

Projekt: Rupert  
  
Anforderungsspezifikation

Daunicht Nils (ndaunich@hsr.ch)

Fröhlich Adrian (a1froehl@hsr.ch)

Hochreutener Joel(jhochreu@hsr.ch)

Peisker Dominic(dpeisk@hsr.ch)

# Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 05.03.2015 | 0.1 | Erstellung | Nils Daunicht, Dominic Pesiker |
| 12.03.2015 | 0.2 | Zusammenfügen der einzelnen Abschnitte | Nils Daunicht |
| 13.03.2015 | 1.0 | Korrektur | Dominic Peisker |
| 16.03.2015 | 1.1 | UC 05 Fully dressed  UC 06 angepasst | Joel Hochreutener |
| 16.04.2015 | 1.2 | Flowcharts einfügen und Überarbeiten | Dominic Peisker |
| 20.04.2015 | 1.3 | Formulierungen angepasst | Joel Hochreutener |
| 29.05.2015 | 2.0 | Letztes Update | Nils Daunicht |

# Inhalt

[Änderungsgeschichte 2](#_Toc420663893)

[Inhalt 3](#_Toc420663894)

[1. Einführung 5](#_Toc420663895)

[1.1 Beschreibung 5](#_Toc420663896)

[1.2 Gültigkeitsbereich 5](#_Toc420663897)

[1.3 Referenzen 5](#_Toc420663898)

[1.4 Übersicht 5](#_Toc420663899)

[2. Allgemeine Beschreibung 6](#_Toc420663900)

[2.1 Produkt Perspektive 6](#_Toc420663901)

[2.2 Produkt Funktion 6](#_Toc420663902)

[2.3 Einschränkungen 6](#_Toc420663903)

[2.4 Annahmen 6](#_Toc420663904)

[2.5 Abhängigkeiten 6](#_Toc420663905)

[3. Use Cases 7](#_Toc420663906)

[3.1 Use Case Diagramm 7](#_Toc420663907)

[3.2 Aktoren & Stakeholder 7](#_Toc420663908)

[3.2.1 Host 7](#_Toc420663909)

[3.2.2 Player2 7](#_Toc420663910)

[3.3 Beschreibungen (Brief) 7](#_Toc420663911)

[3.3.1 UC 01: Spiel erstellen 7](#_Toc420663912)

[3.3.2 UC 02: Lobby suchen 8](#_Toc420663913)

[3.3.3 UC 03: Spiel beitreten 8](#_Toc420663914)

[3.3.4 UC 04: Spiel starten 8](#_Toc420663915)

[~~3.3.5 UC 05: Einheiten platzieren~~ 8](#_Toc420663916)

[3.3.6 UC 06: Runde durchführen 8](#_Toc420663917)

[3.3.7 UC 07: Spiel beenden 9](#_Toc420663918)

[3.4 Beschreibung (Flow Chart) 10](#_Toc420663919)

[3.4.1 UC 01 für den Host 10](#_Toc420663920)

[3.4.2 UC 06 aus Sicht des schnelleren Spielers 11](#_Toc420663921)

[3.5 Beschreibungen (Fully Dressed) 12](#_Toc420663922)

[~~3.5.1 UC 05: Einheiten Platzieren~~ 12](#_Toc420663923)

[3.5.2 UC 06: Runde durchführen 14](#_Toc420663924)

[3.5.3 UC 06.01: Bewegungsphase durchführen 15](#_Toc420663925)

[3.5.4 UC 06.02: Kampfphase durchführen 17](#_Toc420663926)

[4. Weitere Anforderungen 19](#_Toc420663927)

[4.1 Qualitätsmerkmale 19](#_Toc420663928)

[4.2 Schnittstellen 19](#_Toc420663929)

[4.3 Randbedingungen 19](#_Toc420663930)

# Einführung

## Beschreibung

Bei der Software handelt es sich um ein Rundenbasiertes Strategiespiel für 2 Spieler. Das Spiel besteht aus Runden, welche jeweils aus zwei Phasen besteht. Die Phasen heissen Bewegungs- resp. Kampfphase. Pro Phase kann für jede Einheit ein Zug definiert werden. Beide Spieler wählen gleichzeitig ihre Züge für die aktuelle Phase. Die Züge bestehen unter anderem aus der für jede Einheit gewählten Aktion. Aufgrund der gewählten Aktionen wird der Ausgang des Kampfes berechnet. Wir wollen das Spiel dabei so gestalten, dass die Einzelnen Züge gespeichert und das gesamte Spiel somit Rekonstruierbar ist. Als Spielfeld dient eine schachfeldartige Karte. Das Spiel soll eine Client-Server Architektur besitzen. Dabei hostet jeweils ein Spieler das Game auf seinem Rechner.

