Contents

1 Zielbestimmung **3**

1.1 Musskriterien 3

1.2 Wunschkriterien 6

2 Produkt–Einsatz **7**

2.1 Anwendungsbereich 7

2.2 Zielgruppen 7

2.3 Produktumgebung 7

2.3.1 Softwareanforderungen 7

2.3.2 Hardwareanforderungen 7

2.4 Betriebsbedingungen 8

3 Produktfunktionen **9**

3.1 Funktionale Anforderungen 9

3.1.1 Benutzeroberfläche 9

3.1.2 21

3.1.3 Datenverarbeitung 21

3.1.4 Datenspeicherung 21

3.2 Nichtfunktionale Anforderungen 21

3.2.1 Performance 21

3.2.2 Zuverlässigkeit 21

4 Testszenarien **22**

4.1 UI 22

4.2 Verarbeitung 22

4.3 Speichern 22

4.4 Performance 22

4.5 Benutzbarkeit (Schimpanse benötigt) 22

4.6 Stabilität 23

5 Entwicklungsumgebung **23**

5.1 Verwendete Software 23

5.2 Verwendete Hardware 23

5.3 verwendete Organisation 23

1 Zielbestimmung

1.1 Musskriterien

Das Programm soll dazu dienen, Zelluläre Automaten auf einem 2D 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	Es soll ein zellulärer Automat mit möglichst großer Freiheit definiert und simuliert werden können.
M0003	Darstellung Spielfeld	Die Darstellung des Zellulären Automaten erfolgt über eine 2D Matrix aus Quadraten deren Farbe und Helligkeit den Zustand eines Feldes wiedergeben.
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 2D 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 seperat 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 gegenüberliegen 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.
M0014	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.
M0015	Startbedingungen	Beim Programmstart soll ein 80x80 Zellen großes Spielfeld präsentiert werden, auf welches die Spielregeln für Conway's Game of Life verwendet werden.

M0016	Nachbarschaftswahl	Die Verwendung von sowohl Moore als auch Neumann Nachbarschaft muss ermöglicht werden. Am Rand einer endlichen Fläche können nicht alle Nachbarn existieren und werden auch nicht gezählt.
M0017	Numerische Anzeige des Zellzustandes	Der genaue Wert einer Zelle muss irgendwie anzeigbar sein.

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 Zustönde 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

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

2.3.2 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.
- Farbmonitor mit ausreichend Nutzfläche mindestens 80 ∗80 Pixel.
  Empfohlen wird ein HD Ready Display mit 720∗1280 Pixeln.
- Maus die mit Links.–/ Rechtstasten und einem Mausrad bestückt ist.
- Tastatur

2.4 Betriebsbedingungen

- Schreib– und Leserechte für die Speicherstände.
- Verfügbarer Speicherplatz (großzügigerweise werden 500 MB Festplattenspeicher empfohlen).
- Arbeitsspeicher angepasst an die Feldgröße (Standardkonfiguration benötigt 128 MB).

