Universität Bremen
FB 3 – Informatik
Dr. Karsten Hölscher
TutorIn: Euer/Eure TutorIn

# Software-Projekt 2 – SoSe 2020

**VAK 03-BA-901.02**

# Architekturbeschreibung

Clara Maria Odinius odinius@uni-bremen.de
Habib Mergan habib1@uni-bremen.de
Kevin Santiago Rey Rodriguez kev_rey@uni-bremen.de
Liam Hurwitz hurwitz@uni-bremen.de
Mehmet Ali Baykara baykara@uni-bremen.de
Miguel Alejandro Caceres Pedraza mcaceres@uni-bremen.de

*Abgabe: TT. Monat JJJJ — Version 1.1*

# Inhaltsverzeichnis

# Version und Änderungsgeschichte

*Die aktuelle Versionsnummer des Dokumentes sollte eindeutig und gut zu identifizieren sein, hier und optimalerweise auf dem Titelblatt.*

| Version | Datum | Änderungen |
|---|---|---|
| 0.1 | TT.MM.JJJJ | Dokumentvorlage als initiale Fassung kopiert |
| 0.2 | TT.MM.JJJJ | ... |
| ... | | |

# 1 Einführung

## 1.1 Zweck

*Was ist der Zweck dieser Architekturbeschreibung? Wer sind die LeserInnen?*

## 1.2 Status

## 1.3 Definitionen, Akronyme und Abkürzungen

| Begriff | Definition |
|---|---|
| Software-Architektur | ist die grundlegende Organisation eines Systems verkörpert |
| Architektursicht (View) | Repräsentation eines ganzen Systems aus der Perspektive einer kohärenten Menge von Anliegen (IEEE P1471, 2002). |
| Anwendungsfälle | Spezifiziert eine beliebige Menge von Aktionen, die ein System ausführen muss, damit ein Resultat stattfindet, welches für mindestens einen Akteur von Bedeutung ist. |
| Framework | Programmiergerüst, welches den Rahmen der Anwendung bildet. Es umfasst Bibliotheken und Komponenten. |
| UML | Steht für Unified Modeling Language und ist eine grafische Modellierungssprache zur Spezifikation, Visualisierung, Konstruktion und Dokumentation von Modellen für Softwaresysteme. |
| IDE | integrierte Entwicklungsumgebung - Hilft bei der Bearbeitung von Projekten in der Softwareentwicklung |
| Versionskontrolle | Hochgeladenen Versionen werden festgehalten und können wieder hergestellt werden |
| Maven | Java Programme können standardisiert und verwaltet werden |
| Library | Sammlung von verschiedensten vorgefertigten Methoden oder Klassen |
| Interface | Schnittstelle |
| Multiplayer | Spielmodus, mit mehr als einem Spieler |

| | |
|---|---|
| Singleplayer | Spielmodus, mit einem Spieler, der gegen einen computergesteuerten Gegner spielt. |
| JUnit | Frameworke für Tests für die Programmiersprache Java |
| Javadoc | Software-Dokumentationswerkzeug für die Programmiersprache Java. |
| JavaEE | (Java Platform Enterprise Edition) Eine Spezifikation für die transaktionsbasierte Ausführung von in Java programmierten Anwendungen. |
| GUI | Steht für Graphical User Interface und kennzeichnet eine grafische Schnittstelle, über die ein Mensch mit einer Software interagiert. |
| Paketdiagramm | Strukturdiagramm der UML, stellt die Verbindung zwischen Paketen, Paketimports bzw. Verschmelzungen und deren Abhängigkeiten dar. |
| Problemkarte | Beschreiben Probleme im Zusammengang der Einflussfaktoren und stellen Lösungen bzw. entsprechende Strategien dar. |
| Sequenzdiagramm | Strukturdiagramm der UML, stellt den Austausch von Nachrichten zwischen Objekten mittels einer Lebenslinie dar. |
| Klassendiagramm | Strukturdiagramm der UML, stellt die statischen Strukturen eines Systems dar. |
| Komponentendiagramm | Strukturdiagramm der UML, stellt Komponenten und deren Schnittstellen dar. |
| HTTP | Steht für Hypertext Transfer Protocol, einem zustandslosen Protokoll zum synchronen Versenden von Informationen über Rechnernetze. |
| HTTPS | Durch Verschlüsselungstechniken gesichertes HTTP. |
| TCP | Das Transmission Control Protocol ist ein Netzwerkprotokoll, das definiert, auf welche Art und Weise Daten zwischen Netzwerkkomponenten ausgetauscht werden sollen. |
| UDP | Das User Datagram Protocol, kurz UDP, ist ein minimales, verbindungsloses Netzwerkprotokoll, das zur Transportschicht der Internetprotokollfamilie gehört. |

