Modul Programmieren & Datenstrukturen

Battleship - Projektdokumentation

Vorgabe der Dozenten

**Inhaltsverzeichnis**

[1. Einleitung 2](#_Toc353956523)

[1.1. Begriffe, Abkürzungen 2](#_Toc353956524)

[1.2. Referenzen 2](#_Toc353956525)

[2. Anforderungen 3](#_Toc353956526)

[2.1. Systemübersicht 3](#_Toc353956527)

[2.2. Anwendngsfall 4](#_Toc353956528)

[3. Systemspezifikation 5](#_Toc353956529)

[3.1. Zustände 6](#_Toc353956530)

[3.1.1. SelectingOpponent 7](#_Toc353956531)

[3.1.2. PreparingGrid 7](#_Toc353956532)

[3.1.3. Play 7](#_Toc353956533)

[3.1.4. Sequenzdiagramme 7](#_Toc353956534)

[3.2. Funktionale Sicht 8](#_Toc353956535)

[3.3. Verteilungs- und Betriebssicht 8](#_Toc353956536)

[3.4. Datensicht 8](#_Toc353956537)

[3.5. Schnittstellen 8](#_Toc353956538)

[3.6. Kommunikation über Netzwerk 8](#_Toc353956539)

[3.7. Erweiterungen 8](#_Toc353956540)

**Abbildungen**

[Abbildung 1: Kontextdiagramm der Battleship Application bei einem Spiel gegen einen Gegenspieler über das Netzwerk 3](#_Toc353191320)

[Abbildung 2: Kontextdiagramm der Battleship Application bei einem Spiel gegen einen Computergegner 3](#_Toc353191321)

[Abbildung 3: Die vier Komponenten der Battleship Application 5](#_Toc353191322)

[Abbildung 4: Statemaschine der Anwendung 6](#_Toc353191323)

[Abbildung 5: Sequenzdiagramm PreparingGrid 7](#_Toc353191324)

Versionen:

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Datum | Autor | Bemerkungen |
| 0.1 | 27.03.2013 | tisuter | Initiale Version |
| 0.2 | 08.04.2013 | tisuter | Überarbeitung mit P. Sollberger |
| 0.3 | 17.04.2013 | zasollbe | Einleitung angefügt |

# Einleitung

Geschätzte Studierende

in der zweiten Hälfte des Semesters bearbeiten Sie in einer kleinen Gruppe ein kleineres Softwareprojekt. Mit diesem Projekt sind die folgenden Ziele verbunden:

* Sie wenden im Unterricht gelernte Konzepte der Sprache Java in einem grösseren Kontext an.
* Sie wiederholen wesentliche Elemente der Programmiersprache Java.
* Sie implementieren eine Softwarelösung im Team.
* Sie können Kontextdiagramme lesen und interpretieren.
* Sie können Komponentendiagramme lesen und interpretieren.
* Sie können Sequenzdiagramme lesen und interpretieren.
* Sie können Zustandsdiagramme lesen und interpretieren.
* Sie können Sourcecode mit Hilfe von Klassendiagrammen dokumentieren.
* Sie können in einem grösseren Programm die Übersicht wahren.

Das gesamte Projekt ist als Lernprojekt zu verstehen, bei dem Sie Schritt für Schritt Kenntnisse erwerben und anwenden.

Die Dozierenden und Assistierende begleiten Sie während des Projekts und ermöglichen Ihnen, wesentliche Erfahrungen zu reflektieren. Zur Unterstützung dieses Prozesses erstellen Sie jede Woche für die Dozierenden einen kurzen Projekt-Statusrapport mit folgendem Inhalt:

* Welche Arbeiten wurden in der letzten Woche ausgeführt. Was hat gut geklappt, wo hatten oder haben Sie Probleme?
* Welche Tätigkeiten sind für die nächste Woche vorgesehen?
* Welche Knackpunkte (Herausforderungen oder Risiken) bestehen noch? Was gedenken Sie dagegen zu unternehmen?

Eine Vorlage für diesen Projekt-Statusrapport [2] finden Sie im ILIAS.

