Software Engineering 1

Abgabedokument Teilaufgabe 1
(Anforderungsanalyse und Planungsphase)

Persönliche Daten, bitte vollständig ausfüllen:

Nachname, Vorname:	Simkina Margaryta
Matrikelnummer:	01446530
E-Mail-Adresse:	a01446530@unet.univie.ac.at
Datum:	02.11.18

Aufgabe 1: Anforderungsanalyse (2 Punkte)

Analysieren der Spielidee und des Netzwerkprotokolls um 10 Anforderungen (bestehend zumindest aus 3 funktionalen, 3 nichtfunktionalen und einer Designbedingung) nach den folgenden Kriterien zu dokumentieren.

Anforderung 1: Jeder Zug des Spiels soll als Text in Ausgabe angezeigt werden.

Die Züge sollen in Ausgabe in Console angezeigt werden. Daten sollen in grafische Öberfläche vorgestellt werden. Diese soll Spieler und Richtung des Zugs bzw Spielstatus von dem Spiel anzeigen, beispielweise „Der Spieler 1 ist Links gegangen und befindet sich in der Position (3,5). Der Schatz wurde noch nicht gefunden“.

Auskunft: „Hierbei verfolgen die menschlichen Spieler das eigentliche Spielgeschehen ähnlich eines Livestreams nur passiv“ – Spielidee. Und auch meine persönliche Gedächtnisse, sodass menschlicher Spieler kann immer Spielstatus folgen, sogar wenn User interface(GUI) nicht ganz implementiert wurde.

Typ der Anforderung: funktional

Anforderung 2: Die Registrierung des Spielers zu dem Spiel soll durch die Angabe von Name, Last Name und Matrikelnummer realisiert werden.

Um der Benutzer zu identifizieren, es wird Basisinformation angefordert.

Auskunft: „Hierbei ist der Teil <SpielID> mit der während der Spielerzeugung erhaltenen eindeutigen SpielID zu ersetzen. Als Body der Nachricht werden Informationen zum Entwickler des Clients übertragen. Dies wären der Vorname (studentFirstName), der Nachname (studentLastName) und die Matrikelnummer (studentID)“ – Netzwekprotokol.

Anforderung 3: Das Spiel läuft automatisch durch selbstläufende KIs, damit der menschliche Spieler das Spiel als Livestream passiv anschauen kann.

Menschlicher Spieler nimmt in dem Spiel kein teil, er nur beobachtet das Spiel von der Seite.

Auskunft: „Bei einem solchen selbstablaufenden Spiel werden alle Aktionen automatisch durch künstlichen Intelligenzen (kurz KIs) berechnet und ausgeführt“ – Spielidee.

Typ der Anforderung: funktional

Anforderung 4: Es soll zu mindestens 3 Bergfeldern, 4 Wasserfelde und 15 Wiesenfelder pro Kartenhalbe erzeugt werden.

Diese Regel gilt damit die Karte nicht nur von Wiesen besteht, sodass es zu langweilig wäre oder meistens von dem Wasser sodass der Spieler nicht so viel Auswahl von Richtungen hätte.

Auskunft: „Jede Kartenhälfte muss mindestens 3 Bergfelder, 15 Wiesenfelder und 4 Wasserfelder beinhalten“ – Spielidee.

Typ der Anforderung: funktional

Anforderung 5: Es wird genau ein Schatz und eine Burg auf jeden Halbe der Karte erzeugt, damit das Spiel zwischen 2 KIs fair aufgeteilt wurde.

Auf jeder Kartenhalbe befindet sich genau ein Schatz, dass von dem menschlichen Spieler versteckt wurde.

Auskunft: „In Schatz ist hierbei jeweils der KI zugeordnet, welche auf dieser Kartenhälfte startet“ – Spielidee.

Typ der Anforderung: funktional

Anforderung 6: Das Spiel soll nicht mehr als 200 Spielaktionen beinhalten.

Das Spiel soll 200 oder weniger Spielaktionen dauern. Wenn 200 erreicht wurde, verliert KI automatisch, damit das Spiel zu lange nicht dauert.

Auskunft: „Wurde festgelegt, dass ein Spiel nicht länger als 200 Spielaktionen dauern darf und ein KI für jede Aktion (Spielerbewegung, Kartengenerierung etc.) nicht mehr als 3 Sekunden Bedenkzeit zur Berechnung erhält“ – Spielidee.

