2/10/2014

Christian Janeczek, Wolfgang Mair

5AHITT

DEZSYS01

REMOTING PATTERNS

Contents

[Task Description 2](#_Toc399934344)

[Design consideration 3](#_Toc399934345)

[REQUIREMENT: State-Centric State Machine Solution 3](#_Toc399934346)

[Advantages 3](#_Toc399934347)

[Disadvantages 3](#_Toc399934348)

[REQUIREMENT: State-Centric State Machine Solution + Hidden Transitions 4](#_Toc399934349)

[Advantages 4](#_Toc399934350)

[Disadvantages 4](#_Toc399934351)

[REQUIREMENT: Event-Centric State Machine Solution 5](#_Toc399934352)

[Advantages 5](#_Toc399934353)

[Disadvantages 5](#_Toc399934354)

[REQUIREMENT: State Pattern State-Machine Solution 6](#_Toc399934355)

[REQUIREMENT: Table Driven State-Machine Solution 7](#_Toc399934356)

[Apportionment of work with effort estimation 8](#_Toc399934357)

[Final Time Apportionment 9](#_Toc399934358)

[Task execution 10](#_Toc399934359)

[Test report 11](#_Toc399934360)

[Bibliography 12](#_Toc399934361)

# Task Description

Das Framework für Remoting Patterns finden sie unter dem Thema "Resources"!  
  
Gruppenarbeit: 2 Mitglieder (Server/Client)  
  
Analysieren Sie in einer Gruppe von 2 Leuten die mitgelieferte Implementation der verteilten LeelaApplikation. Identifizieren Sie dabei alle verwendeten Elemente der "Basic Remoting Patterns" und erstellen Sie UML-Klassendiagramme für die Pakete comm, comm.socket, comm.soap, evs2009 und evs2009.mapping  
  
Schließen Sie die unfertigen Tests ab, und dokumentieren Sie etwaige Schwierigkeiten.  
  
Was ist zu tun?

* UML Klassendiagramm
* Erweitern der Testfälle (mind. einen Testfall erweitern)
* Kritik und Verbesserungsvorschläge

*Punkte (16):*

*Identifikation von Basic Remoting Patterns ... 1Pkt  
Beschreibung der Applikation ... 4Pkt  
UML-Diagramme ... 3Pkt  
Schreiben von einem neuen Testfall ... 2Pkt  
konstruktive Verbesserungsvorschläge / Kritikpunkte ... 6Pkt*