Viel Erfolg sowie spannende und wertvolle Projekterfahrungen wünschen Ihnen

Ihr Dozierendenteam

Begriffe, Abkürzungen

Die Anwendung wird in Englisch erstellt. Gewisse Begriffe werden aber in dieser SysSpec auf Deutsch benutzt. In der folgenden Tabelle sind die Übersetzungen.

|  |  |
| --- | --- |
| Engl. Begriff | Dt. Begriff |
| Battleship | Schiffe versenken |
| Grid | Spielfeld |
| Engine | Spielsteuerung |
| Opponent | Gegenspieler |

Referenzen

[1] Projektauftrag Schiffe versenken (Projektauftrag.pdf)

[2] Projekt-Statusrapport (PRG2\_Projekt-Statusrapport.docx)

# Anforderungen

Aus der Aufgabenstellung [1] lassen sich zwei User Storys ableiten:

User Story 1:

Als Spieler möchte ich über das Netzwerk gegen einen Gegenspieler Schiffe versenken spielen um Spass zu haben.

User Story 2:

Als Spieler möchte ich gegen einen Computergegner Schiffe versenken spielen um Spass zu haben.

Systemübersicht

Aus diesen User Stories lässt sich das System visualisieren.

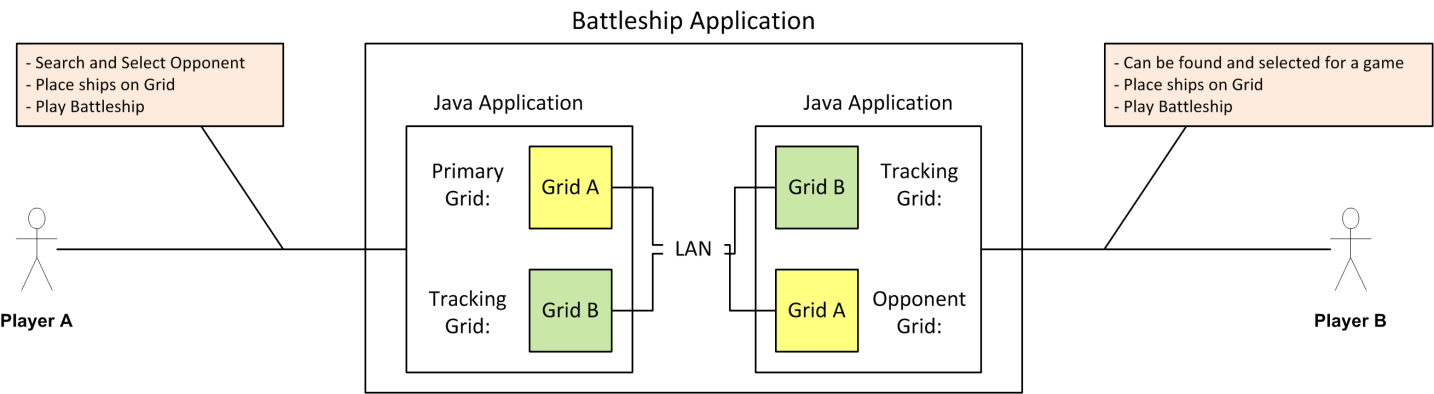


Abbildung : Kontextdiagramm der Battleship Application bei einem Spiel gegen einen Gegenspieler über das Netzwerk

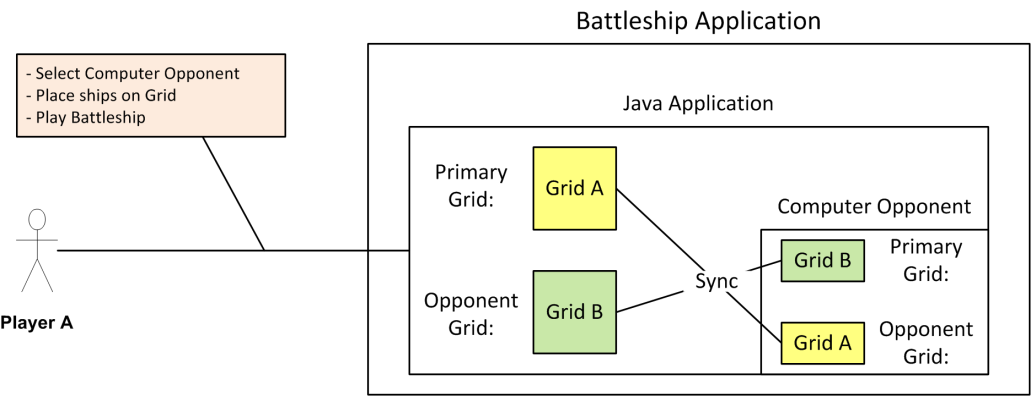


Abbildung : Kontextdiagramm der Battleship Application bei einem Spiel gegen einen Computergegner

