MIKROBØLGEOVN

Obligatorisk aflevering 3

Figure 1

Casper Fevre Hansen 201501949
Michael Juhl 20084844
Jim Sørensen 201602614

Gruppe: 34

Date: 2021-04-05

Contents

1 Indledning

Denne rapport beskriver den 3. obligatoriske handin i faget Softwaretest. Rapporten er blevet udarbejdet i fællesskab af gruppe 34.

Selve rapporten omhandler design, integration test af software til en mikrobølgeovn. Der ud over er der lavet rettelser i udleveret kode og dertilhørende unit tests. Disse rettelser er fundet ved at de lavet integration tests.

Link til GitHub-repositoriet:
https://github.com/TeamSWT34/MicrowaveHandin3

Link til Jenkins-jobbet:
http://ci3.ase.au.dk:8080/job/TeamSWT34Handin3/

2 Designe af integration test

Figur: (2) viser dependency træ over mikrobølge system.

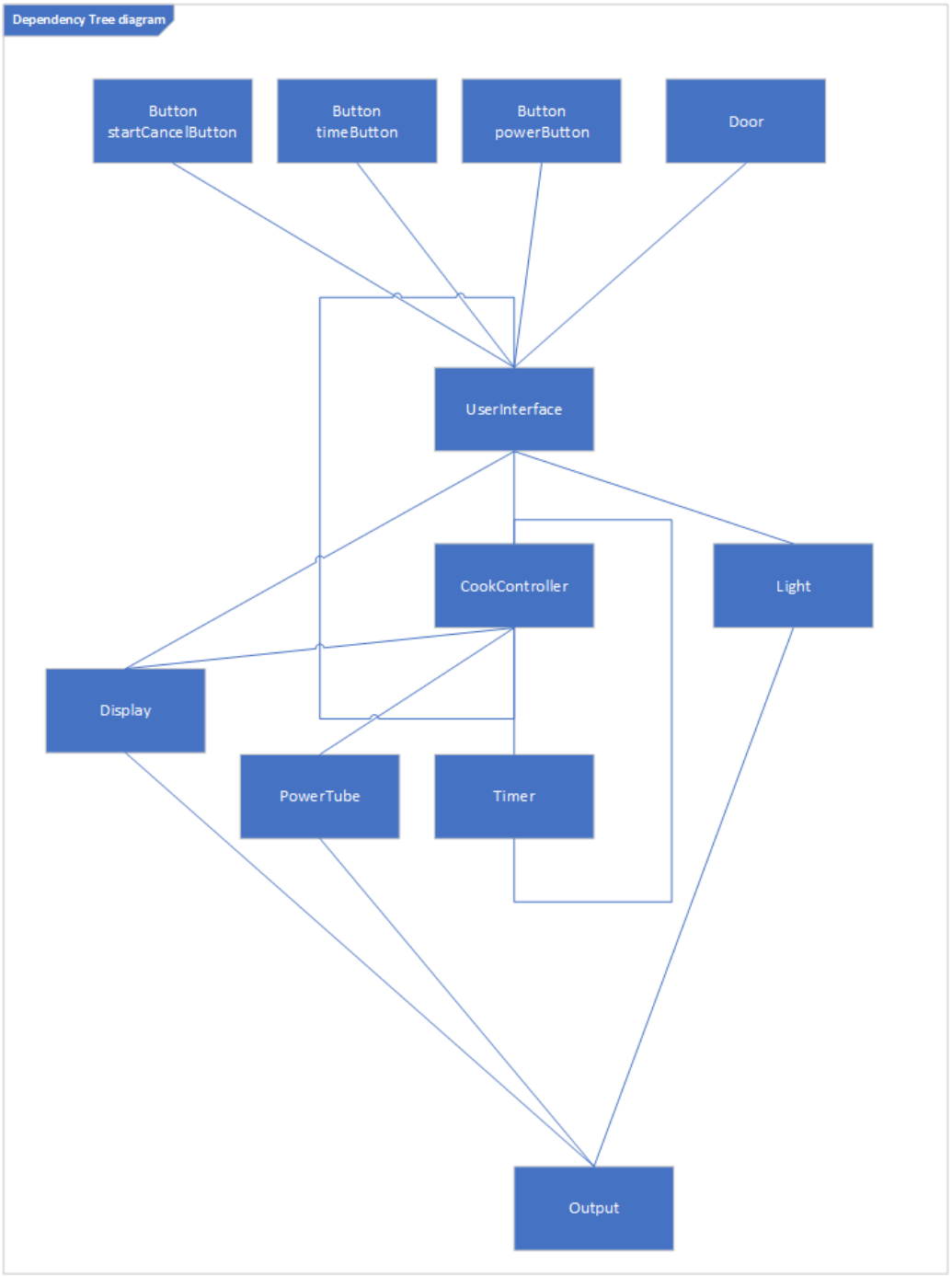

Figure 2: Dependency Tree

Der er brugt top down, integrationtests strategi. Dette er fra fordi det gav et hurtig og nemt overblik over det tabalen, ved at tage udgangspunkt i at bruger knapperne som driver. Da output ligger langt nede i dependency træet: (2), gjorde dette også at ved ButtonUp pattern ville der have været mange string tests.

Step #	Button powerButton	Button timeButton	Door	Button startCancelButton	UserInterface	CookController	Light	Display	Timer	PowerTube	Output
1	T	S	S	S	X	S	S	S			
2	T	T	S	S	X	S	S	S			
3	T	T	T	S	X	S	S	S			
4	T	T	T	T	X	S	S	S			
5	T	T	T	T	X/S	X	S	S	S	S	
6	T	T	T	T	X	X	X	S	S	S	S
7	T	T	T	T	X	X	X	X	S	S	S
8	T	T	T	T	X	X	X	X	X	S	S
9	T	T	T	T	X	X	X	X	X	X	S
10*	T	T	T	T	X	X	X	X	X	X	X
	T: This module is included, it's the/a top module, and the one driven										