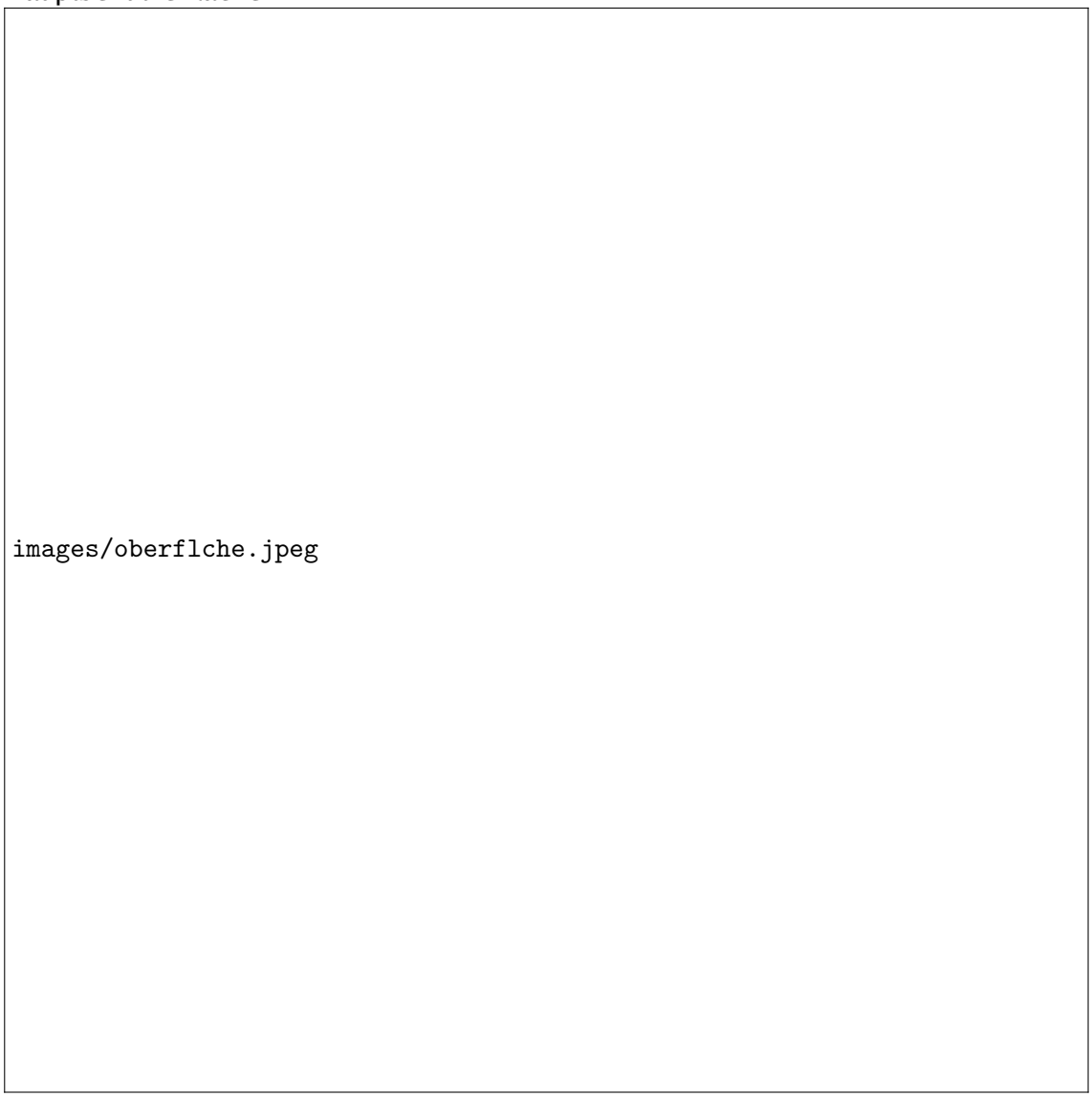
3 Produktfunktionen

3.1 Funktionale Anforderungen

3.1.1 Benutzeroberfläche

Nach dem Start wird folgende Oberfläche als Standard erscheinen. Im folgenden werden die (numerierten) UI–Elemente erläutert.

Hauptbenutzerfläche



AF–01	Spielfeldeditor	Mausklick auf den ”Spielfeldeditor– Button” öffnet das Dropdownmenü Spielfeldeditor.
AF–02	Regeleditor	Mausklick auf den ”Regeleditor– Button” öffnet das Dropdownmenü Regeleditor.

AF–03	Undo/Redo	Mausklick auf ”undo” macht die letzte Eingabe des Spielers rückgängig. ”Redo” stellt sie wieder her (Hochoptional).
AF–04	Simulation starten / unterbrechen	Mausklick auf den Button schaltet die automatische Simulation an oder aus.
AF–05	Stepover	Mausklick 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 Spielfeldeditor–Button
Nachbedingung Erfolg	Öffnen des Fensters ”Spielfeldeditor
Nachbedingung Fehlschlag	Ausgeben der programminternen Fehlermeldung in Dialogfenster.
Ablauf	Nutzer klickt auf Button und das entsprechende Fen–ster 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 Erfolgt	Öffnen des Fensters ”Regeleditor”
Nachbedingung Fehlschlag	Ausgeben der programminternen Fehlermeldung im Dialogfenster.
Ablauf	Nutzer klickt auf den ”Regeleditor– 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 Erfolgt Redo: Wiederherstellen (Hochoptional)	Undo: Rückgängig machen der zuletzt ausgeführten Aktion.
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 Erfolgt	Umschalten der Simulation zwischen ”Simulation läuft” und ”Pausiert”
Nachbedingung Fehlschlag	Ausgeben der programminternen Fehlermeldung im Dialogfenster.
Ablauf	Simulation gestoppt: User klickt auf Button, Simulation startet. Hintergund des Knopfes wird Grün. Simulation läuft: User klickt auf Button, Simulation stoppt. Hintergrund des Knopfes wird Grau.

