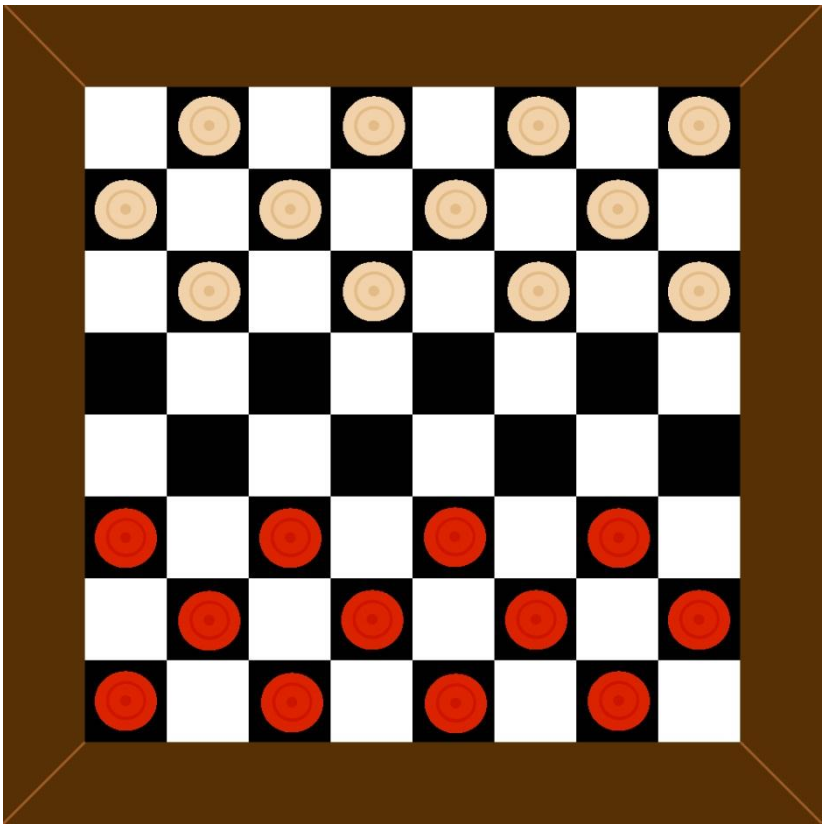
# Projektarbeit M318 - Dame

*Beschrieb, Konzept, Planung*




Verfasser: David Oberholzer
Linh-An Thach
Remo Dörig
Fabio Litscher


Version: 1.0

Datum: 28.02.2014

# Inhalt

# 1. Aufgabenstellung

## 1.1. Ausformulierte Aufgabenstellung

Die Aufgabe besteht darin, unter den vorgegebenen Rahmen- & Randbedingungen, welche im Dokument „Lerninhalte Dame.pdf“ vorgegeben sind, das Spiel „Dame“ in Form eines Gruppenprojekts in C# zu realisieren.

## 1.2. Generelle Randbedingungen

- Meilensteine vorgegeben
- Optionale Modi und Funktionalitäten vorgegeben
- Dokumentationspflicht (Code sowie Projektverlauf)
- Quellenangabe / Quellenverweise
- Milestones müssen nach den von der Schule gegebenen Daten fertiggestellt werden.
- Arbeitsaufwand sollte der Freizeit nicht im Weg stehen.

## 1.3. Organisation der Gruppe

Das Team besteht aus folgenden Mitgliedern:

- Fabio Litscher
- David Oberholzer
- Remo Dörig
- Linh-An Thach

Die Organisation und Aufgabenaufteilung innerhalb der Gruppe gestaltet sich, je nach Pendezen und Vorkenntnissen, flexibel. Die Datenverwaltung soll über GitHub erfolgen. Die Kommunikation findet via Telefon, Email oder Ähnlichem statt.

## 1.4. Ziel

Ziel dieses Projekts ist es ein voll funktionsfähiges Spiel zu programmieren und den Ablauf des Projektes sauber zu dokumentieren. Wir wollen uns Basiswissen in C# sowie im Umgang mit Projektplanung und Realisierung erarbeiten.

# 2. Analyse Istzustand

## 2.1. Spielablauf

Es treten zwei Spieler auf einem 8x8 grossen Feld gegeneinander an. Ziel des Spiels ist es, alle Spielsteine des Gegners zu eliminieren.

