Kiek
Eine Modellierungs- und Simulationsumgebung für Hierarchische Asymmetrische Zellulare Automaten

Jörg Buchholz[1], Benjamin Marwede[1], Timo Bunger, Wilhelm Hasselbring, Andre Hitzschke, Jasminka Matevska-Meyer, Heiko Müller, Achim Olker, Christian Oppermann, Matthias Rohr, Michael Rudner, Michael Sonnenschein, Ingo Stierand, Matthias Uflacker

Carl von Ossietzky Universität Oldenburg, Fachbereich Informatik, 26111 Oldenburg

Zusammenfassung:

Zellulare Automaten (ZA) bieten eine einfache Möglichkeit, räumliche Prozesse zu modellieren. Mit ihrer Hilfe wird deutlich, wie durch einfache Regeln komplexes Verhalten entstehen kann, wie es in der Natur zu finden ist. Allerdings lassen sich viele reale ökologische Systeme nur mit Mühe in dieses Modell überführen, da durch eine Rasterung des Gebietes wichtige räumliche Informationen verloren gehen. Hierarchische Asymmetrische Zellulare Automaten (HAZA) sind eine auf ZA basierende, an der Universität Oldenburg entwickelte Modellierungsmethode mit dem Ziel, den erweiterten Anforderungen von (landschafts-)ökologischen oder sozio-ökonomischen Systemen gerecht zu werden.
Im vorliegenden Aufsatz soll zum einem das Konzept, welches HAZA zugrunde liegt, erläutert und zum anderen unsere auf HAZA basierende Modellierungs- und Simulationsumgebung "Kiek" vorgestellt werden.

1 Einleitung

Im Bereich der Umweltwissenschaften stellt sich immer wieder die Aufgabe, räumliche Strukturen bei der Modellierung eines Systems zu berücksichtigen. Dies gilt im Besonderen für Ausbreitungsmodelle, wie beispielsweise die Entwicklung von Städten oder die natürliche Verbreitung von Tier- und Pflanzenpopulationen. Gerade hier ist der Raumbezug von hoher Bedeutung, will man Wechselwirkungen zwischen unterschiedlichen Landnutzungen untersuchen. Außer von der räumlichen Entfernung hängen solche Wechselwirkungen auch stark von der Form der betrachteten Gebiete ab. Längliche Grundstücke beeinflussen eine größere Anzahl Gebiete als quadratische. Häufig findet eine Wechselwirkung zwischen zwei Gebieten nur indirekt statt. So beeinflusst die Düngung einer Weide nicht direkt die angrenzenden Areale. Eine Beeinträchtigung findet erst durch Versickerung der Düngemittel in den Untergrund statt, wo es durch Wasser ausgeschwemmt und verteilt wird. Es ist einsichtig, dass dies eine gesonderte Modellierung des Untergrunds notwendig macht.

Gesucht wird folglich eine Möglichkeit, räumliche Beziehungen zwischen asymmetrisch geformten Gebieten zu berücksichtigen. Genau diese Bedingungen werden durch Hierarchische Asymmetrische Zellulare Automaten (HAZA) [SV03] erfüllt. Sie

[1] E-mail: {joerg.buchholz,benjamin.marwede}@informatik.uni-oldenburg.de

bieten darüber hinaus die Möglichkeit, Prozesse auf unterschiedlichen Raum- und Zeitskalen in Modellen methodisch zu trennen.

Bereits vor einiger Zeit wurde das Framework EcoScape [Spe02] entwickelt, welches die computergestützte Nutzung von HAZA ermöglicht. Die Benutzungsoberfläche Kiek, die derzeit im Rahmen einer Projektgruppe an der Carl von Ossietzky Universität Oldenburg realisiert wird, setzt auf EcoScape auf und realisiert die Schnittstelle zwischen Anwender und dem eigentlichen Framework. Dieses Dokument soll Kiek näher erläutern. Dabei werden zunächst für das Verständnis wichtige Grundlagen geklärt, um daraufhin Entscheidungen und Konzepte von Kiek aufzuzeigen. Den Abschluss bildet ein Ausblick über mögliche Erweiterungen.

2 Grundlagen

2.1 Zellulare Automaten

Zellulare Automaten [Wol86, TM87] stellen einen klassischen Ansatz dar, zum Zwecke einfacher Modellierung Raum, Zeit und Eigenschaften diskret zu strukturieren. Am bekanntesten ist die zweidimensionale **Form** mit quadratischen Zellen, tatsächlich gibt es aber keine Begrenzung in der Anzahl der **Dimensionen**. Auch die Anzahl der Attribute pro Zelle, die technisch gesehen ebenfalls je eine Dimension darstellen, ist beliebig. Die Zellen können im Prinzip beliebige regelmäßige Formen annehmen.