## 1.4 Referenzen

Vorlesungsfolien

## 1.5 Übersicht über das Dokument

# 2 Anwendungsfälle

## 2.1 Anmeldung

Ein User offen das spielt und dann kommt ein Fenster, wo der User sich mit password und name einloggen kann, wenn er die richtige Daten eintragt. dann das Spielt kann weiter geführt werden. Wenn die Daten nicht richtig sind, dann das User bekommt ein Hinweis, dass er wieder eintragen soll.

## 2.2 Multi-Spieler

Wenn das Users angemeldet ist, dann er sehen kann, welche der andere User auch verbindet sind und starten ein Spiel mit der anderen User oder gegen Computer.

## 2.3 Spielt-Starten

Im ersten Fenster des Spiels kann der Benutzer das Schiff, die Waffen, die er verwenden möchte, die Besatzung und den Schwierigkeitsgrad des Spiels auswählen. danach kann er anfangen.

## 2.4 Spielt-Verlauft

Wenn der Spieler das Spiel beginnt, kann er die Crew in den von ihm gewählten Positionen positionieren. Dann können Sie die Sprungtaste drücken, um die Zielplaneten auszuwählen. Der Benutzer kann dies immer in jeder Runde wiederholen.

Wenn der Benutzer einen Zielplaneten auswählt, kann er auf Gegner treffen und nach einem Dialogfeld mit ihnen kämpfen.

Sobald der Benutzer besiegt hat, kann der Gegner sein Schiff reparieren, seine Crew heilen oder einen Sprung machen.

## 2.5 Kampf-Regeln

Wenn der Benutzer in einen Kampf verwickelt ist, unterliegt dieser bestimmten Regeln.

- Wenn ein Kampf beginnt, kann der Benutzer das Ziel seiner Waffen auswählen, die die Abschnitte des gegnerischen Schiffes sind.
- Waffen brauchen Zeit zum Laden und Schießen.
- Damit eine Waffe das Ziel treffen kann, muss der Gegner Schilde deaktiviert haben.

- Um die Schilde zu deaktivieren, müssen die Waffen ihn treffen. Jedes Mal, wenn er einen Aufprall findet, verliert der Schild an Stärke. Sobald die Stärke des Schildes Null ist, wird der Schild deaktiviert.
- Waffen treffen nicht immer das Ziel.
- Damit eine Waffe verwendet werden kann, muss der Waffenbereich des Schiffes funktionsfähig sein.
- Jedes Mal, wenn eine Waffe ein Schiff trifft, verliert sie ihren Lebenspunkt.
- Sobald eines der beiden Schiffe keine Lebenspunkte mehr hat, verliert dieses Schiff.

## 2.6 Pausenmodus

Während eines Kampfes kann der Benutzer das Spiel in den Pausenmodus versetzen. Dies bedeutet, dass beide Schiffe aufhören zu schießen und die Besatzung aufhört, sich zu bewegen. Wenn das Spiel fortgesetzt wird, werden die Waffen wieder aktiviert und auch die Crew.