# Design consideration

## REQUIREMENT: Identification of Remoting Patterns Usage

Which patterns have been used? Bla bla

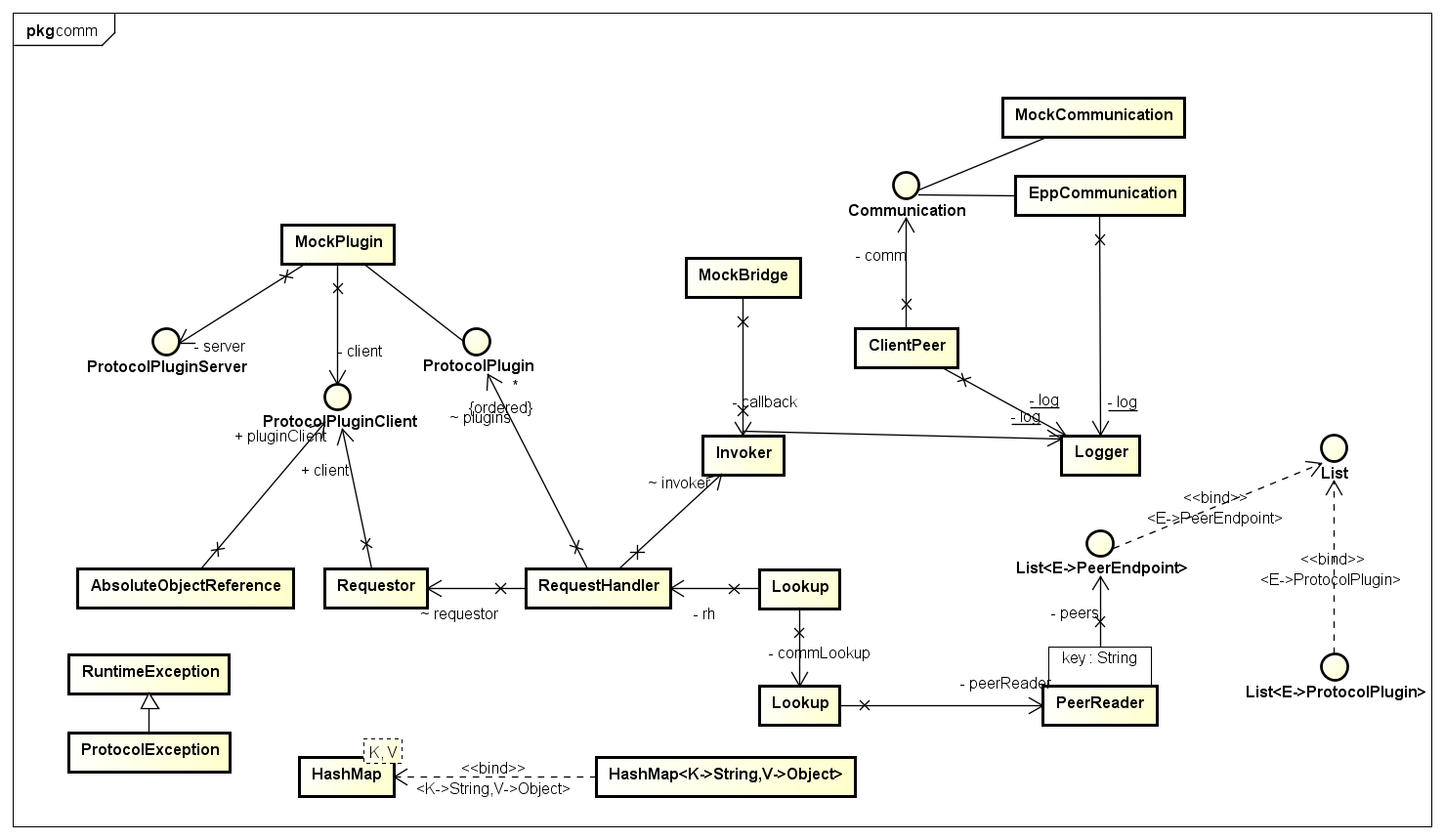
## REQUIREMENT: Description of the application

