UNIVERSITÄT KASSEL

Fachbereich Ökologische Agrarwissenschaften

Diplomarbeit

Integration von individuenbasierten Konzepten in das Dokumentations- und Simulationssystem für ökologische Modelle, ECOBAS

1. Betreuer:
Dr. Joachim Benz

2. Prüfer der Diplomarbeit:
M.Sc. André Hähnke

vorgelegt von:
Jürgen Bierwirth
Sommersemester 2005

Witzenhausen, den 28.03.2005

Ich erkläre hiermit, dass ich diese Arbeit selbständig, ohne fremde Hilfe und unter Benutzung der angegebenen Literatur angefertigt habe.

Witzenhausen, den 28.3.2005

(Jürgen Bierwirth)

„ . . . Mit Begriffen wie Komplexität und Reduktion,
Selbstreferenz und Autopoiesis, oder rekursiv-geschlossene
Reproduktion bei umweltoffener Irritierbarkeit
sind komplizierte theoretische Fragen aufgeworfen, die wir
bei den folgenden Überlegungen nicht ständig im Blick
behalten können . . . “

Niklas Luhmann: *Ökologische Kommunikation*

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung und Stand der Entwicklung

1.1 Ziel der Diplomarbeit

Das Ziel dieser Arbeit ist es, einen Entwurf für ein hinreichend exaktes und umfassendes System zur Erstellung, Spezifikation und Dokumentation individuenbasierter ökologischer Modelle zu erstellen, welches einerseits noch intuitiv erfassbar und nachvollziehbar ist und wodurch sich andererseits vollständig und konsistent alle relevanten Modellinformationen in einer formal exakten, deklarativen Form darstellen lassen. Die Herangehensweise geschieht dabei nicht aus der Sicht des Informatikers bzw. Programmierers, sondern aus der Sicht des Modellierers aus einem angewandten Bereich wie z.B. der Ökosystemforschung, der eine unterstützende Umgebung für eine standardisierte und vollständige Modellbeschreibung verwenden möchte. Es wird hierbei angestrebt, so weit wie möglich die Modellerstellung und -spezifikation auf die eigentlichen Modellierungselemente, die mathematisch-formalen Strukturen und die ökologischen Hintergrundinformationen zu konzentrieren und software- bzw. programmiersprachennahe Konstruktionselemente hiervon zu trennen und in den Bereich des Simulationssystems zu verlagern. Die Umsetzung dieses Entwurfes für den Bereich der individuenbasierten Modelle geschieht in Form der Erweiterung des bestehenden Systems ECOBAS (http://eco.wiz.uni-kassel.de/ecobas.html), welches bereits für verschiedene Modelltypen der „klassischen“ mathemathischen Modellierung[1] einen mathematisch-formalen und ökologisch-deklarativen Rahmen zur Modellerstellung, Dokumentation und Simulation bereitstellt.
Als erste Etappe zu diesem Ziel wurden Erweiterungen der Eingabeschemata für den *ECOBAS Modelling Assistant* definiert. Die Schemata werden in Form von XML-Dokumenttyp-Definitionen (XML-DTDs) für die ECOBAS_XML Modelldateien erstellt. Der *ECOBAS Modelling Assistant* stellt hierbei die entsprechende XML-Editoroberfläche zur Darstellung und Überprüfung der Formate zur Verfügung.

1.2 Hintergrund und Einordnung der Forschungsgebiete der individuen- und agentenbasierten Modellierung

Die Methoden der sogenannten „individuenbasierten“ bzw. der „agentenbasierten“ Modellierung entwickeln sich in mehreren wissenschaftlichen Disziplinen, die erst im Laufe des letzten Jahrzehnts zunehmend miteinander verknüpft wurden: Im Bereich der theoretischen Ökologie

[1] Der Begriff „klassische mathematischen Modellierung“ bezieht sich hierbei auf Modelle, die im Wesentlichen auf etablierten mathematischen Formulierungs- und Analysemethoden mit Hilfe von Differential- bzw. Differenzengleichungen basieren.