# 3 Globale Analyse

## 3.1 Einflussfaktoren

| Abgeleitet aus | Einflussfaktor | Flexibilität und Veränderlichkeit | ++/ - - | Auswirkungen |
|---|---|---|---|---|
| O1 : Organisation | | | | |
| O1.1 Time-To-Market | | | | |
| | Die Auslieferung erfolgt am 02.08.2020. | Keine Veränderlichkeit oder Flexibilität, da Vorgaben bestehen. | - -/ - - | Nicht alle Funktionen können realisiert werden bzw. implementiert werden. |
| O1.2 Architektur-Abgabe | | | | |
| | Die Auslieferung erfolgt am 31.05.2020. | Keine Veränderlichkeit oder Flexibilität, da Vorgaben bestehen. | - -/ - - | Durch den Zeitdruck könnte die Architektur mangelhaft werden. Wenn wir uns nicht genug Zeit lassen, könnten Aspekte, die relevant für die Architektur sind, vergessen werden. |
| O1.3 Entwickler | | | | |
| | Die Projektgruppe besteht aus 6 Entwicklern | Falls ein Entwickler (temporär) ausfällt kann ein anderer Entwickler einspringen. Keine neuen Entwickler können eingestellt werden. Gruppenmitglieder können wegfallen oder austreten | +/ - - | Die Architektur kann wegen Zeitmangel (es können nicht mehr als die ursprünglichen sechs Entwickler mitarbeiten) und fehlenden Fähigkeiten Mängel enthalten. Unvollständige Implementierung droht |
| O1.4 Fähigkeiten Entwickler | | | | |
| | Nicht alle Entwickler haben die gleiche Programmier-erfahrung | Hohe Veränderlichkeit und Flexibilität durch Ausführen des Projekts und Recherche. TODO! | ++/ ++ | Die Implementierung kann Mängel enthalten. |

| O1.5 Teamarbeit in Corona-Zeiten | | | | |
|---|---|---|---|---|
| | Das persönliche Treffen kann nicht stattfinden, aufgrund von Kontakt-beschränkung | Digitale Treffen über z.B. Discord. Keine Veränderlichkeit aufgrund der Gesetzeslage. | ++/ - - | Missverständnisse können öfter auftreten. Teamarbeit könnte erschwert werden und dadurch die Qualität der der Architektur beeinträchtigen |

| Abge-leitet aus | Einflussfaktor | Flexibilität und Veränderlichkeit | ++/ - - | Auswirkungen |
|---|---|---|---|---|
| T1: Technik | | | | |
| T1.1: Programmiersprache | | | | |
| | Java 8 oder höher ist vorgegeben. | Die Programmier-sprache wird vorgegeben. Wir können aber zwischen verschiedenen Versionen auswählen. | - -/ o | Das Architektur muss in Java umgesetzt werden. |
| T1.2 Betriebssystem | | | | |
| | Die Anwendung muss auf den gängigen Betriebssystemen (MacOS, Windows, Linux) laufen. | Keine Veränderlichkeit oder Flexibilität, da Mindestanforderungen bestehen. Aber wir können noch weitere Betriebssysteme hinzufügen | - -/ - - | Bei der Implementierung müssen die gängigen Betriebssysteme berücksichtigt werden. Wenn die Anwendung auf mehr Betriebssystemen laufen soll, dann muss mehr Zeit investiert werden. |
| T1.3 Client-Server-Architektur TODO! | | | | |
| | Zur Implementierung muss JavaEE 11 benutzt werden. | Keine Veränderlichkeit oder Flexibilität, da Teil der Mindestanforderungen. | - -/ - - | Das Projekt muss komplett in Java umgesetzt werden. |

| | | | | |
|---|---|---|---|---|
| T1.4: Framework libgdx TODO! | | | | |
| | Als Framework zur Erstellung der Oberfläche muss JSF verwendet werden. | Keine Veränderlichkeit oder Flexibilität, da Teil der Mindestanforderungen. | - -/<br>- - | Das Projekt muss in Java umgesetzt werden. |
| T1.5: Persistenz TODO! | | | | |
| | Zur sicheren Speicherung der Daten soll die relationalen Datenbank, leichtgewichtige verwendet werden. z.B (H2, SQLite, Derby) | Keine Veränderlichkeit oder Flexibilität, da Teil der Mindestanforderungen. | - -/<br>- - | Bei der Implementierung muss eine leichtgewichtige, relationale Datenbank (ohne Installation verwendbar) verwendet werden. |
| T1.6: Build System TODO! | | | | |
| | Maven oder Gradle müssen als Build-System verwendet werden. | Auswahl zwischen Maven und Gradle möglich. Keine Veränderlichkeit möglich, da Mindestanforderungen bestehen. | +/<br>- - | Das Projekt muss Maven/Gradle-build fähig sein. |