1. **Jeder Zug muss nicht mehr als 3 Sekunden dauern, anders verliert KI.**

Angenommen, dass das jeder Zug des Spielers soll innerhalb von 3 Sekunden genommen werden.

Auskunft: „Wurde festgelegt, dass ein Spiel nicht länger als 200 Spielaktionen dauern darf und ein KI für jede Aktion (Spielerbewegung, Kartengenerierung

etc.) nicht mehr als 3 Sekunden Bedenkzeit zur Berechnung erhält“ – Spielidee.

Typ der Anforderung: nicht - funktional

2. **Die gegnerische Schätze sind von Anfang zur KI nicht erkannt, um das Spiel spannend und sinnvoll zu machen.**

Die Schätze und die Burgen, die generiert wurden. Werden aber nicht auf der Karte angezeigt, um Spiel spannende zu machen.

Auskunft: „Der Schatz einer KI wird immer auf der Kartenhälfte versteckt, welche von der jeweiligen KI erstellt wurde“ – Spielidee.

Typ der Anforderung: nicht-funktional

3. **Wenn KI das Feld mit dem Wasser erreicht hat, verliert er automatisch.**

   KI soll Wasser vermeiden, da ein Zug zu den Wasserfeld bedeutet, dass die KI verloren hat.

Auskunft: „Hierbei gilt, dass Wasser unter keinen Umständen betreten werden darf. Bewegt sich die Spielfigur einer KI in ein Wasserfeld, verliert die KI automatisch“ - Spielidee.

Typ der Anforderung: funktional

4. **Die Kartenhalbe soll regelngemäßig entstellt werden.**

Die Generierung der Karte soll von dem KI realisiert werden und alle Regeln sollen da beobachtet werden, anders verliert KI automatisch.

Auskunft: „Sollte eine der beiden von den KIs erstellten Kartenhälften gegen diese Bedingungen verstoßen, verliert die KI, welche die verstoßende Kartenhälfte generiert hat. Die andere KI gewinnt, wenn deren Kartenhälfte korrekt aufgebaut ist, ansonsten verliert diese ebenfalls“ – Spielidee.

Typ der Anforderung: nicht - funktional

5. **Ressourcen des Servers so optimal benutzt werden, dass er nicht mehr als 999 Spielen gleichzeitig läufen lasst.**

Die Ressourcen des Servers sind begrenzt und deswegen sollen die Anzahl von gleichzeitig läufende Spiele nicht großer als 999 Spiele sein.

Auskunft: „Um die Ressourcen des Servers zu schonen gilt die Einschränkung, dass maximal 999 Spiele parallel ausgeführt werden dürfen. Sobald die Zahl der gleichzeitig laufenden Spiele die 999 Spiele übersteigt werden ältere Spiele automatisch entfernt und mit den neueren danach erstellen Spielen ersetzt“ - Netzwerkprotokol.

Typ der Anforderung: nicht - funktional

6. **im Falle einer inkorrekten Anfrage Spieler soll eine zusätzlich retournierte Fehlermeldung kriegen.**

Inkorrekte Anfrage des Clients sollen als Fehlermeldung angezeigt werden.

Auskunft: „Darüber hinaus wird im Falle einer inkorrekten Anfrage typischerweise eine Fehlermeldung zusätzlich retourniert“ – Spielidee.

Typ der Anforderung: funktional

7. **Das Projekt soll in REST Architektur entwickelt werden.**

Auskunft: „Die technologische Basis des Nachrichtenaustauschs stellt eine Restschnittstelle dar, daher es wird das HTTP Protokoll verwendet sowie die zugehörigen Operationen GET und POST“ – Netzwerkprotokoll.

Typ der Anforderung: Designbedingung.

Aufgabe 2: Anforderungsdokumentation (3 Punkte)

Anforderung 1: Jeder Zug des Spiels soll als Text in Ausgabe angezeigt werden.

Dokumentation einer Beispielanforderung

• **Beschreibung und Priorität:** Ablauf des Spiels soll als Text in Ausgabe angezeigt werden. Jeder Zug soll auf dem Bildschirm für den Anwedner sichtbar sein. Diese Züge beeinhalten den Nummer des Spielers und die Richtung des Zugs.

