Architecture and Design of

Embedded Real-Time

Systems

Journal on Exercises 3

Group 10

Authors:

Mads Krabsen, 201507805

Jonas K. Rask, 2015xxxxxx

Frederik P. Madsen, 2015xxxxx

Supervisor:

Jalil Bxxxxxxxxxxxx

Contents

[1 Introduction 2](#_Toc25400565)

[1.1 Intro to requirements for the exercises 2](#_Toc25400566)

[1.2 Patterns used in the solution 2](#_Toc25400567)

[2 Solution 3](#_Toc25400568)

[2.1 Introduction to architecture and decisions 3](#_Toc25400569)

[2.2 Use Case View 3](#_Toc25400570)

[2.3 Logical View 4](#_Toc25400571)

[2.3.1 Class diagram(s) 4](#_Toc25400572)

[2.3.2 Sequence diagram(s) 4](#_Toc25400573)

[2.3.3 State Diagram(s) 4](#_Toc25400574)

[2.4 Implementation View 5](#_Toc25400575)

[2.4.1 Implementation details 5](#_Toc25400576)

[3 Discussion of results 5](#_Toc25400577)

[4 Conclusion 5](#_Toc25400578)

[Appendix A 5](#_Toc25400579)

**Revision History**

|  |  |  |
| --- | --- | --- |
| Revision | Date/Authors | Description |
| 1.0 | 23.11.2019/MK | Document created |
| 1.1 | 23.11.2019/MK | Introduction, Requirements, Patterns and UC view initiated |
|  |  |  |
|  |  |  |

# Introduction

This Journal is made as an assignment for the Embedded Real-time systems course at Aarhus University. The journal will consist of a short description of the requirements of the system, then an identification of the design patterns used to realise the system and then a short description of the architectude and design using the 4+1 software engineering model.

## Intro to requirements for the exercises

The requirements of the exercise will be stated here:

1. The EmbeddedSystemX must be implemented using GoF State Pattern
2. Each state from the GoF State Pattern must implemented using Singleton pattern
3. The command pattern must implement the user interface

## Patterns used in the solution

The solution for the system relies on three design patterns. To realise the state machine on Figure 1 the state pattern is used. The state pattern as described by the GoF follows the generic class diagram on Figure 1. The implementation realises two state patterns, one for the overall state machine with states: *PowerOnSelfTest, Initializing, Failure, Operational.* The second STM handles the internal states of the operational state. The Operational internal state machine has states: *Ready, Configuration, RealTimeLoop, Suspended.*

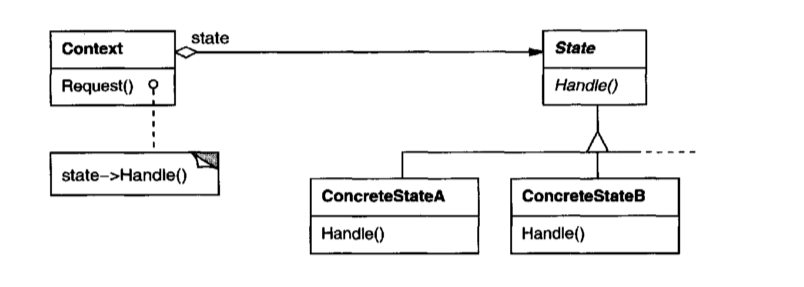


Figure 1 Generic Structure of State Pattern

To ensure that each state object of the state patterns isn’t created and destroyed every time a new state is entered and exited each inheritor of the state class is implemented using the Singleton pattern. This ensures as described by GoF that an static instance of the class is created the first time ::getInstance function is called.

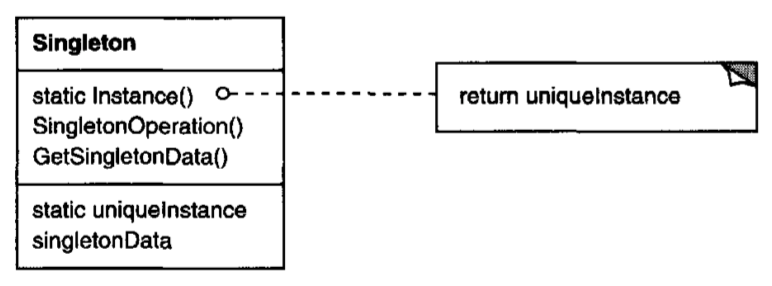


Figure 2 Generic Structure of Singleton Pattern

To abstract the user interface from the internal workings of the EmbeddedSystemX a command pattern is implemented so that each action performed by the user is abstracted away from the implementation of the action.

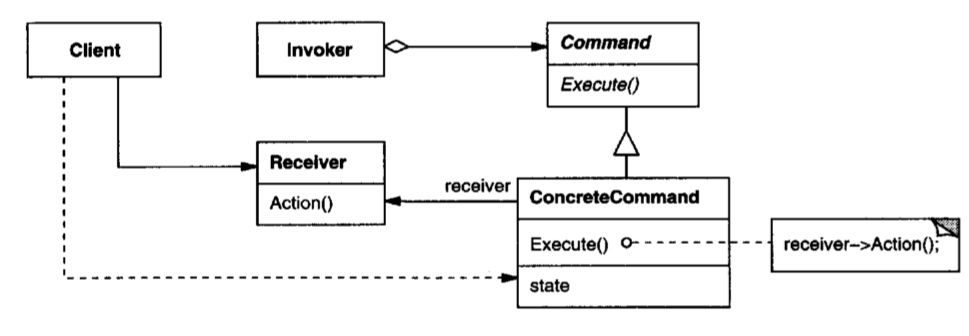


Figure 3 Generic Structure of Command Pattern

# Solution

## Introduction to architecture and decisions

## Use Case View

The EmbeddedSystemX offers three main use cases for the user to interact with, these can be seen on the use case diagram on Figure 4. The use cases are derived from the system behaviour described in state machine diagram on Figure 5. Here is can be identified that the user is responsible for triggering the state transitions:

|  |  |
| --- | --- |
| State transition | Description |
| Failure -> PowerOnSelfTest | If the initial PowerOnSelfTest is failed the user must trigger either a system exit or a restart. |
| Ready\_-> Configure | When the system is in operational ready state the user must trigger transition to configure state. |
| Ready -> RealTimeLoop | When the system is in operational ready state the user must trigger transition to RealTimeLoop state. This could be a sensor read, actuator actuate operational loop |
| RealTimeLoop->Ready | The user must stop the system from executing |
| RealTimeLoop \_-> Suspend | When the system is executing the real time loop the user must trigger the suspend |
| Suspend -> RealTImeLoop | When the system is suspended the user must trigger the resume. |
| Operational -> PowerOnSelfTest | The user must trigger a restart so that the system can go from operational state to PowerOnSelfTest state |

The functionality described in the table above is summed up in the Actor context diagram below

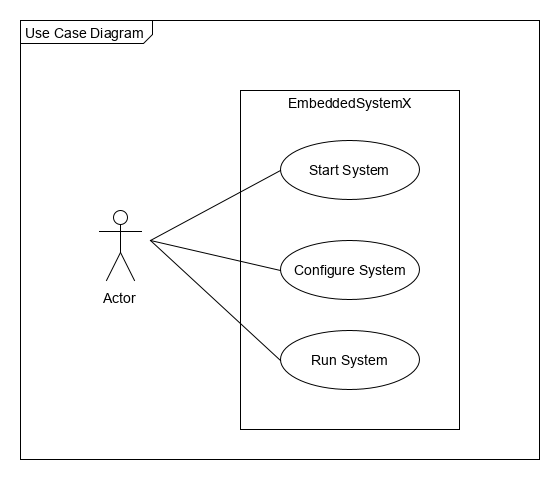


Figure 4 Actor-context diagram for the EmbeddedSystemX

### Start System

The Start System use case follows the general use case description below:

|  |  |
| --- | --- |
| UC 1: Start System | Description |
| Actor | User |
| PostCondition | System is in Operational Ready state |
| Trigger | User power on system |
| Main flow |  |
|  | User power on system |
|  | System performs PowerOnSelfTest |
|  | System performs Initializing |
|  | System enters Operational Ready state |
|  | User is informed of System state |
| Exceptions |  |
| selfTest Fail | PowerOnSelfTest fail |
|  | System informs user that system self test fails |
|  | User restarts system |
|  | UC continues form step 2. |

### Configure System

|  |  |
| --- | --- |
| UC 1: Start System | Description |
| Actor | User |
| Precondition | System is in Operational Ready state |
| PostCondition | System has updated configuration |
| Trigger | User trigger configuration |
| Main flow |  |
|  | User enters configuration |
|  | System enters Configuration state |
|  | System reads configuration information |
|  | System informs user that configuration is done |
|  | System returns to Operational ready state |

### Run System

|  |  |
| --- | --- |
| UC 1: Start System | Description |
| Actor | User |
| Precondition | System is in Operational Ready state |
| PostCondition | System returns to Operational Ready state |
| Trigger | User trigger start/run |
| Main flow |  |
|  | User trigger start/run |
|  | System enters RealTimeLoop state |
|  | System runs real time loop indefinitely |
|  | User stops system |
| Exceptions | User suspends system |
|  | User trigger suspend real-time loop |
|  | System enters Suspended state |
|  | User trigger Resume real-time loop |
|  | UC continues from step 2 |

## Logical View

### Class diagram(s)

### Sequence diagram(s)

### State Diagram(s)

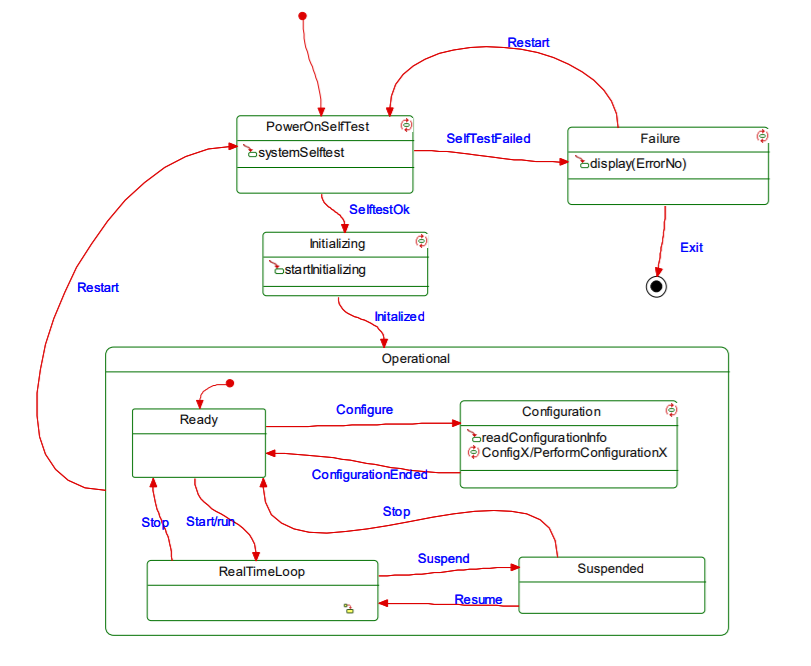


Figure 5 A state Diagram of EmbeddedSystemX

## Implementation View

### Implementation details

# Discussion of results

# Conclusion

# Appendix A