	X: This module is included										
	S: This module is faked: stubbed or mocked										
	*Step # 10 is difficult to automate because of Output to console!										

Figure 3: MicrowaveTopDownPlan

- Step 1 - 4
  - Test kommunikation imellem knapper og Userinterface
- Step 5
  - Test kommunikation imellem Userinterface og CookCtroller
  - Bruger mock af UserInterface for at teste double dependency
- Step 6
  - Test kommunikation imellem Userinterface og Light
- Step 7
  - Test kommunikation imellem Userinterface/Cookcontroler og Display
- Step 8
  - Test kommunikation imellem Cookcontroler og Timer
- Step 9
  - Test kommunikation imellem Cookcontroler og PowerTube
- Step 10
  - Test kommunikation imellem Light/Display/PowerTube og Output
  - Da Output skriver på konsollen er dette svært at teste ved integrationtests og bliver derfor testet manuelt

3 Integration test

3.1 Fejl i Timer

Ved testen af Timer nedefor, blev der fundet en fejl: (4)

Listing 3.1: Test af timer

```
1   [TestCase(1,1,0,59)]
2   public void Timer_Sleeps_LogLine_Time_Output(int startMin, int waitTimeInSec,int expectMin,int expectSec)
3   {
4       const int LATENCY_MS = 100;
5       int zeroSec = 0;
6
7       sut_PowerButton.Press();
8
9       for (int i = 0; i < startMin; i++)
10          sut_TimeButton.Press();
11
12      sut_StartCancelButton.Press();
13
14      fakeOutput.Received().OutputLine($"Display shows: {startMin:D2}:{zeroSec:D2}");
15
16      Thread.Sleep(waitTimeInSec*1000+LATENCY_MS);
17
18      fakeOutput.Received().OutputLine($"Display shows: {expectMin:D2}:{expectSec:D2}");
19  }
```

