Strukturering
Softdrink-Automat

Fag:	Projekt – E1PRJ1
Emne:	Strukturering – Softdrink-Automat
Gruppe:	6
Dato:	20. marts 2006
Medlemmer:	Benjamin Sørensen, Jacob Nielsen, Klaus Eriksen, Mikkel Larsen og Tomas Stæhr Hansen

Indholdsfortegnelse

1. Indledning

1.1 Formål

Formålet med struktureringen er at opdele opgaven i realiserbare delopgaver. Vi har opdelt struktureringen i to hovedemner, software og hardware. Inden vi gik i gang med selve designet af de to emner, gennemgik vi grænsefladen imellem dem.
Grænsefladen mellem hardware og software vil blive gennemgået i kapitel 2. Kapitel 3 og 4 går i dybden med struktureringen af henholdsvis hardware og software.

1.2 Referencer

1.3 Revisionshistorie

1. maj 2003 revisionsnr.: 1.0

1.4 Bilagsliste

Bilag 1: Interface connector
Bilag 2: Digital connector
Bilag 3: Analog Connector
Bilag 4: Funktionsbeskrivelser.
Bilag 5: Sekvensdiagrammer

2. Generel beskrivelse

2.1 Systembeskrivelse

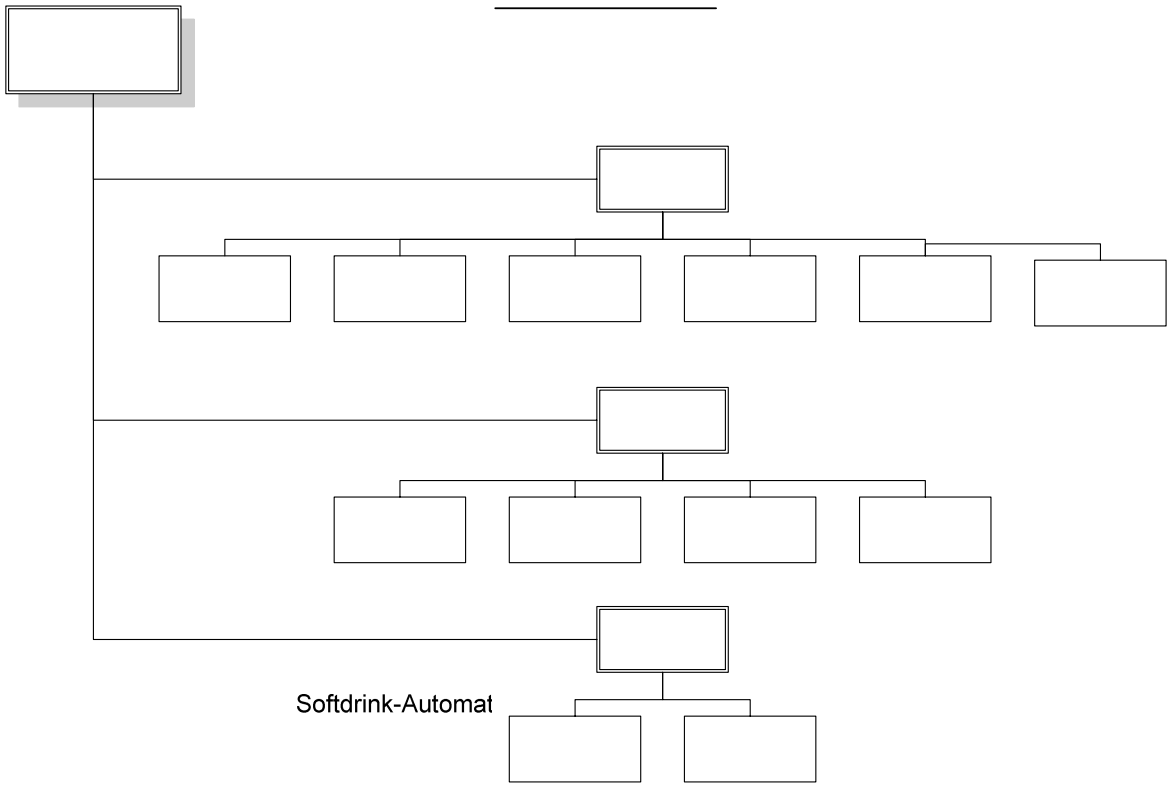


Fig 2.1. Funktionsdiagram
Diagrammet viser softdrinkautomatens funktionalitet delt op i mindre overskuelige afsnit.

2.2 System grænseflader

Grænsefladerne er mellem software og hardware er opstillet i bilag 1. Her ses det, at interfacekortet (Humusoft AD 512 I/O) modtager analog signaler for VAND_STATUS, SAFT_STATUS og TEMP_STATUS, digitale signaler for KNAP_START_U/P, KNAP_RESET_U/P, POS_START. POS_DOS og KOP_I_HOLDER. Outputsignaler fra interfacekortet er MOTOR_START, MOTOR_RETNING, LYSDIODE_SAFTVENTIL, LYSDIODE_VANDVENTIL og LYSDIODE_VÆSKE_VARM.