und der Ökosystemforschung wurde seit den 80er Jahren in erster Linie der Begriff der individuenbasierten Modellierung geprägt, so wie auch in den häufig zitierten „Basispublikationen“ zu diesem Thema von HUSTON *et al.* (1988) und DEANGELIS und GROSS (1992). In den Forschungsbereichen Soziologie, Ökonomie und Ressourcenmanagement wurde vornehmlich der Begriff agentenbasierte Modellierung verwendet. In beiden Bereichen wird das Konzept des 'bottom-up'- Ansatzes hervorgehoben, bei dem versucht wird, aus der Modellierung der Eigenschaften der einzelnen, unabhängig voneinander agierenden Elemente eines Systems (Individuen/Agenten) durch deren Verhaltensweisen und Interaktionen eine zutreffende Beschreibung des Verhaltens des Gesamtsystems erzeugen zu können. Dieses Konzept wird häufig als neuer Modellierungstyp dem „klassischen“ mathematischen Modellierungsansatz gegenübergestellt, bei dem eine mathematisch-analytische Beschreibung (in Differenzen- bzw. Differentialgleichungen) des beobachteten Gesamtverhaltens von Systemen angestrebt wird. Hierbei werden unterschiedliche Eigenschaften Individuen ausgeblendet oder in Form statistischer Verteilungen eingebracht und das Systemverhalten wird mit Durchschnitts- bzw. Summengrößen simuliert. Weiterhin sind in engem Zusammenhang mit der agentenbasierten Modellierung die Begriffe „Multi-Agenten Modell“ bzw. „Multi-Agenten-System“ zu nennen, die wesentlich in der Informatik geprägt wurden unter Themengebieten wie „Artificial Intelligence“ bzw. „Distributed Artificial Intelligence“ und „Complex Adaptive Systems“ (siehe z.B. FERBER (1999)). Der Begriff „Multi-Agenten-System“ bzw. „Multi-Agenten-Simulation“ wird in den letzten Jahren zunehmend auch im Rahmen der interdisziplinären Zusammenarbeit in den Anwendungsbereichen der Ökonomie, der Verhaltens- und Lernforschung, der Spieltheorie und der ökolologischen Modellierung verwendet (EPSTEIN und AXTELL, 1996; BOUSQUET und LE PAGE, 2004; OECHSLEIN, 2004). Der Begriff „individuenbasiert“ betont stärker die heterogenen Eigenschaften von Individuen innerhalb einer Gruppe, wohingegen der Begriff „agentenbasiert“ die Modellierung der Aktionen bzw. Interaktionen, der Verhaltensweisen, Entscheidungsprozesse und Rollen von Individuen innerhalb eines Systems betont. Der Agenten- und der Individuenbegriff wird nachfolgend in dieser Arbeit synonym, allerdings mit der genannten leicht unterschiedlichen Betonung verwendet.

Eine wesentliche technische Grundlage für die Entwicklung der individuen-/agentenbasierten Modelle stellt die Entwicklung objektorientierter Programmiersprachen und -methoden dar, wie z.B. Java, SmallTalk, Objective C und C++. Zum Teil wurde daher auch für ökologische Modelle der aus der Informatik stammende Begriff der objektorientierten Modellierung verwendet. Es besteht insbesondere bei individuenbasierten Modellen die Problematik, dass das Modellierungskonzept so eng und strukturell mit der verwendeten objektorientierten Programmiersprache verknüpft ist, dass eine Trennung von Modelldokumentation und -konzeption und Quellcode der Software schwierig ist und daher sowohl die Modelldarstellung als auch die wissenschaftliche Validierung durch andere Wissenschaftler schwer möglich ist (GRIMM, 2002; LOREK und SONNENSCHEIN, 1999).

Als weitere Forschungsbereiche, die sich mit individuen-/agentenbasierten Modellen beschäftigen sind die (physikalisch-chemische) Chaostheorie und die soziologische Systemtheorie zu nennen, die grundlegende theoretische Fragestellungen wie die Bildung von Systemstrukturen und -mustern, Systemstabilität und Kommunikationstrukturen behandeln. Verschiedene Aspekte hierzu werden beispielsweise in den Abhandlungen von COVENEY und HIGHFIELD (1990) und von SOLÉ *et al.* (1999) zum Themenbereich der Chaostheorie und Selbstorganisation und von LUHMANN (1990) zum Thema Kommunikation zwischen Individuen und ihrer Umwelt dargestellt.