## 2.2. Spielregeln

### 2.2.1. Platzierung des Spielbretts und der Spielsteine

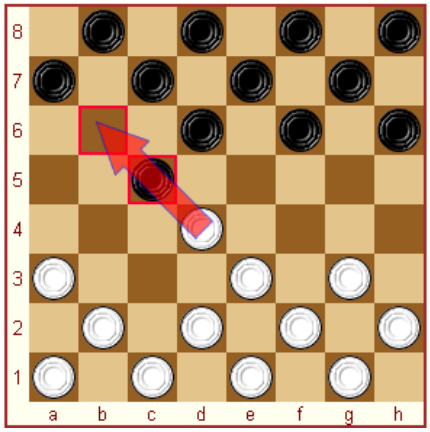
Das Spielbrett muss zwingend so platziert werden, dass unten links ein schwarzes Feld liegt.
Die Spielsteine werden bis zur dritten Reihe auf die schwarzen Felder gelegt.

### 2.2.2. Das Ziehen

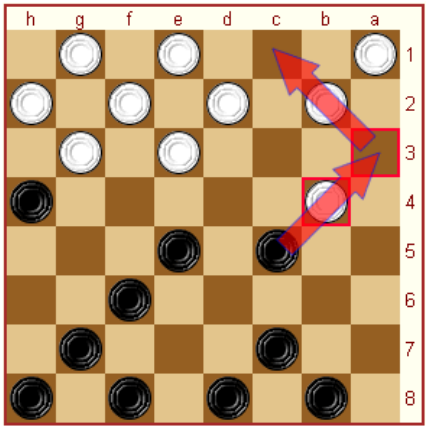
Die Spielsteine dürfen nur diagonal auf ein leeres schwarzes Feld gezogen werden. Normale Spielsteine dürfen sich nur vorwärts bewegen.

### 2.2.3. Das Schlagen

Eine Schlagmöglichkeit besteht, falls ein gegnerischer Stein übersprungen werden kann. Dieser wird anschliessend aus dem Spiel entfernt. Es besteht zusätzlich die Möglichkeit mehrere Steine hintereinander im gleichen Zug zu überspringen. Die Voraussetzungen sind die gleichen wie beim einfachen Schlagen.




http://www.brettspielnetz.de/spielregeln/dame.php




http://www.brettspielnetz.de/spielregeln/dame.php

### 2.2.4. Spielstein in Dame umwandeln

Um ein Spielstein in eine Dame wandeln zu können, muss der Stein die gegnerische Grundlinie des Gegners erreichen. Dabei spielt es keine Rolle ob dies durch einen einfachen Zug oder durch Schlagen erreicht wird.

Die Dame ihrerseits kann von nun an sowohl vorwärts wie auch rückwärts gezogen werden. Genauso kann sie auch in diese Richtungen gegnerische Spielsteine schlagen.

### 2.2.5. Spielende

Gewonnen hat der Spieler, welche alle gegnerischen Steine geschlagen hat oder dessen Spielsteine so blockiert hat, das dieser keinen weiteren Zug unternehmen kann.

## 2.3. Varianten

Es gibt einige verschiedene Varianten vom Damenspiel. Hier sind einige davon aufgelistet.

### 2.3.1. Klassische Dame

Die klassische Dame ist die Variante, die wir programmieren werden, da diejenigen von uns, die das Spiel bereits kennen, diese Variante spielen.

Diese Variante entspricht exakt den Spielregeln von Punkt 2.

### 2.3.2. Französische Dame

Hier zieht man nur vorwärts, allerdings darf man nur rückwärts schlagen. Die Dame kann auch hier vor- und rückwärts schlagen.

### 2.3.3. Englische Dame

Bei der Englischen Dame darf die Dame nicht nur vor- und rückwärts auf den Diagonalen, sondern auch geradeaus, rechts und links auf den geraden Linien schlagen.

### 2.3.4. Italienische Dame

In der italienischen Variante wird so gespielt, dass eine Dame nicht von einem normalen Stein geschlagen werden kann, sondern nur von einer anderen Dame.
Wenn mehrere Möglichkeiten vorhanden sind, muss diejenige gewählt werden mit der man am meisten gegnerische Steine schlagen kann.