Bei beiden Anwendungen wird deutlich, dass es vier Grids gibt, wobei jeweils zwei miteinander synchronisiert werden müssen. Diese Synchronisation ist die einzige Netzwerkaktivität während dem Spiel.

Anwendngsfall

Hier ist der Anwendungsfall, welcher aus der User Story 1 abgeleitet wurde.

|  |  |
| --- | --- |
| **Name** | Spiel gegen Netzwerkgegenspieler |
| **Beschreibung** | Netzwerkgegenspieler werden gesucht und danach wird gegen einen dieser Gegenspieler Battleship gespielt. |
| **Beteiligte Akteure** | User (Player A)  Netzwerkgegenspieler (Player B) |
| **Auslöser** | Zwei Spieler wollen gegeneinander Battleship spielen. |
| **Vorbedingungen** | Beide Spieler haben die Applikation gestartet  Die beiden Spieler befinden sich im selben Netzwerk |
| **Ergebnis** | Einer der beiden Spieler gewinnt das Spiel. |
| **Standardablauf** | 1. Mögliche Gegenspieler werden im Netzwerk gesucht 2. Ein möglicher Gegenspieler wird selektiert, damit Battleship gespielt werden kann. 3. Wenn der Gegenspieler akzeptiert, wird ein neues Spiel erzeugt. 4. Beide Spieler setzen ihre Schiffe auf dem eigenen Spielfeld. Dabei gelten folgende Regeln:  * Jedes Schiff besetzt eine bestimmte Anzahl von horizontal oder vertikal aufeinanderfolgenden Feldern. * Die Schiffe können sich nicht überlappen * Beide Spieler haben dieselben Schiffe  1. Sobald alle Schiffe gesetzt sind kann der Spieler sein eigenes Spielfeld als bereit markieren. Ab diesem Zeitpunkt können die Schiffe nicht mehr bewegt werden. 2. Sobald beide Spieler bereit sind beginnt das eigentliche Spiel. 3. Der Spieler der an der Reihe ist, attackiert ein Feld des Gegners. Bei einem Angriff gelten folgende Regeln:  * Der Gegner signalisiert ob sich auf diesem Feld ein Schiff befindet (Treffer) oder ob es Meer ist (Fehlschuss). * Bei einem Treffer darf der attackierende Spieler nochmals schiessen. * Bei einem Fehlschuss ist der andere Spieler dran. * Sobald alle Felder eines Schiffs getroffen worden sind, gilt dieses als versenkt. * Der attackierte Spieler teilt bei einem Treffer mit, ob sein Schiff versenkt ist.  1. Punkt 7 wird solange wiederholt, bis bei einem Spieler alle Schiffe versenkt wurden. Dann ist vorbei und der Spieler, welcher noch Schiffe besitzt, hat gewonnen. |
| **Exceptions**  **(Spielabruch)** | Beide Spieler selektieren sich gleichzeitig für ein Spiel.  Die Verbindung zwischen zwei Spielern wird beendet.  Einer der Spieler beendet die Applikation. |

# Systemspezifikation

Für die Anwendung wird das **Model-View-Controller** **Muster** verwendet:

* View: GUI
* Controller: Engine und Opponent
* Model: Grid

Die Komponente Opponent ist ein Stellvertreter (Proxy) für einen Gegenspieler. Alle Kommandos die an diese Komponente gesendet werden, werden weitergeleitet und via Netzwerk von einem Netzwerkgegenspieler oder von dem Computergegenspieler ausgeführt.