1.3 Besondere Charakteristika individuenbasierter Modelle im Vergleich zu „klassischen“ mathematischen Modellen

Eine eindeutige Abgrenzung zwischen „traditioneller“ mathematischer Systemmodellierung und individuenbasierter Modellierung ist kaum möglich. Bei jeder Form der Modellierung müssen Grenzen für verschiedene Systeme und Subsysteme festgelegt werden, so dass sich diese jeweils als Einheit verhalten und sich von anderen Systemen und ihrer Umgebung unterscheiden. Die zu modellierenden Einzelsysteme innerhalb eines Gesamtsystems können je nach wissenschaftlicher Fragestellung dabei sehr unterschiedlich definiert werden. Es ist a priori nicht eindeutig, welche Systemeinheiten als Individuen einer spezifisch individuenbasierten Modellierung gewählt werden müssen, auch wenn es z.B. im Fall der Betrachtung einer Tierpopulation naheliegt, die einzelnen Tiere als biologisch vorgegebene Individueneinheiten zu wählen. Werden soziale oder ökologische Systeme modelliert, können je nach Fragestellung funktionale Gruppen (Familie, Berufsgruppe, Population, Pflanzenart etc.) oder jedes einzelne Mitglied der Gruppe als Individueneinheit modelliert werden. Letztlich lässt sich jedes System als Gesamtheit von Subsystemen physikalischer oder funktionaler Art darstellen bis hinunter in den atomaren Bereich (wo sich dann sowohl die Definition von Einheiten als auch die eindeutige, mechanische Betrachtungsweise im unendlich Kleinen verliert). Die gewählte Abgrenzung der einzelnen Systemeinheiten kann also nicht als Kriterium für eine spezifisch individuenbasierte Modellierung ausreichen. Jede Sytemeinheit stellt immer ein „klassisches“ Modell dar mit Merkmalen und Verhaltenseigenschaften, die sich als „Durchschnitt“ oder „Gesamtsumme“ der Subsysteme darstellen.

Typische individuenbasierte Modellierungsansätze lassen sich durch nachfolgende Merkmale charakterisieren, sowie durch die theoretischen Fragestellungen, die mit einem Modell behandelt werden sollen.

1.3.1 Typische Merkmale individuenbasierter Modelle

(a) Definition von Individuenklassen
Ein typisches Element ist Definition von „Klassen“ von Individuen mit jeweils für eine Klasse gemeinsam definierten Einzeleigenschaften und Verhaltensweisen. Die Einzelindividuen innerhalb einer Klasse haben eine identische Definition von Merkmalen und Verhaltensweisen (Aktionen und Reaktionen), können aber individuell unterschiedliche „Parametrisierungen“ aufweisen (z.B. für die Eigenschaften Geburtsgewicht, Farbe, Wachstumspotenzial etc.). Aus den Definitionsklassen werden beliebig viele Inderviduenobjekte (Instanzen) während der Simulation erzeugt, sodass hierdurch die Modellierung relativ großer Anzahlen von Individuen unterstützt wird, die in klassischen Modellierungssystemen eine Schwierigkeit darstellt. Die mögliche Anzahl von Individuen, die in einer Simulation erzeugt und simuliert werden können, ist im wesentlichen nur durch die Leistungsfähigkeit der Hardware begrenzt.
Die Herangehensweise der Definition von Klassen von Individuen gleichen Typs entspricht sowohl der 'Grammatik' objektorientierter Programmiersprachen, als auch der Denkweise in der Biologie (hierarchische Taxonomie). Ebenso entsteht hierdurch die Möglichkeit der Schaffung einer Hierarchie von Klassen, bei welcher Eigenschaften einer übergeordneten Klasse an die untergeordneten Klassen 'weitervererbt' werden, sodaß prinzipiell neue Individuenklassen leichter erstellt werden können und übergeordnete Klassendefinitionen in verschiedenen Modellen verwendet werden können.

(b) Definition von Interaktions- und Kommunikationsabläufen
Im Unterschied zur klassischen mathematischen Modellierung entsteht bei einem 'bottom-up'-Ansatz das Verhalten eines Gesamtsystems als Ergebnis aus dem Verhalten und den Interaktionen der unabhängig voneinander agierenden Individuen. Dies hat zur Folge, dass die Kommunikationsmöglichkeiten der Individuen mit ihrer Umwelt und anderen Individuen in besonderer Weise von der Modellierungsumgebung unterstützt werden müssen. Die Veränderung von Zustandsdaten und Aktionen werden innerhalb eines Simulationsablaufes nicht nur direkt durch das Voranschreiten der Zeit aufgerufen, sondern insbesondere auch durch Versenden von Aufrufen (Nachrichten) zwischen Individuen und Umweltobjekten. Hierzu muss für jede Individuenklasse und für die Umweltobjekte festgelegt werden, auf welche Nachrichten mit welcher Handlung reagiert werden kann und welche Nachrichten an andere Objekte versendet werden können. Unterschiedliche Individuenklassen können mit sehr unterschiedlichen Reaktionen (Methoden) auf die gleiche Nachricht reagieren. Die Darstellung von Aktionsabläufen und Interaktionsformen wurde insbesondere in Arbeiten auf dem Gebiet der agentenbasierten Modellierung bzw. Multi-Agenten-Syteme behandelt und entwickelt (Railsback, 2001; Janssen, 2002; Oechslein, 2004; Bousquet und Le Page, 2004).