| Abgeleitet aus | Einflussfaktor | Flexibilität und Veränderlichkeit | ++/ - - | Auswirkungen |
|---|---|---|---|---|
| T1.8: Multiplayer TODO! | | | | |
| | Der Server muss einen Multiplayer Modus ermöglichen. | Keine Flexibilität oder Veränderlichkeit, da Mindestanforderungen bestehen. Wir können es aber ermöglichen, dass mehr als zwei Spieler gleichzeitig spielen. | - -/ - - | Die Anwendung darf nicht von gleichzeitiger Verwendung von zwei Nutzern überfordert sein. Trotzdem müssen mindestens 2 Spieler gleichzeitig das Spiel spielen können, um die Mindest-anforderungen zu erfüllen. |
| P1: Produktfaktoren | | | | |
| P1.1.1: Pause TODO! | | | | |
| | Die Anwendung muss in der Lage sein, ein zuvor von einem bestimmten Benutzer gestartetes Spiel zu speichern und dem Benutzer die Möglichkeit zu geben, es zu einem anderen Zeit-punkt fortzu-setzen. | Keine Flexibilität und keine Veränderlichkeit, da Mindestanforderungen bestehen. | - -/ ++ | Der Server muss es ermöglichen, dass das Spiel unterbrochen werden kann |
| P1.8: Schwierigkeitsstufen TODO! | | | | |
| | Für Benutzer müssen mindestens zwei unter-schiedliche Schwierig-keitsgrade implementiert werden. | Keine Veränderlichkeit,da Mindestanforderungen bestehen.Es können aber mehr als zwei Schwierigkeitsgrade implementiert werden. Außer der leichtesten Stufe können wir den Schwierigkeitsgrad der anderen Stufe beliebig auswählen. | - -/ - - | Das Spiel soll in mindestens zwei verschiedenen Schwierigkeits-stufen gespielt werden können |

| Abgeleitet aus | Einflussfaktor | Flexibilität und Veränderlichkeit | ++/ - - | Auswirkungen |
|---|---|---|---|---|
| P1.2: Struktur und Eingeschäften des Raumschiff | | | | |
| P1.2.1: Aufteilung in Sektionen | | | | |
| | Pro Sektion gibt es die relevanten Systeme: Antrieb, Waffen, Schutzschild | Keine Flexibilität oder Veränderlichkeit, da, Mindestanforderungen bestehen. | - -/ - | Die Sektionen in den Raumschiffen müssen die relevanten Systeme, Antrieb,Waffen,Schutzschild, beinhalten |
| P1.2.2: Schäden der Sektionen | | | | |
| | Jede Sektion kann beschädigt werden, wodurch das von ihm gesteuerte System beschädigt wird. | Keine Flexibilität oder Veränderlichkeit, da Mindestanforderungen bestehen. | - -/ - - | Spieler können gegnerische Raumschiffe beschädigen und das eigene Raumschiff reparieren. |
| P1.3: Ressourcen des Spiels | | | | |
| | Ressourcen müssen implementiert werden, um die Spiellogik auszuführen, zum Beispiel: Geld Energie Hüllenintegrität | Keine Veränderlichkeit, da Mindestanforderungen bestehen. Wir können aber verschiedene Spielressourcen verwenden. | - -/ - - | Die Spielressourcen werden gebraucht um das Spiel zu spielen |
| P1.3: Raumschiff Eigenschaften | | | | |
| | Eigenschaften können durch Geld verbessert werden. | Keine Veränderlichkeit oder Flexibilität, da Teil der Mindestanforderungen. | - -/ - - | Die Spiellogik sollte dem Spieler erlauben die Eigenschaften des Raumschiffs zu verbessern. |