## Gültigkeitsbereich

Das Dokument ist während des SE2 Projektes im Frühlingsemester 2015 und im Zusammenhang mit der Arbeit am Projekt Rupert gültig

## Referenzen

|  |  |
| --- | --- |
| Dokument | Inhalt |
| Projektplan | Beschreibt die Planung des Projekts |

## Übersicht

Im Abschnitt 2, Allgemeine Bedingungen, werden die Funktionen im Allgemeinen definiert. Im Folgenden Abschnitt Use Cases wird dann auf die einzelnen Use-Cases des Projekts eingegangen. Und im letzten Abschnitt Weitere Anforderungen werden die nicht funktionalen Anforderungen definiert.

# Allgemeine Beschreibung

## Produkt Perspektive

Die Anforderungen an das Projekt Rupert werden im Projektplan Genauer erläutert

## Produkt Funktion

Die Primäre Funktion des Produktes das unter dem Namen Rupert entwickelt wird ist es ein rundenbasiertes Spiel zu bieten das gelangweilten Studenten und weiteren Personen kurzweilige Ablenkung in spannenden 2 Spieler Schlachten bietet.

## Einschränkungen

Das Spiel ist für 2 Spieler ausgelegt.

## Annahmen

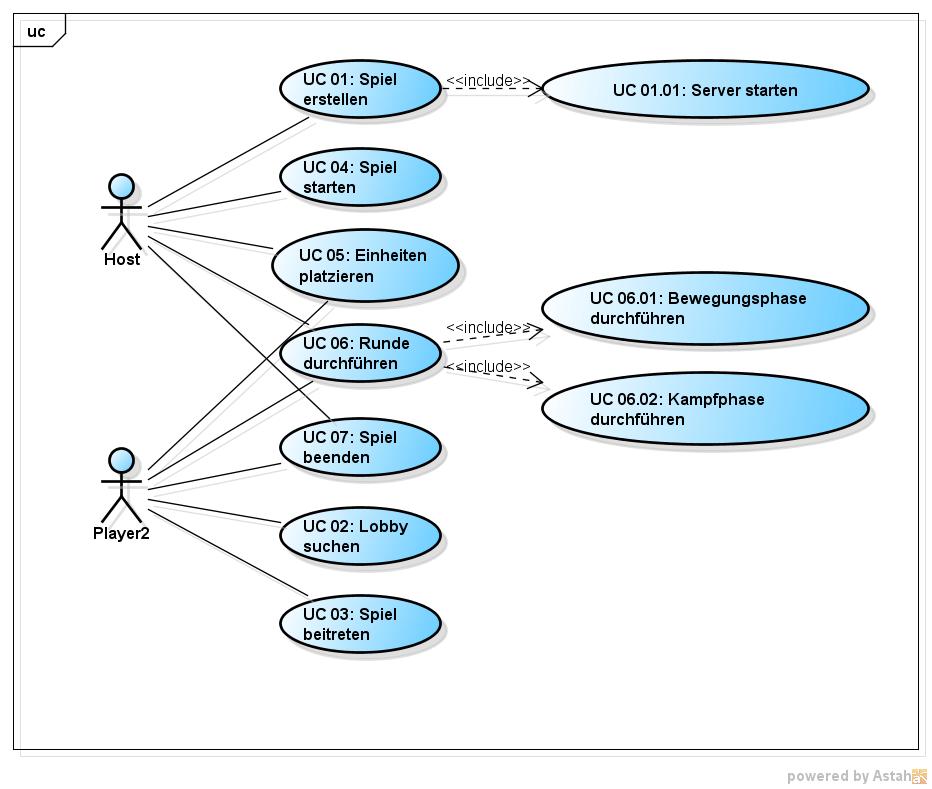
Der Nutzer verfügt über eine installierte Java Version.

## Abhängigkeiten

Das Produkt hat keine externen Abhängigkeiten

# Use Cases

## Use Case Diagramm



## Aktoren & Stakeholder

Die im Folgenden definierten Aktoren Host und Player2 werden während dem Spiel beide als Spieler bezeichnet.