# 3. Lastenheft

## 3.1. Milestones

- M0: Prototyp 21.03.2014
  Prototyp, der Spielbar ist
- M1: Basisversion 25.04.2014
  Spiel, welches das gesamte Regelwerk implementiert
- M2: Ausbaustufe 1 16.05.2014
  Diverse Zusatzfunktionen
- M3: Ausbaustufe 2 06.06.2014
  Diverse Zusatzfunktionen

## 3.2. Detaillierte Anforderungen Meilensteine

### 3.2.1. M0: Prototyp

- Spielfeld mit Spielsteinen. Spielsteine sollten bewegbar sein.
- Spielmodus: Nur Menschliche Spieler, ohne Computerspieler oder Spieler über das LAN.
- Spielregeln die implementiert sein sollten:
  - Schlagen
  - Nach vorne fahren (Nur diagonal)
  - Mehrfach fressen
  - Spielende bei Verlust aller Spielersteine

### 3.2.2. M1: Basisversion

- Komplettes Regelwerk implementiert.
- Spielmodus: Computer vs. Mensch (Computer nicht sehr intelligent)
- Animationen von Bewegungen und schlagen.
- GUI erweitern um Beschriftungen.
- Tests schreiben

### 3.2.3. M2: Ausbaustufe 1

Je nach verfügbaren Ressourcen

- High-Score implementieren
- Spieleinstellungsmenü
- Töne implementieren
- Computerintelligenz erweitern (Schwierigkeitsstufen)
- Spielmodus: Computer vs. Computer

### 3.2.4. M3: Ausbaustufe 2

Je nach verfügbaren Ressourcen

- LAN Spiele
- Spielaufzeichnung
- Undo
- Spielstand speichern

## 3.3. Bewertungskriterien

### 3.3.1. M0

Ein einfaches Spiel zwischen zwei Spielern ist möglich. Es müssen nur die aufgelisteten Regeln (Kapitel: Detaillierte Anforderungen Meilensteine – M0: Prototyp) beachtet werden. Folgende Regeln werden nicht beachtet:

- Dame (Doppelsteine bei Erreichen des Spielfeldendes)
- Wenn Fressen vergessen wurde, Strafstein entfernen

### 3.3.2. M1

Das Spiel muss mit komplettem Regelwerk gegen einen Computer oder einen Mitspieler gespielt werden können. Die Animationen sollten dem Spielverlauf entsprechend dargestellt werden.

Der Spielstand sollte beschriftet sein und sich dem Spielverlauf entsprechend anpassen.

### 3.3.3. M2

Es soll ein Auswahlmenü für Spieleinstellungen existieren, in welchen folgende Einstellungen getätigt werden können:

- Ton ein/aus
- Spielernamen
- Schwierigkeitsstufen
- High-Score anschauen
- Spielmodus auswählen

Der High-Score kann mit dem Spielernamen abgespeichert werden.

Töne bei Bewegung eines Steines und beim Fressen eines gegnerischen Steines funktionieren. Bei Erstellen einer Dame und bei Spielende gibt es auch einen entsprechenden Ton.

### 3.3.4. M3

Gegeben falls kann im Menü ein Spiel über das Netzwerk ausgewählt werden.

Ein Zug kann rückgängig gemacht werden. Diese Funktion kann ausgeschaltet werden.

Das Spiel kann während dem Spiel abgebrochen und abgespeichert werden, damit man zu einem anderen Zeitpunkt weiterspielen kann.

Aufgezeichnete Spiele sollen abgespielt werden können.

## 3.4. Randbedingungen

Das Programm sollte von Windows 7 an aufwärts an funktionieren.

