Architecture and Design of

Embedded Real-Time

Systems

Journal on Exercises 13/11/2018

Group 9

Authors:

David Jensen

Henrik Bagger Jensen

Supervisor:

Jalil Boudjadar

Contents

[1 Introduction 2](#_Toc529440984)

[1.1 Intro to requirements for the exercises 2](#_Toc529440985)

[1.2 Patterns used in the solution 2](#_Toc529440986)

[2 Solution 2](#_Toc529440987)

[2.1 Introduction to architecture and decisions 2](#_Toc529440988)

[2.2 Use Case View 2](#_Toc529440989)

[2.3 Logical View 2](#_Toc529440990)

[2.3.1 Class diagram(s) 2](#_Toc529440991)

[2.3.2 State Diagram(s) 5](#_Toc529440992)

[2.4 Implementation View 7](#_Toc529440993)

[2.4.1 Implementation details 7](#_Toc529440994)

[3 Discussion of results 7](#_Toc529440995)

[4 Conclusion 7](#_Toc529440996)

**Revision History**

|  |  |  |
| --- | --- | --- |
| Revision | Date/Authors | Description |
| 1.0 | dd.mm.yyyy/xxx | Document XX |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

In this exercise we will implement a State Machine with the GoF State Pattern and the GoF Singleton Pattern

## Intro to requirements for the exercises

## Patterns used in the solution

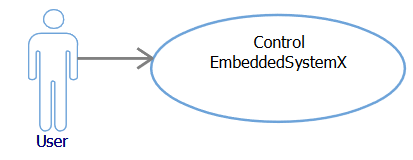
We used the GoF State and Singleton State.

# Solution

## Introduction to architecture and decisions

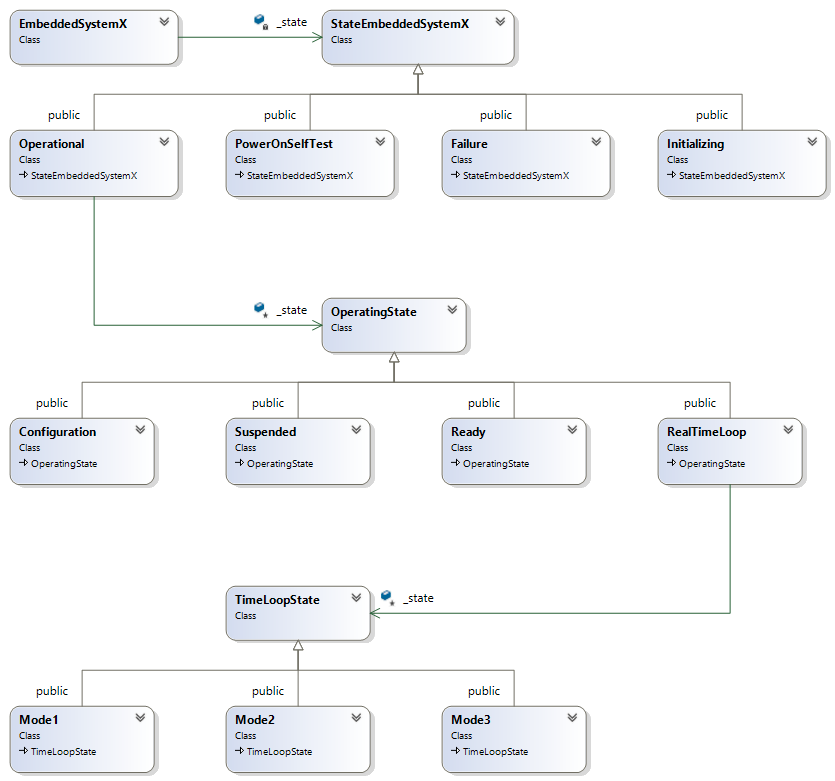
By design we are using the Gof State to model our system states and to make sure we keep track of what state we are in the Singleton helps us. This makes sure what even if we revisit a state we will use the same instance.