Die einzelnen Komponenten können auf der folgenden Abbildung betrachtet werden:

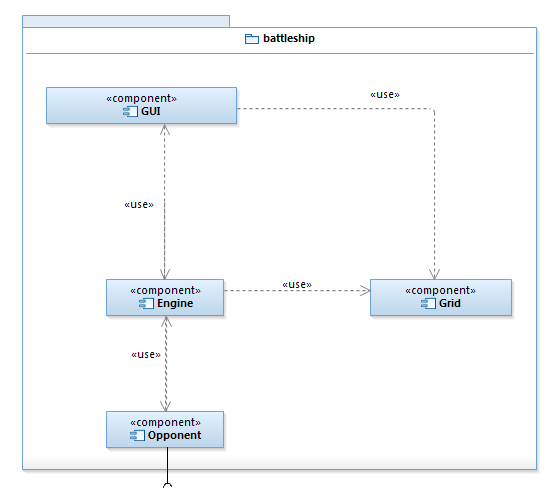


Abbildung : Die vier Komponenten der Battleship Application

Zustände

Währen der Laufzeit der Applikation treten vier Zustände ein:

* Gegenspieler auswählen (SelectingOpponent)
* Spielfeld vorbereiten (PreparingGrid)
* Spielen (Play)
* Spiel beendet (Finished)

Diese 4 Zustände sind im folgenden Diagramm dargestellt. Was in den einzelnen Zuständen passiert, ist im Folgenden beschrieben. Weiter existiert für jeden Zustand auch noch ein Sequenzdiagram.

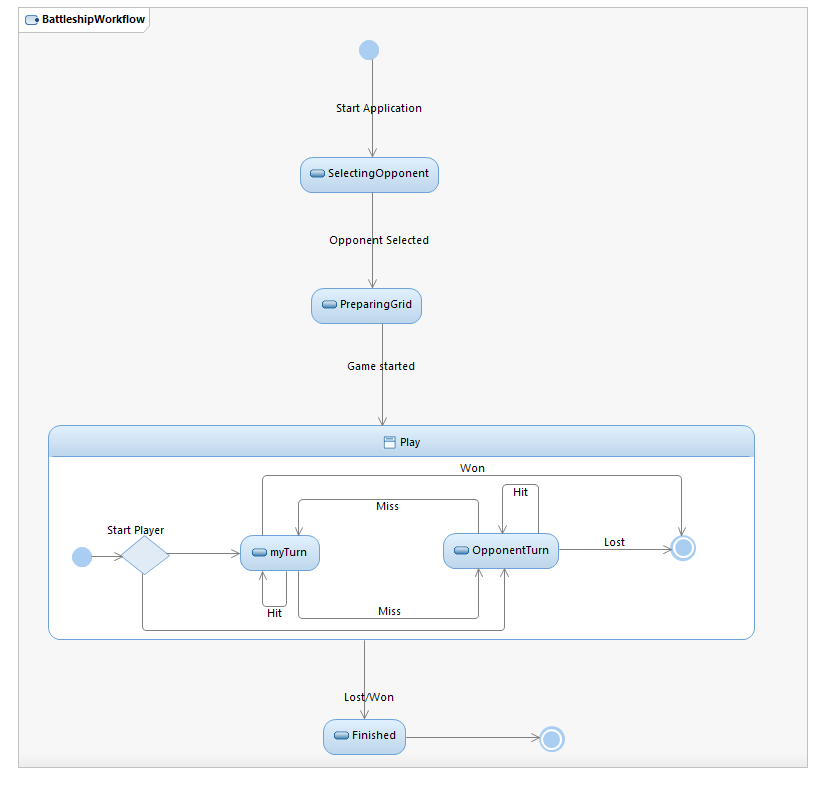


Abbildung : Statemaschine der Anwendung

### Selecting Opponent

Bei der Option „Play vs AI“ wird lokal gegen eine künstliche Intelligenz gespielt

Wählt man „Create Server“ wird ein Server kreiert. Ein anderer Spieler kann sich dann mit „Connect to Network Player“ mit dem Server verbinden, in dem er die IP des Spielers eingibt, der den Server erstellt.