Priorität: Hoch

• **Impuls/Ergebnis Listen:**
o Impuls: Der Anwender 1 drückt ClientApplication „Run application“ und hierbei sendet die Anfrage nach Server um das Spiel zu starten.
o Ergebnis: Der Server sendet die Nachricht mit entsprechenden ID zu den Anwender.

o Impuls: Der Anwender 1 fragt die Registrierung des Clients an. Im Anschluss an die Registrierung eines Spielers muss der Client eine Kartenhälfte generieren und an den Server übertragen.

Fall 1: Client hat die Karte entsprechend Regeln erstellt

o Ergebnis: Server validiert die Reegeln und sendet die ResponseEnvelope mit entsprechenden SpielerID zu den Anwender, state wird mit dem Wert Okay definiert.

Fall 2: Client hat die Karte NICHT entsprechend Regeln erstellt.

Ergebnis: im Fehlerfall sind exceptionName und exceptionMessage mit Details zum Fehler befüllt sowie das state Element mit dem Text Error definiert.

o Impuls: Der Anwender 1 teilt SpielID mit Anwender 2.
o Ergebnis: Client 2 sendet die Anfrage mit dem SpielID nach dem Server und VORHERIGES Schritt wurde durchgeführt. Im fehlerfreien Fall wird das Spiel gestartet.

• **Benutzergeschichten:**
o Als Anwender ich möchte das Spiel als LifeStream in Text in der Console beobachten
o Als Anwender möchte ich über den Sieg oder Failure berichtet werden.

• **Benutzerschnittstelle:**
o Einfache Darstellung im Form des Textangabe in Console.

• **Externe Schnittstellen:**
o Schnittstelle Libraries: Es wird einen util.Logger dafür benutzt.

Schnittstell UI-Framework : CLI für automatisierte Ausgabe jedes Zuges und des Spielablauf.

Anforderung 2: die Kartenhalbe soll regelngemäßig entstellt werden.

• **Beschreibung und Priorität:** Die beide Kartenhalbe sollen so vom Server generiert werden, dass sie alle Regeln der Spielidee entsprechen. Falls einige der Regeln nicht beachtet wurde, verliert KI automatisch.

Priorität: hoch

• **Impuls/Ergebnis Listen:**

o Impuls: Der Client 1 sendet nach Server um ein neues Spiel zu starten

o Ergebnis: Der Server sendet die Antwort mit entsprechenden ID zu den Client.
o Impuls: Der Anwender 2 startet das Diagramm Tool erstmalig.
o Ergebnis: Das System sendet die Nachricht mit entsprechenden ID zu den Anwende und startet das Spiel. Da wird das automatische Ablauf gezeigt

o Impuls: Einen von Anwendern gewinnt oder verliert das Spiel
o Ergebnis: Ein Ausgabetext wird über den Sieg berichten

•**Benutzergeschichten:**

o Als Anwender ich möchte, dass Client automatisch ein regelngemäßiges Spiel erstellt.
o Als Anwender ich möchte im Fehlerfall exceptionMessage mit Details zum Fehler kriegen.•**Benutzerschnittstelle:** [Geben Sie hier, falls möglich und sinnvoll, ein Beispiel für die zugehörige Benutzerschnittstelle an, um darzustellen welchen Einfluss diese Anforderung auf die CLI Benutzeroberfläche hat.]

•**Externe Schnittstellen:** o Einfache Darstellung im Form des Textangabe in Console.

o Schnittstelle Nachrichtaustausch : XML wurde für diese Nachricht benötigt. Dazu wird http – POST Anfrage benutzt, um die Daten vom Client nach Sender zu versenden. Server prüft und validiert die Daten.

o Schnittstell UI-Framework : CLI für die Exceptions Nachrichten und die Details von denen

Anforderung 3: Die Registrierung des Spielers zu dem Spiel soll durch die Angabe von Name, Last Name und Matrikelnummer realisiert werden.

• **Beschreibung und Priorität:** Registrierung des Spielers erreicht in der Nachricht, die Name, Last Name und Matrikelnummer beinhaltet. Da wir keinen Spieler speichern, die Namen sind unwichtig.