(c) Modellierung einer räumlich expliziten heterogenen Umwelt
In individuenbasierten Modellen agieren Individuen typischerweise in einer räumlich expliziten,

heterogenen (meist 2-dimensionalen) Umwelt. Eine räumlich explizite Modellierung findet mit Hilfe von partiellen Diffentialgleichungen ebenso bei klassischen mathematischen Modellen in der Ökologie statt, allerdings kommt, wie an späterer Stelle noch mehrfach diskutiert wird, der Raumdefinition und der Lokalisation der Objekte bzw. Individuen bei einer individuenbasierten Modellerieung eine besondere Beachtung zu.

(d) Objektverwaltung
Da Individuen während des Simulationsablaufes erzeugt werden und sterben können, räumlich verteilt sind und häufig große Anzahlen simuliert werden, findet im allgemeinen eine explizite Modellierung der Individuenverwaltung (Listen, zenrale Steuerungsfunktionen) statt.

1.3.2 Spezifische Fragestellungen individuenbasierter Modellierung

Die Wahl eines individuenbasierten Modellierungsansatzes kann sowohl aufgrund programmiertechnischer Erwägungungen (pragmatisch), als auch auf Grund der zugrundeliegenden, theoretischen Fragenstellungen (paradigmatisch) erfolgen (Grimm, 1999). Ein pragmatischer Grund ist meistens die Notwendigkeit der Modellierung räumlich verteilter Objekte. Der Modelltypus der zellulären Automaten kann in diesem Zusammenhang als Modellform mit weniger komplexen Agenten angesehen werden (Klügl *et al.*, 2002; Oechslein, 2004, S. 46/47), die nach wie vor häufig zur Modellierung räumlich verteilter Vorgänge verwendet wird. Eine Diskussion der Verwendung individuenbasierter Modellierung aufgrund programmiertechnischer (pragmatischer) Überlegungen im Gegensatz zu prinzipiellen (paradigmatischen) Fragestellungen der theorertischen Ökologie findet sich u.a. bei Grimm (1999) mit einer Gegenüberstellung einer Reihe von Modellen, bei Łomnicki (1999) in Bezug auf Fragestellungen zur Populationsdynamik und bei Railsback (2001) in Bezug auf das Konzept der “complex adaptive systems”.

Folgende generelle theoretische Fragestellungen können (ohne Anspruch auf Vollständigkeit) einem individuenbasierten Modellansatz zu Grunde liegen:

a) Wie entsteht aus direkt beobachtbaren oder hypothetischen Eigenschaften von Individuen und den Interaktionen zwischen Individuen und ihrer Umwelt das Verhalten eines komplexen Gesamtsystems? Welche Einzeleigenschaften bzw. Verhaltensregeln von Individuen sind von kritischer Bedeutung für das Verhalten des Gesamtsystems?

b) Welche Auswirkung hat die Variationsbreite der Eigenschaften von Individuen auf das Gesamtverhalten des Systems? (Eine Erörterung dieser Frage findet sich im Bereich Populationsdynamik u.a. bei Łomnicki (1999)).

c) Welche Auswirkungen haben die Interaktionen eines heterogenen Lebensraumes (z.B. räumlich inhomogene Ressourcenverteilung) mit der Variationsbreite der Individuen innerhalb einer Population?

1.4 Beispiele bestehender Modellierungsumgebungen für agentenbasierte Modelle

Als zwei aktuell bedeutende Modellierungsumgebungen für agentenbasierte Modelle werden an dieser Stelle das Multiagenten-Simulationswerkzeug SWARM und die Simulationsumgebung für Multiagentensimulationen SeSAm kurz vorgestellt.
SWARM (Minar *et al.*, 1996), das seit den 90er Jahren am Santa Fé Institut entwickelt wird, ist das bekannteste Simulationswerkzeug für Multi-Agenten-Systeme und diente als Vorbild für zahlreiche konzeptionell ähnliche Softwareprojekte. Kernstück von SWARM ist eine in Objective-C geschriebene Klassenbibliothek für event-basierte Simulationen, sowie in Tcl/Tk implementierte Benutzerschnittstellen-Module. SWARM-Modelle bestehen meist aus einer räumlich expliziten zweidimensionalen Umgebung, in der sich verschiedene Klassen von mobilen Agenten bewegen. Agenten können zu einem Schwarm zusammengefasst werden, der dann wiederum als Agent betrachtet werden kann (hierarchische Agenten). Swarm-Modelle liegen in Form von Objective-C Quellcode vor, ein sicherer Umgang mit Compilern und Debuggern wird Vorausgesetzt. Für SWARM gibt es eine rudimentäre Makro-Sprache (MAML: Multi Agent Modeling Language), die jedoch im Wesentlichen aus Präprozessor-Makros besteht und deshalb lediglich Abkürzungen für wiederkehrende Codefragmente, nicht aber eine in sich geschlossene Hochsprache darstellt. Für den Bereich ökologischer Modelle wurden verschiedene Erweiterungen in Form von Klassenbibliotheken unter dem Namen EcoSwarm (http://www.humboldt.edu/ ecomodel/software.htm) an der Humboldt State University, Kalifornien unter Leitung von Steve Railsback entwickelt. Eine Übersetzung und Erweiterung des Systems von Klassenbibliotheken in Java-Bibliotheken wurde in dem Repast-Projekt durchgeführt (http://repast.sourceforge.net).
Die Hauptschwierigkeit bei der Modellierung mit SWARM besteht darin, dass die Modellerstellung sehr eng mit der Verwendung der Programmiersprachensyntax und der Klassenbibliotheken verbunden bleibt wodurch zwar prinzipiell eine grosse Flexibilität erhalten bleibt, aber andererseits auch ein hoher Einarbeitungsaufwand in die Programmiertechnik und die Gefahr einer engen Verknüpfung von Modellkonzeption und Softwarekonzeption.