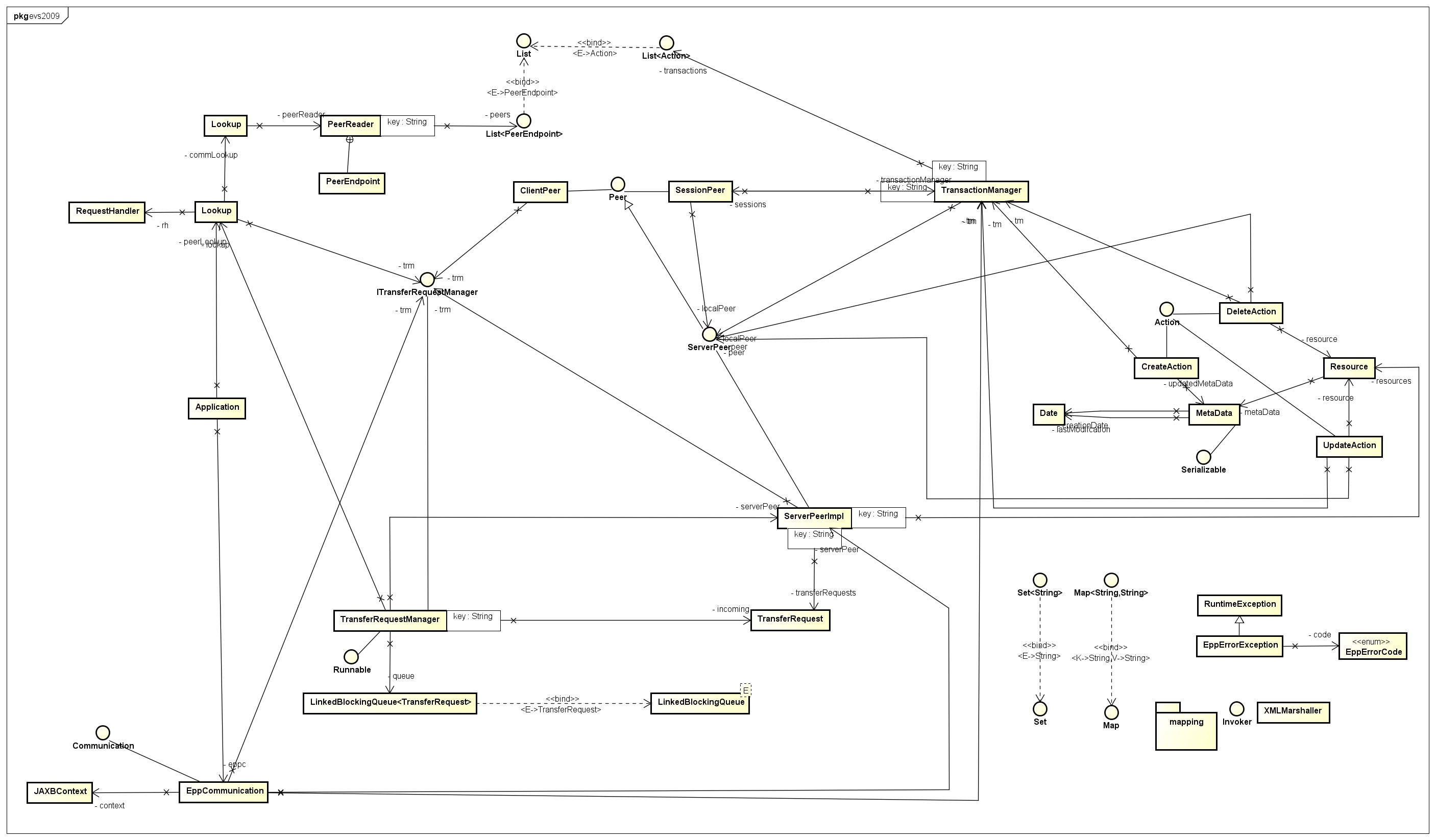
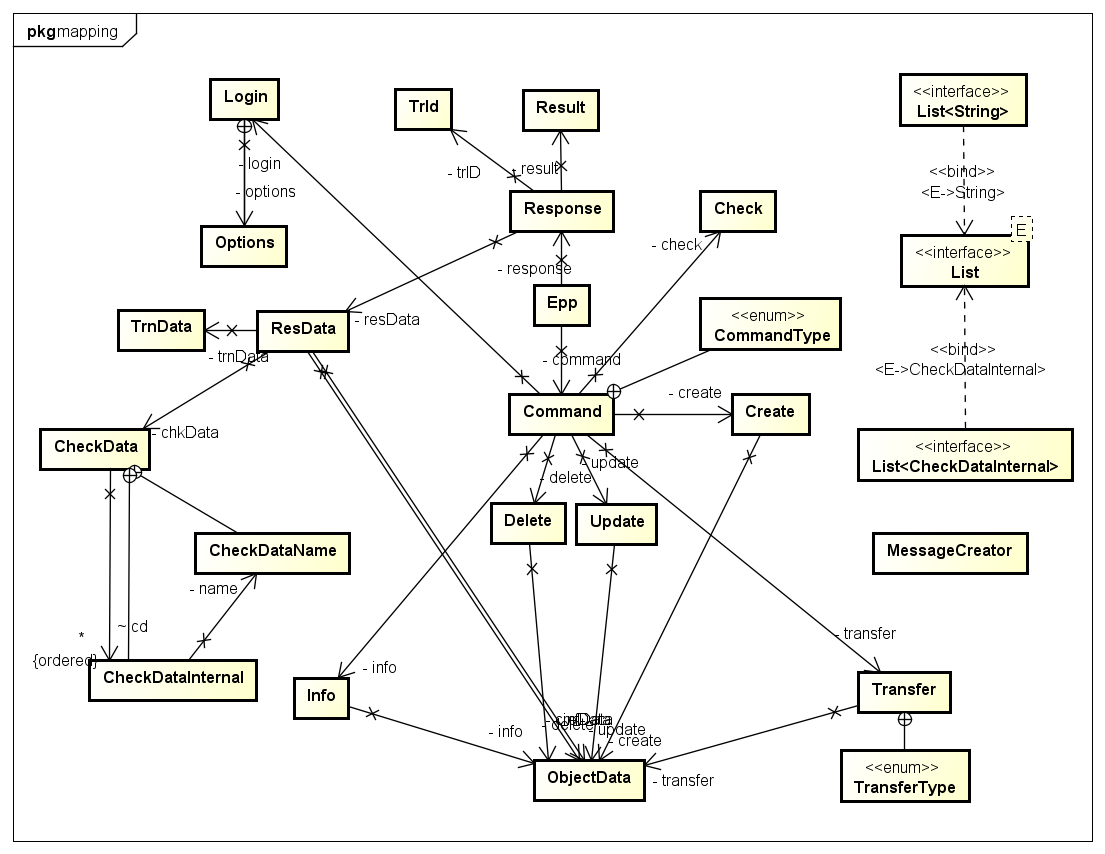
The application does this and that, also it has the functionality of bla bla