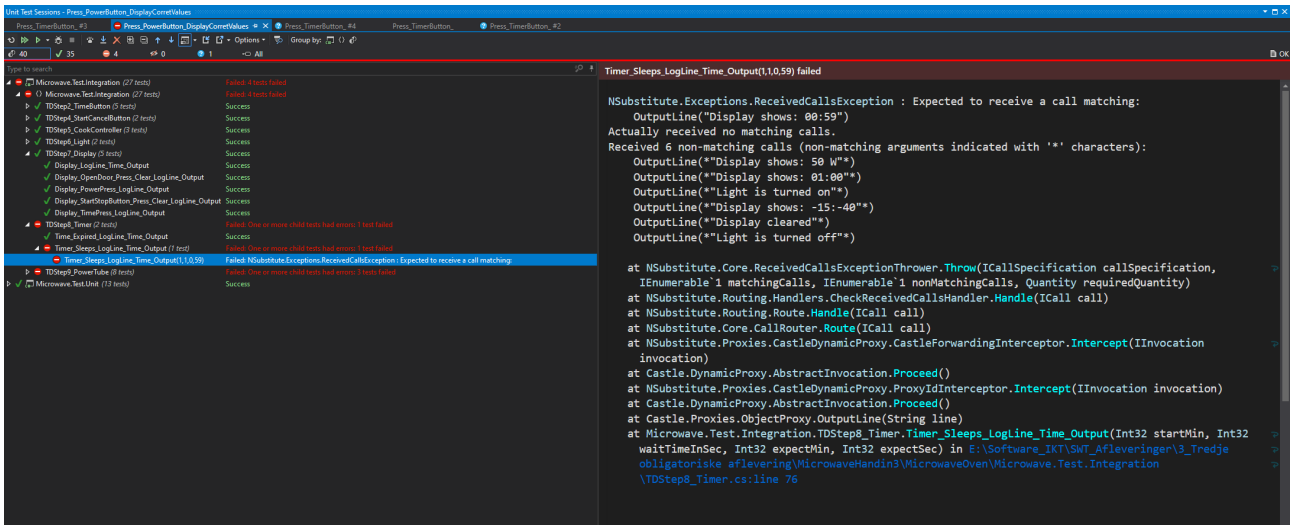

Figure 4: TDStep8_Timer

Det ses på figur: (4) at der blev udskrevet "-15:-40" i 'OutputLine' men skulle havde været "00:59". Dette skyldes at i 'Timer.cs' at timer arbejder med millisekunder og hvor 'CookControler' arbejder med sekunder. For at løse denne fejl er 'TimerRemaining' i 'Timer.cs' rettet til at - med 1 og ikke 1000 som kan ses i code: (3.3)

Listing 3.2: Timer

```
1   private void OnTimerEvent(object sender, System.Timers.ElapsedEventArgs args)
2   {
3       // One tick has passed
4       // Do what I should
5
6       // --------------------
7       //TimeRemaining -= 1000; //Old Fail is fix
8       TimeRemaining --;
9
10      TimerTick?.Invoke(this, EventArgs.Empty);
11
12      if (TimeRemaining <= 0)
13      {
14          Expire();
15      }
16  }
```

Efter at koden er blevet ændret for at rette op på fejl, er UNIT-testene blevet kørt igen, der kun naturligvis fejl. UNIT-teste herunder fejle før da den var sat til en Timer på 2000, men efter at koden er ændre fra millisekunder til sekunder vil den ikke udløbet efter 2100ms derfor er testen nu ændret til 2s. Dette gør sig også gældende for alle test hvor Timer bliver startet.

Listing 3.3: Timer

```
1   [Test]
2   public void Start_TimerTick_CorrectNumber()
3   {
4       ManualResetEvent pause = new ManualResetEvent(false);
5       int notifications = 0;
6
7       uut.Expired += (sender, args) => pause.Set();
8       uut.TimerTick += (sender, args) => notifications++;
9
10      //uut.Start(2000);
11      uut.Start(2);
12
13
14      // wait longer than expiration
15      Assert.That(pause.WaitOne(2100));
16
17      Assert.That(notifications, Is.EqualTo(2));
18  }
```

3.2 Fejl i PowerTube

Ved testen af PowerTube nedefor, blev der fundet en fejl: (5)

Listing 3.4: Test af PowerTube

```
1    [TestCase(1,50)]
2    [TestCase(2,100)]
3    [TestCase(5, 250)]
4    [TestCase(10, 500)]
5    [TestCase(14, 700)]
6    [TestCase(15, 50)]
7    public void PowerTube_StartStopButton_Start_Output(int press, int power)
8    {
9        for (int i = 0; i < press; i++)
10           sut_PowerButton.Press();
11
12
13       sut_TimeButton.Press();
14       sut_StartCancelButton.Press();
15       fakeOutput.Received().OutputLine($"PowerTube works with {power}");
16   }
```

I udskrift fra PowerTube testen ses det at den ønskede værdig for power skal være imellem 1 og 100 men var på 500. Ved at se på PowerTube kan det ses at PowerTube.TurnOn() har pre-condition af power til et interval mellem 1-100. Men i dokumentationen er det vist at den skal stige med 50 og op til 700. For at rette dette er if-sætninger her under (3.5) ændret fra interval 1-100 til 50-700.