Die Multiagenten-Simulationsumgebung mit dem aktuell am weitreichendsten verwirklichten Ziel der Verbesserung der Standardisierung, Kommunizierbarkeit und Analysefähigkeit agentenbasierter Modelle stellt wahrscheinlich das SeSAm-System (Oechslein, 2004) dar. SeSAm besteht aus den Komponenten SeSAm-UML als Repräsentationssprache, SeSam-IMPL als Implementierungsumgebung und den dazugehörigen Simulations- und Analysetools. SeSAm-UML stellt eine agentenspezifische Erweiterung des UML-Standards sowie der Agent-UML Erweiterung dar. Verwendet werden Klassendiagramme zur Darstellung der statischen Klassenbeziehungen und der Klassenattribute, Aktivitätsgraphen und optional die in Agent-UML

definierten Interaktionsdiagramme. Aus der Modellkonzeption und ersten Repräsentation in SeSAm-UML wird eine möglichst übergangslose Übertragung in die Implementierungssprache SeSAm-IMPL umgesetzt. Diese dient der Verfeinerung und genaueren Modellspezifizierung sodass hieraus direkt die Modellsimulation gestartet werden kann. SeSAm-IMPL kann über eine graphische Oberfläche verwendet werden und stellt eine Vielzahl sogenannter primitiver Funktionen zur Verfügung mit der die SeSAm-UML-Spezifizierung eines Modells soweit verfeinert wird, dass das Simulationswerkzeug hieraus eine ausführbare Modellübersetzung erstellen kann.
Die Zielrichtung der Entwicklung von SeSAm entspricht in weiten Teilen der Zielrichtung dieser Arbeit (Verbesserung der Standardisierung und Kommunizierbarkeit von agentenbasierten Modellen), allerdings sind die Ansatzpunkte aus Sicht einer mathematisch-ökologischen Standardisierung und aus Sicht einer softwaretechnischen Standardisierung unterschiedlich. Da der UML-Standard mit der Hauptausrichtung auf die Softwareentwicklung und hierbei insbesondere der Darstellung von Programm/Benutzer-Interaktionen und Porgrammablaufdarstellungen entwickelt wird, erscheint zum augenblicklichen Zeitpunkt der Nutzen von UML zur Modellierung ökologischer Systeme mit großen Anzahlen von Agenten mit tendenziell weniger komplexen Verhaltensrollen noch sehr begrenzt. Die Möglichkeiten und Grenzen der Verwendung von agentenspezifischen UML-Erweiterungen zur Dokumentation und Konstruktion von individuenbasierten Modellen muß aber zweifellos weiter im Auge behalten werden.

2 Anforderungen an die Unterstützung typischer individuenbasierter Modellelemente innerhalb eines Systems zur Erstellung und Dokumentation ökologischer Modelle

Insbesondere bei der individuenbasierten Modellierung spielt die enge Verknüpfung von Programmiertechnik und von Konzepten objektorientierter Programmiersprachen mit der Erstellung individuenbasierter Modelle eine besondere Rolle (Ginot, 2002; Lorek und Sonnenschein, 1999). Die Problematik mangelnder Tools zur Erleichterung der Modellerstellung und zur Verbesserung der Kommunizierbarkeit, Überprüfbarkeit und Analyse individuenbasierter Modelle wird in diesem Zusammenhang in der Literatur vielfach diskutiert, wie u.a. bei Lorek und Sonnenschein (1999); Grimm (1999, 2002); Ginot (2002); Manson (2002); Ropella *et al.* (2002) und Oechslein (2004, S. 9-11). Modellierer sind häufig zudem nicht auch gleichzeitig Programmierer, sodass der Programmquellcode nicht allgemein nachvollzogen werden kann. Die Verifikation und Analyse von Modellen Simulationsergebnissen ist nur möglich, wenn eine hinreichend exakte, verständliche und umfassende Modellspezifikation und Dokumentation erfolgt.