Ein Spieler verbindet sich dabei mit der Server Engine indem er eine Register-Message schickt. Anhand dieser Message entscheidet die Engine ob der Spieler zugelassen ist oder nicht. Die Engine lässt momentan 2 Spieler zu. Falls zwei Spieler erfolgreich registriert wurden, wechselt die Engine in den Status PreparingGrid und benachrichtigt auch die Players

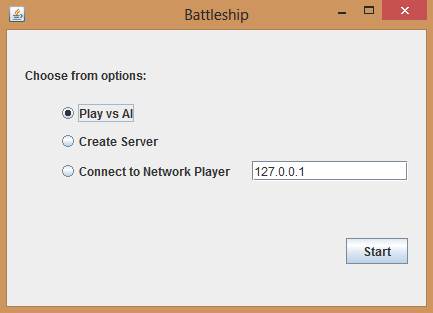


Abbildung 5: GUI – SelectingOpponent

### PreparingGrid

Beide Spieler setzen ihre Schiffe auf ihrem Grid. Sobald ein Spieler alle Schiffe gesetzt hat, wird signalisiert, dass der Spieler bereit ist. Ab diesem Zeitpunkt, dürfen die Positionen der Schiffe nicht mehr verändert werden. Sobald beide Spieler bereit sind beginnt das eigentliche Spiel.

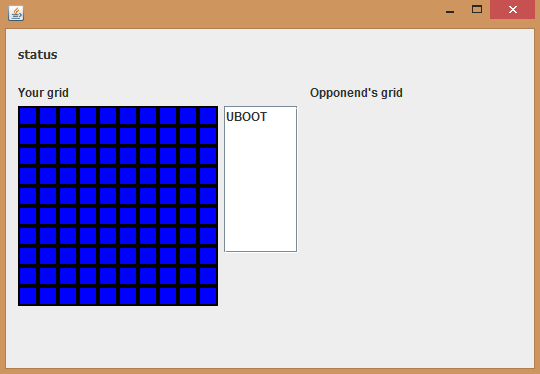


Abbildung 6: GUI – PreparingGrid (PlacingShips)

### Play

### Die Engine benachrichtigt die Player wenn das Spiel beginnt. Sie sendet den Event Play. Das heisst, beide Spieler haben ihre Schiffe plaziert.

### Die Engine signalisiert weiter über den Event Your\_Turn und Opponents\_Turn welcher Spieler an der Reihe ist.

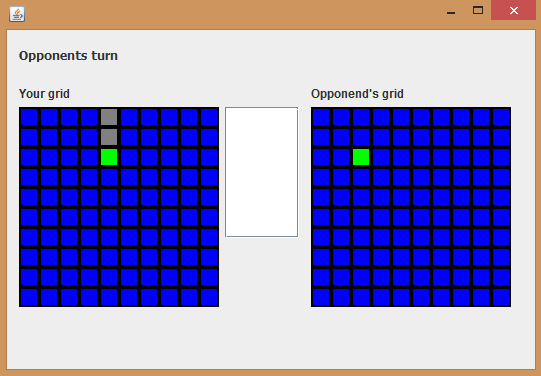


Abbildung 7: GUI – play:opponentTurn

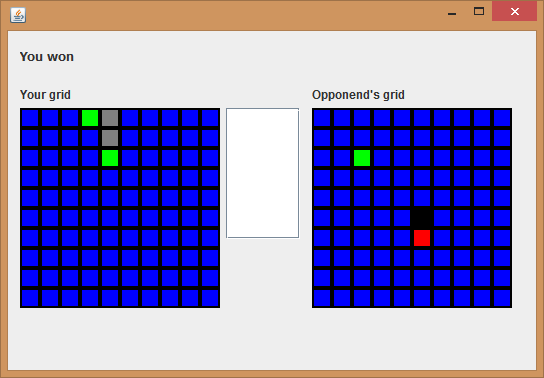


Abbildung 8: GUI – youWon

### Sequenzdiagramme

Die Abläufe in den Zuständen werden in Sequenzdiagrammen dargestellt. Diese zeigen die Interaktionen zwischen den Komponenten.