3. Hardware

Denne beskrivelse af hardware er baseret på, at alle mekaniske processer til softdrinkautomaten er udarbejdet på forhånd. Med disse rammer givet, påbegynder vi arbejdet step for step.
Efter flere udkast til diverse diagrammer, lykkedes det at skabe fornuftige diagrammer over systemet, der har gjort det muligt at overskue selv de mindste mekaniske funktioner.
Det egentlige udviklingsarbejde vil være implementering af hardwaren på veroboardet. Dette er beskrevet i afsnit 3.2 Veroboard, beskrivelsen er dog begrænset til funktionalitet og grænseflader. Den konkrete implementering er blevet diskuteret, men ikke beskrevet i denne fase af projektet. Den mekaniske opstilling, der er beskrevet i afsnit 3.1, er givet på forhånd, og kræver derfor ikke yderligere bearbejdelse i det videre forløb af projektet.

3.1 Mekanisk opstilling

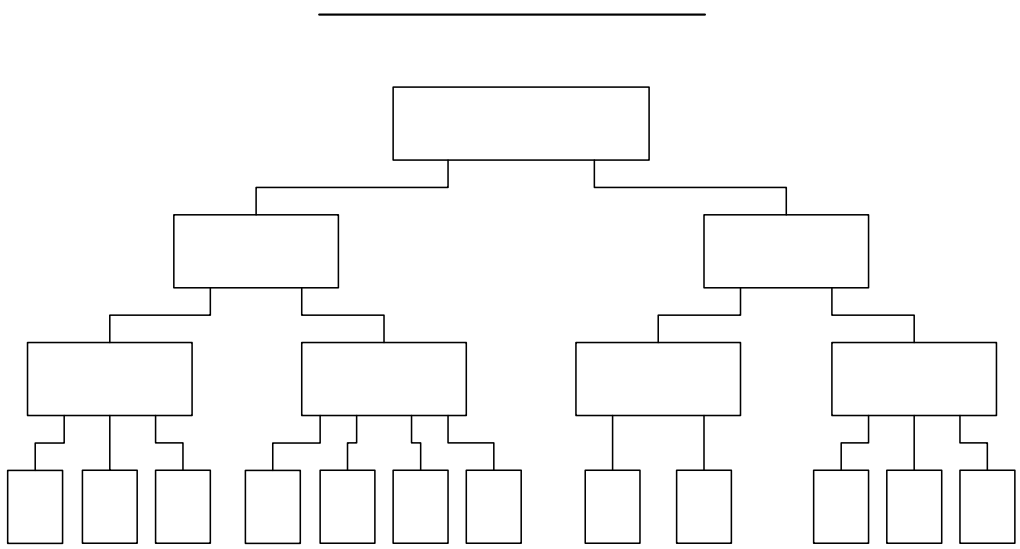


Fig. 3.1. Topdown diagram.
Diagrammet viser en overordnet opdeling af softdrinkautomaten. Det nederste niveau i mekanikdelen af diagrammet er slutblokke. Disse blokke er beskrevet i dette afsnit.

Følere: Fig. 3.1 I1

Temperaturføler:

Funktionalitet:

Temperaturføleren er en komponent med en elektrisk egenskab, der varierer strømmen med temperaturen i vandbeholderen. Variationen skal kunne omsættes entydigt til temperatur (grader celsius) indenfor det i kravspecifikationen angivne temperaturinterval.

Grænseflader:

Modulet har grænseflade til analog connector (V2 fig. 3.2), pins 1 (katode) , 2 (anode)

Væskestandsfølere:

Funktionalitet:

På den mekaniske opstilling er der monteret væskestandsfølere på beholderne til hhv. vand og saft. Væskestandsføleren er et potmeter, der varierer med væskeniveauet i vandbeholderen. Variationen er en ohmsk impedansvariation idet der er tale om 10kΩ lineære potentiometre påmonteret arm og flyder. Impedansvariationen omsættes til en spændingsvariation med en Ohm/Volt konverter. Spændingsvariationen sendes til pc'ens processkontrol for styring af magnetventilernes åbnetider.

Grænseflader:

Modulet har grænseflade til Ohm/Volt konverteren (V4 fig. 3.2). Der etableres separat kabelføring.

Kopholderføler:

Funktionalitet:

Føleren i kopholderen skal kunne angive hvorvidt der er placeret en kop i kopholderen.

Grænseflader:

Modulet har grænseflade til digital connector(V3 fig. 3.2), pin 12.

Positionsfølere

Funktionalitet:

Positionsfølerne angiver, om kopholderen er i position hhv. startposition og doseringposition. Ved start- og doseringsposition sendes signalet videre til lysdioderne via digital connector.

Grænseflader:

Modulet har grænseflade til digital connector (V3 fig. 3.2), pin 9 (start), 10 (dosering).

Start: Fig. 3.1 I2

Funktionalitet:

Ved tryk på startknappen afbrydes den elektrisk kontakt (n.c.), og processen startes.

Grænseflader:

Modulet har grænseflade til digital connectoren, pin 1(V3 fig. 3.2).

Reset: Fig. 3.1 I3

Funktionalitet:

Ved tryk på resetknappen skabes elektrisk kontakt (n.c.), og processen resettes.

Grænseflader:

Modulet har grænseflade til digital connectoren, pin 8 (V3 fig. 3.2).

Temperatur Display: Fig. 3.1 O1

Funktionalitet:

Brugeren skal kunne aflæse vandtemperaturen i grader celsius. Driverkredsløbet til displayet omsætter input spændinger mellem -2 og 2 volt lineært til en visning på displayet -200 og 200.