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 Erfolgt	Simulieren und anzeigen der nachfolgenden Spielfeldgeneration
Nachbedingung Fehlschlag	Ausgeben der programminternen Fehlermeldung im Dialogfenster.
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 Erfolgt	Anpassung des Simulationsschritt–Delays
Nachbedingung Fehlschlag	Ausgeben der programminternen Fehlermeldung im Dialogfenster.
Ablauf	Nutzer zieht den Slider auf eine Stelle im Intervall von 0–5 Sekunden

Anwendungsfall ID	AF–07
AF Name	Zellmodifikation
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, zu editierende Zelle mit Linksklick ausgewählt, Benutzer lebendig und mindestens leicht angetrunken. Pegel wird durch KI analyse der Mausbewegungen ermittelt.
Auslösendes Ereignis	Mausklick auf das Zustandstextfeld oder auf das Spielfeld
Nachbedingung Erfolgt	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 folgt eine Fehlermeldung und setzen auf 0 (zur Sicherheit)
Ablauf	Textfeld: Nutzer klickt auf Textfeld. Nutzer gibt ein, welcher Zielzustand (32Bit signed Integer) 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 bear–beiten auf.

Anwendungsfall ID	RF–01
AF Name	Laden
Akteur	Benutzer des Programms
Vorbedingung	Programm gestartet, Benutzer lebendig, Dropdown–menü ”Regeleditor” ausgewählt
Auslösendes Ereignis	Mausklick auf den ”Laden–Button”
Nachbedingung Erfolgt	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 Erfolgt	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, Dropdown–menü ”Regeleditor” ausgewählt
Auslösendes Ereignis	Mausklick auf den ”Topologie– Radio– Button”
Nachbedingung Erfolgt	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 Erfolgt	Öffnen des Popupfensters ”Regeleditor” (siehe oben).
Nachbedingung Fehlschlag	Fehlermeldung und Rückkehr zur Haupt–Oberfläche.
Ablauf	Öffnen des ”Übergangsregel–Editors”. Beim 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 diese Weise kann der Zustand der akuell betrachteten Zelle für die nächste Iteration auf Basis ihrer Nachbarn berechnet werden

Regeleditor

SE–01	Laden	Ruft Filechooser auf, worüber ein bereits gespeichertes Spielfeld geladen wird.
SE–02	Einfügen	Einen kleineren Spielfeldzustand in das Aktuelle wird an einer beliebigen Stelle eingefügt.
SE–03	Speichern	Aktueller Zustand des Spielfeldes wird in einer Datei gesichert.
SE–04	Größe	Dimensionen: Das erste Eintragskästchen gibt die Breite an, das Zweite die Höhe des gewünschten Spielfeldes.
SE–05	Anwenden	Das aktuelle Spielfeld wird auf die gewünschten Dimensionen gebracht.

SE–06	Zufallsgenerator	Werkzeug um das Spielfeld mit Zufälligen werten zu füllen.
SE–07	Clear	Generiert ein leeres Spielfeld.

Anwendungsfall ID	SE–01
AF Name	Laden
Akteur	Benutzer des Programms
Vorbedingung	Spielzustand über den ”Play/ Pause– Button” pausieren. Hintergrundfarbe des Buttons färbt sich grau.
Auslösendes Ereignis	Mausklick auf den ”Laden–Button”
Nachbedingung Erfolgt	Öffnen des Java– Swing– Fenster mit Filechooser zum öffnen eines gespeicherten Spielfeldes
Nachbedingung Fehlschlag	Ausgeben der programminternen Fehlermeldung im Dialogfenster.
Ablauf	Nutzer klickt auf den Laden–Button und wählt die jeweilige Datei aus, die in das Spiel eingebunden werden soll.