### Host

Der Host ist derjenige Spieler, der den Server startet und das Spiel erstellt.

### Player2

Player2 ist ein beliebiger Spieler, der einem Spiel beitritt.

## Beschreibungen (Brief)

### UC 01: Spiel erstellen

Der Benutzer startet aus dem Client heraus den Server. Er kann ggf. noch einige Voreinstellungen vornehmen. Dies ermöglicht dem zweiten Spieler den Beitritt.

#### UC 01.01: Server starten

Durch das Erstellen des Spiels wird der Server gestartet und der Host automatisch als Spieler 1 mit dem Server verbunden.

### UC 02: Lobby suchen

Dem Client wird im Menü eine Liste mit allen im Netz gefundenen Servern und den darauf laufenden Spielen angezeigt.

### UC 03: Spiel beitreten

Der Benutzer selektiert ein Spiel aus der Liste von offenen Spielen in der Lobby. Der Spieler wählt seinen Namen und der Client verbindet sich dann mit dem Server. Clients können einer Lobby beitreten, sofern sich noch nicht zwei Spieler darin befinden, beziehungsweise das Spiel noch nicht läuft.

### UC 04: Spiel starten

Sobald ein weiterer Spieler mit dem Server verbunden ist kann der Host das Spiel starten. Anschliessend teilt der Server dem Client alle für das Spiel nötigen Informationen mit.

### ~~UC 05: Einheiten platzieren~~

~~Die Spieler platzieren ihre Einheiten im Startbereich bis alle Einheitenpunkte verbraucht sind oder beide Spieler bereit sind. Sobald der erste Spieler bereit ist wird ein Timer gestartet. Spätestens nach Ablauf des Timers startet das Spiel.~~

Aus Zeitgründen verschoben auf spätere Version.

### UC 06: Runde durchführen

Eine Runde besteht aus zwei Phasen, der Bewegungsphase und der Kampfphase. Zuerst kommt die Bewegungsphase und ausgehend von der neuen Situation nach der Bewegungsphase die Kampfphase. Am Ende einer Runde steht jede Einheit wieder alleine auf einem Feld.

#### UC 06.01:Bewegungsphase durchführen

Beide Spieler legen für jede gewünschte Einheit fest, auf welches Feld sich diese bewegen soll. Auf einem Feld kann jeweils nur eine Einheit stehen und eine Einheit hat eine Typenabhängige Anzahl Felder, welche sie pro Phase überqueren kann. Hat jeder Spieler für jede gewünschte Einheit eine Bewegung erfasst oder die Phase beendet wird vom Server die Situation nach den Bewegungen berechnet. Sollten beide Spieler eine Einheit auf dasselbe Feld ziehen wird die höher Priorisierte Einheit auf diesem Feld stehen und die andere Einheit ein Feld davor.

#### UC 06.02: Kampfphase durchführen

Aufgrund der Ausgangslage der Bewegungsphase kann jeder Spieler für seine Einheiten eine Kampfaktion auswählen. Während der Kampfphase bleiben alle Einheiten auf Ihrem Feld stehen. Die Reichweite einer Einheit bestimmt, wie viele Felder eine anzugreifende Einheit maximal entfernt sein darf. Neben Angreifen kann man auch Verteidigen. Einheiten die Verteidigen erhalten beim berechnen des Kampfausganges eine höhere Priorität und erleiden für die Dauer dieser Kampfphase weniger Schaden von angreifenden Einheiten. Alle Angriffe werden simultan berechnet.

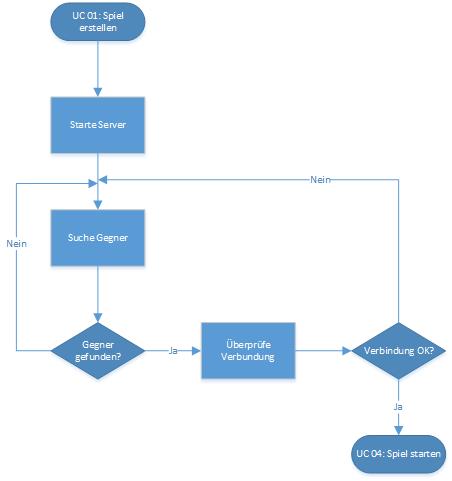
### UC 07: Spiel beenden

Das Spiel ist beendet sobald ein Spieler keine Einheiten mehr besitzt ~~oder aufgibt.~~ ~~Sowohl der Host, als auch der zweite Spieler können das Spiel jeder Zeit abbrechen und beenden.~~