| Abgeleitet aus | Einflussfaktor | Flexibilität und Veränderlichkeit | ++/ - - | Auswirkungen |
|---|---|---|---|---|
| P1.2: Besatzung | | | | |
| P1.2.1: Raumschiff hat Besatzung TODO! | | | | |
| | Der Spieler kann die Besatzung auf die Sektionen verteilen | Keine Veränderlichkeit, da Mindestanforderungen bestehen.Wir können aber die Spiellogik so implemenetieren, sodass nur ein Besatzungs-mitglied sich zur selben Zeit in einer Sektion aufhalten kann. | - -/ - | Die Interaktion der Besatzung mit verschiedenen Einheiten der Anwendung kann die Durchführung des Projekts erschweren |
| P1.2.2: Besatzung Eigenschaften TODO! | | | | |
| | Die Besatzung des Schiffes kann Systeme/ Sektionen reparieren, kann sterben kann eingestellt/ angeheuert werden,hat Fähigkeiten | Keine Veränderlichkeit, da Teil der Mindestanforderungen. Aber wir können noch weitere Fähigkeiten implementieren | - -/ - - | Dieser Faktor kann die Komplexität des Spiels beeinflussen und die Entwicklung der Anwendung verlangsamen. |
| P1.3: Besatzung Fähigkeiten | | | | |
| | Die Fähigkeiten der Besatzung können verbessert werden und beeinflussen die Systeme des Schiffes, auf dem sie arbeiten. | Keine Veränderlichkeit oder Flexibilität, da Teil der Mindestanforderungen. | - -/ - - | Die Implementierungslogik dieser Ressourcen und ihre Beziehung zu anderen Entitäten können Implementierungsprobleme und Probleme bei der Bereitstellung verursachen. |

| Abgeleitet aus | Einflussfaktor | Flexibilität und Veränderlichkeit | ++/ - - | Auswirkungen |
|---|---|---|---|---|
| P1.2: Universum | | | | |
| P1.2.1: Struktur des Universums | | | | |
| | Das Universum muss Stationen und Planeten haben, die vom Raumschiff des Spielers durchquert werden. | Keine Veränderlichkeit oder Flexibilität, da Teil der Mindestanforderungen. | - -/ - | Die Architektur muss ermöglichen, dass sich Raumschiffe im Universum auf Stationen und Planeten fliegen können. |
| P1.2.2: Station/Planet Eigenschaften | | | | |
| | Jede Station oder jeder Planet muss Aktionen haben, die das Spiel negativ oder positiv beeinflussen müssen. | Keine Veränderlichkeit oder Flexibilität, da Teil der Mindestanforderungen. | - -/ - - | Die Architektur muss vorsehen, dass Aktionen auf Stationen und Planeten Einfluss auf den Spielverlauf haben |
| P1.3: Feindliche Schiffe TODO!!!! | | | | |
| | Im Universum müssen mindestens drei verschiedene Raumschiffe geben, die unterschiedliche Anzahl an Sektionen und Layout,unterschiedliche Eigenschaften und unterschiedliche Anzahl an Besatzungsmitgliedern haben. | Wir können mehr als drei verschiedene Raumschiffe implementieren. Keine Veränderlichkeit, da Mindestanforderungen, bestehen. | ++/ - - | Die Architektur muss mindestens drei verschiedene Raumschiffe vorsehen um die Mindestanforderungen zu erfüllen |