Eine solche Struktur kann auf verschiedene Weisen aufgefasst werden: Entweder als über den Raum gespanntes Gitter, an dessen Kreuzungspunkten zu bestimmten Zeitpunkten Messwerte (Samples) gewonnen werden, oder als lückenlose Aggregation von Flächen, wobei für jede Teilfläche ein einheitlicher Zustand gilt.

Zustandsübergänge eines zellularen Automaten werden im Allgemeinen von Rechenmaschinen mit Hilfe von Differenzen- oder Differenzialgleichungen berechnet. Häufig genügen allerdings sehr einfache Regeln. Jede einzelne Zelle ist dann ein ”endlicher Automat”. Kurz gesagt hat ein solcher Automat eine beschränkte Anzahl von möglichen Zuständen, die in diskreten Zeitschritten ineinander übergehen. Im Falle der zellularen Automaten können diese Übergänge in Abhängigkeit sowohl vom jeweils eigenen Zustand als auch vom Zustand anderer Automaten (= Zellen) geschehen.

Wesentlich ist die sogenannte **Nachbarschaftsregion** einer Zelle. Das sind jene anderen Zellen, die einen Einfluss auf die Zelle ausüben, deren Zustand also in die Berechnung eines neuen Zustandes eingeht. Diese Nachbarschaftsregion muss nicht zwingend aus den direkt angrenzenden Zellen bestehen.

Traditionell ist die **Taktung** global, d.h. der Zustand jeder Zelle wird synchron in regelmäßigen Zeitschritten neu berechnet. Es ist jedoch prinzipiell auch eine ereignisbasierte Steuerung mit Hilfe von Warteschlangen auszuführender Zustandsübergänge möglich.

Weiterhin bedürfen die **Ränder** eines zellularen Automaten einer Betrachtung. Der modellierte Raum kann endlich oder (theoretisch) unendlich sein, und im ersteren Falle, gekrümmt oder nicht, in verschiedenen Dimensionen, etwa in Form eines Zylinders oder Torus' im zweidimensionalen Raum.

Weitere Informationen zur Verwendung zellularer Automaten in den Umweltwissenschaften und insbesondere in der Ökologie finden sich etwa in [Hog88, Wit00].

2.2 Erweiterungen: Asymmetrie und Hierarchie

"Klassische" zellulare Automaten eignen sich nicht für jeden Zweck räumlich expliziter Modellierung von Umweltsystemen optimal. Sie sind zwar einfach zu realisieren und zu analysieren. Räumlich heterogene Strukturen sowie Prozesse auf unterschiedlichen Raum- und Zeitskalen lassen sich auf diese Weise jedoch nicht einfach, wenn überhaupt, modellieren.

In [SV01, SV03] werden zellulare Automaten zunächst um die Möglichkeit der Definition asymmetrisch geformter Zellen und irregulärer Verbindungsstrukturen erweitert. Außerdem wird vorausgreifend eine "globale Ebene" eingeführt, um globale veränderliche Information für alle Zellen bereitzustellen.

Konkrete Veränderungen im Vergleich zu "klassischen" zellularen Automaten sind:

- Der Raum wird (nicht zwingend lückenlos) in paarweise disjunkte Flächen zerlegt, d.h. ein Punkt im Raum gehört zu maximal einer Zelle. Diese Zellen sind durch Polygonzüge begrenzte, beliebig geformte Flächen.

