Heisprosjekt Gruppe 55

Timo August Brun, Fredrik Warø

21. mars 2024

Sammendrag

Hensikten ved heisprosjektet var å kode handlingsmåten til en heis. Skjelett-koden til heisen var utlevert, hvor denne rapporten først strukturerer, så koder og deretter reflekterer på handlingsmåten til en singel heis. Rapporten vektlegger også bruk av V-modellen for planlegging, Unified Modelling Language (UML) for kode oppbygging, C som kodespråk og GitHub for prosjektplattform. I tillegg er fokuset til heisprosjektet å erfare disse forskjellige verktøyene som et delmål ved siden av implementasjonen. Resultatet var en fullstendig fungerende heis etter spesifikasjoner gitt i oppgaveteksten [3] med oppsett og dokumentasjon.

1 Introduksjon

Denne rapporten har som hensikt å implementere systemet til en heis. Den tar for seg planlegging, utvikling, implementering og testing av systemet. Her vektlegger rapporten i tillegg bruk og erfaring av V-modellen, Unified Modelling Language (UML) og programmeringsspråket `C`. Ettersom dette prosjektet inngår i emnet Tilpassede Datasystemer ved Norges Teknisk Naturvitenskapelig Universitet. Systemet i denne rapporten har som mål å oppnå spesifikasjonene gitt i oppgaveteksten Lab 2: Heisprosjekt [3] og bygger på sjelett-kode utgitt i samsvar med denne. Fokuset er derfor på handlemåten til en heis. Rapporten er strukturert i samråd med den kronologiske prosessen av utviklingen av en heis i henhold til V-modellen. Først struktureres systemet ved forskjellige UML verktøy, deretter moduldesign, implementasjon og refleksjon. Det fullstendige prosjektet er vedlagt i vedlegg [2] og etterfølger Factory Acceptance Testen gitt i oppgaveteksten [3]. Filene som er skapt eller endret for dette prosjektet er `elev_operator.c`, `elev_operator.h` og `main.c`. Dokumentasjon på disse filene finnes vedlagt i vedlegg [2].

1.1 V-Modellen

V-modellen er en utviklingsprosess som fokuserer på å segmentbasert utvikling delt inn i abstraksjonsnivå. Først ved å designe basert på system spesifikasjonene, deretter programvare spesifikasjoner og til slutt programvare design. Hver del implementeres motsatt av den nevnte rekkefølgen med sin tilsvarende test. KILDE PÅ V-MODELL. Hensikten ved å bruke V-modellen er fordelen det gir i modularisering og implementasjon. Det blir lettere å utvikle kode til en spesifikk test. I denne rapporten utvikles heisen i henhold til FAT testen i oppgaveheftet [3].

1.2 Unified Modelling Language

UML blir brukt som et grafisk utviklingsverktøy. Her opprettes forskjellige deler basert på behov. Her nevnes klassediagram, tilstandsdiagram og sekvensdiagram. Klassediagrammet representerer struktur og samhandling mellom forskjellige deler. Abstraksjonsnivået er høyt og ønsker å utvikle forståelse for et produkt sin endelige implementasjon.[1]
Et tilstandsdiagram viser sammenhengen mellom forskjellige tilstander er system befinner seg i. Dette inneholder overganger forårsaket av ulike handlinger eller innfridde betingelser [1]. Tilstandsdiagrammet har et lavere abstraksjonsnivå en klassediagrammet og fokuserer i større del på software arkitektur.
Det siste diagrammet denne rapporten tar hensyn til er et sekvensdiagram. Diagrammet er kronologisk i rekkefølge og visualiserer hvordan forskjellige objekter i et system samhandler, hvordan informasjonflyten beveger seg og eventuelle unntakstilstander. [1] Sekvensdiagrammet er det siste steget brukt før design av moduler. Dermed er det viktig at sekvensdiagrammet visualiserer logikken og handlingene systemet bruker på en oppgave. I henhold til en heis betyr dette fullføring av en ordre.