Anwendungsfall ID	SE–02
AF Name	Einfügen
Akteur	Benutzer des Programms
Vorbedingung	Spielzustand über den ”Play/ Pause– Button” pausieren. Hintergrundfarbe des Buttons färbt sich grau.
Auslösendes Ereignis	Mausklick auf den Einfügen–Button
Nachbedingung Erfolgt	Öffnen des Fensters ”Einfügen”
Nachbedingung Fehlschlag	Ausgeben der programminternen Fehlermeldung im Dialogfenster.
Ablauf	Nutzer klickt auf den Einfügen–Button und wählt aus dem Pop–Up–Fenster den jeweiligen Spielzustand aus, der in das Spielfeld eingebunden werden soll.

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 Testszenarien

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. ”abeuiae” ist keine valide Spielfeldgröße. Ferner muss geprüft werden, ob alle Knöpfe ausschließlich das tun, was sie sollen und nicht spaßige Nebeneffekte erzeugen. Knöpfe, die was am Spielfeld ändern, dürfen während der Simulation nicht betätigbar sein. Man muss fähig sein Zustände wieder zu verlassen, ohne das Programm zu beenden. Überschreiten des Wertebereiches von int darf nicht zum Absturz oder unvorhersehbarem Verhalten führen.

4.2 Verarbeitung

Der Regelinterpreter bekommt den Regelsatz für Game of Life und es werden bekannte For–mationen eigegeben und geschaut, ob diese sich auf Langzeit stabil verhalten. Es wird eine Regelsatz entworfen, welcher zu Überschreitung des Zahlenbereichs int führt. Das Programm muss weiter Funkitionieren. Es wird empirisch getestet ob bei dem Zufallsgenerator auss–chließlich erwünschte Zustände mit jeweiligen Verteilungen generiert werden. Die Funktion der Topologien wird überprüft.

4.3 Speichern

Der Java Filechooser wird wohl funktionieren, nicht wahr?
Zugriffsfehler (Rechte oder Speicherplatz bedingt) müssen abgefangen werden.

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

4.4 Performance

Flüssige Interaktion mit dem UI ist gewährleistet, wenn die Hardwareanforderungen (s. 2.3.3) eingehalten werden.

4.5 Benutzbarkeit (Schimpanse benötigt)

Ausflug in den Zoo. Laptop in den Käfig.

4.6 Stabilität

Langzeittest mit Testperson.

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 Hubraumk–lassen.

5.3 verwendete Organisation

Haben Sie wirklich den Eindruck, dass hier irgendwas organisiert abläuft? Aber gut, ein Ver–such: 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 je–mand Photoshop macht, dann Diaa.

Glossar

Begriff	Erklärung
Performance	Geschwindigkeit der Software.
JRE	Java Runtime Environment. Ein Stück frei erhältliche Software, die es Ermöglicht Java Programme auszuführen.
.csv	”Comma separated values“ simples Tabellen–Dateiformat. Trennung von Spalten durch Kommata und Zeilen durch Umbrüche.
.txt	Dateiendung für Textdateien.
Zellulärer Automat	Ein Konzept zur Modellierung dynamischer Systeme. Zellen die eine bestimmte Menge von zuständen einnehmen können befinden sich in einem Raum. Die Räumlich nächsten Zellen bilden die Nachbarschaft. Aus dem eigenen Zustand und dem der Nachbarn ergibt sich über eine Transitionsregel der Folgezus–tand.
Transisitonsregel	Vorschrift die unter Verwendung vorhandener Daten den Zustand einer Zelle in den Nächsten überfuhrt.
Simulation	Dynamische Abbildung, meist realer Sachver–halte, anhand eines Modells, durch Anwen–dung des Modells über Zeit.
Spielfeld	Ein abgegrenztes 2D Feld aus Zellen.
Zelle	Ist ein zellular Automat (Zelle), die zwei Zustände einnehmen kann: lebendig oder tot.
Zustand	Zelle befindet sich im Zustand: lebendig oder tot.
Nachbarn	
Moor Nachbarnschaft	
Neumann Nachbarschaft	
Hauptbenutzeroberfläche	Dies ist die Oberfläche mit der sich der Be–nutzer und der Computer interagieren.

Spielfeldeditor	
Regeleditor	
Invers Polnische Notation	Beste Erfindung der Welt. Klammerfreie Nota–tion von Algorithmen, die auf einem Stack aus–geführt werden können.
Syntax	Regeln zur Anordnung und Reihenfolge von Zeichensystemen
Multithreading	