Es existieren mittlerweile eine Reihe von Entwicklungstools und -umgebungen zur Unterstützung individuenbasierter Modellerstellung und Simulation. Diese bewegen sich in einem sehr weiten Spektrum von der Bereitstellung programmiersprachenspezifischer Toolkits (Bibliotheken) bis hin zu einfachen baukastenähnlichen Systemen mit grafischer Oberfläche, die eine Modellerstellung per Mausklick ermöglichen. Es entsteht im Allgemeinen der Zwiespalt, dass die Tools bzw. Entwicklungsumgebungen in den Modellierungsmöglichkeiten entweder zu einschränkend und nur für spezielle Anwendungsbereiche nutzbar sind, oder aber stark programmiersprachenspezifische oder zumindest prgrammiersprachennahe Softwarebausteine (Bibliotheken) zur Verfügung stellen, sodass eine aufwändige Einarbeitung in das System bzw. die entsprechende Programmiersprache einen ähnlichen Aufwand darstellen, wie den, das Modell entsprechend der spezifischen Anforderungen von Grund auf neu zu programmieren bzw. programmieren zu lassen (siehe z.B. Topping *et al.* (2003)).
Einige der wichtigsten aktuell verwendeten Modellierungstools im Bereich der agentenbasierten Modellierung sind SWARM/RePAst (Minar *et al.*, 1996), MOBYDIC/Cormas (Ginot, 2002) und SeSAm (Oechslein, 2004) und StarLogo (Resnick, 1995; Klopfer, 2003).

2.1 Allgemeine Zielvorgaben

Entsprechend HOCH *et al.* (1998), BENZ *et al.* (2001) und OECHSLEIN (2004) können folgende allgemeine Zielvorgaben für ein geeignetes System zur Modellspezifikation und -dokumentation genannt werden:

a) Die Modellbeschreibung muss in einer hinreichend exakten, vollständigen und mathematisch-formalen Spezifikationssprache erfolgen, sodass eine weitgehend automatisierte Implementierung, d.h. Überführung in eine Progammiersprache bzw. ein Simulationssystem, möglich ist.

b) Die Modellspzifikation sollte zugleich intuitiv verständlich, deklarativ und übersichtlich bleiben, sodass ein detailliertes Modellstudium auch dem Nicht-Informatiker möglich ist, d.h. ohne die Notwendigkeit, den Programmquellcode nachzuvollziehen.

c) Aus der Modellspezifikation sollte neben der Softwareimplementierung ebenso auch in einem Schritt eine automatisierte Modelldokumentation erzeugt werden können. Hierzu müssen zusammen mit der mathematisch-formalen Spezifikation auch Möglichkeiten zur Eingabe von Randbedingungen und Hintergrundinformationen wie der ökologische Gültigkeitsbereich, erklärende textuelle Beschreibungen der Bedeutung von Zustandsvariablen und mathematischen Gleichungen, Messmethoden usw. bestehen.

2.2 Spezielle Anforderungen zur Unterstützung individuenbasierter ökologischer Modelle

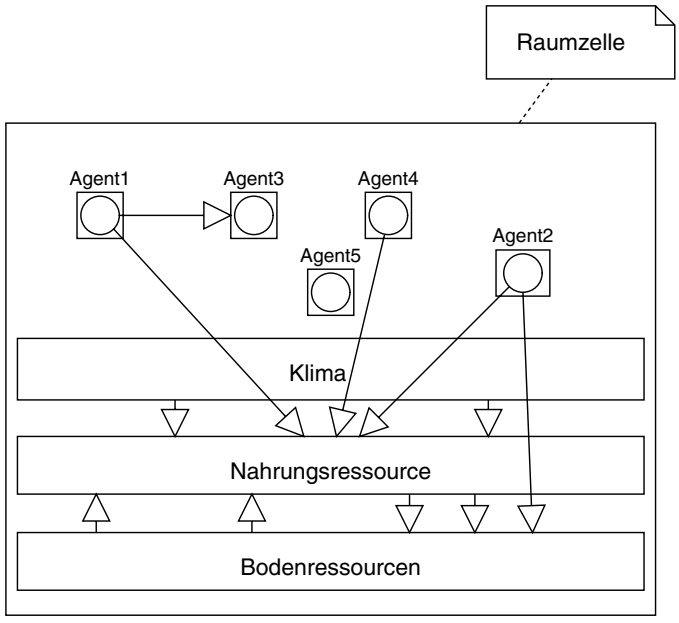
2.2.1 Definition der räumlich expliziten Umwelt (Raum-Ressourcen-Objekte)