| Abgeleitet aus | Einflussfaktor | Flexibilität und Veränderlichkeit | ++/ - - | Auswirkungen |
|---|---|---|---|---|
| P1.2: Struktur des Kampfes | | | | |
| P1.2.1: taktische Entscheidungen | | | | |
| | Der Kampf mit einem gegnerischen Raumschiff erfolgt rundenbasiert. | Keine Flexibilität oder Veränderlichkeit, da Mindestanforderungen bestehen. | - -/ - - | Die Architektur muss ermöglichen, dass nur rundenbasierte Kämpfe stattfinden. |
| P1.2.2: Waffenverhalten | | | | |
| | Abfeuern einer bestimmten Waffe auf eine bestimmte Sektion des Gegners. | Keine Flexibilität oder Veränderlichkeit, da Mindestanforderungen bestehen. | - -/ - - | Die Architektur muss ermöglichen, dass Sektionen gegnerischer Raumschiffe angegriffen werden können. |
| P1.3: Verteilen von Besatzungsmitgliedern | | | | |
| | Während eines Kampfes kann die Position der Besatzung auf dem Schiff verändert werden. Der Sektionswechsel wird einige Zeit in Anspruch nehmen. | Die Besatzung kann die Sektionen wechseln, aber muss nicht. Jedoch kann die Zeit die, in Anspruch genommen wird flexibel ausgewählt werden von uns. Keine Veränderlichkeit,da Mindestanforderungen bestehen. | - -/ - - | Die Architektur muss ermöglichen, dass Besatzungsmitglieder die Sektionen wechseln können. |

## 3.2 Probleme und Strategien

| Problem 1: Gruppenmitglieder fallen weg |
|---|
| **Beschreibung:**<br>Während des Projekts kann es dazu kommen, dass Gruppenmitglieder die Gruppe verlassen müssen oder eigenständig verlassen oder das Gruppenmitglieder temporär ausfallen. Das würde dazu führen, dass der Umfang des Projekts/Mindestanforderungen angepasst werden würden. |
| **Einflussfaktoren:**<br>O1.1: Time-To-Market<br>O1.2: Architektur-Abgabe<br>O1.3: Entwickler |
| **Strategien:**<br>S-1: Reorganisation der Gruppe<br>S-2: Umverteilung der Aufgaben, die der entfallene Gruppenmitglied hatte<br>S-3: Mit dem Tutor sprechen und ggf. die Mindestanforderungen anpassen lassen |
| **Entscheidung:** S-1 |

| Problem 2: Zeitliche Unterschätzung des Aufwands |
|---|
| **Beschreibung:**<br>Aufgrund von fehlender Erfahrung kann es dazu kommen, dass wir die Aufgaben und deren zeitlichen Aufwand unterschätzen. Das kann dazu führen, dass wir die Abgabetermine nicht einhalten können. |
| **Einflussfaktoren:**<br>O1.1: Time-To-Market<br>O1.2: Architektur-Abgabe<br>O1.4: Fähigkeiten Entwickler |
| **Strategien:**<br>S-1: So früh wie möglich anfangen, die Aufgaben zu bearbeiten<br>S-2: Selbst gestellte Deadlines einhalten<br>S-3: Fokus auf Mindestanforderungen und falls Zeit übrig bleibt, dann erst Features bearbeiten |
| **Entscheidung:** S-1 |

| Problem 3: Covid-19 |
|---|
| **Beschreibung:**<br>In Zeiten der Corona-Krise können keine persönlichen Treffen stattfinden, aufgrund des Kontaktverbots und somit entfallen Teambuildung-Events und persönliche Treffen mit dem Tutor/Kunden. |
| **Einflussfaktoren:**<br>O1.5: Teamarbeit in Corona-Zeiten |
| **Strategien:**<br>S-1: Regelmäßige digitale Treffen mit Gruppenmitgliedern und dem Tutor<br>S-2: Regelmäßig den Fortschritt unserer Arbeiten dem Tutor zeigen<br>S-3: Videokonferenz zum Kennenlernen als Teambuildung Maßnahme |
| **Entscheidung:** S-1 |

| Problem 3: Missverständnisse von Mindestanforderungen |
|---|
| **Beschreibung:**<br>Die Gruppe entwickelt eine falsche Anforderung, aufgrund von Missverständnissen. |
| **Einflussfaktoren:**<br>O1.1: Time-To-Market<br>P1 : Produktfaktoren |
| **Strategien:**<br>S-1: Evolutionäre Prototypen entwickeln<br>S-2: Inkrementelle Auslieferungen des Produkts<br>S-3: Verschiedene Akzeptanztests durchführen |
| **Entscheidung:** S-1 |