Grænseflade:

Analog connector (V2 fig. 3.2) pin 6 +/- 2 volt

Statusindikatorer/Lysdioder: Fig. 3.1 O2

Pos start

Funktionalitet:

Lyser når koppen er i startposition.

Grænseflade:

Lysdiodens anode på digital connector, Pin 16 (V3 fig. 3.2), katode til stel. Kræver 2 volt ca. 10 mA.

Frem

Funktionalitet:

Lyser når kopholderen kører mod doseringsposition.

Grænseflade:

Lysdiodens anode på digital connector, Pin 17 (V3 fig. 3.2), katode til stel. Kræver 2 volt ca. 10 mA.

Pos. Dos

Funktionalitet:

Lyser når koppen er i doseringsposition.

Grænseflade:

Lysdiodens anode på digital connector, Pin 18 på (V3 fig. 3.2), katode til stel. Kræver 2 volt ca. 10 mA.

Saft

Funktionalitet:

Lyser når der doseres saft.

Grænseflade:

Lysdiodens anode på digital connector, Pin 19 (V3 fig. 3.2), katode til stel. Kræver 2 volt ca. 10 mA.

Vand

Funktionalitet:

Lyser når der doseres vand.

Grænseflade:

Lysdiodens anode på digital connector, Pin 20 (V3 fig. 3.2), katode til stel. Kræver 2 volt ca. 10 mA.

Tilbage

Funktionalitet:

Lyser når koppen kører mod startposition.

Grænseflade:

Lysdiodens anode på digital connector, Pin 21 (V3 fig. 3.2), katode til stel. Kræver 2 volt ca. 10 mA.

Væske varm

Funktionalitet:

Lyser når vandet er over 16,0 grader.

Grænseflade:

Lysdiodens anode på digital connector, Pin 22 (V3 fig. 3.2), katode til stel. Kræver 2 volt ca. 10 mA.

Stepmotor: Fig. 3.1 O3

Funktionalitet:

Kører kopholderen frem og tilbage afhængig af indgangssignalernes værdi.

Grænseflade:

Pin 4 og 5 til henholdsvis styresignalerne a og b. CMOS kompatibelt signal.

Ventiler: Fig. 3.1 O4

Funktionalitet:

Skal kunne åbne og lukke for henholdsvis vand- og saftbeholderen. Lukker ved lav styrespænding.

Grænseflade:

Pin 19 for saft og pin 20 for vand på digital connector (V3 fig. 3.2).