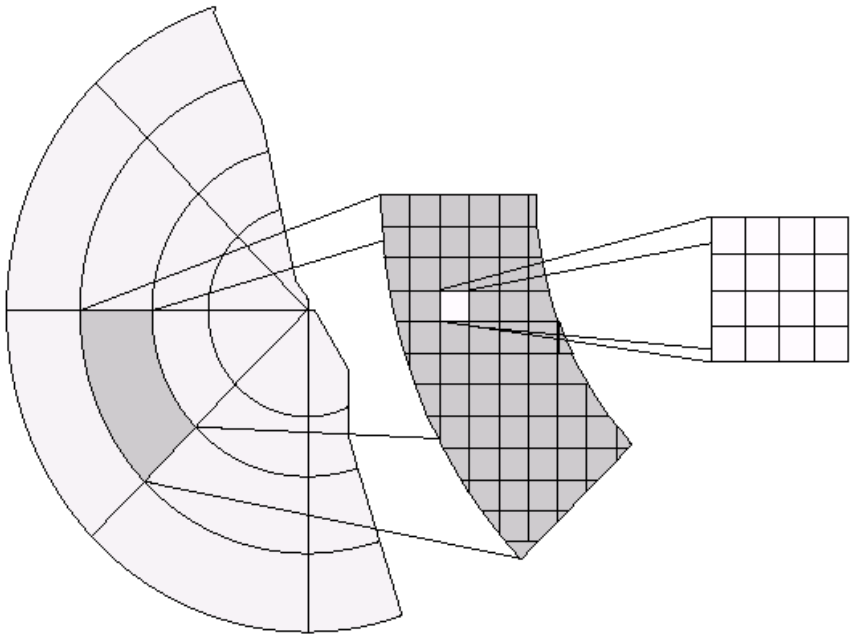

Abbildung 1: Struktur eines HAZA

- Zellen exportieren Zustandsinformationen an ihre Nachbarn aktiv. (Klassisch werden stattdessen die Nachbarn von Zellen ”beobachtet”.)
- Die Nachbarschaftsrelationen und deren Grade können frei definiert werden. Sie legen fest, mit welcher Stärke die Einflüsse benachbarter Zellen für die betrachtete Zelle im nächsten Zustand berücksichtigt werden müssen.

Die Autoren zählen weiterhin

- einen synchronen Zeitfortschritt und
- die Homogenität der Zellen bezüglich der für das Modell relevanten Systemeigenschaften

zu den typischen Eigenschaften eines Asymmetrischen Zellularen Automaten (AZA).

Die erwähnte ”globale Ebene” kann wiederum als Zelle eines übergeordneten Automaten aufgefasst werden. Auf diese Weise lässt sich das Prinzip rekursiv bzw. baumartig auf mehrschichtige Automaten verallgemeinern. Ein **Hierarchischer Asymmetrischer Zellularer Automat** (HAZA) besteht aus einer endlichen Anzahl von Ebenen (auch Schichten oder Layers genannt), die jeweils durch einen Asymmetrischen Zellularen Automaten beschrieben werden. Die Zellen bilden so eine baumförmige Struktur mit einem einzelnen AZA als Spitze (bzw. "Wurzel" im Informatik-Jargon). Allen Automaten ist eine Zeitbasis gemein, jedoch nicht notwendigerweise ein gemeinsamer Takt. Zwischen den Ebenen sorgen sogenannte Aggregations-Funktionen für die Abbildung der Zellenstrukturen der Automaten aufeinander.

2.3 Das EcoScape-Framework

Das EcoScape-Framework ist im Rahmen einer Diplom-Arbeit der Abteilung Umweltinformatik an der Carl von Ossietzky Universität Oldenburg entstanden [Spe02]. Ziel des letztlich in C++ umgesetzten Projektes war die Entwicklung eines Frameworks, das die Hierarchischen Asymmetrischen Zellularen Automaten vollständig und korrekt abbilden und berechnen kann. EcoScape umfasst dabei sowohl Mechanismen zur Modellbildung als auch zur Simulation.

Wichtig sowohl für die direkte Implementierung eines Computermodells als auch für die Entwicklung eines interaktiven Modellierwerkzeuges war die vollständige Modellierbarkeit zur Laufzeit. Dies erforderte für die funktionalen Elemente der Modellstruktur Mechanismen zur Spezifikation der Berechnungsvorschriften.

EcoScape ist in drei Hauptkomponenten eingeteilt:

- Die **Modell**-Komponente umfasst eine Datenstruktur und eine Menge von Methoden, mit denen beliebige HAZA beschrieben werden können. Insbesondere sind Eigenschaften, wie z.B. das Vorhandensein mehrerer Ebenen, sowie die besonderen Zellattribute und -geometrien berücksichtigt.
- Aufgabe der **Simulator**-Komponente ist hingegen, ein übergebenes Modell in diskreter Zeit und diskretem Raum zu simulieren, indem sie für alle Zellen ei-

ner bestimmten Ebene zu gegebenen Zeitpunkten den Automatenzustand nach den bestehenden Regeln neu berechnet.

- Die **Datenbank**-Komponente erlaubt es, Informationen aus beliebigen Geo-Informations-Systemen zu importieren und die gewonnenen Simulationsergebnisse abzulegen.