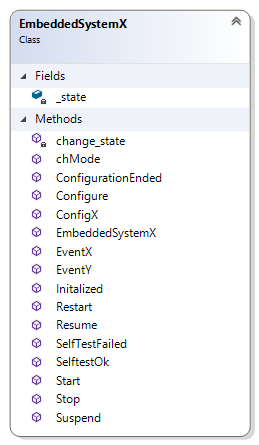
## Use Case View

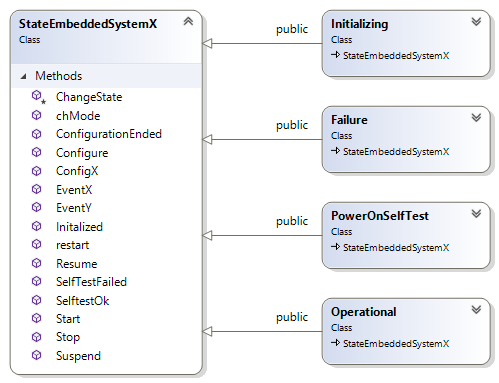


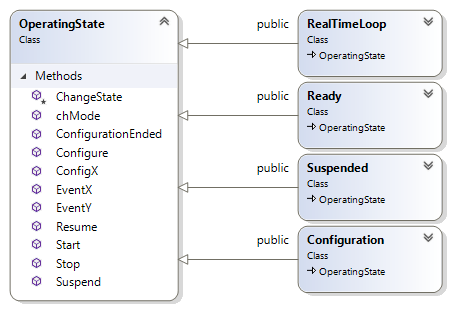
## Logical View

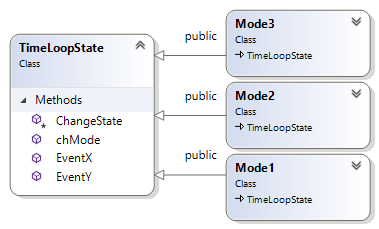
### Class diagram(s)