Bei der individuenbasierten Modellierung ist insbesondere im Bereich der Ökologie die räumliche Verteilung der Individuen und der Umweltobjekte (Ressourcen) von elementarer Bedeutung. Individuen interagieren mit Ihrer Umgebung und anderen Individuen vielfach in Abhängigkeit von der räumlichen Entfernung (z.B. Räuber-Beute-Situation und Konkurrenzsituation bei der Ressourcennutzung wie Wasser-, Nährstoff- und Lichtkonkurrenz in Pflanzenbeständen und die Nutzung von Trinkwasser/Bewässerungswasser in der Landwirtschaft). Es muss also ein Schema zur Verfügung gestellt werden, mit dem möglichst einfach „Raumobjekte“ mit Größenangaben (Fläche/Volumen und Maßeinheit) sowie die Diskretisierung (Rasterauflösung) definiert werden können. Die bis dato am häufigsten verwendeten Räume sind in der Struktur als zweidimensionale reguläre Gitter angelegt und können als Ausgangspunkt dienen. Für „Raumobjekte“ müssen konstante Eigenschaften (z.B. Höhenmeter, Hangneigung, Hangausrichtung, Evaporationspotential, Wassertiefe etc.) sowie veränderliche Eigenschaften (z.B. Ressourcenvorräte wie Bodenfruchtbarkeit, Nährstoffkonzentrationen und Wasser) für jeden Raumpunkt definiert werden können. Hierbei müssen nicht-statistisch bzw. mathematisch verteilte Rauminformationen aus Datenbanken/Matrizen, z.B. aus einem GIS-System, als Startwerte einge-

lesen werden können. Zusätzlich müssen einfache reaktive Operationen zur Interaktion mit Agenten und zur Verarbeitung von räumlich fixierten Prozessen formuliert werden können. Die diskretisierten Raumpunkte werden häufig auch als Raumzellen beschrieben, welches die anschauliche Vorstellung und praktische Definition eines Raumes mit vielfältigen Attributen und reaktiven Eigenschaften erleichtert. Wie auch in dem Vergleich von Modelltypen bei KLÜGL *et al.* (2002) beschrieben, stellt diese Form der Definition von Raum- bzw. Ressourcenobjekten als Submodell die Form von *zellulären Automaten* dar, da die Spezifikationen (Regeln und Verhaltensweisen) für alle Raumpunkte (Raumzellen) gleich sind (abgesehen von heterogenen Parameterwerten). Hierbei werden gegenüber der Modellierung der Agenten weniger komplexe, agentenspezifische Eigenschaften wie vor allem die Erzeugung/Reproduktion von Agenten während der Simulationslaufzeit und die Bewegung im Raum modelliert.

Werden in einem Modell mehrere Klassen von Raum-Ressourcen erstellt, lassen sich diese als übereinander gelagerte Ebenen vorstellen, deren einzelne Zellen an einem Raumpunkt miteinander kombiniert zusammen die Umgebungsbeschreibung der Agenten darstellen. In Abbildung 2.1 wird dieses mit den drei exemplarischen Ressourcenklassen 'Klima', Nahrungsressourcen' und 'Bodenressourcen' skizziert.

Abbildung 2.1: Gedachte Raumzelle mit mehreren Ressourcenebenen.



2.2.2 Definition von Individuen-/Agenten-Klassen

Zur Definition und Beschreibung der Klassen von Individuen/Agenten des Modells sollten folgende Elemente definiert werden können:

(a) Deklaration der Attribute (Variablen)
Variablen können in einem mathematischen Modell unterteilt werden in Zustandsvariablen, abhängige Variablen und Konstanten. Die grundlegende Charakterisierung von Variablen sollte folgende Angaben umfassen: Variablenname, Datentyp (Integer, Float, Alphanumer), Skala (metrisch, ordinal, nominal), Dimension (Skalar, Vektor, Matrix), physikalische Einheit, Bedeutung und Gültigkeitsbereich. Diese Angaben wurden in der bestehenden Spezifikationssprache ECOBAS_MIF (ECOBAS Modell Interchange Format) bereits angelegt. Speziell für agentenbasierte Modelle müssen darüber hinaus die Datentypen 'Agenten-ID' sowie 'Liste von Agenten-IDs' zur Verfügung stehen, mit denen eine Adressierung der Agenteninstanzen während der Simulationslaufzeit ermöglicht wird.

Bei allen Variablen bzw. Konstanten muss festgelegt werden können, ob die Werte nur für das Individuum selbst, oder auch für andere Modellobjekte erkennbar, d.h. 'intern' oder 'öffentlich' sind. Die direkte Veränderung von Variablenwerten in anderen Modellobjekten sollte explizit über einen Operationsaufruf, wie nachfolgend unter (c) beschrieben, geschehen.