3 Entwurfsentscheidungen

In diesem Kapitel sollen die unterschiedlichen Entscheidungen, die bei der Implementierung der Benutzungsoberfläche Kiek getroffen wurden, vorgestellt werden. Dabei geht es weniger darum, jedes Feature bis ins Detail zu erläutern, als vielmehr darum, einen umfassenden Überblick zu geben.

Ein sehr wichtiger Aspekt bei der Entwicklung der Benutzungsoberfläche war die Plattformunabhängigkeit, da der angesprochene Nutzerkreis mit sehr unterschiedlichen Betriebssystemen arbeitet. Aus diesem Grund wurde die Oberfläche von Kiek vollständig in Java realisiert. Um dem Benutzer eine gewohnte Arbeitsumgebung zu vermitteln, wurde das Aussehen bekannter Software nachempfunden. Dafür bot sich die Swing-Klassenbibliothek [Swing] an, aus der viele Komponenten übernommen werden konnten.

Wie bereits erwähnt bildet Kiek die Benutzungsoberfläche zum EcoScape-Framework. Aus Performance-Gründen wurde dieses in C++ realisiert. Somit galt es, eine Kommunikationsmöglichkeit zwischen Java und C++ zu schaffen. Um dieses zu realisieren, wurde JNI [JNI] als beste Alternative ausgemacht. Besonders in punkto Kommunikationsgeschwindigkeit ist es gegenüber Konkurrenzlösungen, wie etwa SOAP [SOAP], weit überlegen.

Entscheidend für die Akzeptanz von Kiek ist eine komfortable Bedienung der Benutzungsoberfläche. Wichtig hierfür ist vor allem die Art, wie Parameteränderungen sichtbar und nachvollziehbar gemacht werden. Deegree [Deegree], eine Bibliothek speziell zur Visualisierung von Polygonen, erfüllt all diese Anforderungen und bietet zusätzlich Funktionen, die aus herkömmlicher Bildbetrachtungssoftware bekannt sind. So ist es beispielsweise möglich, an Zellen heran zu zoomen, um Details besser zu erkennen.

Da asymmetrische Zellen, wie der Name schon andeutet, eine beliebig hohe Formvarianz aufweisen, ist es für eine sinnvolle Anwendung unabdingbar, eine geeignete Importfunktion für solche Zellen anzubieten. Nur so ist es dem Anwender möglich, die von ihm gewünschte Zellform bei der Modellierung zu berücksichtigen. Kiek bietet für diesen Fall die Möglichkeit, GIS-Karten in Form von ShapeFiles zu importieren. Solche GIS-Karten lassen sich mit entsprechender Software, beispielsweise ArcGis, einfach und komfortabel erstellen – eine wichtige Bedingung für die Benutzbarkeit von Kiek.

Die aufgeführten Entscheidungen bilden die Grundlage für die Funktionalität von Kiek. Ihr Einfluss wird insbesondere bei den Konzepten der Benutzungsoberfläche deutlich, die im folgenden Kapitel vorgestellt werden sollen.

4 Konzepte der grafischen Benutzungsoberfläche

Bei der Entwicklung einer grafischen Benutzungsoberfläche für das EcoScape-Framework mussten unterschiedlichste Ziele miteinander vereint werden. So ist Kiek beispielsweise für die Nutzung durch einen sehr heterogenen Anwenderkreis ausgelegt. Daraus resultiert, dass Parametereingaben nicht zu restriktiv gewählt werden dürfen, um Experten nicht zu bremsen - aber auch nicht zu frei, um Anwender ohne tiefere „PC-Kenntnisse“ nicht zu überfordern. Es galt somit, Flexibilität, Einfachheit und Sicherheit unter einer Oberfläche zu vereinen.

4.1 Assistenten

Der Entwicklungsprozess innerhalb von Kiek wird vollständig durch Assistenten unterstützt. Damit ist es auch unerfahrenen Benutzern möglich, in kürzester Zeit Modelle oder Szenarien zu entwickeln, ohne dass wichtige Einstellungen übersehen werden. Assistenten vereinfachen komplexe und häufig fehlerträchtige Parametereingaben und ermöglichen dadurch dem Entwickler, sich auf seine eigentliche Tätigkeit, die Entwicklung, zu konzentrieren. Der Ablauf der einzelnen Assistenten wurde bewusst einfach und vor allem stets ähnlich zueinander gestaltet.