Priorität: Mittel (ohne Angabe den Daten werden die Daten in Kostruktor trotzdem gegeben)

• **Impuls/Ergebnis Listen:**
o Impuls: Der Client 1 sendet die Anfrage sich zu registrieren „ClientRegistration“ mit den persönlichen Daten vom Anwender.
o Ergebnis: Der Server sendet die Nachricht (ResponseEnvelope) mit entsprechenden SpielerID.

o Impuls: Der Client 1 sendet die Anfrage sich zu registrieren „ClientRegistration“ mit den fehlerhaften Daten vom Anwender.
o Ergebnis: Server validiert die Anfrage und für diesen Fall ist der Namen einer zugehörigen Fehlermeldung im exceptionName Element zu finden und ein Hinweistext über den aufgetretenen Fehler im Element exceptionMessage.

• **Benutzergeschichten:**
Als Anwender möchte ich eine Möglichkeit haben, mich zu dem Spiel zu registrieren

Als Anwender möchte ich über den fehlerhaften Fall in meinen Daten benachrichtigt werden.

• **Benutzerschnittstelle:**
Einfache Darstellung im Form des Textangabe in Console.

• **Externe Schnittstellen:**
Schnittstelle Nachrichtaustausch : XML wurde für diese Nachricht benötigt. Dazu wird http – POST Anfrage benutzt, um die Daten vom Client nach Sender zu versenden.

Aufgabe 3: Definition von Meilensteinen und Aufwandsabschätzung (Projektplanung) (3 Punkte)

• MS 1: Dokumentation erstellen 10.10.2018 – 01.11.2018

AP 1: Spielidee kennenlernen und verstehen , Netzwerkprotokol verstehen

Aufwand: 4 Stunden

AP 2: Anforderungsanalyse: die Anforderung formulieren und diese detailiert mit Schnittstellen und User-Stories beschreiben

Aufwand: 3,5 Stunden

AP 3: Die Klassen und Packages formulieren, die Beziehungen zwischen Client und Server skizzieren, um besser zu verstehen

Aufwand: 2 Stunden
AP 4: TDD Development von Packages, Classen und Methoden

Aufwand: 2,5 Stunden

AP 5: Klassendiagramme im Tool skizzieren, benötigte Methoden dazu entwickeln, um danach Beziehungen klar zu verstehen

AP 6: Von den Beziehungen zwischen Klassen einen Sequenzdiagramm Entwurf realisieren

• MS 2: Client Basis 05.11.2018 – 30.12.2018

Netzwerk:

AP 1: Netzwerkkomunikation Basis Nachrichten zu schreiben, auf dem Postman zu testen
Aufwand: 6 Stunden
AP 2: Client Application Struktur in Verbindung mit Rest Nachrichten
Aufwand: 4 Stunden

AP 3: XML Nachrichten für SpielID bzw PlayerRegistration

Aufwand: 2 Stunden

MAP:

AP 4: MapBase Logik der Karte erstellen und Schema Programmieren
Aufwand: 3 Stunden

AP 5: MVC strukturieren und implementieren

Aufwand: 4 Stunden

AP 6: Map Generation fortsetzten, Klasse MapBase noch entwickeln, Nachricht für Map und Response des Servers anzupassen

Aufwand: 5-7 Stunden

KI:

AP 7: Dijkstra Algorithm verstehen

Aufwand 2-3 Stunden

AP 8: KI Schritte und Algorithmen entwickeln

Aufwand: 3 Tagen

• MS 4: Client GUI und Optimization 30.11.2018-02.12.2018

AP 1: Karte erstellen - in GUI zeichnen Aufwand:

6 Stunden

AP 2: Unit-Tests für 50% Funktionalität hinzufügen , diese anprobieren

Aufwand: 3-4 Stunden

AP 3: Fehlerabhebung und Logging

Aufwand: 1 Stunde

AP 4: Entsprechend zeitliche und andere Anforderungen Spiel optimieren

Aufwand: 1 Stunde

AP 5: Documentation entsprechend den Veränderungen zu aktualisieren

Aufwand: bis 1 Stunde

• MS 5: Server 03.12.2018 – 15.12.2018

AP 1: Server Erzeugung lernen und mit Rest Anfragen anprobieren

Aufwand: 5 Stunden
AP 2: XML – Nachrichten entsprechend Netzwerkprotokoll entwickeln