# 4. Konzept

## 4.1. Lösungsvarianten

### 4.1.1. Sprache

- Java: Multiplatform Programmiersprache
- VBA:  Scriptsprache für Office Wekrzeuge
- C#: Programmiersprache von Microsoft

### 4.1.2. IDE

- Eclipse: Java basierte IDE
- VisualStudio: IDE von Microsoft

### 4.1.3. Versionsverwaltung

- Git: Versionsverwaltung ursprünglich für den Linux-Kernel
- CVS: Alte bekannte Versionsverwaltung
- Subversion:  Oft benutzte Versionsverwaltung

### 4.1.4. Testing

- TDD: Test Driven Developement
- Tests im Nachhinein schreiben

### 4.1.5. GUI

- WindowsForms: von Lehrer empfohlen
- GTK: Multiplatform
- QT: Multiplatform, viele Firmen wechseln von GTK zu QT

## 4.2. Bewertung Lösungsvarianten

### 4.2.1. Vorteile Nachteile

| | Vorteile | Nachteile |
|---|---|---|
| **Sprache** | | |
| VBA | Gewisse Vorkenntnisse | Inkonsistent |
| Java | Multiplatform | Keiner von uns hat Erfahrung mit Java |
| C# | Gute Integration in Microsoft Umfeld | Microsoft abhängig |
| **IDE** | | |
| Eclipse | Super für Java | Langsam |
| VisualStudio | Super für C# | Läuft nur auf Windows |
| **Versionsverwaltung** | | |
| Git | Skaliert super, wird oft benutzt | Relative kompliziert |
| CVS | Einfach aufgebaut | Community weniger hilfsbereit wie Git |
| Subversion | | Community weniger hilfsbereit wie Git |
| **Testing** | | |
| TDD | Nach dem die Tests geschrieben sind ist es relative einfach zu programmieren | Man kann auf unerwartete Probleme schlechter Reagieren |
| Im Nachhinein | Sehr agil | Programmierung des Programms ein bisschen aufwändiger |
| **GUI** | | |
| WindowsForms | Von Lehrer empfohlen, Einfach | Nicht „mächtig“ |
| GTK | Grosse Funktionalität | Ästhetik schlecht |
| QT | Grosse Funktionalität | |

### 4.2.2. Entscheid

#### *4.2.2.1. Sprache*

VBA konnten wir von Anfang an ausschliessen, da wir gerne etwas Neues lernen möchten und niemand ein richtiger VBA-Fan ist.

Schlussendlich haben wir uns für  C# entschieden, da schon ein paar Mitglieder Erfahrung damit haben und alle Interesse haben in diesem Bereich mehr zu lernen.

#### *4.2.2.2. IDE*

Nach kurzer Recherche im Internet stellte sich heraus, dass alle VisualStudio für C# empfehlen. Dementsprechend benutzen wir VisualStudio.

#### *4.2.2.3. Versionsverwaltung*

Hier haben wir uns für Git entschieden und das aus folgenden Gründen:

- Github.com ist sehr übersichtlich und wird oft benutzt
- Unser Projekt kann fast endlos gross werden, es genügt sogar für den Linux-Kernel
- Einige Leute aus unserer Gruppe haben bereits Erfahrung damit

#### *4.2.2.4. Testing*

Wir haben uns dazu entschieden die Tests im Nachhinein zu Schreiben. Niemand von uns hat jemals mit TDD gearbeitet. Wir befürchten, dass wir mit TDD zu wenig agil auf unerwartete Probleme reagieren können.