3.2 Beskrivelse af Veroboard

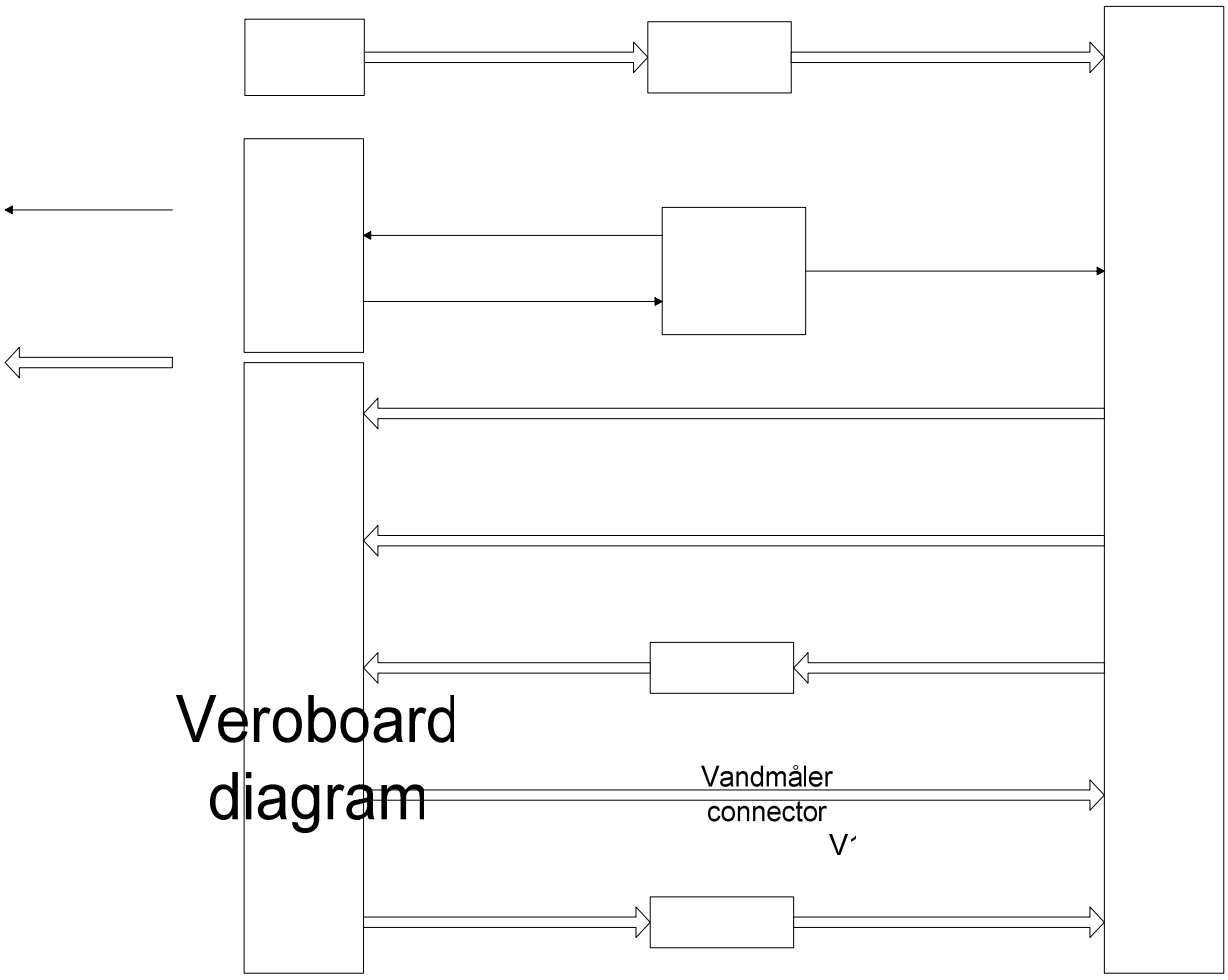


Fig. 3.2. Veroboard diagram

Diagram over veroboardet til styring af automaten.

Bilag 2 viser et overordnet diagram for veroboardet, hvor alle connections og deres respektive pin numre er vist.

Enkeltstrenget signal

Analog
Connector

V2

Flerstrenget signal

Væskestandsfølerconnector: Fig. 3.2 V1

Funktionalitet:

Forbinder væskefølerne på væskebeholderne med veroboardet.

Grænseflade:

DB9 stik.

Analog connector: Fig. 3.2 V2

Funktionalitet:

Forbinder det analoge stik fra automatens frontpanel til veroboardet.

Grænseflade:

26 polet SUB-D, fladkabel.

Digitale connector: Fig. 3.2 V3

Funktionalitet:

Forbinder det digitale stik fra automatens frontpanel til veroboardet.

Grænseflade:

26 polet SUB-D, fladkabel.

Væskemålere Ω/V Convertorer: Fig. 3.2 V4

Funktionalitet:

Konverterer modstandsværdien fra væskestandsfølerne til en analog spænding.

Grænseflade:

Input:	Væskestandsføler connector.
Output:	2 Analoge spændinger (0-5V) en for hver væskermåler: Vand: Veroboard interface connector, pin 1 Saft: Veroboard interface connector, pin 3

Temperaturmåler Ω/V Convertor: Fig. 3.2 V5

Funktionalitet:

Konverterer modstandsværdien fra Temperaturmåleren til en analog spænding .

Grænseflade:

Input:	Føleranode på Pin 2 på digital connector (V3 fig. 3.2) og følerkatode på pin 1.
Output	to Analoge spænding: 0 - 0,36 V til display. 0 - 0,36 V til pin 6 på analog connector og 0 – 3,6 V til veroboard interface connector pin 5 (V2 fig. 3.2).

Motorstyring: Fig. 3.2 V6

Funktionalitet:

Leverer styringssignaler til stepmotorerne ved hjælp af signaler fra interfacekortet (Humusoft AD 512 I/O).

Grænseflade:

Input:	Veroboard interface connector pin 22.
Output:	Motor A, Digital connector pin 4 og Motor B, Digital connector pin 5 (CMOS).

Knap stabilisator: Fig. 3.2 V7

Funktionalitet:

Fjerner prel fra automatens kontakter.

Grænseflade:

Input:	Digital connector pin 1 (start) hhv. Digital connector pin 8 (reset)
Output:	2: Start og Stop, uden prel (TTL).

Interface connector: Fig. 3.2 V8

Funktionalitet:

Forbinder computerens interfacekortet (Humusoft AD 512 I/O) med Veroboardet.

Grænseflade :

37 polet SUB-D, fladkabel.

4. Software

4.1 Indledning

Dette afsnit beskriver struktureringen af den del af styringen, der foregår på pc. Forud for struktureringen af softwaredelen ligger en fast grænseflade mellem software og hardware.
Kapitlet vil indeholde beskrivelser af de to modeller vi har arbejdet med, hvilken model vi har valgt og hvorfor. Vi vil derefter gå i dybden med de enkelte klasser og sammenspillet mellem klasserne.

4.2 Indledende overvejelser

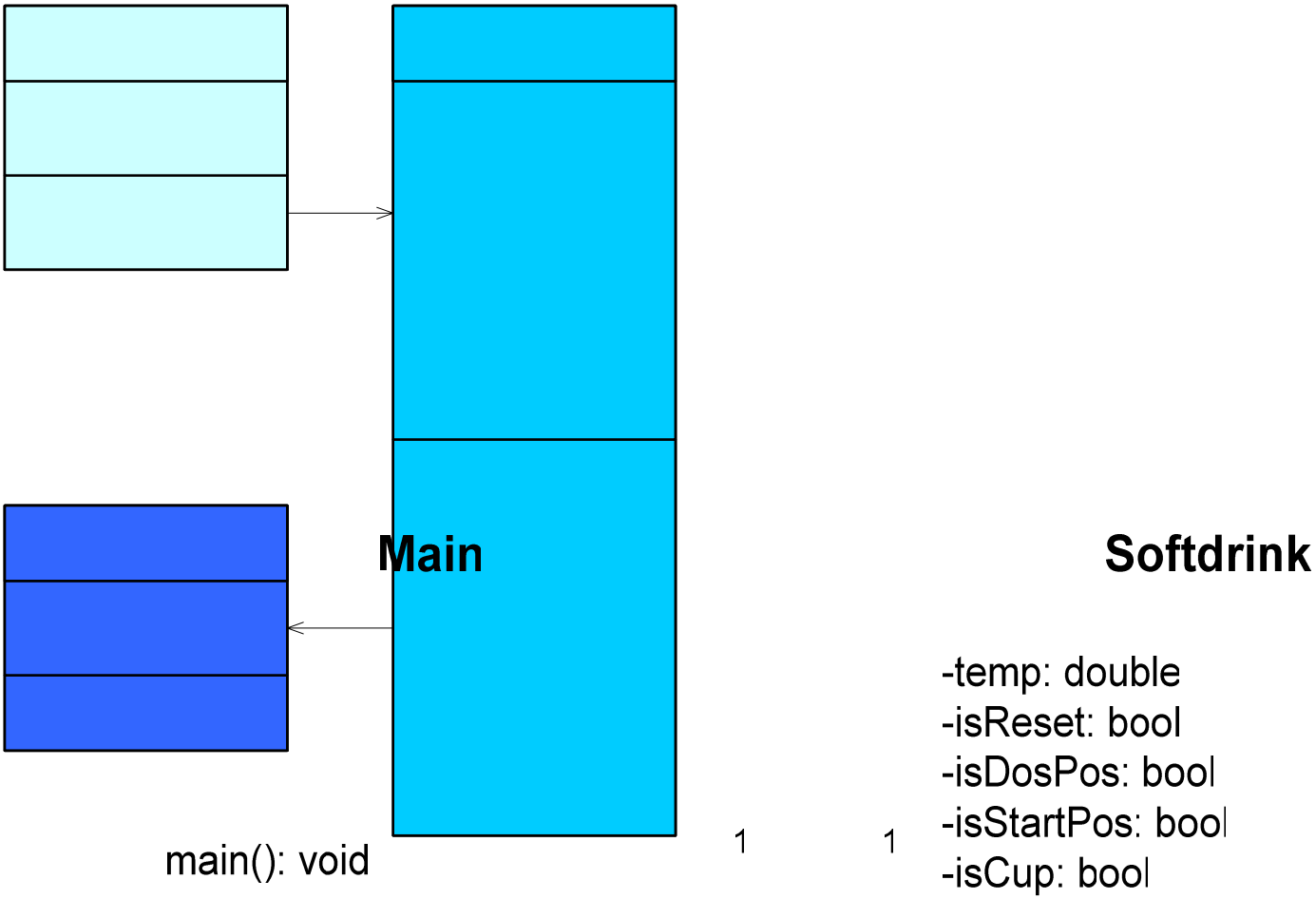
Man kan strukturere på mange forskellige måde. Der ligger mange design patterns til rådighed, som man kan strukturere efter. Den mest anvendte er Model-View-Controller (MVC) modellen. Det er da også denne vi har valgt at bygge vores implementation op omkring.
Før vi fastlagde os på MVC modellen, overvejede vi en simplere model. Vi vil kort beskrive ideen.

4.2.1 Simpel model

Man kunne strukturere koden omkring en enkelt klasse, softDrink og supplere med klasser som diode, motor og ventil. softDrink klassen skulle modsvare selve automaten, og de supplerende klasser skulle repræsentere de dele softdrinkautomaten er opbygget af. Al kommunikation til hardwaren skulle så foregå i de supplerende klasser, og softDrink klassen skulle stå for selve styringen. De umiddelbare fordele ved den struktur er, at man undgår en række unødvendige kald objekter imellem, som tager tid, og man opretholder en vis form for adskillelse af kode. Af ulemper kan nævnes, at koden bliver rodet og svær at opdatere, da mange klasser kalder til hardware interface kortet.

På fig. 4.1 er modellen beskrevet i et UML klassediagram. Vi har kun diode klassen med som supplerende klasse.

Fig. 4.1 Simpel model.



Vi droppede dog hurtigt den simple model og gik i gang med MVC modellen. Inden vi beskriver vores model, beskriver vi den grundlæggende ide i MVC og fordele/ulemper.

4.3 Beskrivelse af MVC struktur

Da dette ikke er en bog om design patterns, så vil dette afsnit gå meget overfladisk igennem MVC. Ideen med MVC er, at adskille koden i 3 klasser, som har hver deres funktion.

Modellen

Modellen står for selve databehandlingen. Modellen leverer data til view'et uden at bekymre sig om, hvordan data skal repræsenteres. Data fra modellen er display-neutralt, så det kan interfaces til mange views uden at der opstår redundant kode. Dette gør det nemmere at vedligeholde koden og nedbringer

antallet af fejl. Modellen svarer på forespørgsler fra controlleren, behandler data og giver besked til aktive views, om at opdatere.

Controller:
Controller er ansvarlig for styringen, og interaktion fra brugeren i form af mus og keyboards. Interaktion fra brugeren trigger de events som skal ændre modellen, og controlleren fortæller registrerede view at de skal opdatere deres display.

View:
Viewdelen står for den grafiske data repræsentation. View'et er isoleret fra komplekse data operationer, eg. database adgang eller tilgang til hardware.
View'et modtager simpelthen de rå data fra modellen og foretager den grafiske formatering og repræsentation af disse data. Systemets interface til interaktion med brugeren ligger som regel også i view'et.