Aufwand: 8-9 Stunden

AP 3: Fehlerabhebung beachten, Unit-Tests schreiben

Aufwand: 2 Stunden

AP 4: Logging hinzufügen

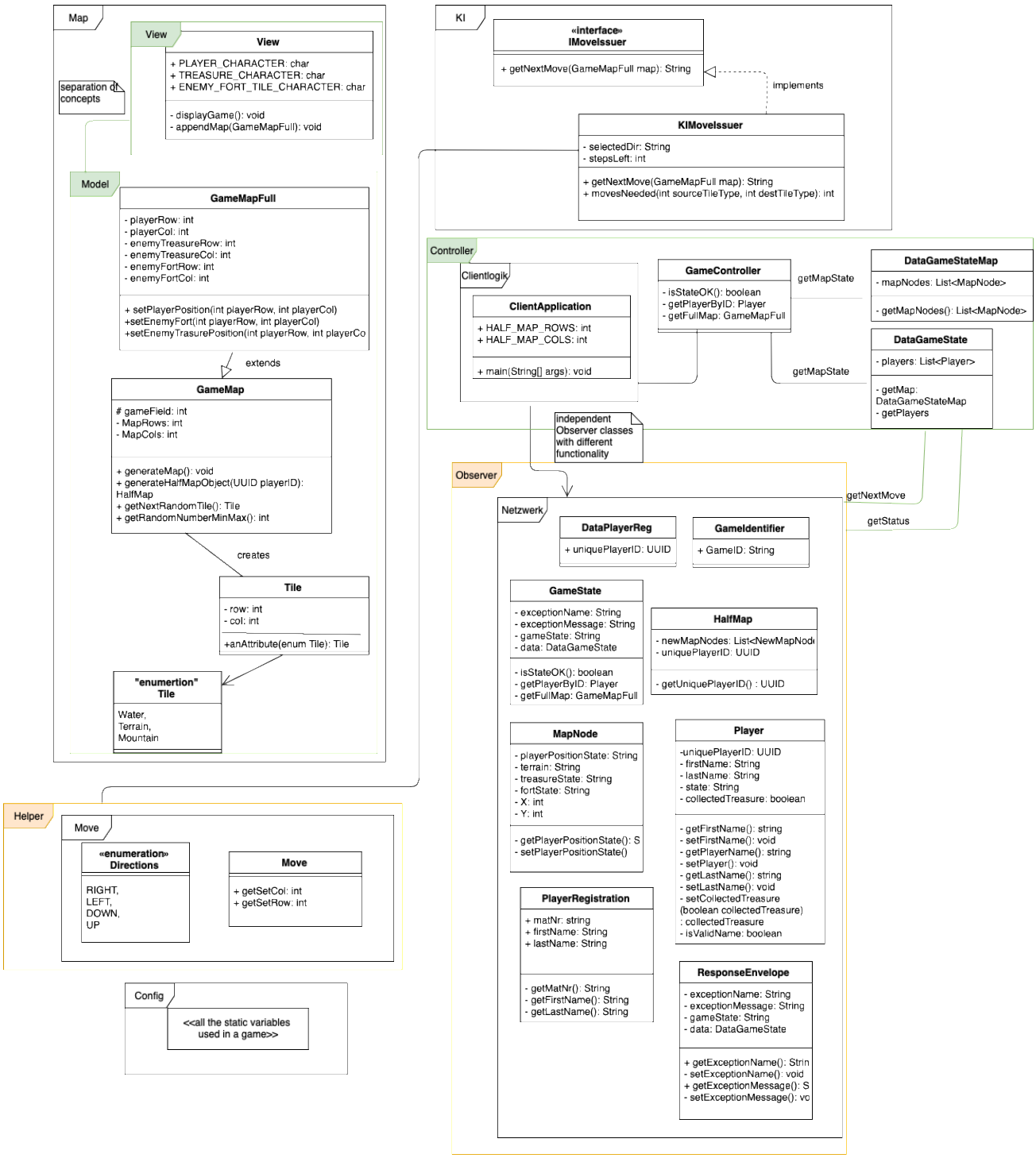
Aufwand: 1 Stunde

AP 5: Documentation entsprechend den Veränderungen zu aktualisieren

Aufwand: 2 Stunden

Aufgabe 4: Architektur entwerfen, modellieren und validieren

4.1 Class Diagram



4.2 Sequence Diagram