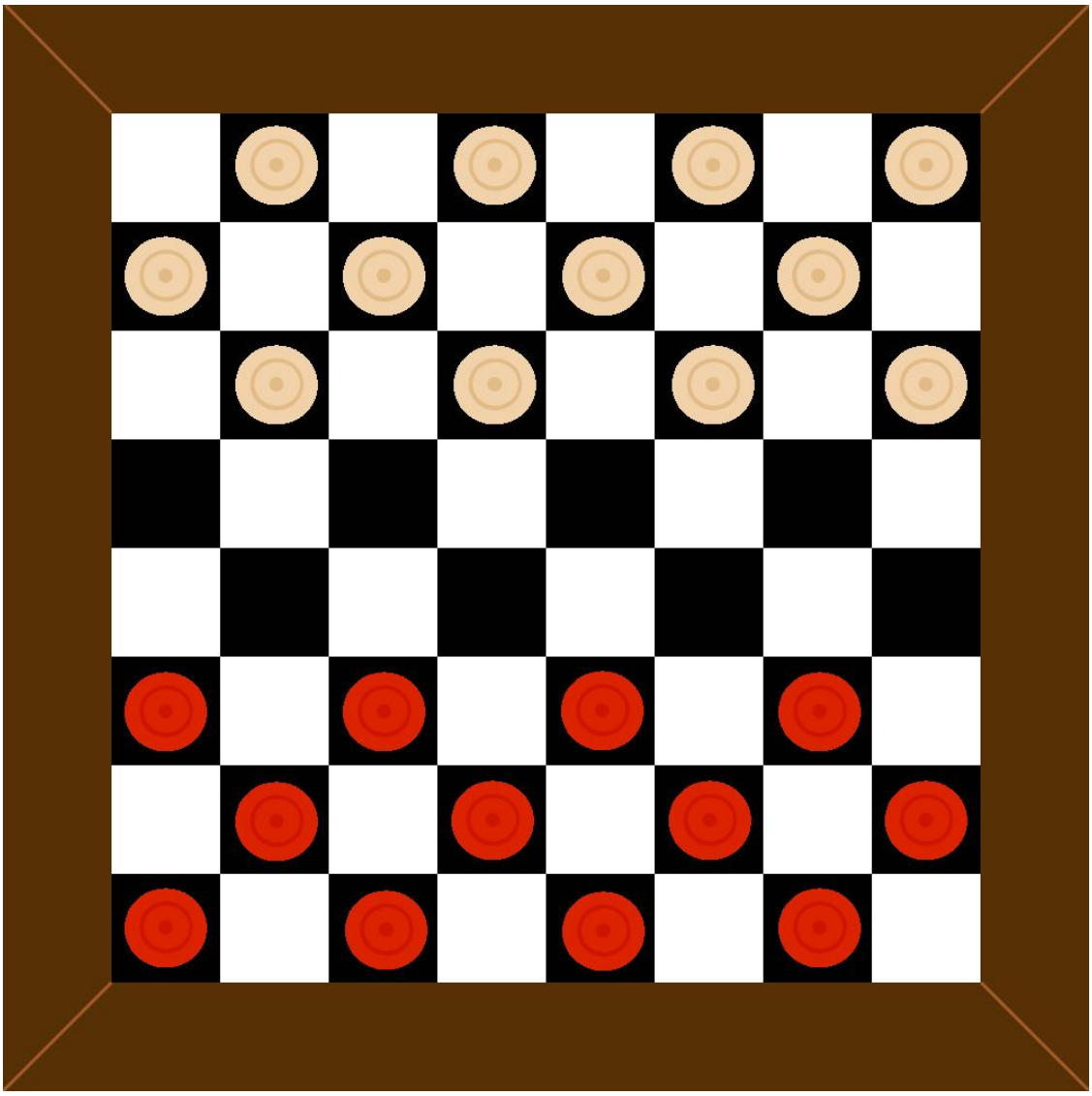
#### *4.2.2.5. GUI*

Wir haben uns für WindowsForms entschieden, da es vom Lehrer empfohlen wurde und wir bereits einige Erfahrungen damit haben.

## 4.3. Entwurf Benutzeroberfläche

### 4.3.1. Look & Feel

Das Spielbrett könnte wie folgt aussehen:



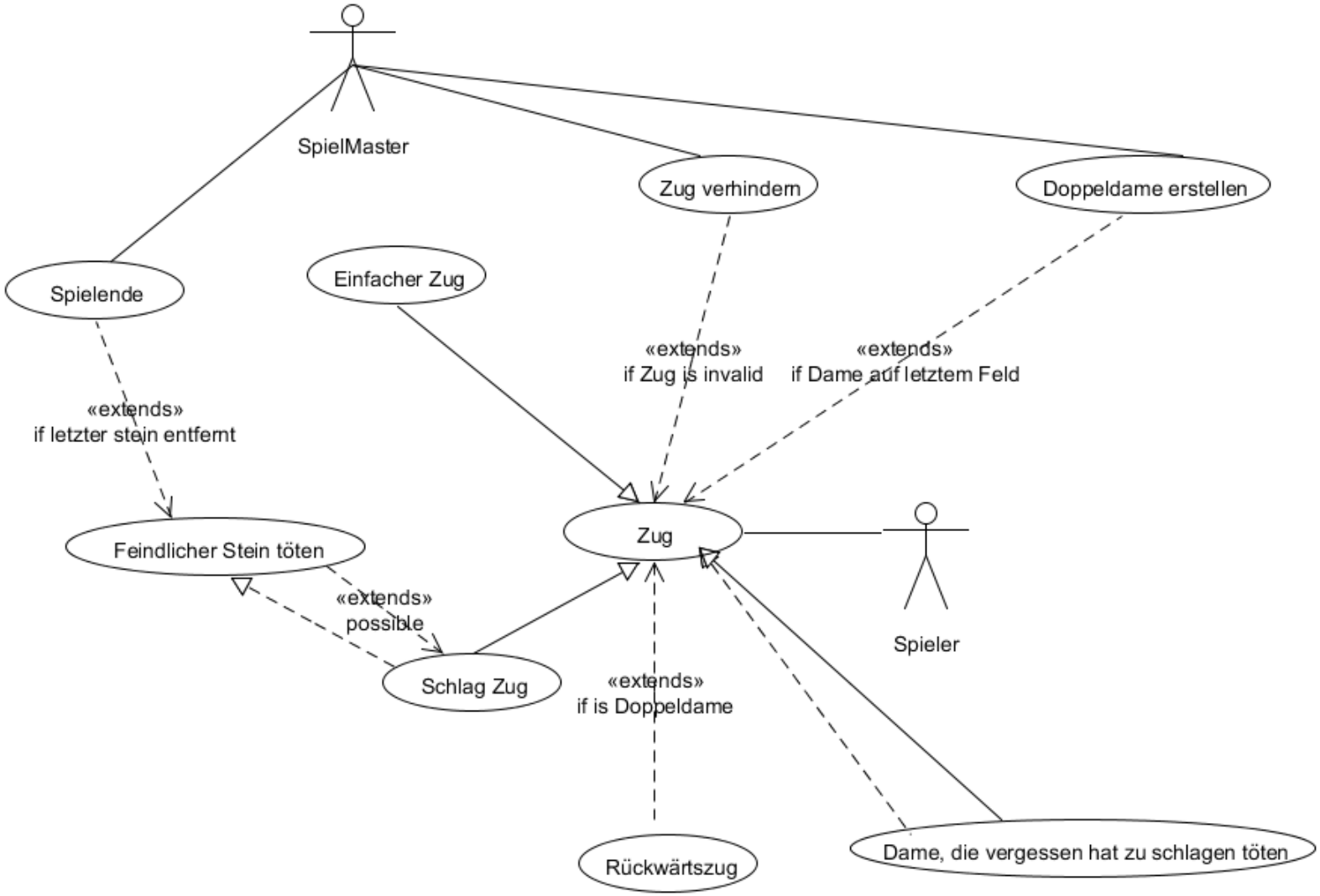
### 4.3.2. Bedienkonzept (Eingabegeräte)

Es wird die Maus als Eingabegerät unterstützt, indem man die Steine anklicken und dann auf das gewünschte Feld klicken kann um mit dem Stein zu fahren.

Falls Undo implementiert wird, kann man diese Funktion alternativ auch mit Ctrl + Z verwenden.

Wenn der Spieler nicht am Zug ist, wird jeglicher Input auf das Spielfeld ignoriert.

## 4.4. Use Case Diagramm



## 4.5. Testkonzept

### 4.5.1. Vorgehen für Test

Wir werden Unit Tests schreiben.  Wir werden http://msdn.microsoft.com/en-us/library/ms243147%28v=vs.80%29.aspx benutzen. Wir werden die Tests im Nachhinein schreiben, also kein Test Diven Development. Wir werden jedoch zu jeder Klasse sicher einen Test schreiben.

### 4.5.2. Testfälle

Folgende Tests werden sicher gemacht. Weitere Tests werden Agile beim Programmieren erstellt.

#### *4.5.2.1. Test Spielfeld*

- Einfacher Zug nach vorne sollte fehlschlagen
- Einfacher Zug auf leeres Feld
- Zug auf Feld ausserhalb des Spielfelds sollte fehlschlagen
- Einfacher Zug rückwärts sollte fehlschlagen
- Einfacher Zug mit Dame rückwärts sollte funktionieren
- Einfacher Zug auf Feld in letzter Linie sollte Stein in einen Damenstein umwandeln
- Einfacher Zug auf besetztes Feld schlägt Fehl
- Angriffszug sollte zwei Felder nach vorne und Feindlichen Spieler töten
- Angriffszug über freundlichen Spieler sollte nicht funktionieren
- Angriffszug über leeres Feld sollte nicht funktionieren
- Angriffszug auf Feld in letzter Zeile ergibt eine Dame
- Nach Angriffszug sollte ein weiterer Angriffszug möglich sein
- Nach Angriffszug sollte kein einfacher Zug mehr möglich sein
- Wenn ein Spieler eine Angriffszug verpasst, sollte dieser gefressen werden können, jedoch nicht wenn der Spieler einen anderen Angriffszug gemacht hat

