAT-test

FAT-testen gikk veldig bra, men vi hadde misforstått to av heisspesifikasjonene, H3 og S7. I forhold til spesifikasjon S7 hadde vi ikke implementert at heisen skulle bli stående stille etter at knappen var trykket, men gikk til en definert tilstand. Dette fikk vi enkelt rettet opp i, og den har nå den ønskede responsen.

Vi oppdaget etter FAT-testen at vi lagde programvaren mye mer komplisert enn den hadde behøvd å være, da vi *ikke* antok at alle gikk på heisen i en etasje. Vi tok høyde for at noen som ville ned ikke ville gått på en heis som var på vei oppover. Vi noterer oss dette bak øret til neste heislab hvor det er flere heiser; da vil kanskje denne løsningen være nyttig. Vi endret koden slik at den oppfylte H3 etter testen, og fikk dermed en enklere og kortere kode.

4 Refleksjon

4.1 UML og V-modellen

Bruken av UML og V-modellen har hatt påvirkning i implementasjonen i ulike stadier av prosjektet. UML-diagrammene påvirket nok mest i startfasen, og var et godt verktøy for å få oversikt over oppgaven og hvilke moduler vi kom til å trenge. Dette gjorde selve implementasjonen mye enklere enn det ville vært om man bare hadde begynt rett på programmeringen. Det er likevel viktig å ikke bruke for lang tid på diagrammene, da man mest sannsynlig vil komme til å endre på ting underveis.

Vi skulle forsøke å bruke den pragmatiske V-modellen i dette prosjektet. I praksis ble det for oss brukt en mer videreutviklet modell hvor vi gikk mye fram og tilbake mellom implementasjon og enhetstesting, før integrasjonstestingen til slutt. Dette virket mest hensiktsmessig for oss. Hadde vi fulgt den pragmatiske V-modellen til punkt og prikke, ville oppgaven vært vanskeligere å løse da det var så mye interaksjon mellom de ulike modulene, og en endring i én av de ville ført til at vi måtte endret på flere.

Da vi lagde det opprinnelige designet vårt hadde vi ikke enda sett på hva som allerede fantes i den utdelte koden, derfor ble det noen endringer underveis. Vi trengte for eksempel ikke å definere alle funksjonene vi først hadde tenkt, knappene var allerede implementert i 'elvio' og motor-modulen ble den utførende modulen. Vi endte også opp med å ha to kø-matriser slik som vi hadde som kommentar i det opprinnelige designet. Dette gjorde det enklere å ha kontroll på hva slags type bestillinger som ble gjort og sikret riktig prioritering.

4.2 Bruk av KI

I dette prosjektet har vi ikke benyttet oss av KI. I ettertid ser vi likevel at det kunne ha vært en fordel å gjøre det, da vi endte opp med å bruke 1,5 timer på å lete etter en feil i en av funksjonene våre. Det viste seg at løsningen var å flytte en definisjon ut av funksjonen, noe KI kanskje ville ha oppdaget betydelig raskere enn oss. Likevel føler vi at vi har hatt stort utbytte av å feilsøke på egenhånd, og har lært mye av å prøve og feile. KI kunne også med fordel ha blitt brukt til å effektivisere koden, men dette er ikke noe vi valgte å prioritere.

4.3 Skalarbarhet, vedlikehold og robusthet

UML-diagrammene har bidratt til skalarbarhet ved at de er så generelle, og dermed enkelt burde kunne brukes til et større system. Diagrammene har også gjort det enklere å lage mer oversiktlig og lesbar kode, noe som vil føre til enklere vedlikehold. De er også et godt hjelpemiddel for at andre enkelt skal forstå hva hensikten med koden er.

I forhold til robusthet mener vi at V-modellen har bidratt til dette fordi man først har enhetstesting før integrasjonstesting. På den måten sikret vi oss at modulene fungerte bra både alene og i samhandling med andre.

4.3.1 Effektivitet og kodekvalitet

Vi opplever at koden vår fungerte bra og var stabil, men vi ser at vi kunne ha effektivisert den ved å implementere en tilstandsmaskin. Da kunne vi også gjort main-koden kortere og enklere. Det ville nok også vært nyttig å prøve og lage en tilstandsmaskin da det er relevant for faget og tilpassede datamaskiner generelt.

Vi ser også at å bruke structs ville ført til større robusthet og sikkerhet, ved at man kunne gjort variabler og funksjoner private der hvor det er relevant.

I forhold til å avslutte programmet var vi ikke klar over at man kunne bruke "Ctrl + c". Derfor implementerte vi en funksjon hvor det å trykke på stopp- og '1. etasje'-knappen samtidig stoppet heisen, tømte køen og slukket alle lys.

5 Konklusjon

I dette prosjektet har vi lært å bruke ulike UML-diagrammer for et mer strukturert og effektivt arbeid, og opplevde dette som et nyttig verktøy. Å bruke en mer videreutviklet V-modell fikk vi også en god erfaring med. Å programmere i C har vært lærerikt, og det å se en idé komme til virkelighet med en fungerende heis til slutt har vært veldig motiverende.