Besondere Aufmerksamkeit verdient die Eingabemaske zur Definition der Zustandsübergangsfunktion. Hier wird die gleichzeitige Berücksichtigung erfahrener und

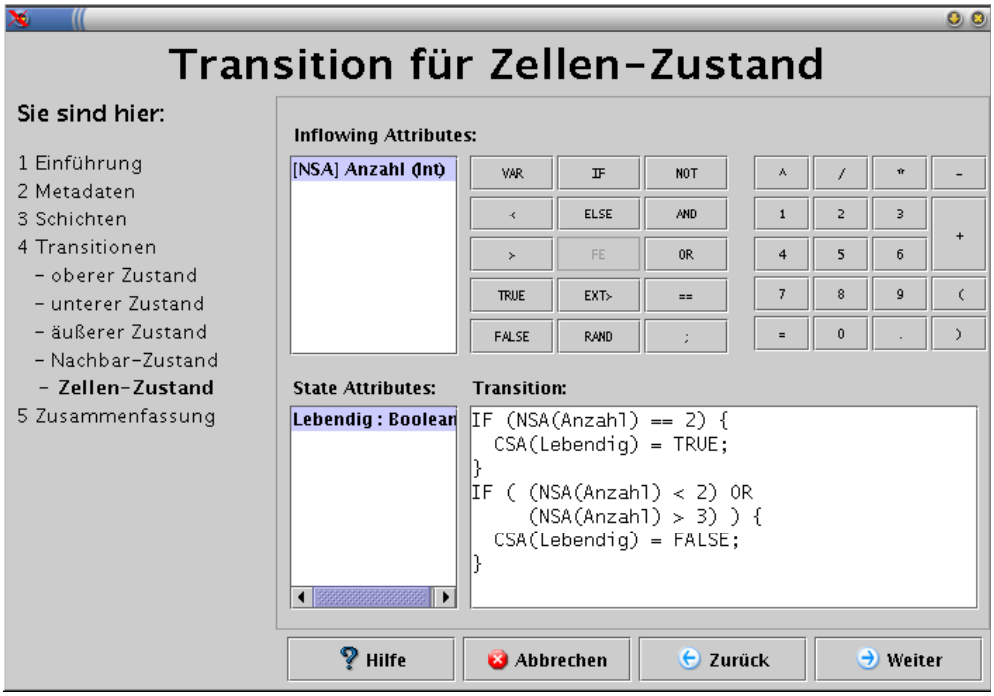

Abbildung 2: Eingabe einer Zustandsübergangsfunktion

unerfahrener Entwickler besonders deutlich. Der unerfahrene Anwender kann durch einfaches Klicken eine vollständige Zustandübergangsfunktion zusammenstellen, während dem erfahrenen Anwender die Eingabe von komplexen Programmstrukturen in einer eigenen Skriptsprache per Tastatur ermöglicht wird.

Abbildung 2 zeigt die Eingabemaske für die Zustandübergangsfunktion. Zu erkennen ist neben den typischen Navigationselementen eines Assistenten auch ein Beispiel für die erwähnte Skriptsprache.

4.2 Projektbaum

Der in Kiek enthaltene Projektbaum dient der übersichtlichen Darstellung von Projekten. Über die entwickelte Baumstruktur kann der Benutzer alle aktuell geladenen Experimente, Modelle und Szenarien einfach und schnell verwalten, wie es auch beispielsweise schon im Werkzeug Meta-X [FL03] der Fall ist.

Die Baumdarstellung spiegelt die Hierarchie, in der Experimente, Modelle und Szenarien angeordnet sind, wider. Wie in Abbildung 3 zu erkennen ist, werden die verschiedenen Einträge im Baum durch eindeutige und intuitive Icons symbolisiert. Somit lassen sich Experimente, Modelle und Szenarien bereits optisch unterscheiden.