SelectingOpponent

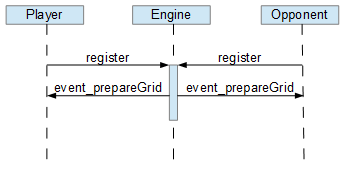


Abbildung 9: Sequenzdiagramm SelectingOpponent

PreparingGrid

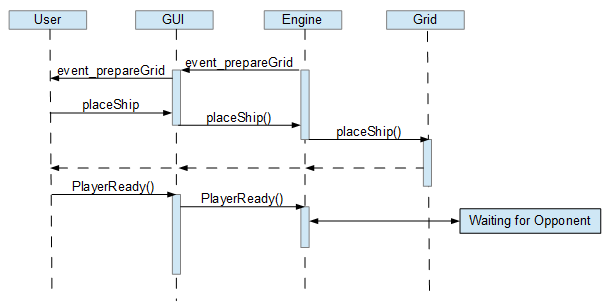


Abbildung 10: Sequenzdiagramm PreparingGrid

Play

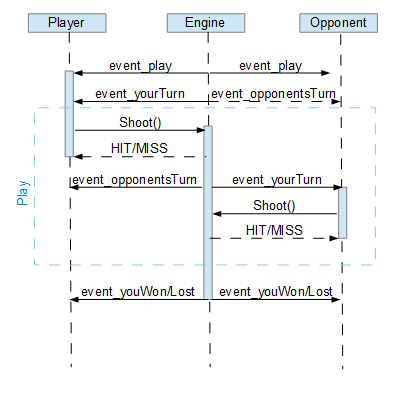


Abbildung 11: Sequenzdiagramm Play

Funktionale Sicht

Die Engine wird als Server gestartet und wartet bis sich zwei Player registrieren.

Zwischen den Komponenten wird über ein Protokoll kommuniziert.

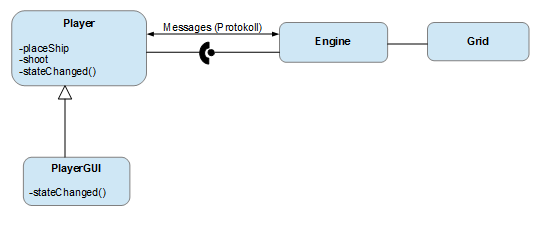


Abbildung 12: Funktion

### Message Protokoll:

Protokoll:

|  |  |  |  |
| --- | --- | --- | --- |
| Magic | PID | MSG Type | Payload length |
| 16 Bit | 16 Bit | 16 Bit | 16 Bit |

Header

|  |  |  |
| --- | --- | --- |
| Header | length | Name[] |
| 8 Byte | 8 Bit | 8 Bit |

Register

|  |  |
| --- | --- |
| Header | state |
| 8 Byte | 8 Bit |

Event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Header | Ship Type | x | y | vertical |
| 8 Byte | 8 Bit | 8 Bit | 8 Bit | 8 Bit |

PlaceShip

|  |  |
| --- | --- |
| Header | ShootState |
| 8 Byte | 8 Bit |

PlaceShip\_Response

|  |  |  |
| --- | --- | --- |
| Header | x | y |
| 8 Byte | 8 Bit | 8 Bit |

Shoot

|  |  |
| --- | --- |
| Header | ShootState |
| 8 Byte | 8 Bit |

Shoot\_Response

Verteilungs- und Betriebssicht

Es muss die JVM 1.7 installiert sein. Die Kommunikation erfolgt über das TCP/IP Protokoll. Der Port lautet 5001.

Datensicht

Die Daten werden auf dem Server gehalten. Die Engine ist für die Daten zuständig. Die Player wissen nicht wo der Gegner die Schiffe platziert hat.

Schnittstellen

Die Schnittstelle Player implementiert den TCP-Client und kann mit der Engine kommunizieren.

Um ein GUI zu implementieren wird also einfach von der Schnittstelle Player abgeleitet.

Kommunikation über Netzwerk

Siehe Funktionale Sicht

Erweiterungen

Die Engine sollte eine Message-Queue haben und nicht einfach Threads für jede ankommende Message starten. Weiter sehen wir das Battleship-Spiel nicht als eine fertige Lösung, sondern vielmehr als ein Prototyp.