It is working like jada jada

## REQUIREMENT: UML-Diagrams

The UMLs are located right heeeeeeere



This UML is way too bigThis UML ... I like it

## REQUIREMENT: Writing a new Test Case

This Test was tested in the AuthenticationTests class.

/\*

\* A method which tests what happens when there is someone going to login twice

\*/

@Test

**public** **void** correctDoubleLogin() **throws** Exception {

peer.login(Helper.***correctPassword***, Helper.***correctPassword***);

peer.login(Helper.***correctPassword***, Helper.***correctPassword***);

peer.logout();

}

These Tests were tested in the CRUDTests class.

/\*

\* A method to test the reaction of the server when he receives a null object

\*/

@Test

**public** **void** correctCreationAndReadZero() {

String identifier = **null**;

insertObject(identifier);

**byte**[] readBytes = serverPeer.read(identifier);

*assertEquals*(getBytes().length, readBytes.length);

*assertEquals*(testString, **new** String(readBytes));

}

/\*

\* A method to test the reaction of the server when he receives a lot of read tasks at once

\*/

@Test

**public** **void** correctCreationAndReadloop() {

String identifier = "Wow";

insertObject(identifier);

**byte**[] readBytes = **null**;

**for**(**int** i = 0; i<500; i++){

readBytes = serverPeer.read(identifier);

}

*assertEquals*(getBytes().length, readBytes.length);

*assertEquals*(testString, **new** String(readBytes));

}

I am a happy new test case and this is my story….

Once upon a time there was a test case

It was written by Wolfgang Mair and lived happily ever after

THE END

## REQUIREMENT: Constructive Tips for Improvement / Critics

Some of the less efficient hotpoints of the application are:

Bla bla

Jada jada

Here?