#### *4.5.2.2. Test KI*

- Spiel gegen andere KI ohne einmal Regeln zu verletzten

#### *4.5.2.3. Test LAN*

Wir müssen noch abklären wie gut man Netzwerktests in C# machen kann.

- LAN Spieler in Netzwerk sollte gefunden werden
- LAN Spieler kann sich Verbinden
- LAN Spieler erfüllt alle Tests die in „Test Spielfeld“ beschrieben sind

# 5. Arbeitsplanung:

Die Arbeiten der jeweiligen Milestones werden agil an die bestgeeigneten Person vergeben. Die bestgeeignete Person zeichnet sich dadurch aus, dass sie über das jeweilige Thema Kenntnisse hat und nicht mit anderen Tätigkeiten beschäftigt ist. Zeitbedingungen innerhalb der Milestones haben wir nicht.

## 5.1. Arbeitspakete, Tätigkeiten, Phasen

### 5.1.1. M0 Prototyp:

Für den Prototyp brauchen wir ein Klassendiagramm, welches in Umlet realisiert wird. Einen Dokumentationsprototypen muss auch schon erstellt werden. In diesem werden alle Arbeitsschritte genau dokumentiert.

Für die Programmierung brauchen wir jemanden der das GUI macht und jemanden der die ganze Logik im Hintergrund realisiert. Die Schnittstelle zwischen GUI und Logik wird im Klassendiagramm bereits definiert.

### 5.1.2. M1 Basisversion

Die Basisversion wird auf dem Prototyp aufgebaut. An dem GUI wird nun nichts mehr geändert. Die Dokumentation des Prototyp muss vervollständigt und während des Ausbaus detailliert erweitert werden. Jemand muss das komplette Regelwerk in der Logik einbauen. Auch müssen Tests geschrieben werden, um die Basisversion zu kontrollieren.

Der Basisversion wird eine künstliche Intelligenz hinzuprogrammiert.

Jemand muss den Spielmodus Computer vs. Mensch implementieren.

### 5.1.3. M2 Ausbaustufe 1

Je nach verfügbaren Ressourcen wird ein Highscore implementiert. Dies soll jemand auf dem GUI anzeigen lassen. Nun muss ein GUI für das Spieleinstellungsmenü von jemandem erstellt werden, welches von der anderen Person die Logik dazu programmiert wird. Töne müssen nun für die Aktionen ausgesucht und implementiert werden.

Jemand programmiert die Computerintelligenz weiter, sodass es möglich wird im Spieleinstellungsmenü eine Schwierigkeitsstufe einzustellen, um so seinen Level anzupassen.

Jemand muss den Spielmodus Computer vs. Computer implementieren.

Alle neuen Funktionalitäten müssen von der jeweiligen Person detailliert in der bestehenden Dokumentation festgehalten werden.

### 5.1.4. M3 Ausbaustufe 2

Je nach verfügbaren Ressourcen wird das spielen über das LAN ermöglicht. Für dies muss man das Klassendiagramm zuerst erweitern.

Jemand implementiert die Funktion „Undo“, welche dazu dient eine Aktion rückgängig zu machen. Die Aktion „Undo“ soll mit der Tastenkombination „Ctrl + Z“ realisiert werden.

Jemand muss nun die Funktion „Spielaufzeichnung speichern“ hinzufügen. Gespeicherte Spielstände sollten wieder abgespielt werden können. Dementsprechend muss eine History von jemandem implementiert werden.

Schlussendlich muss noch von jemandem die Funktion „Spielstand speichern“ implementiert werden.

Alle neuen Funktionalitäten müssen von der jeweiligen Person detailliert in der bestehenden Dokumentation festgehalten werden.

## 5.2. Zeitplan mit Meilensteinen:

So sieht der Zeitplan der Meilensteine aus: