

obligatorisk aflevering 3 Microwaveoven

Gruppe 14

29. april 2020



|  |  |
| --- | --- |
| **Navn** | **Studienummer** |
| Caroline Rhode Nissen | 201806156 |
| Ina Bargmann Meyer | 201808400 |
| Maria Louise Pedersen | 201704927 |

URL til Jenkins:

* <http://ci3.ase.au.dk:8080/job/SWT14_HandIn3/>

URL til GitHUb:

* <https://github.com/BestGroup14/MicrowaveOven>

Indholdsfortegnelse

[Indledning 2](#_Toc38619706)

[Dependency tree 2](#_Toc38619707)

[Bottom Up Plan 4](#_Toc38619708)

[Bottom Up – Step 1 5](#_Toc38619709)

[Fundne fejl 6](#_Toc38619710)

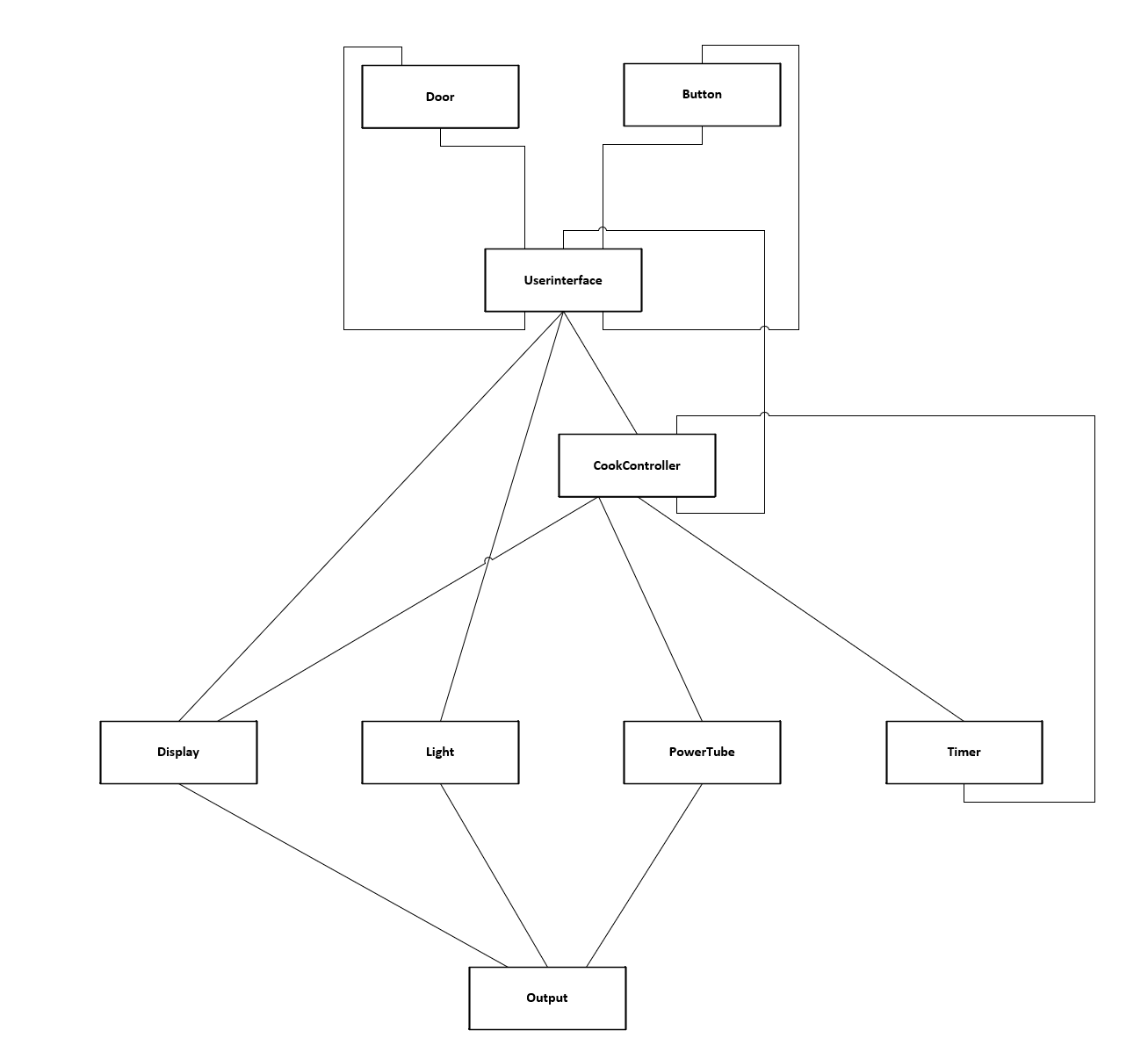
[Fejl 1 6](#_Toc38619711)

# Indledning

Denne journal vil først og fremmest indeholde et dependency tree for MicrowaveOven. Derudover vil der være en integrations plan for integrationstesten af systemet, samt en kort forklaring der forklare valget af integrationstest strategi. Der implementeres integrationstest for mindst 1 step, og for dette step vil de moduler der testes og deres afhængigheder blive markeret i dependency tree’et.

# Dependency tree

I forbindelse med at planlægge integrations test af MicrowaveOven startes der med at udarbejde et dependency tree, som hjælper til at give et overblik over alle afhængighederne mellem de forskellige moduler i koden. Stregerne i mellem modulerne angiver afhængighederne mellem modulerne. Afhængigheden går fra bunden af et modul til toppen af et andet modul, hvilket betyder at det modul, hvor stregen går fra bunden er afhængig af det modul, hvor stregen er placeret i toppen. Måden hvorpå man finder dependency tree er ved blandt andet at kigge på klassediagrammer og sekvensdiagrammer. På figur 1 ses vores dependency tree for MicrowaveOven.



Figur 1. Dependency Tree for MicrowaveOven

Som det ses på figur 1, så er der flere loops. Loops kommer enten fra, at begge moduler er afhængige af hinanden, eller ud fra events. Grunden til at event er en loop skyldes, at events er en dobbeltrettet association. Derudover er Door og Button valgt til at være øverst, da det er disse som styrer systemet. Derefter vælges klasserne alt efter, hvordan afhængighederne går ned gennem koden.

# Bottom Up Plan

I teorien burde vi teste Output, som 1. step, hvor Display, Light og PowerTube ville være top moduler og Ouput ville være inkluderet, men da det er en console.writeline metode er denne svær og teste, og derfor venter vi med denne til sidst. Det vil sige, at der er alt andet testet og derfor hvis programmet fejler, ved vi, at det er Output, som fejler. Vi springer derfor Output over, som 1. step, og Output modulet bliver derfor erstattet med en stub, hvilket også gælder de resterende steps. Som det sidste step, der vil blive testet vil det derfor Button og Door, som er topmoduler samt at alle andre moduler er inkluderet. Enten kan vi teste Output manuelt, hvor vi tjekker consollen, om der bliver udskrevet det ønskede, ellers kan man benytte en StringWriter til at teste, om der udskrives det korrekte i consollen.

På tabel 1 ses integrationsplanen, som har til formål at teste alle interfaces mellem modulerne. Ved hjælp af denne plan testes alle afhængigheder og dermed hele koden, og går alle tests godt vil vi have et komplet virkende system.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Button | Door | UserInterface | CookController | Display | Timer | PowerTube | Light | Output |
|  |  |  |  | T | X | X | X |  | S |
|  |  |  | T |  | X |  |  | X | S |
|  |  |  | T | X | X | X | X |  | S |
|  |  | T | X | X | X | X | X | X | S |
|  | T |  | X | X | X | X | X | X | S |
|  | T | T | X | X | X | X | X | X | X |

Tabel 1. Bottom Up Plan

Nedenfor ses en forklaring af de forskellige forkortelser, der bruges i tabellen:

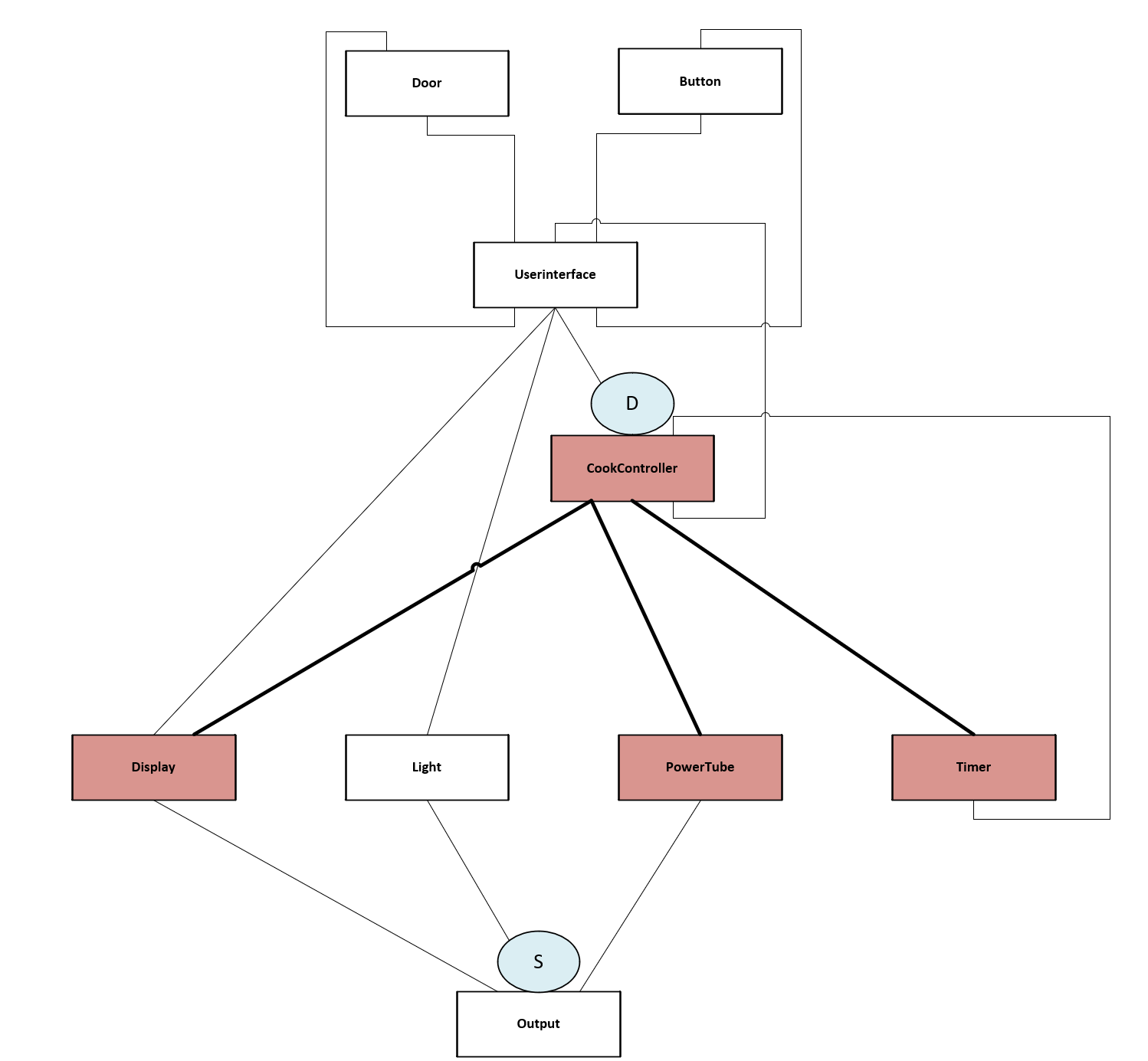
T 🡪 Dette modul er inkluderet, og er top modulet, som test driveren interagerer med

X 🡪 Dette modul er inkluderet

S 🡪 Dette modul er en fake – En stub eller en mock

# Bottom Up – Step 1

I forbindelse med integrationstesten skal mindst et integrationsstep indeholdende mindst tre af modulerne i designet testes. På figur 2 ses step 1 af vores integrationsplan med Bottom Up metoden.



Figur 2. Bottom Up - Step 1

De røde bokse på viser de moduler, som er inkluderet i steppet. D betyder, at det er testdriveren, og dermed er denne, som systemet testes gennem. S betyder, at det er en stub.

# Fundne fejl

Herunder vil de fejl, der er blevet fundet undervejs i integrationstesten blive beskrevet, samt hvordan de blev fundet.

## Fejl 1