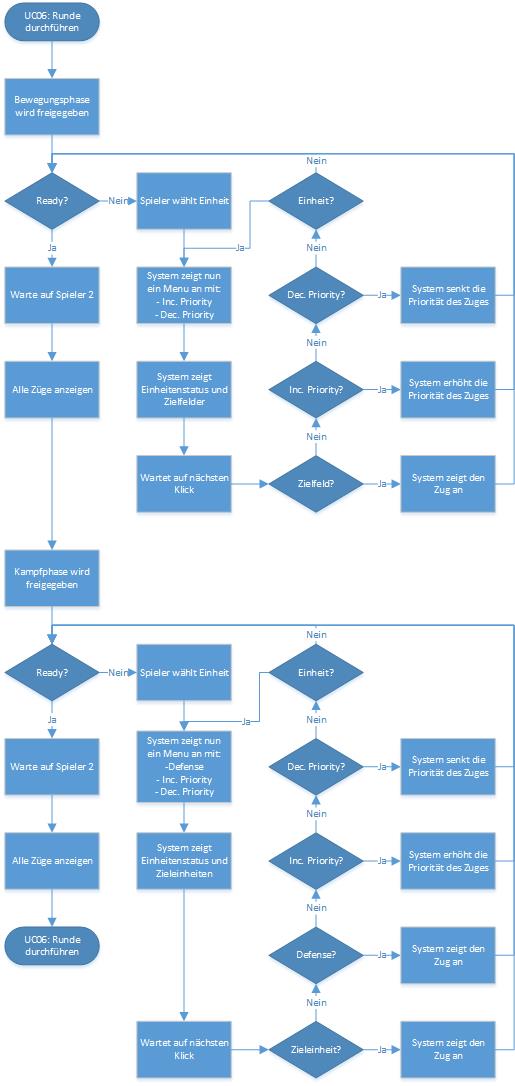
Das Spielende wird erkannt, aber nicht dargestellt. Das Spiel muss manuel geschlossen werden. Aus Zeitgründen konnte dieser Use Case nicht fertig implementiert werden.

## Beschreibung (Flow Chart)

### UC 01 für den Host



### UC 06 aus Sicht des schnelleren Spielers



## Beschreibungen (Fully Dressed)

### ~~UC 05: Einheiten Platzieren~~

~~Goal: Die Spieler platzieren ihre Einheiten im Startbereich~~

~~Level:~~

~~Primary Actor Spieler~~

~~Trigger Start eines Spiels~~

~~Stakeholders and Spieler: Will seine Einheiten setzen.  
Interests Spieler: Will die Einheitstypen selbst wählen.  
 Spieler: Will, dass der Gegner die eigene Aufstellung nicht sieht.  
 Spieler: Will nicht zu lange auf den Gegner warten.  
 Spieler: Will die Informationen für jeden Einheitstyp aufrufen können.~~

~~Preconditions~~

~~Postconditions Alle Einheiten stehen auf einem eigenen Feld im Startbereich.~~

~~Main Success Scenario:~~

1. ~~System gibt die Platzierungsphase frei.~~
2. ~~Spieler wählt einen Einheitstyp aus.~~
3. ~~System zeigt die Eigenschaften an.~~
4. ~~Spieler wählt ein Startfeld.~~
5. ~~System platziert die Einheit.~~
6. ~~Beide Spieler wiederholen unabhängig voneinander Schritt 2-5 bis einer OK drückt.~~
7. ~~System startet den Zähler vom zweiten Spieler.~~
8. ~~Der zweite Spieler drückt OK.~~
9. ~~System zeigt das ganze Spielfeld an.~~
10. ~~System startet UC 06: Runde durchführen.~~

~~Extension:~~

~~\*a. Wenn ein Spieler offline geht:~~

1. ~~System beendet das Spiel~~

~~2a. Wenn der Spieler eine Einheit auswählt:~~

1. ~~System zeigt das Aktionsmenu an.~~
2. ~~System zeigt die Eigenschaften an.~~
3. ~~Spieler wählt "Move".~~
4. ~~Spieler wählt ein Startfeld.~~
5. ~~System verschiebt die Einheit.~~
6. ~~Weiter bei Punkt 6.~~

~~Oder:~~

1. ~~Spieler wählt "Remove".~~
2. ~~System löscht die Einheit.~~
3. ~~Weiter bei Punkt 6.~~

~~4a. Wenn der Spieler einen Einheitstyp wählt:~~

1. ~~Weiter bei Punkt 3.~~

~~4b. Wenn der Spieler ein belegtes oder unerlaubtes Feld wählt:~~

1. ~~Weiter bei Punkt 6.~~

~~8a. Wenn der Zähler auf null steht:~~

1. ~~Weiter bei Punkt 9.~~

~~Special Requirements~~

~~Technology and Data~~

~~Frequency Einmal pro Spiel~~

~~Open Issues~~

### UC 06: Runde durchführen

Goal: Einheiten bewegen und kämpfen.

Level:

Primary Actor Spieler

Trigger Start der nächsten Runde

Stakeholders and Spieler: Will seine Figuren bewegen und den Gegner angreifen.  
Interests Spieler: Will nicht zu lange auf den Gegner warten.  
 Spieler: Will den Status seiner Einheiten sehen.

Preconditions Einheiten sind gesetzt  
 Optional: Spielstand ist nicht fehlerhaft

Postconditions Jede Einheit steht auf einem eigenen Feld

Main Sucess Scenario:

1. System gibt die Bewegungsphase frei.
2. Spieler führen die Bewegungsphase durch (UC 06.01: Bewegungsphase durchführen).
3. System gibt die Kampfphase frei.
4. Spieler führen die Kampfphase durch (UC 06.02: Kampfphase durchführen).
5. System startet die nächste Runde.

Extensions:

~~\*a. Wenn ein Spieler offline geht oder die Verbindung abbricht:~~

1. ~~System beendet das Spiel~~

Special Requirements:

Technology and Data:

Frequency: Kontinuierlich bis zum beenden des Spieles

Open Issues:

### UC 06.01: Bewegungsphase durchführen

Goal: Beide Spieler wählen neue Positionen für ihre Einheiten

Level: Subfunktion

Primary Actor Spieler

Trigger Bewegungsphase freigegeben

Stakeholders and Spieler: Will alle seine Figuren bewegen.  
Interests Spieler: Will, dass der Gegner seine Züge noch nicht sieht.  
 Spieler: Will die Zugreihenfolge bestimmen.  
 Spieler: Will die Informationen für jede Einheit aufrufen können.

Preconditions

Postconditions Jede Einheit steht auf einem eigenen Feld.

Main Sucess Scenario:

1. Spieler wählt eine Einheit aus.
2. System zeigt Menu und Stats.
3. System hebt die möglichen Zielfelder hervor.
4. Spieler wählt das Zielfeld.
5. System erfasst den Zug und zeigt ihn an.
6. Beide Spieler wiederholen unabhängig voneinander Schritt 1-5 bis einer „ready“ drückt.
7. System startet den Zähler vom zweiten Spieler.
8. Der zweite Spieler drückt „ready“.
9. System berechnet die neue Position für jede Einheit
10. System zeigt die neuen Positionen an.

Extensions:

~~\*a. Wenn ein Spieler offline geht:~~

1. ~~System beendet das Spiel~~

~~4a. Wenn der Spieler „Inc. priority" wählt:~~

1. ~~System erhöht die Priorität des Zuges der ausgewählten Einheit um 1.~~
2. ~~Weiter bei Punkt 6.~~

~~4b. Wenn der Spieler „Dec. priority" wählt:~~

1. ~~System senkt die Priorität des Zuges der ausgewählten Einheit um 1.~~
2. ~~Weiter bei Punkt 6.~~

4c. Wenn der Spieler eine Einheit auswählt:

1. Weiter bei Punkt 2.

~~4d. Wenn der Spieler einen Rechtsklickt macht:~~

1. ~~System beendet aktiven Zug~~
2. ~~Weiter bei Punkt 6.~~

~~8a. Wenn der Zähler auf null steht:~~

1. ~~System beendet aktiven Zug.~~
2. ~~Weiter bei Punkt 9.~~

Special Requirements: Punkt 9 und 10 müssen innerhalb von 15 Sekunden fertig sein.

Technology and Data:

Frequency: Einmal pro Runde

Open Issues:

### UC 06.02: Kampfphase durchführen

Goal: Beide Spieler wählen Kampfaktionen für ihre Einheiten

Level: Subfunktion

Primary Actor Spieler

Trigger Kampfphase freigegeben

Stakeholders and Spieler: Will für alle seine Einheiten eine Kampfaktion wählen.  
Interests Spieler: Will, dass der Gegner seine Züge noch nicht sieht.  
 Spieler: Will die Zugreihenfolge bestimmen.  
 Spieler: Will die Informationen für jede Einheit aufrufen können.

Preconditions

Postconditions Kämpfe sind ausgeführt.

Main Sucess Scenario:

1. Spieler wählt eine Einheit aus.
2. System zeigt Menu und Stats.
3. System hebt die möglichen Zielfelder hervor.
4. Spieler wählt das Zielfeld.
5. System erstellt den Zug und zeigt ihn an.
6. Beide Spieler wiederholen unabhängig voneinander Schritt 1-5 bis einer „ready“ drückt.
7. System startet den Zähler vom zweiten Spieler.
8. Der zweite Spieler drückt „ready“.
9. System berechnet den Ausgang der Kämpfe
10. System aktualisiert das Spielfeld.

Extensions:

~~\*a. Wenn ein Spieler offline geht:~~

1. ~~System beendet das Spiel~~

4a. Wenn der Spieler „defense" wählt:

1. System erstellt den Zug mit einer höheren Priorität und zeigt ihn an.
2. Weiter bei Punkt 6.

4b. Wenn der Spieler eine eigene Einheit auswählt:

1. Weiter bei Punkt 2.

~~4c. Wenn der Spieler einen Rechtsklick macht:~~

1. ~~System beendet aktiven Zug~~
2. ~~Weiter bei Punkt 6.~~

~~8a. Wenn der Zähler auf null steht:~~

1. ~~System beendet aktiven Zug.~~
2. ~~Weiter bei Punkt 9.~~

Special Requirements: Punkt 9 und 10 müssen innerhalb von 15 Sekunden fertig sein.

Technology and Data:

Frequency: Einmal pro Runde

Open Issues:

# Weitere Anforderungen

## Qualitätsmerkmale

* [Funktionalität](http://de.wikipedia.org/wiki/Funktionalit%C3%A4t):
* [Konformität](http://de.wikipedia.org/wiki/Konformit%C3%A4t): Wir halten uns an die Richtlinien von Google zu Java. (<https://google-styleguide.googlecode.com/svn/trunk/javaguide.html>)
* [Zuverlässigkeit](http://de.wikipedia.org/wiki/Zuverl%C3%A4ssigkeit_(Technik)):
* Bei Inkonsistenz beendet der Server das Spiel. Inkonsistenz kann durch Verbindungsabbruch eines Clients entstehen.
* Falscheingaben werden vom Spiel ignoriert.
* [Benutzbarkeit](http://de.wikipedia.org/wiki/Gebrauchstauglichkeit_(Produkt)):
* Das Programm ist mit Maus und Tastatur bedienbar und soll von einem Mitstudenten ohne vorherige Einführung gespielt werden können.
* Das Spiel soll von neuen Benützern direkt durch benützen erlernt werden können
* Das Spiel soll leicht bedienbar sein.
* [Effizienz](http://de.wikipedia.org/wiki/Effizienz_(Informatik)):
* [Zeitverhalten](http://de.wikipedia.org/wiki/Leistung_(Informatik)): Das Programm soll innerhalb von 10 Sekunden einen Kaltstart durchführen können. Dies auf einem Laptop mit aktueller Hardware (nicht älter als 3 Jahre) und installiertem Windows 7 oder höher.
* Traffic: Kommunikation zwischen Client und Server auf ein Minimum beschränken und die Qualität der ausgetauschten Daten so auswählen das die Packet Grösse minimal ist (<0.5 MB)
* [Übertragbarkeit](http://de.wikipedia.org/wiki/Plattformunabh%C3%A4ngigkeit):
  + Installierbarkeit: Für die Software soll ein Installer mitgeliefert werden

## Schnittstellen

Die Lösung verfügt über keine fachlichen Schnittstellen.

## Randbedingungen

* Maximal 10 Einheiten pro Person
* ~~Zeitbeschränkung für einen Zug sind 2 Min~~
* Startbereich: 10x4 Felder
* Karte: 14x20 Felder
* Zug Zeit und Einheitenzahl skalierbar modulieren
* Schere Stein Papier System