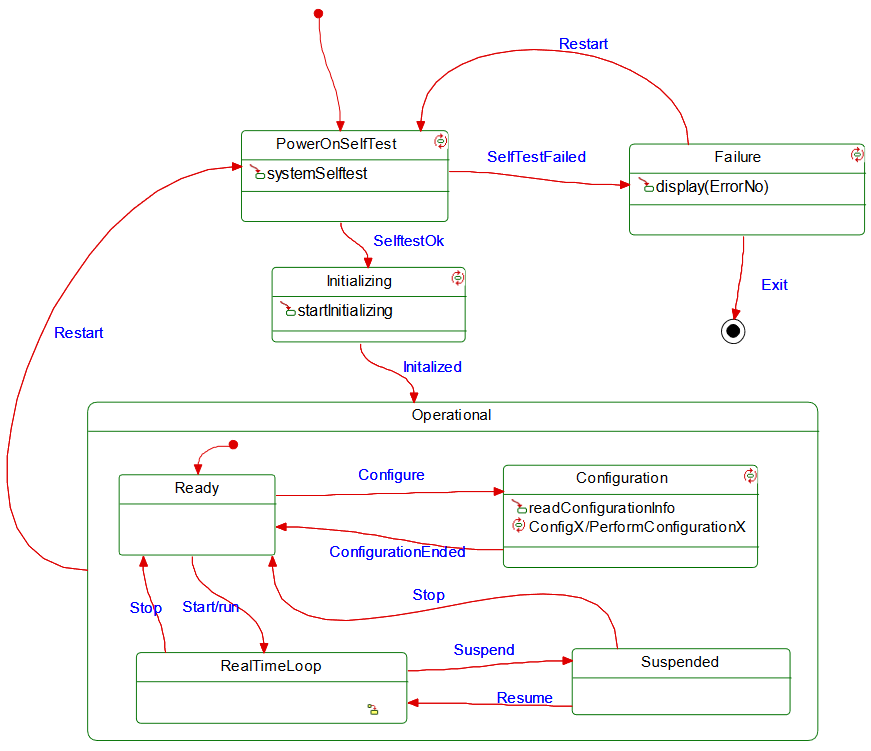


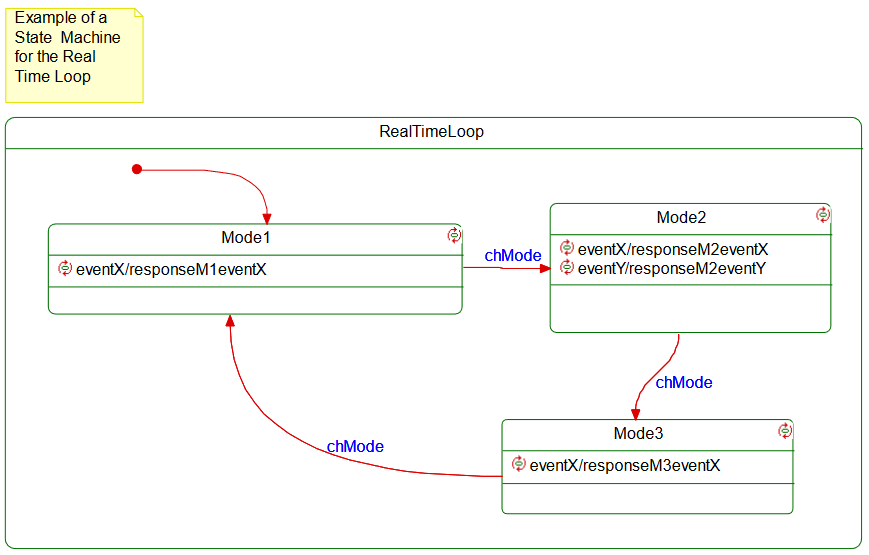






### State Diagram(s)



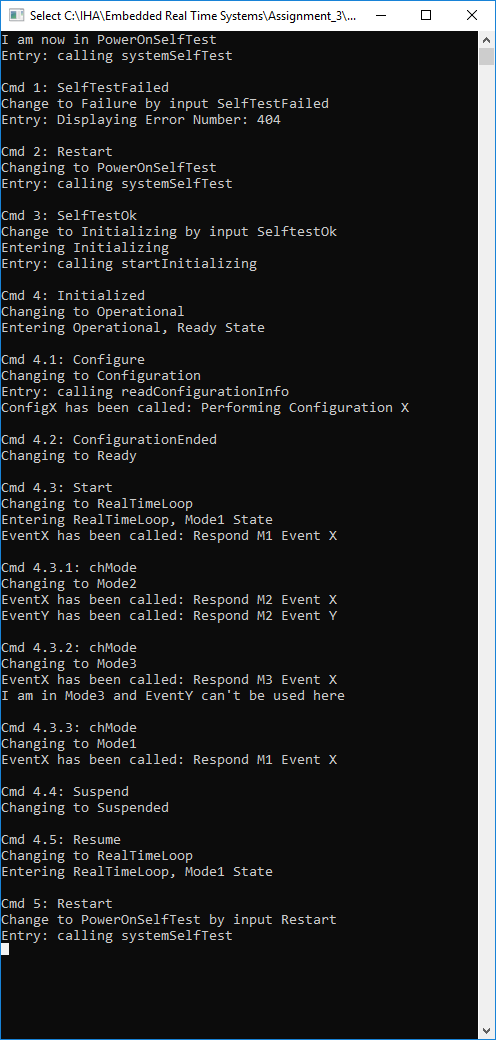


## Implementation View

### Implementation details

# Discussion of results

As seen in the screenshots, we can move around in the state machine and get the wanted output.



# Conclusion

We have shown how two patterns can be used together.