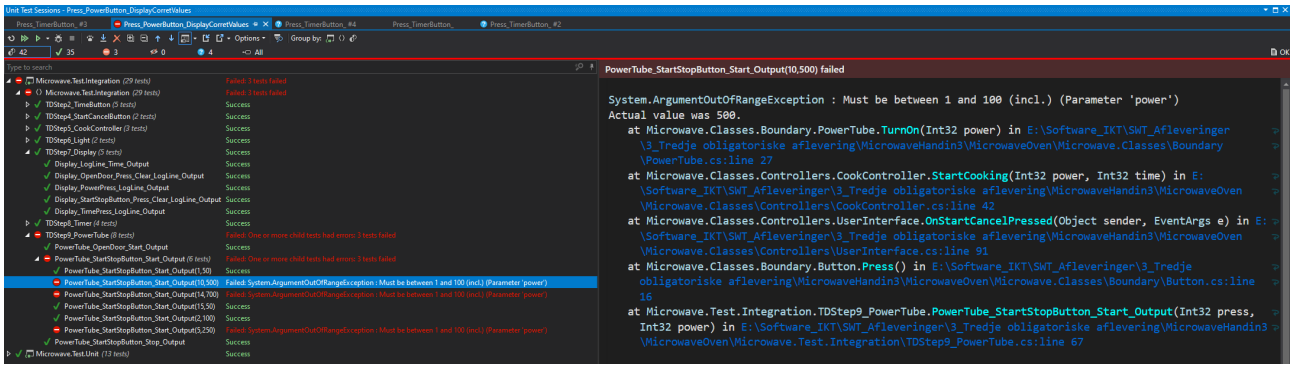

Figure 5: TDStep9_Powertube

Listing 3.5: PowerTube

```
1    public void TurnOn(int power)
2    {
3        /*
4        if (power < 1 || 100 < power)
5        {
6            throw new ArgumentOutOfRangeException("power", power, "Must be between 1 and 100 (incl.)");
7        }
8        */
9        // Fix code below - Boundary error for power:
10       if (power < 50 || 700 < power)
11       {
12           throw new ArgumentOutOfRangeException("power", power, "Must be between 50 and 700 (incl.)");
13       }
14
15       if (IsOn)
16       {
17           throw new ApplicationException("PowerTube.TurnOn: is already on");
18       }
19
20       myOutput.OutputLine($"PowerTube works with {power}");
21       IsOn = true;
22   }
```

UNIT-testen herunder er efter ændringen i koden blevet rettet, ved at fjerne den første TestCase hvor den tester med 1. Da dette nu vil give en exception, da det ikke er muligt at have power på en uden for pre-condition på 50-700.

Listing 3.6: PowerTube Test1

```
1       //[TestCase(1)]
2       [TestCase(50)]
3       [TestCase(100)]
4       [TestCase(700)]
5       public void TurnOn_WasOffCorrectPower_CorrectOutput(int power)
6       {
7           uut.TurnOn(power);
8           output.Received().OutputLine(Arg.Is<string>(str => str.Contains($"{power}")));
9       }
```

UNIT-testen herunder tester de steder hvor der kommer exception ved at power er en værdig der ligger uden for pre-condition. Efter at koden er blevet ændret til at power nu skal være imellem 50 og 700 er nogle af TestCase blevet rettet til.

Listing 3.7: PowerTube Test2

```
1       [TestCase(-5)]
2       [TestCase(-1)]
3       [TestCase(0)]
4       [TestCase(49)]
5       //[TestCase(101)]
6       //[TestCase(150)]
7       [TestCase(701)]
8       [TestCase(750)]
9       public void TurnOn_WasOffOutOfRangePower_ThrowsException(int power)
10      {
11          Assert.Throws<System.ArgumentOutOfRangeException>(() => uut.TurnOn(power));
12      }
```