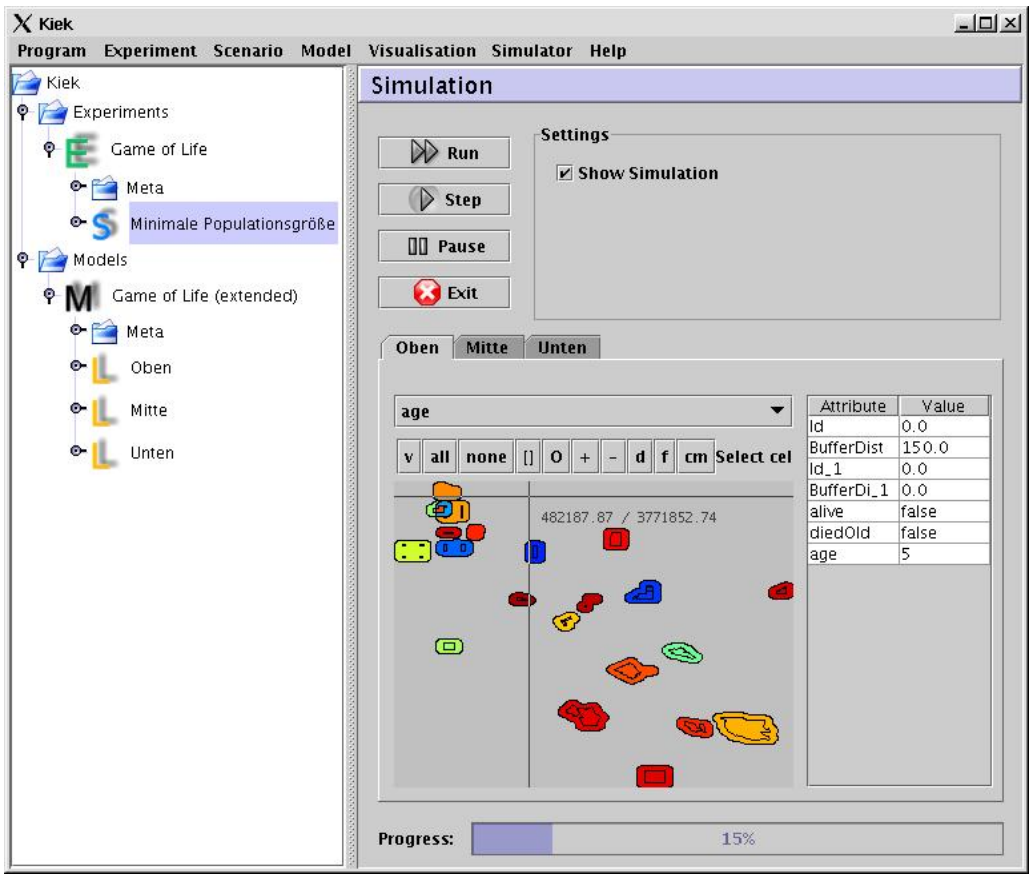

Abbildung 3: Projektbaum und Simulationsfenster

Jeder Knoten im Baum verfügt über ein eigenes Kontextmenü. Über dieses lassen sich alle wichtigen Funktionen von Kiek, wie beispielsweise das Erstellen oder Editieren eines Modells, direkt ansteuern. Zudem ermöglicht es den Zugriff auf detaillierte Informationen des durch den Knoten repräsentierten Objekts.

4.3 Interaktiver Simulator

Die innerhalb von Kiek erstellten Szenarien können simuliert werden. Dabei steht neben einer automatischen auch eine schrittweise Simulation zur Verfügung, wobei der aktuelle Simulationsstand auch grafisch angezeigt werden kann (Abbildung 3).

Da HAZA üblicherweise aus mehreren Ebenen bestehen, kann zwischen diesen Ebenen ausgewählt werden. Die Darstellung des Zustands der einzelnen Zellen einer Ebene geschieht durch eine geeignete Farbwahl. Diese ist durch den Benutzer bestimmbar und verbessert dadurch das Verständnis für die aktuelle Entwicklung der Simulation. Durch die Visualisierung der Zustände gewinnt die interaktive Simulation eine große Bedeutung bei der Validierung des entwickelten Modells. Wird bereits bei der Simulation eine Fehlentwicklung festgestellt, so kann die Simulation abgebrochen und das Modell verbessert werden.

Die Simulationsdaten werden in sogenannten Zeitreihen abgelegt. Diese werden vor der Simulation vom Benutzer erstellt (Abbildung 4) und ermöglichen durch Setzen von einfachen, aber sehr flexiblen Filtern die exklusive Speicherung der gewünschten Simulationsdaten. Nach Beendigung der Simulation kann der Benutzer bestimmen, welche Zeitreihen er behalten möchte. Diese werden dann unterhalb des simulierten Szenarios abgelegt und können ausgewertet werden.

4.4 Auswertung

Kiek beschränkt sich bei der Auswertung der Simulationsergebnisse auf die Darstellung der zeitlichen Entwicklung ausgewählter Zustandsattribute. Dafür stehen diverse Diagramme zur Verfügung, unter denen der Benutzer das für seinen Zweck sinnvollste Diagramm aussuchen kann. Es lässt sich schnell ein erster Eindruck über die Simulationsergebnisse und damit über die Qualität des Modells gewinnen.