1.3 Programmeringspråket C

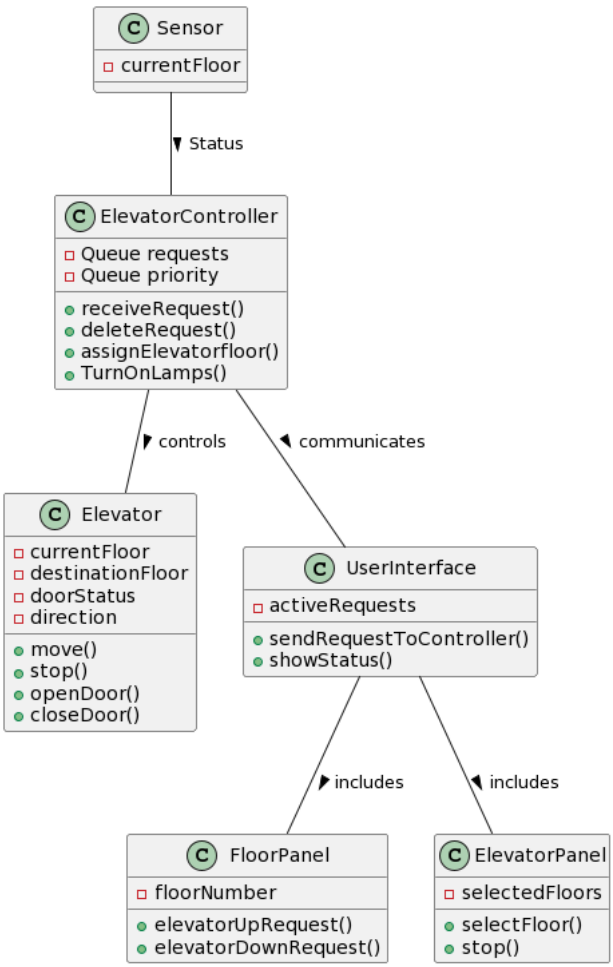
C er et prosedyre orientert programmeringspråk. Språket gir tilgang på lavnivå programmering, er raskt og har lavt minneforbruk.

2 Unified Modelling Language

Som kjent bruker denne rapporten forskjellige UML verktøy for å utvikle og planlegge heissystemet. Her er det tre forskjellige UML verktøy strukturert i

synkende abstraksjonsnivå med vedlagte figurer fra de forskjellige utviklingsprosessene.

2.1 Klassediagram



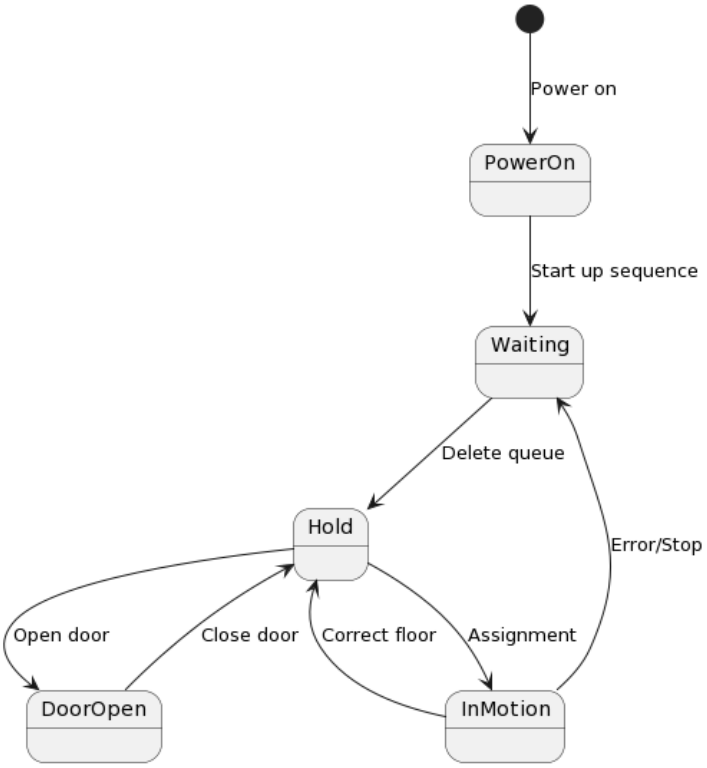
Figur 1: Klassediagram

Under arkitekturfasen i oppgaven ble det laget ett klassediagram 1. Diagrammet ble brukt til å se sammenhengen mellom klassene i programmet, og forsikrer at den resulterende heismodellen er logisk riktig i høyeste abstraksjonsnivå, altså at det er en modell som har mulighet til å oppfylle de gitte kravene. Dersom heisen ikke er logisk riktig på dette nivået vil de tilhørende modulene som senere skal implementeres ikke fungere sammen. Heismodellen skal som nevnt oppfylle de gitte krevene. På dette nivået resulterte dette

eksempelvis i at det ble implementert egne funksjoner for å åpne døren og at heisen skulle bevege seg, da dette forsikret av de to hendelsene aldri ville skje samtidig.

2.2 Tilstandsdiagram

Etter Klassediagrammet ble det laget ett Tilstandsdiagram 2. Dette diagrammet videre bekriver funksjonaliteten til heismodellen. Diagrammet viser tydelig ett skille på tilstandene hvor døren er åpen og når heisen er i bevegelse, i tillegg til at både stopp-funksjonen og oppstartsekvensen er separate fra den vanlige handlingsloopen. Dette er viktig logikk som må være på plass før implementasjonsfasen, slik at man forsikrer at den resulterende heismodellen har ønsket funksjonalitet.

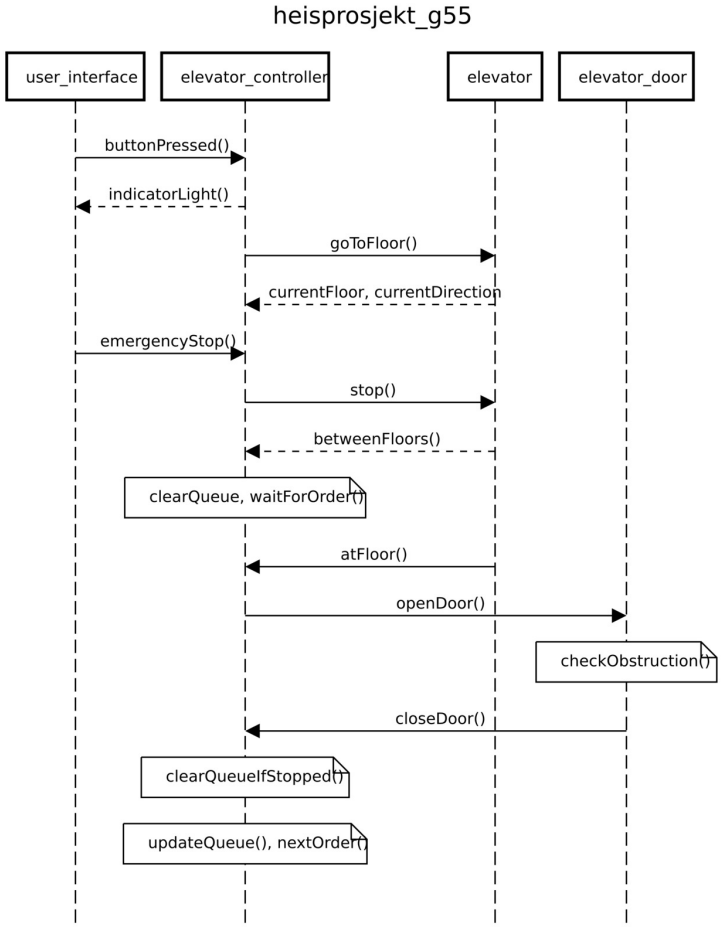


Figur 2: Tilstandsdiagram

2.3 Sekvensdiagram

Det siste diagrammet brukt for systemutviklingen i denne rapporten er sekvensdiagrammet sett i Figur 3. Dette diagrammet viser hvordan en bruker-

input overføres til kontrollmekanismen. I dette tilfellet betyr det programvaren og køsystemet. Dette vil videre gi et signal til heisen. Kontrolleren må til enhver tid vite retning og siste etasje heisen var på, av den grunn at om heisen stopper i en udefinert tilstand må systemet vite hvor det er i forhold til en ny ordre. To andre hovedmomenter er nødstopp og heisdøren. Ved nødstopp handler heisen forskjellig om den er i en etasje eller ikke. I tillegg skal åpning av døren kun skje ved strenge vilkår, altså på en etasje. Her er det viktig at systemet ivaretar slik informasjon.



Figur 3: Sekvensdiagram

3 Moduldesign

Etter arkitekturfasen skal de forskjellige modulene som er nødvendig for å oppfylle den nå etablerte arkitekturen lages. Disse modulene skal altså støtte den virkemåten som ble etablert i tilstands- og sekvens diakrammet og samtidig følge klassesammenhengen identifisert i klassediagrammet. Modulene bør opprettes slik at hver av de har en individuell spesifikk funksjon. Funksjonen bør kunne testes speparat fra resten av modellen, altså bør modulene i liten grad være avhengig av hverandre for å fungere.
Først ble det deffinert en modul for heisbevegelse, denne skulle altså ta inn en ønsket etasje og bruke dette og en etasjesensor for å bevege heisen fra nåværende til ønsket etasje. Modulen kan testes ved å gi den en tilfeldig etasje og observere om heisen går til gitt etasje.
Etter heisen ankommer riktig etasje skal døren åpnes og lukkes, det defineres dermed en modul for dette. Modulen er avhengig av at heisen er i en definert etasje, og kan testes ved og manuelt flytte heisen til en deffinert etasje, observere handlemåte og deretter gjøre det samme når heisen er i en udefinert etasje. Modulen skal også ta hensyn til eventuell obstruksjon ved åpen dør.
Når dette er etablert behøver heisen en heiskontroller. Denne har i oppgave og ta inn ett køsystem og gi ut neste ønskede etasje. Testing av denne gjøres ved å opprette en kunstig kø med ett gitt innhold la kontrolleren ta den inn og observere om det gis ut riktig etasje.
Videre behøver heisen ett køsystem. Det er nødvendig at heisen skal betjene alle bestillinger fra bruker-panelene og disse bestillingene skal kunne opprettes når som helst. Denne modulen skal dermed fungere i alle faser av heisens normale driftsykel. Køsystemet skal også fjerne bestillinger når de er ekspedert. Testing av denne modulen vil være avhengig av heiskontrolleren og heisbevegelsen, og kan gjennomføres ved å opprette to eller flere bestillinger og se om heisen gjennomfører gitte bestillinger i ønsket rekkefølge.
Deretter trenger heisen en stoppfunksjon. Denne modulen har i oppgave å stoppe heisens eventuelle moment og deretter gi heisen en midlertidig ny tilstand. Denne tilstanden skal være utenfor heisen normale driftsykel, mens stoppfunksjonen skal kunne kalles i alle tilstander innenfor heisens normale driftsykel.
Til slutt må det integreres ett lampesystem som skrur av og på alle de riktige lampene underveis. Dette systemet vil sannsynlig bli delvis integrert i alle modulene og det vil ikke være nødvendig å ha en egen modul for lyssetting. Etter lampesystemet er integrert kan systemet testes i alle faser, og nå bør det observeres om lampene oppfører seg som forventet.