Fordele:
Der er mange fordele ved MVC strukturen. Nogle af punkterne er

- Høj modularitet. Det er let at udskifte f.eks. et tekstuelt view med et grafisk. Vi skal kun lave et nyt view.
- Overskuelighed. Koden bliver overskuelig og mere forståelig.
- Vedligeholdelse. Det er nemt at vedligeholde koden. Lokale ændringer i f.eks. controlleren har ingen indflydelse på andre moduler.
- Udvikling. Det er muligt at kode modulerne parallelt, når interfaces mellem modulerne ligger fast.

Ulemper:
Der er også nogle ulemper ved strukturen.

- Det stiller krav til grundig analyse af brugerens behov før designet går i gang.
- Det er tidskrævende at designe og analysere.
- Det er overkill for små applikationer.

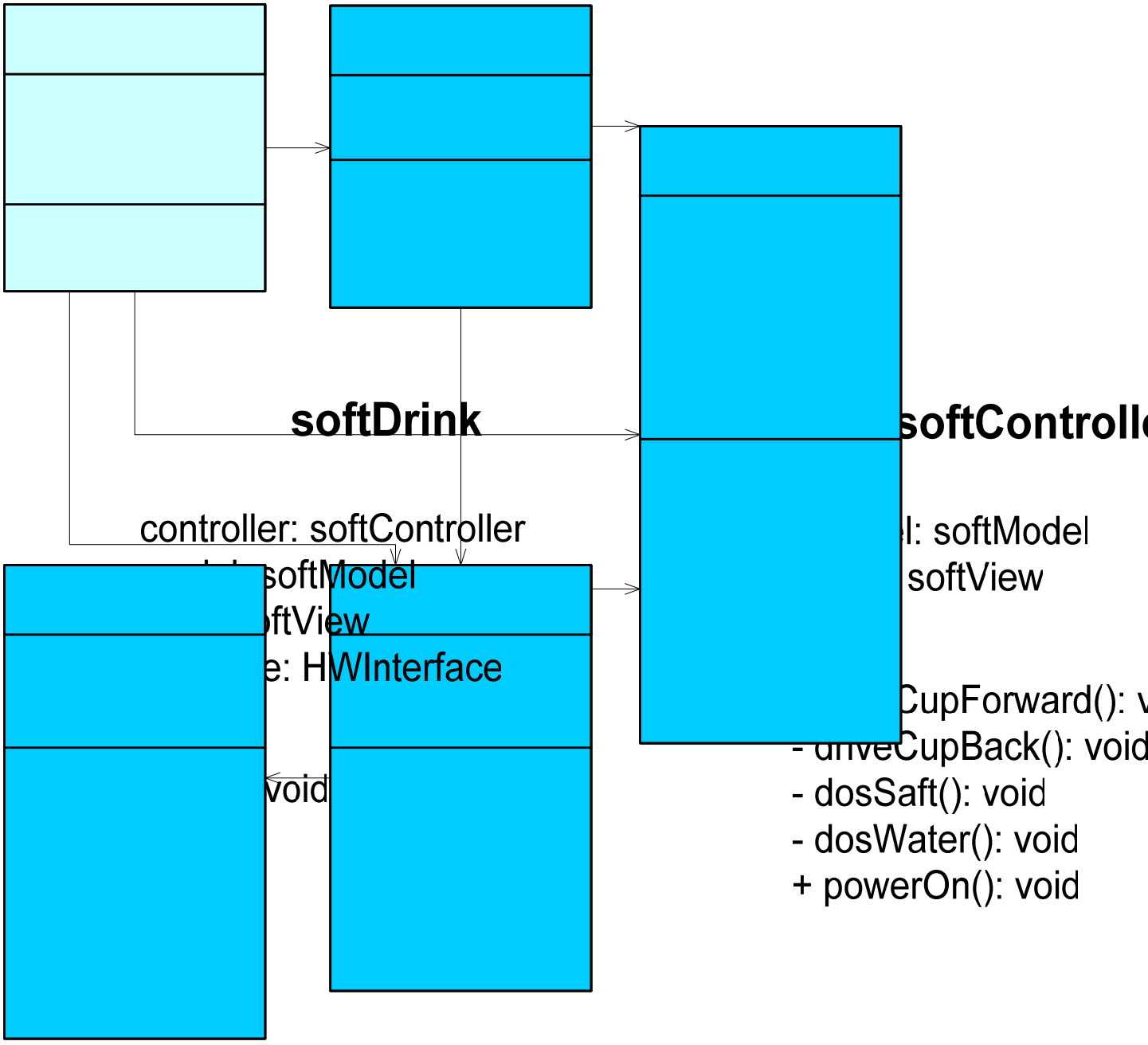
4.4 Fastlæggelse af MVC struktur

Vi valgte som sagt den store MVC model. Vi kunne godt have kørt videre med den simple model, men fordelene, specielt modulariteten, ved MVC var mange. Med MVC får vi delt koden op, så modulerne kan kodes parallelt. Desuden får vi mulighed for at nemt at udvide softwaren til at styre flere automater. En anden klar fordel er, at vi nemt kan teste de enkelte klasser, interaktion mellem klasserne, og specielt interaktion med driveren.
MVC strukturen medfører at afviklingen af softwaren skaber en masse funktionskald modulerne imellem. Der er klart et trade-off mellem hastighed og modularitet af koden. Alligevel mener vi, at modularitet og overskuelighed er at foretrække. Specielt da der ikke er strenge krav til hastigheden af maskinen, eller en nødstop funktion på maskinen.