Häufig möchte der Anwender die Simulationsergebnisse weitergehend analysieren. Für diesen Fall stellt Kiek eine flexible Exportfunktion bereit. Neben dem verbreiteten CSV-Datenformat, welches unter Kiek frei gestaltet werden kann, wird auch XML unterstützt. Somit lassen sich die Simulationsergebnisse ohne Probleme in Analyse- bzw. Statistikwerkzeuge, wie beispielsweise "R" [R], exportieren und dort auswerten.

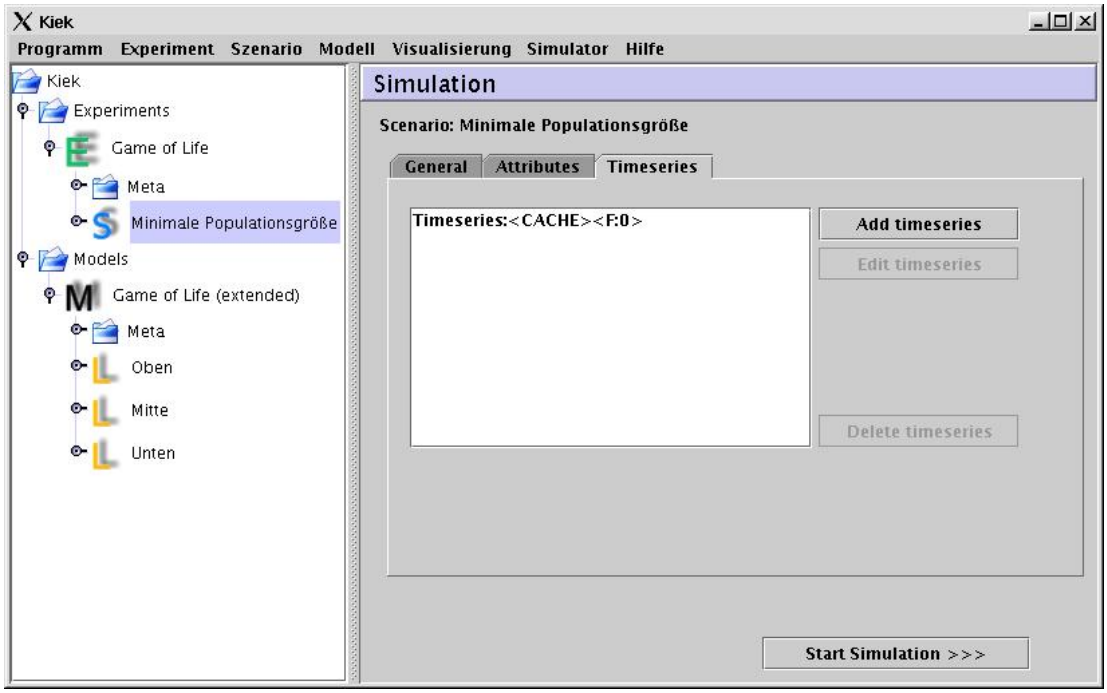

Abbildung 4: Konfiguration der Zeitreihen vor der Simulation

5 Ausblick

Mit Kiek besteht eine einfache und anwenderbezogene Möglichkeit, Szenarienvergleiche auf der Grundlage von HAZA werkzeugunterstützt durchzuführen. Kiek befindet sich im Endstadium der Realisierung, bietet jedoch noch einige Erweiterungsmöglichkeiten, die im Rahmen der Projektgruppe nicht mehr vollständig realisiert werden konnten. Eine der möglichen Erweiterungen ist die Schaffung einer Benutzungsoberfläche auf Grundlage des Struts-Frameworks [Struts]. Dies erlaubt den Zugriff des Benutzers auf einen Kiek-Server über einen Browser mit Hilfe von Servlets.

Geplant ist außerdem eine ausführliche Validierung des gesamten Programmpaketes anhand des Beispielmodells [Bie00], in dem die Ausbildung und das Überleben von Metapopulationen in einer räumlich strukturierten Umwelt untersucht werden.

Da Kiek im Rahmen einer Projektgruppe entstanden ist, endet die Entwicklung mit dem Abschluss der Gruppenarbeit im März 2004. Eine Fortführung in nachfolgenden BSc- und Diplom-Abschlussarbeiten ist geplant.

