

# **Pflichtenheft**

## **Conways's Game of Life**

**„Eine universelle Software zur Simulation zellulärer Automaten“**

### **Auftraggeber:**

- Hochschule Bochum
- Ansprechpartner: Dipl.-Inform. Christian Düntgen
- Raum: D 3-30

### **Auftragnehmer:**

- Die 5 Kranken Schwestern
- Weder krank noch Frauen
- Definitionsphasenmanager: Jörg Galilee Uwimana
- Architekt (Entwurfsbeauftragter): Felix Reinhardt
- Gruppenschuldiger, Spezifikationsbeauftragter: Alex Chojnatzki
  - Implementierungs-Beauftragter: Nicholas Schuran
- Kundenbetreuer, Außenminister, Abnahmebeauftragter: Diao El Bathich

**Stand: 28.10.2021**

# Contents

<b>1 Zielbestimmung</b>	<b>3</b>
1.1 Musskriterien . . . . .	3
1.2 Wunschkriterien . . . . .	6
<b>2 Produkt-Einsatz</b>	<b>7</b>
2.1 Anwendungsbereich . . . . .	7
2.2 Zielgruppen . . . . .	7
2.3 Produktumgebung . . . . .	7
2.3.1 Softwareanforderungen . . . . .	7
2.3.2 Softwareanforderungen . . . . .	7
2.3.3 Hardwareanforderungen . . . . .	7
2.4 Betriebsbedingungen . . . . .	7
<b>3 Produktfunktionen</b>	<b>9</b>
3.1 Funktionale Anforderungen . . . . .	9
3.1.1 Benutzeroberfläche . . . . .	9
3.1.2 . . . . .	17
3.1.3 Datenverarbeitung . . . . .	17
3.1.4 Datenspeicherung . . . . .	17
3.2 Nichtfunktionale Anforderungen . . . . .	17
3.2.1 Performance . . . . .	17
3.2.2 Zuverlässigkeit . . . . .	18
<b>4 Testszenarien</b>	<b>19</b>
4.1 UI . . . . .	19
4.2 Verarbeitung . . . . .	19
4.3 Speichern . . . . .	19
4.4 Performance . . . . .	19
4.5 Benutzbarkeit (Schimpanse benötigt) . . . . .	19
<b>5 Entwicklungsumgebung</b>	<b>20</b>
5.1 Verwendete Software . . . . .	20
5.2 Verwendete Hardware . . . . .	20
5.3 verwendete Organisation . . . . .	20

# 1 Zielbestimmung

## 1.1 Musskriterien

Das Programm soll dazu dienen, Zelluläre Automaten auf einem 2-D orthogonalen Spielfeld darstellen zu können. Dazu werden als Beispiel die Regeln für Conway's Game of Life verwendet. Hierzu sind unbedingt die folgenden Features erforderlich:

M0001	UI	Das Programm muss eine graphische Oberfläche haben.
M0002	Scope	<ul style="list-style-type: none"><li>• Es soll ein zellulärer Automat mit möglichst großer Freiheit definiert und simuliert werden können.</li></ul>
M0003	Darstellung Spielfeld	<ul style="list-style-type: none"><li>• Die Darstellung des Zellulären Automaten erfolgt über eine 2 Dimensionale Matrix aus Quadraten deren Farbe und Helligkeit den Zustand eines Feldes wiedergeben.</li></ul>
M0004	Transitionsregeleditor	Die Transitionsregeln sollen über eine definierte und im Handbuch dokumentierte Syntax (invers Polnische Notation, ggf. auch mathematische Schreibweise) formuliert werden können. Der neue Zustand einer Zelle darf dabei von der Zelle selbst, sowie von den umliegenden acht benachbarten Zellen abhängen. Ihr Status wird in Variablen bereitgestellt.
M0005	Spielfeldaufbau	Das Spielfeld soll als 2-D Array von Integerwerten ausgeführt sein, welche den Zellzustand repräsentieren.
0006	Spielfeldgröße	Die Spielfeldgröße soll vor Simulationsstart vom Benutzer über (Text-)Eingabefelder festgelegt werden können.
M0007	Speichern & Laden	Spielfeldzustand und Transitionsregeln sollen separat gespeichert und geladen werden können.

M0008	Einfügen	Es sollen Figuren in das Spielfeld eingefügt werden können. Dies soll so geschehen, dass Figuren als Spielstände mit kleinerer Feldgröße als ganzes geladen und eingefügt werden können.
M0009	Navigation	es soll möglich sein, das Spielfeld mit Zoom und Pan verschieden zu betrachten.
M0010	Spielfeldmanipulation	Der Zustand einer Zelle soll durch Mausklick darauf auf einen wählbaren Wert einstellbar sein. Das Wählen des Werts soll durch ein Texteingabefeld auf der Benutzeroberfläche erfolgen. Details in der Beschreibung der Benutzeroberfläche.
M0011	Topologie	Das Randverhalten des Spielfelds soll zwischen begrenztem Rechteck und Torus (Zellen an den Kanten sind mit den ihnen gegenüberliegenden Zellen benachbart) wählbar sein.
M0012	Automatische Simulation	Die Simulationsgeschwindigkeit soll über einen Slider einstellbar sein. Die Simulation soll über einen Button gestartet und unterbrochen werden können.
M0013	Manuelle Simulation	Über einen Button soll die nächste Generation berechnet und angezeigt werden können.
M0013	Zufälliger Anfangszustand	Der Spielfeldzustand soll zufällig generierbar sein. Dazu soll einem Zellzustand eine Wahrscheinlichkeit zugewiesen werden können, mit dem Default-Zustand 0, sodass jede Zelle genau einen Zustand erhält.
M0014	Anzeige	Die Anzeige des Spielfeldzustands soll durch Farben erfolgen, wobei einem Zustand eine Farbe zugeordnet wird.

M0015	Startbedingungen	Beim Programmstart soll ein 80x80 Zellen großes Spielfeld präsentiert werden, auf welches die Spielregeln für Conway's Game of Life verwendet werden.
-------	------------------	---

## 1.2 Wunschkriterien

W0001	Undo	Es sollen Eingaben rückgängig gemacht werden können.
W0002	Regeleditor	Eingabe der Regeln in für Menschen gut lesbarer Mathematischer Schreibweise, mit Grundrechenarten und logischen Operationen
W0003	Performance	Multithreading parallelisierbarer Prozesse
W0003	Farbanpassung	Wenn möglich soll die Farbe eines Zustands durch den Benutzer einstellbar sein.

## **2 Produkt-Einsatz**

### **2.1 Anwendungsbereich**

Das Programm soll dazu dienen, Zelluläre Automaten mit recht großer Freiheit bauen zu können. Ob es sich dann um Game of Life, einen Waldbrandsimulator handelt, ist dann außen vor.

### **2.2 Zielgruppen**

Die Verwendung dieses Programms für Conway's Game of life ist einfach, da die Spielregeln mitgeliefert werden. Dies kann von allen interessierten ausprobiert werden, da die Manipulation des Spielfelds zum ausprobieren einlädt.

Leider ist es nicht möglich, den Regeleditor intuitiv bedienbar zu gestalten, da es für eine effiziente Verarbeitung notwendig ist, den Zustand einer Zelle in der nächsten Generation als Mathematische Funktion der Zustände der Nachbarzellen darzustellen. Aus diesem Grund gibt es zwar einen Leitfaden, um Mathematische Funktionen mit den Umliegenden Zellen als Ausgangsdaten zu erstellen, es ist jedoch nicht einfach, dies zu tun. Deal with it.

### **2.3 Produktumgebung**

#### **2.3.1 Softwareanforderungen**

#### **2.3.2 Softwareanforderungen**

- Ein "Java Runtime Environment" der Version 1.8.x oder neuer. Ältere Versionen werden nicht getestet.
- Betriebssystem, was in der Lage ist, besagte JRE auszuführen.

#### **2.3.3 Hardwareanforderungen**

- Ein Computer aus diesem Jahrtausend mit einer Prozessorarchitektur für die eine JRE verfügbar ist. Dual-Core oder besser empfohlen, Dienstalter nicht über 1,6 Dekaden.

### **2.4 Betriebsbedingungen**

- Schreib- und Leserechte für die Speicherstände.
- verfügbarer Speicherplatz. (500 MB Festplattenspeicher großzügigerweise empfohlen)

- Arbeitsspeicher angepasst an die Feldgröße (128 MB sollten für die Standardkonfiguration ausreichen)



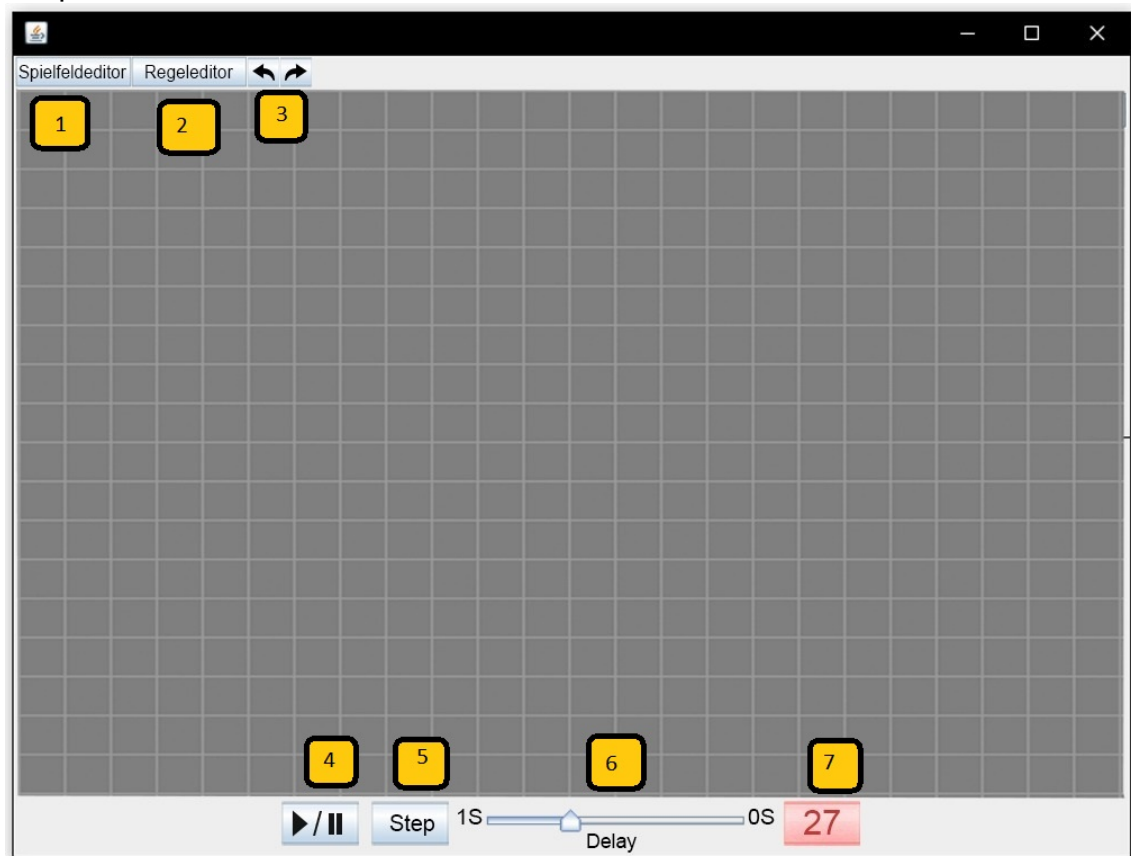
## 3 Produktfunktionen

### 3.1 Funktionale Anforderungen

#### 3.1.1 Benutzeroberfläche

Nach dem Start soll folgende Oberfläche als Standard auftauchen. Im folgenden werden die (numerierten) UI-Elemente erläutert.

Hauptbenutzerfläche



AF-01	Spielfeldeditor	Klick auf den Button Spielfeldeditor öffnet das Dropdownmenü "Spielfeldeditor".
AF-02	Regeleditor	Klick auf den Button "Regeleditor" öffnet das Dropdown-Menü "Regeleditor"
AF-03	Undo/Redo	Klick auf "undo" macht die letzte Eingabe des Spielers rückgängig. "Redo" stellt sie wieder her. Hochoptional.
AF-04	Simulation starten / unterbrechen	Klick auf den Button schaltet die automatische Simulation an oder aus.

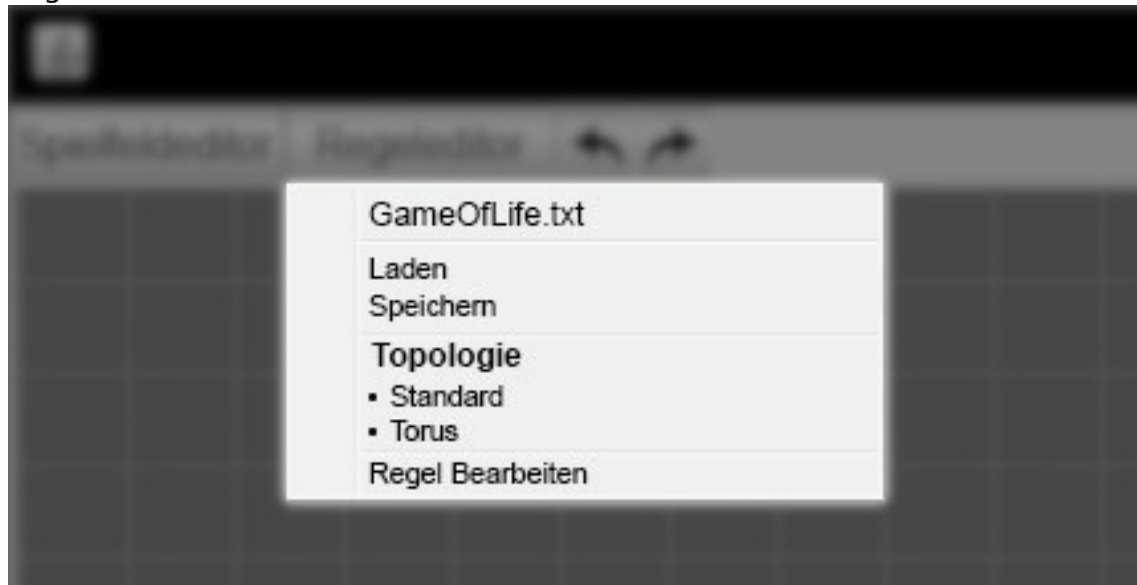
AF-05	Stepover	Klick auf den Button "STEP" führt genau einen Simulationsschritt aus.
AF-06	Delay-Slider	Mit diesem Slider kann die Verzögerung zwischen zwei Generationen zwischen 1 und 0 sekunden stufenlos ausgewählt werden.
AF-07	Zellmodifikation	In diesem Textfeld kann (nur int) der Wert festgelegt werden, auf den eine Zelle gesetzt werden soll, falls man mit der Maus darauf klickt.

Anwendungsfall ID	AF-01
AF Name	Spielfeldeditor
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig
Auslösendes Ereignis	Mausklick auf den „Name“-Button
Nachbedingung Erfolg	Öffnen des Fensters "Spielfeldeditor"
Nachbedingung Fehlschlag	Programm stürzt ab
Ablauf	Nutzer klickt auf Button und das entsprechende Fenster öffnet sich.
Anwendungsfall ID	AF-02
AF Name	Regeleditor
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig
Auslösendes Ereignis	Mausklick auf den Regeleditor-Button
Nachbedingung Erfolg	Öffnen des Fensters "Regeleditor"
Nachbedingung Fehlschlag	Programm stürzt ab
Ablauf	Nutzer klickt auf Button und das entsprechende Fenster öffnet sich.
Anwendungsfall ID	AF-03
AF Name	Undo/Redo
Akteur	Benutzer des Programms
Vorbedingung	Irgendeine Aktion im Programm wurde bereits durchgeführt.
Auslösendes Ereignis	Mausklick auf den Undo- bzw. Redo-Button
Nachbedingung Erfolg	Undo: Rückgängig machen der zuletzt ausgeführten Aktion. Redo: Wiederherstellen. Hochoptional.
Nachbedingung Fehlschlag	Undo: Fehlermeldung : "Nichts zurückzusetzen", Redo: Fehlermeldung: "nichts wiederherzustellen".
Ablauf	Nutzer klickt auf Undo, die zuletzt ausgeführte Aktion wird zurückgesetzt. REDO: die zuletzt ausgeführte Aktion wird wiederhergestellt.

Anwendungsfall ID	AF-04
AF Name	Play/Pause
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig
Auslösendes Ereignis	Mausklick auf den Play/Pause-Button
Nachbedingung Erfolg	Umschalten der Simulation zwischen "Simulation läuft" und "Pausiert"
Nachbedingung Fehlschlag	Programm stürzt ab
Ablauf	Simulation gestoppt: User klickt auf Button, Simulation startet. Simulation läuft: User klickt auf Button, Simulation stoppt.
Anwendungsfall ID	AF-05
AF Name	STEPOVER
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig und im Vollbesitz seiner Maus
Auslösendes Ereignis	Mausklick auf den STEPOVER-Button
Nachbedingung Erfolg	Simulieren und anzeigen der nachfolgenden Spielfeldgeneration
Nachbedingung Fehlschlag	Programm stürzt ab
Ablauf	Nutzer klickt auf Button und der Zelluläre Automat bewegt sich genau einen Simulationsschritt weiter.
Anwendungsfall ID	AF-06
AF Name	Delay-Slider
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig
Auslösendes Ereignis	Klicken und Ziehen auf dem Delayslider
Nachbedingung Erfolg	Anpassung des Simulationsschritt-Delays
Nachbedingung Fehlschlag	Programm stürzt ab
Ablauf	... Nicht im Ernst...

Anwendungsfall ID	AF-07
AF Name	Zellmodifikation
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig und mit einem Alkoholpegel $\geq 5\%$
Auslösendes Ereignis	Mausklick auf das Zustandstextfeld, oder auf das Spielfeld
Nachbedingung Erfolg	Textfeld: User kann neuen Edit-Zielzustand angeben und mit Enter bestätigen. Spielfeld: Zustand der angeklickten Zelle wird auf den Zustand im Textfeld gesetzt.
Nachbedingung Fehlschlag	Textfeld: Bei eingabe eines Nicht-Integers Fehlermeldung und setzen auf 0 (zur Sicherheit)
Ablauf	Textfeld: Nutzer klickt auf Textfeld. Nutzer gibt ein, welcher Zielzustand gewünscht ist. Nutzer bestätigt mit Enter. Spielfeld: Nutzer klickt beliebige Zelle an. Zustand der Zelle wird überschrieben durch Zustand im Textfeld

## Regeleditor



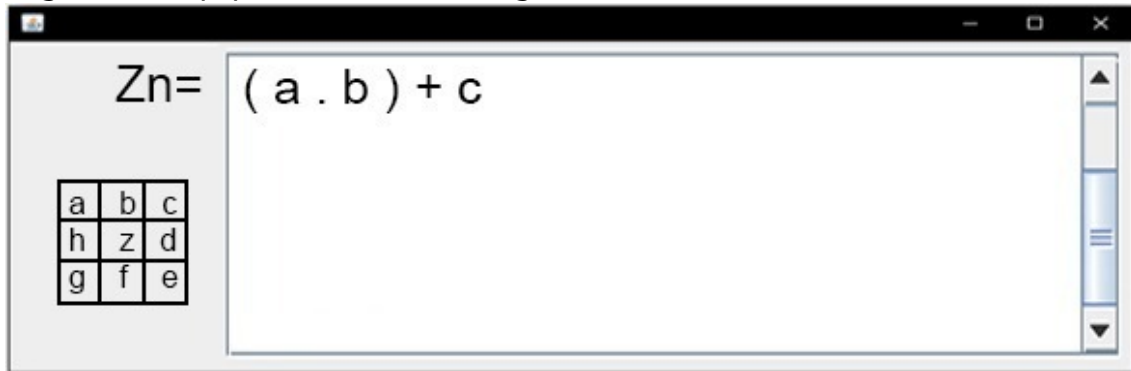
RF-01	Laden	Ruft den Filechooser zum Laden eines anderen Regelausdrucks auf
RF-02	Speichern	Ruft den Java-Swing-Filechooser zum Speichern des aktuellen Regelausdrucks auf.
RF-03	Topologiewechsler	Auswahlschalter für das Spielfeldrandverhalten.
RF-04	Regel Bearbeiten	Ruft das Popup-Fenster zum Regelausdruck bearbeiten auf.

Anwendungsfall ID	RF-01
AF Name	Laden
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig, Regeleditor Dropdownmenü ausgewählt
Auslösendes Ereignis	Mausklick auf den Laden-Button
Nachbedingung Erfolg	Öffnen des Java-Swing-Fensters mit FileChooser zum öffnen einer Regeldatei
Nachbedingung Fehlschlag	Fehlermeldung und Rückkehr zur Haupt-Oberfläche.
Ablauf	Nutzer klickt auf Button und öffnet das Fenster zum laden einer Regeldatei. Durch auswählen und Bestätigen durch klick auf "Öffnen". wird die aktuell aktive Regel durch die geladene ersetzt.
Anwendungsfall ID	RF-02
AF Name	Speichern
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig, Regeleditor Dropdownmenü ausgewählt
Auslösendes Ereignis	Mausklick auf den Speichern-Button
Nachbedingung Erfolg	Öffnen des Java-Swing-Fensters mit FileChooser zum speichern einer Regeldatei
Nachbedingung Fehlschlag	Fehlermeldung und Rückkehr zur Haupt-Oberfläche.
Ablauf	Nutzer klickt auf Button und öffnet das Fenster zum Speichern einer Regeldatei. Durch auswählen und Bestätigen durch klick auf "Öffnen". wird die aktuell aktive Regel an besagter Stelle gespeichert.

Anwendungsfall ID	RF-03
AF Name	Topologie
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig, Regeleditor Dropdownmenü ausgewählt
Auslösendes Ereignis	Mausklick auf den Topologie-RadioButton
Nachbedingung Erfolg	Setzen der Spielfeldkantenbehandlung auf Torus oder Beschränkt, je nach Wunsch.
Nachbedingung Fehlschlag	Fehlermeldung und Rückkehr zur Haupt-Oberfläche.
Ablauf	Durch Klick auf "Standard" wird das Spielfeld als endliches Spielfeld behandelt, an den Kanten werden alle nachbarzellen als 0 angenommen. Durch klick auf "Torus" werden die Zellen an den Kanten die Zellen an gegenüberliegenden Kanten als Nachbarn behandeln.



Regeleditor Popup-Fenster mit Texteingabe:



Anwendungsfall ID	RF-04
AF Name	Regel Bearbeiten
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig, Dropdownmenü "Regeleditor" ausgewählt.
Auslösendes Ereignis	Mausklick auf den "Regel Bearbeiten"-Button im Regeleditor-Dropdownmenü
Nachbedingung Erfolg	Öffnen des Popupfensters "Regeleditor" (siehe oben).
Nachbedingung Fehlschlag	Fehlermeldung und Rückkehr zur Haupt-Oberfläche.
Ablauf	Öffnen des Übergangsregel-Editors. Bei Öffnen steht im Textfeld die zurzeit verwendete Transitionsregel. Durch Tastatureingabe kann der String im Textfeld verändert werden, womit die Transitionsregel angepasst wird. Ferner gibt es ein Bild links, welches als Hilfestellung die Variablen angibt, welche die Zellzustände von Nachbarzellen angeben. Auf die Weise kann der Zustand der aktuell betrachteten Zelle für die nächste Iteration auf Basis ihrer Nachbarn berechnet werden

### 3.1.2

### 3.1.3 Datenverarbeitung

### 3.1.4 Datenspeicherung

## 3.2 Nichtfunktionale Anforderungen

### 3.2.1 Performance

- Lineare Laufzeit der Generationsberechnung pro Spielfeldgröße

### **3.2.2 Zuverlässigkeit**

- This is bleeding edge technology. Report bugs to Jehova's Witnesses, Ortsgruppe Westfalen-Lippe.

Hinweis: Für die Sicherheit des Nutzers wird nicht garantiert.

## **4 Testszzenarien**

### **4.1 UI**

Das UserInterface bietet einige Möglichkeiten für Probleme. Für alle Texteingabefelder muss zur Laufzeit geprüft werden, ob der User-Input in Ordnung ist und bei Bedarf Fehlermeldungen ausspucken. Beispiel: Die Spielfeldgröße muss unbedingt vom Typ int sein. "abeuia" ist keine valide Spielfeldgröße. Ferner muss geprüft werden, ob alle Knöpfe ausschließlich das tun, was sie sollen und nicht spaßige Nebeneffekte erzeugen.

### **4.2 Verarbeitung**

Es muss geprüft werden, ob der Regelinterpreter vernünftig funktioniert. Es muss geprüft werden ob die Spielfeldvariation vernünftig funktioniert. Es muss geprüft werden ob die Arrayverarbeitung für den Spielstand vernünftig funktioniert.

### **4.3 Speichern**

Der Java Filechooser wird wohl funktionieren, nicht wahr?

ggf. kann man einen Speicherzustand laden und unter anderem Namen speichern und dann den Inhalt vergleichen....

### **4.4 Performance**

Awatt, Performance wird auf modernen Systemen in ausreichender Menge vorhanden sein. Bitte keine Speicherlecks bauen...

### **4.5 Benutzbarkeit (Schimpanse benötigt)**

Bin kein Schimpanse, es

## 5 Entwicklungsumgebung

### 5.1 Verwendete Software

Betriebssysteme:	MacOS X, Windoof X, Linux X
Bildbearbeitung & Diaagramme	GIMP, Photoshop, Modelio
Programmierung & Versionierung	Eclipse, Eclipse Window builder, GIT

### 5.2 Verwendete Hardware

Intelligente Frühstücksbrettchen mit abwaschbarer Benutzeroberfläche verschiedener Hubraumkassen.

### 5.3 verwendete Organisation

Haben Sie wirklich den Eindruck, dass hier irgendwas organisiert abläuft? Aber gut, ein Versuch: Wenn etwas schief geht, ist Alex schuld. Wenn jemand Ahnung hat, dann Nico. Wenn jemand Protokoll schreibt, dann Felix. Wenn jemand gute Laune hat, dann Jörg. Wenn jemand Photoshop macht, dann Diaa.

## Glossar

Begriff	Erklärung
Performance	Geschwindigkeit der Software.
JRE	Java Runtime Environment. Ein Stück frei erhältliche Software, die es Ermöglicht Java l