4.5 Beskrivelse af strukturen.

Vores software kan beskrive ud fra fig. 4.2.

Fig. 4.2 MVC klassediagram



Det er værd at bemærke HWInterface klassen, som er den eneste klasse, der snakker med driveren. Vi har ønsket at lægge al interaktion over i denne klasse, da vi ikke på nuværende tidspunkt er sikre på hvordan vi tilgår driveren.

Selve styringen af automaten ligger i softController. Den sørger for kald til modellen og update af view'et.
softView står for opbyggelse og update af displayet.
softModel tilgår HWInterface og kald om update til softView.
De enkelte klasser og interaktionen mellem dem er uddybende beskrevet efterfølgende.

4.6 Beskrivelse af de enkelte klasser

softController:
Klassen har til opgave at styre modellen (klassen softModel) og udskriften til skærmen (klassen softView). softController holder styr på hvornår der skal kaldes ned i de andre klasser vha. en overordnet powerOn funktion og en række private operationer, som kalder videre til modellen og view'et.

softView:
Denne klasses eneste formål er, at holde skærmen opdateret med hvad programmet og selve softdrink-automaten foretager sig. Den repræsenterer en grafisk fremstilling af frontdisplayet på automaten, dvs. hvad der sker på frontdisplayet kan også ses på computerskærmen.

softModel:
Klassens operationer bliver kaldt fra softController-klassen. softModel's forskellige operationer kalder ned i interfacet (klassen HWInterface). Modellen instansieres ved at give et view og en hardware interface med som argument. Dermed kan man selv vælge hvilket view og interface man ønsker.

HWInterface:
Denne klasse er ansvarlig for al kommunikation med hardware I/O-kortet. HWInterface modtager operationskald fra softModel-klassen, hvorefter den udfører den ønskede handling ved at kalde I/O-kortet.

4.7 Gennemgang af klassernes funktioner

Vi gennemgår de vigtige funktioner fra de forskellige klasser. Alle funktionerne er medtaget i bilag 4.

softController

```
void powerOn();
Parametre:
        Ingen
Returværdi:
        Ingen
Beskrivelse:
```

Funktionen styrer automaten gennem kald til de øvrige operationer i modellen og styringesrutiner i klassen selv. View'et styres også herfra.

void driveCupForward();
Parametre:
Ingen
Returværdi:
Ingen
Beskrivelse:
Funktionen styrer koptransporten fra startposition til doseringsposition.

void dosWater();
Parametre:
Ingen
Returværdi:
Ingen
Beskrivelse:
Funktionen styrer doseringen af vand ved at kalde softModel::dosWater().

softModel

bool isStartPressed();
Parametre:
Ingen
Returværdi:
True/false
Beskrivelse:
Funktionen kalder HWInterface::checkStart() og returnerer true/false

bool isTempOk();
Parametre:
Ingen
Returværdi:
True/false
Beskrivelse:
Funktionen kalder HWInterface::getTemp() og returnerer på baggrund af den returnerede værdi true/false

bool isReset();
Parametre:
Ingen

```
Returværdi:
        True/false
Beskrivelse:
        Funktionen kalder HWInterface::checkReset() og returnerer
        true/false
```

void dosWater();

```
Parametre:
        Ingen
Returværdi:
        Ingen
Beskrivelse:
        Funktionen kalder HWInterface::dosWater()
```

void driveForward();

```
Parametre:
        Ingen
Returværdi:
        Ingen
Beskrivelse:
        Funktionen kalder HWInterface::driveForward()
```

void setHotWater(bool);

```
Parametre:
        bool
Returværdi:
        Ingen
Beskrivelse:
        Funktionen kalder HWInterface::setHotWater(bool)
```

softView

void printView();

```
Parametre:
        Ingen
Returværdi:
        Ingen
Beskrivelse:
        Funktionen skal udskrive grunddisplayet til skærmen.
```

void setStart(bool);

```
Parametre:
        True/false
Returværdi:
```

```
        Ingen
Beskrivelse:
        Funktionen skal ændre værdien af Start på skærmen.

void setTemp( int );
Parametre:
        Temperatur-værdi
Returværdi:
        Ingen
Beskrivelse:
        Funktionen skal ændre værdien af temperaturen på skærmen.
```

HWInterface

```
void driveForward();
Parametre:
        Ingen
Returværdi:
        Ingen
Beskrivelse:
        Funktionen skal sørger for at starte motoren i fremad
        retning.

void setHotWater( bool );
Parametre:
        bool
Returværdi:
        Ingen
Beskrivelse:
        Funktionen skal sørger for at tænde dioden på displayet

bool checkStart();
Parametre:
        Ingen
Returværdi:
        True/false
Beskrivelse:
        Funktionen skal checke om der er trykket på start
```