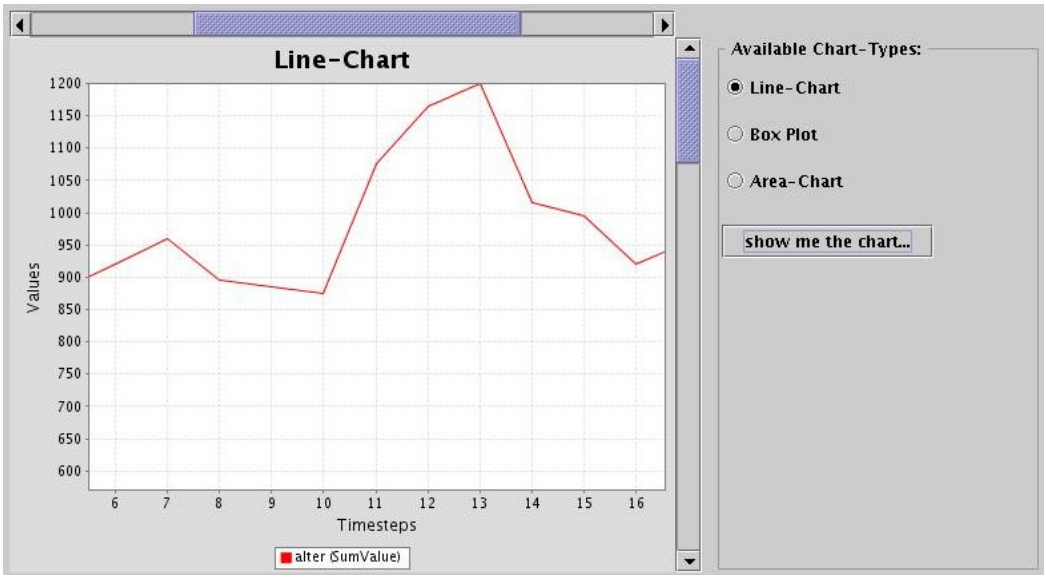

Abbildung 5: Visualisierung

Literaturangaben

[Bie00] R. Biedermann. Metapopulation dynamics of the froghopper *Neophilaenus albipennis* (F., 1798) (Homoptera, Cercopidae) - what ist the minimum viable metapopulation size? Journal of Insect Conservation 4: 99-107, 2000.

[Deegree] http://deegree.sourceforge.net/ - referenziert am 01.03.2004.

[FL03] K. Frank, H. Lorek, F. Köster, M. Sonnenschein, Ch. Wissel, V. Grimm. Meta-X - Software for Metapopulation Viability Analysis. Springer Verlag, 2003.

[Hog88] P. Hogeweg. Cellular automata as a paradigm for ecological modeling. Applied Mathematics and Computation, 27:81-100, 1988.

[JNI] http://java.sun.com/j2se/1.4.2/docs/guide/jni/ - referenziert am 01.03.2004.

[R] http://www.r-project.org/ - referenziert am 01.03.2004.

[SOAP] http://www.w3.org/TR/soap12-part1/ - referenziert am 01.03.2004.

[Spe02] B. Speckels. Entwurf und Realisierung eines Frameworks für hierarchische asymmetrischen zellulare Automaten mit Anbindung an räumliche Datenbanken. Diplomarbeit, Universität Oldenburg, 2002.

[SV01] M. Sonnenschein, U. Vogel. Asymmetric Cellular Automata for the Modelling of Ecological Systems. In L. M. Hilty, P. W. Gilgen (Eds.). "Sustainability in the Information Society" 15th International Symposium Informatics for Environmental Protection. Metropolis-Verlag, pp. 631-636, 2001.

[SV03] M. Sonnenschein, U. Vogel. Hierarchische asymmetrische Zellulare Automaten zur Modellierung ökologischer Systeme auf mehreren Skalen. In: A. Gnauck (Hrsg.): Theorie und Modellierung von Ökosystemen. Workshop Kölpinsee 2001, ASIM Mitteilungen AMB 80, Shaker Verlag, pp. 37-50, 2003.

[Struts] http://jakarta.apache.org/struts/ - referenziert am 25.2.2004.

[Swing] http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/package-summary.html - referenziert am 01.03.2004.

[TM87] T. Toffoli, N. Margolus. Cellular Automata Machines. A New Environment for Modeling. Cambridge, MA: MIT Press, 1987.

[Wit00] J. Wittmann. Zellulare Automaten in der Umweltmodellierung. In D.P.F. Möller (Hrsg.), Frontiers in Simulation, Simulationstechnik - 14. Symposium in Hamburg, ASIM, Seiten 45-50, 2000.

[Wol86] S. Wolfram. Theory and Application of Cellular Automata. World Scientific, Singapore, 1986.