*“I feel so lonely”*

# Apportionment of work with effort estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **Competent person(s)** | **Task** | **Description** | **Estimated time in h** |
| Janeczek | Design consideration | How do I realize the state machines in C? | 2 |
| Janeczek | State-Centric Solution | Implementing the State-Centric solution | 3 |
| Janeczek | State-Centric Hidden Solution | Implementing the State-Centric solution with Hidden Transitions | 2 |
| Janeczek | Event-Centric Solution | Implementing the Event-Centric solution | 2 |
| Janeczek | State Pattern Solution | Implementing the State Pattern Solution | 3 |
| Janeczek | Makefile | Creating a Makefile for the project | 1 |

# Final Time Apportionment

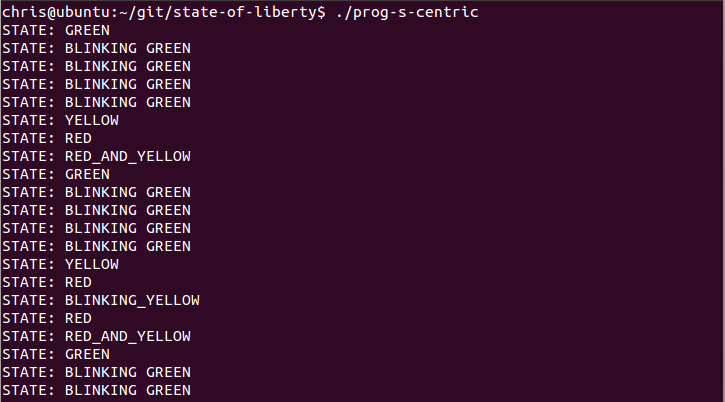
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Competent person(s)** | **Task** | **Estimated time in h** | **Actual time in h** | **Comment** |
| Janeczek | Design consideration | 2 | 3 | I need to get better at Software-Planning! |
| Janeczek | State-Centric Solution | 3 | 4 | It took me quite some time to get comfortable working in C |
| Janeczek | State-Centric Hidden Solution | 2 | 3 | Hidden Transactions required some newly defined methods, which consumed quite an amount of time |
| Janeczek | Event-Centric Solution | 2 | 2 | „Same“ as the State-centric, just with events |
| Janeczek | State Pattern Solution | 3 | ? | ? |
| Janeczek | Makefile | 1 | 0,5 | Makefiles are sexy |
|  |  | 13 | 12,5 |  |

# Task execution

1. Brainstorming with other people: “How do I realize all this in C?”
2. Installing all needed packages, for example: clang, sublime text editor, etc.
3. Getting started with C again. Some warming-up with older tasks.
4. Writing the state-centric solution. Took more time than expected, I got quite sloppy in C.
5. Got more familiar with C and wrote the state-centric hidden solution in an adequate amount of time.
6. The event-centric solution was an advanced solution of the already written state-centric solution.
7. From what I have heard, the State Pattern solution is going to be “zach”.
8. I already know how to define data types in C, but object-oriented programming will be more difficult than expected.
9. I need to learn, how to efficiently build my program. Component-based programming is a must-do, because code duplication is more than just affecting the style of my code.
10. I was successful in creating a header file with the method-heads.
11. One .c – file was redefining each method, which was defined in functions.h
12. Creating the Makefile was an easy task, and I came to this conclusion: Makefiles make the whole thing sexy

# Test report

## State Centric State-Machine



## State Centric State-Machine Hidden Transitions

## 

## Event Centric State-Machine

## 

# Bibliography

|  |
| --- |
| **Title:** How to define an enumerated type in C?  **Author:**Stéphane Gimenez  **Source:** <http://stackoverflow.com/questions/1102542/how-to-define-an-enumerated-type-enum-in-c>  **Last modified:**2011/09/13  **Last seen:** 2014/09/30 |
| **Title:** Structs in C  **Author:** P. Böhme  **Source:**<http://www2.informatik.uni-halle.de/lehre/c/c_struct.html>  **Last modified:**1996/02/15  **Last seen:** 2014/10/01 |
| **Title:** Creating your own header file in C  **Author:** Oliver Charlesworth  **Source:**<http://stackoverflow.com/questions/7109964/creating-your-own-header-file-in-c>  **Last modified:**2011/09/18  **Last seen:** 2014/10/01 |
| **Title:** Makefiles  **Author:**C-HowTo  **Source:**<http://www.c-howto.de/tutorial-makefiles.html>  **Last modified:**?  **Last seen:** 2014/10/01 |
|  |