| Problem 4: Fehlende Erfahrung mit den Technologien |
|---|
| **Beschreibung:**<br>Es kann sein, dass ein Teil oder die ganze Gruppe keine Erfahrungen in der Arbeit mit den vorgegebenen Technologien besitzen und somit in Verzug kommen mit den Abgabeterminen. Das kann zu mangelhaften Implementierungen führen. |
| **Einflussfaktoren:**<br>O1.1: Time-To-Market<br>O1.4: Fähigkeiten Entwickler<br>T1 : Technik |
| **Strategien:**<br>S-1: Einzelne Gruppenmitglieder fokussieren sich auf einzelne Technologien und schaffen sich Expertise und geben das Wissen weiter an andere Gruppenmitglieder<br>S-2: Vorzeitig anfangen und in die Technologien einarbeiten mithilfe von Kursen, Tutorials und Videos |
| **Entscheidung:** S-2 |

| Problem 5: Persistenz der Daten |
|---|
| **Beschreibung:**<br>Es muss eine Persistenz der Daten gewährleistet werden, damit Daten nicht verloren gehen und Spieler können ihre Spiele fortsetzen. |
| **Einflussfaktoren:**<br>T1.5: Persistenz<br>P1.1.1 : Pause |
| **Strategien:**<br>S-1: H2 relationale Datenbank<br>S-2: SQLite relationale Datenbank<br>S-3: Derby relationale Datenbank |
| **Entscheidung:** S-1 |

| Problem 6: Überlastung des Servers |
|---|
| **Beschreibung:**<br>Der Server muss ermöglichen, dass mindestens zwei verschiedene Spieler das Spiel gleichzeitig spielen können. |
| **Einflussfaktoren:**<br>T1.3: Client-Server-Architektur<br>T1.8 : Multiplayer |
| **Strategien:**<br>S-1: Teilung der Anwendung in eine Client-Server-Architektur<br>S-2: Client und Server sind enthalten im selben Programm<br>S-3: Multithreaded Server implementieren |
| **Entscheidung:** S-1 |

| Problem 7: |
|---|
| **Beschreibung:**<br>Der Server muss ermöglichen, dass mindestens zwei verschiedene Spieler das Spiel gleichzeitig spielen können. |
| **Einflussfaktoren:**<br>T1.3: Client-Server-Architektur<br>T1.8 : Multiplayer |
| **Strategien:**<br>S-1: Teilung der Anwendung in eine Client-Server-Architektur<br>S-2: Client und Server sind enthalten im selben Programm |
| **Entscheidung:** S-1 |

# 4 Konzeptionelle Sicht

*Diese Sicht beschreibt das System auf einer hohen Abstraktionsebene, d. h. mit sehr starkem Bezug zur Anwendungsdomäne und den geforderten Produktfunktionen und -attributen. Sie legt die Grobstruktur fest, ohne gleich in die Details von spezifischen Technologien abzugleiten. Sie wird in den nachfolgenden Sichten konkretisiert und verfeinert. Die konzeptionelle Sicht wird mit UML-Komponentendiagrammen visualisiert.*

# 5 Modulsicht

*Diese Sicht beschreibt den statischen Aufbau des Systems mit Hilfe von Modulen, Subsystemen, Schichten und Schnittstellen. Diese Sicht ist hierarchisch, d. h. Module werden in Teilmodule zerlegt. Die Zerlegung endet bei Modulen, die ein klar umrissenes Arbeitspaket für eine Person darstellen und in einer Kalenderwoche implementiert werden können. Die Modulbeschreibung der Blätter dieser Hierarchie muss genau genug und ausreichend sein, um das Modul implementieren zu können.*