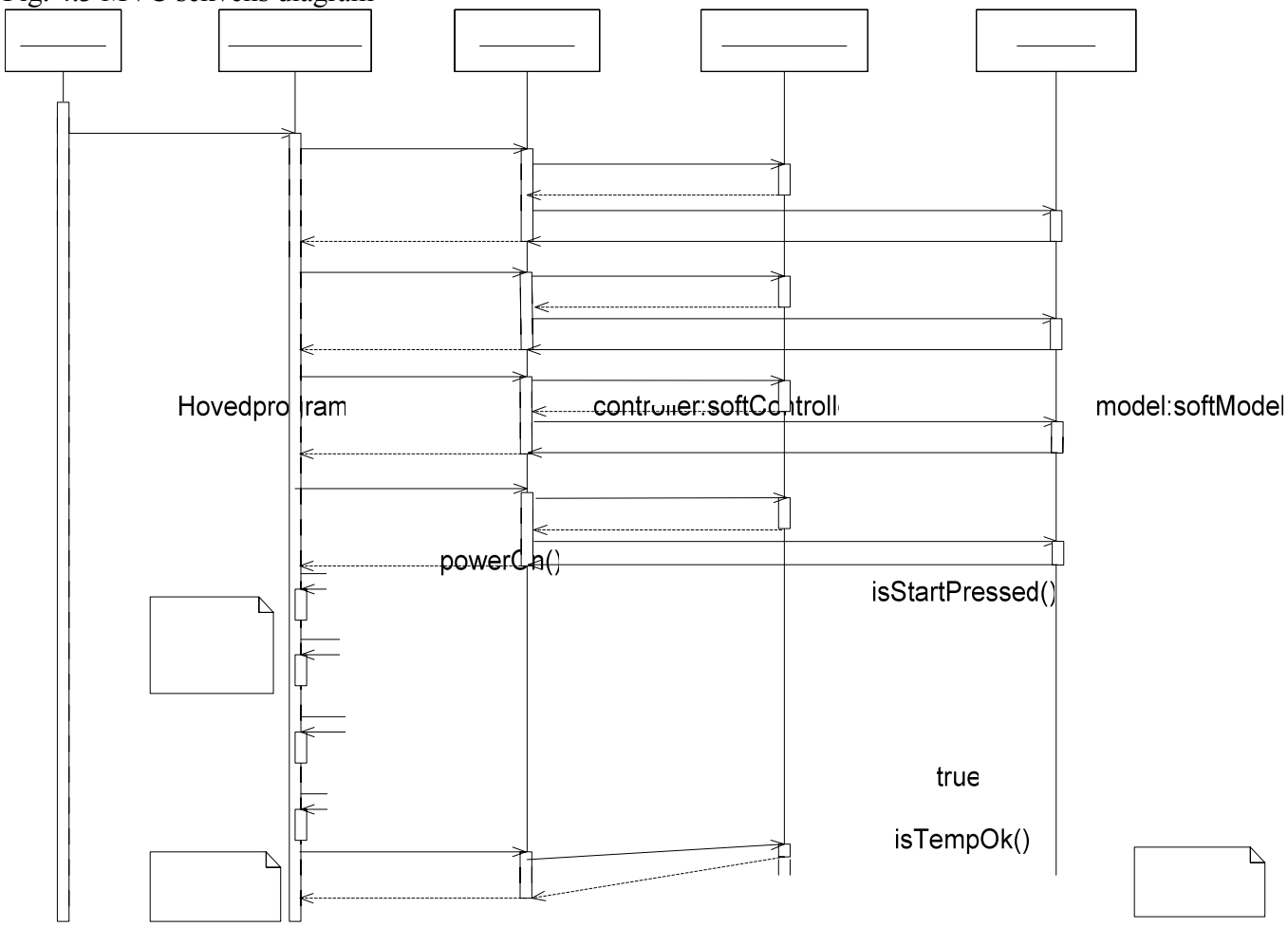
4.8 Beskrivelse af sammenspil mellem klasser

Som det fremgår af klassediagrammet, fig 4.2, er der forskellige forbindelser mellem klasserne.

Vi har en main-blok, der opretter et objekt af hver af klasserne: softController, softModel, HWInterface og softView. Det første der bliver oprettet, er en instans af er softView. Derefter opretter vi en instans af HWInterface'et. Herefter bliver softModel oprettet med pointere til softView og HWInterface objekter som argumenter. Til sidst bliver objektet softController oprettet med pointere til softModel og softView som argument. Når objekterne er oprettet kaldes powerOn() i softController.
SoftControllerens constructor sørger for at updatere viewet, så det modsvare automatens opstartstilstand.
SoftController klarer styringen ved at arbejde med modellen.
Alt interaktion mellem software og hardware sker gennem HWInterfacet. Vi har isoleret alt den hardware specifikke kode i denne klasse, for at gøre koden overskuelig. Ved at give modellen et interface og et view med som argument, får vi mulighed for at udskifte HWinterfacet med et testInterface og view'et med et grafisk GUI.

Sammenspillet mellem klasserne er beskrevet i dette UML-sekvensdiagram.

Fig. 4.3 MVC sekvens diagram



Funktionerne driveCupFoarward, driveCupBack, dosWater og dosSaft er beskrevet i bilag 5.

4.9 Test

Under implementationen vil vi lave en blackbox test af de enkelte funktioner. Disse test vil blive beskrevet i implementationsdokumentationen.
Derudover vil vi i testfasen gå ind og se på interaktionen mellem klasserne.

4.10 Afslutning

Efter denne analyse af strukturen burde vi være klar til at implementere software styringen.

5. Underskrift

Kravspecifikationen er udleveret til kunden d. 13. marts 2003

_____

_____

_____

_____

_____

Gruppe 6