4 Implementasjon

Vedlagt finnes tilhørende C-kode og dokumentasjon.

5 Refleksjon

Gjennom prosjektet har det vært flere fordeler, men også ulemper av å bruke UML og V-modellen. I begynnelsen av arkitekturfasen var det vanskelig å se hvordan prosjektet skulle bli når det var ferdigstilt, og dermed vanskelig og lage gode modeller som kunne implementeres i C. Det å bruke ULM og V-modellen krever høy forståelse av både oppgave og krav før man har begynt med selve implementasjon, men det er ofte i selve implementasjonsfasen, gjennom å prøve og feile, at man faktisk oppnår denne høye forståelsen. Dette gjorde at det tok lang tid før man var fornøyd med arkitekturen og kunne begynne å designe moduler.

Når derimot arkitekturen var ferdigstilt og modulene begynte å ta form var det henholdvis enkelt å implementere modellen i C. Det at både hensikt og grensesnitt er kjent fra modul-designfasen gjorde implementeringen av kode både intuitiv, oversiktlig og effektiv. Det å sette modulene sammen i etterkant viste seg derimot ikke å være like intuitivt. Hovedproblemet var at stoppfunksjonen og «legg til bestilling» funksjonen måtte kjøre hele tiden. Da koden var implementert prosedyreorient kunne ikke to deler av koden kjøre samtidig.

Løsningen på dette var å kjøre disse funksjonene flere steder i systemet, og isteden for at heisen for eksempel var på vent ble det brukt en «while-loop» som alltid sjekket om det kom bestillinger eller om stopp-knappen ble trykket på. Dette problemet ble ikke forutsett i arkitekturfasen, men da koden i første omgang ble implementert oversiktlig var problemet fremdeles relativt enkelt å løse. Dette skyldes ett godt forarbeid ved bruken av ULM og V-modellen.

Gjennom prosjektet har det blitt brukt kunstig intelligens (KI) i hovedsak til å diskutere løsninger på problemer og feilkoder i implementasjonsfasen. Det var også KI som hjalp med syntax i programmet «PlantText ULM» som opprettet både klassediagrammet og tilstandsdiagrammet i rapporten. Selvom KI ikke lagde diagrammet brukt i rapporten, så ble det spart mye tid ved å umiddelbart ha tilgang til eksempler på riktig syntax i ett nytt program isteden for å behøve og søke det opp på internett. Bruker man KI

med slike hensiker er det ett nyttig verktøy for effektivisering av arbeidet.

6 Konklusjon

ULM og V-modellen ga heismodellen ett robust fundament som var lett å bygge videre på når problemer oppstår. Ulempen med fremgangsmåten er at den behøver høy forståelse av oppgaven som skal løses, noe som kan gjøre den unødvendig tidkrevende i mindre prosjekter hvor prøving og feiling mulig kan være en mer effektiv løsning. Bruken av KI i oppgaver som denne effektiviserer arbeidet, men bør ikke brukes som ett direkte verktøy for implementasjon, men heller som en «smart» nettleser.

Referanser

[1] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Professional, 2004.

[2] *heisprosjekt*$_g$55. Attachement 1. 2024. URL: `https://github.com/fredrswa/embeddedsystems_g55`.

[3] Natlandsmyr T. Jacobsson T. H. «Lab 2: Heisprosjektet». I: (2024).