*Die Modulsicht wird durch UML-Paket- und Klassendiagramme visualisiert.*

*Die Module werden durch ihre Schnittstellen beschrieben. Die Schnittstelle eines Moduls $M$ ist die Menge aller Annahmen, die andere Module über $M$ machen dürfen, bzw. jene Annahmen, die $M$ über seine verwendeten Module macht (bzw. seine Umgebung, wozu auch Speicher, Laufzeit etc. gehören). Konkrete Implementierungen dieser Schnittstellen sind das Geheimnis des Moduls und können vom Programmierer festgelegt werden. Sie sollen hier dementsprechend nicht beschrieben werden.*

*Die Diagramme der Modulsicht sollten die zur Schnittstelle gehörenden Methoden enthalten. Die Beschreibung der einzelnen Methoden (im Sinne der Schnittstellenbeschreibung) geschieht allerdings per Javadoc im zugehörigen Quelltext. Das bedeutet, dass Ihr für alle Eure Module Klassen, Interfaces und Pakete erstellt und sie mit den Methoden der Schnittstellen verseht. Natürlich noch ohne Methodenrümpfe bzw. mit minimalen Rümpfen. Dieses Vorgehen vereinfacht den Schnittstellenentwurf und stellt Konsistenz sicher.*

*Jeder Schnittstelle liegt ein Protokoll zugrunde. Das Protokoll beschreibt die Vor- und Nachbedingungen der Schnittstellenelemente. Dazu gehören die erlaubten Reihenfolgen, in denen Methoden der Schnittstelle aufgerufen werden dürfen, sowie Annahmen über Eingabeparameter und Zusicherungen über Ausgabeparameter. Das Protokoll von Modulen wird in der Modulsicht beschrieben. Dort, wo es sinnvoll ist, sollte es mit Hilfe von Zustands- oder Sequenz-diagrammen spezifiziert werden. Diese sind dann einzusetzen, wenn der Text allein kein ausreichendes Verständnis vermittelt (insbesondere bei komplexen oder nicht offensichtlichen Zusammenhängen).*

*Der Bezug zur konzeptionellen Sicht muss klar ersichtlich sein. Im Zweifel sollte er explizit erklärt werden. Auch für diese Sicht muss die Entstehung anhand der Strategien erläutert werden.*

# 6 Datensicht

*Hier wird das der Anwendung zugrundeliegende Datenmodell beschrieben. Hierzu werden neben einem erläuternden Text auch ein oder mehrere UML-Klassendiagramme verwendet. Das hier beschriebene Datenmodell wird u. a. jenes der Anforderungsspezifikation enthalten, allerdings mit implementierungsspezifischen Änderungen und Erweiterungen. Siehe die gesonderten Hinweise.*

# 7 Ausführungssicht

*Die Ausführungssicht beschreibt das Laufzeitverhalten. Hier werden die Laufzeitelemente aufgeführt und beschrieben, welche Module sie zur Ausführung bringen. Ein Modul kann von mehreren Laufzeitelementen zur Laufzeit verwendet werden. Die Ausführungssicht beschreibt darüber hinaus, welche Laufzeitelemente spezifisch miteinander kommunizieren. Zudem wird bei verteilten Systemen (z. B. Client-Server-Systeme) dargestellt, welche Module von welchen Prozessen auf welchen Rechnern ausgeführt werden.*

# 8 Zusammenhänge zwischen Anwendungsfällen und Architektur

*In diesem Abschnitt sollen Sequenzdiagramme mit Beschreibung(!) für zwei bis drei von Euch ausgewählte Anwendungsfälle erstellt werden. Ein Sequenzdiagramm beschreibt den Nachrichtenverkehr zwischen allen Modulen, die an der Realisierung des Anwendungsfalles beteiligt sind. Wählt die Anwendungsfälle so, dass nach Möglichkeit alle Module Eures entworfenen Systems in mindestens einem Sequenzdiagramm vorkommen. Falls Euch das nicht gelingt, versucht möglichst viele und die wichtigsten Module abzudecken.*

# 9 Evolution

*Beschreibt in diesem Abschnitt, welche Änderungen Ihr vornehmen müsst, wenn sich Anforderungen oder Rahmenbedingungen ändern. Insbesondere würden hierbei die in der Anforderungsspezifikation unter „Ausblick“ genannten Punkte behandelt werden.*

...