Als weitere spezielle Variablentypen sollten deklariert werden können:

Zeitvariable:
Die während der Simulation in (diskreten Zeitschritten) fortlaufende Zeit muss als eigener Variablentyp deklariert werden können, damit eine eindeutige Definition zeitabhängiger Prozesse erfolgen kann.

Raumkoordinaten:
Die Raumkoordinaten sollten ebenfalls zur Formulierung raumabhängiger Funktionen als eigener Typ deklariert werden können. Die Angaben aus der Raumdeklaration spezifizieren hierbei die Eigenschaften der Raumkoordinaten wie Dimension, Diskretisierung und Grenzen.
Die eigenen Positionskoordinaten der Agenten sollten als spezieller Variablentyp deklariert werden können, um die Definition von positionsgebundenen Funktionen (z.b Interaktion mit aktueller Raumzelle) zu vereinfachen.

Inputvariablen:
Inputs treten aufgrund der nachfolgend beschriebenen Darstellungsform von Interaktionen zwischen Objekten als Versendung und Empfang von Nachrichten in Form von Empfangsvariablen ('Method-Inputs') oder erwarteten Rückgabevariablen ('CALL-Returns') auf. Zur Überprüfbarkeit korrekter Interaktionsaufrufe sollten die Empfangs- und Rückgabevariablen explizit deklariert werden.

(b) Initialisierung und Festlegung von Startwerten

Entsprechend Abschnitt 1.3.1 müssen Individuen-Instanzen bei der Erzeugung für verschiedene Attribute typischerweise individuell unterschiedliche Startwerte zugewiesen bekommen. Es muss hierbei festgelegt werden, welche Variablen bzw. Konstanten Werte von anderen Modellobjekten bzw. aus Parameter-Dateien bzw. Datenbanken zugewiesen bekommen. Als Beispiel sei hier die Übernahme von Startwerten von Eltern-Individuen (vererbte Merkmale, Raumposition etc.) genannt.

Mathematisch/statistisch verteilte und zufällige Startwerte können in Form von Initialisierungsroutinen definiert werden.

(c) Definition von Operationen (Verhaltensweisen)

Operationen können in folgende drei Kategorieen unterteilt werden: (i) Operationen, die bei der Initialisierung ausgeführt werden, (ii) Operationen, die bei jedem Zeitschritt ausgeführt oder überprüft werden und (iii) Operationen, die bei Bedarf aufgerufen werden können.

Bei Letzteren muss festgelegt werden können, ob diese nur vom Individuum selbst, oder von anderen Modellobjekten aufgerufen werden können und es müssen die Übergabe- und Rückgabeparameter deklariert werden. Im Sprachgebrauch der agentenbasierten Modellierung und ebenso der objektorientierten Programmierung werden Operationen in einem Objekt (hier Agent oder Raumzelle) durch Versenden einer Nachricht aufgerufen, die den Namen der aufzurufenden Operation sowie die Inputparameter und die erwarteten Rückgabeparameter enthält. Hierbei muss bei dem Empfänger die Fähigkeit zur Bearbeitung in Form des Vorhandenseins der benannten Operation sowie der Erlaubnis des Operationsaufrufs für den Absender der Nachricht definiert sein, damit eine Reaktion ausgeführt werden kann.

3 Erweiterungen von ECOBAS zur Untersützung individuenbasierter ökologischer Modelle

3.1 Kurzbeschreibung von ECOBAS

Das ECOBAS-System besteht aus mehreren Ebenen: (i) zur Erstellung und Verwaltung von Modellen und Modellmodulen (ECOBAS Modelling Assistant - EMA), (ii) zur Überführung der erstellten Modellspezifikationen in ein Simulationssystem (ECOBAS SIMUL) sowie (iii) zur Datenanalyse und Darstellung von Simulationsergebnissen. Im Rahmen dieser Arbeit wird ein ECOBAS-Format zur Modellerstellung und -spezifizierung von individuenbasierten Modelltypen entworfen. Eine Kurzübersicht über die für diesen Bereich relevanten Elemente der Ebene (i) von ECOBAS wird nachfolgend dargestellt. ECOBAS wird unter Leitung von Herrn Dr. J. Benz, Universität Kassel, Fachbereich Ökologische Agrarwissenschaften, Witzenhausen entwickelt. Weitere Informationen stehen unter http://eco.wiz.uni-kassel.de zur Verfügung. Die weiteren Ebenen von ECOBAS – die Anbindung eines graphischen Modelleditors (GME) und des Simulationssytems (ECOBAS-SIMUL) – werden auf der Projekthomepage unter anderem in den Präsentationen „What is ECOBAS?“ und „Modularization of ecological models“ vorgestellt.

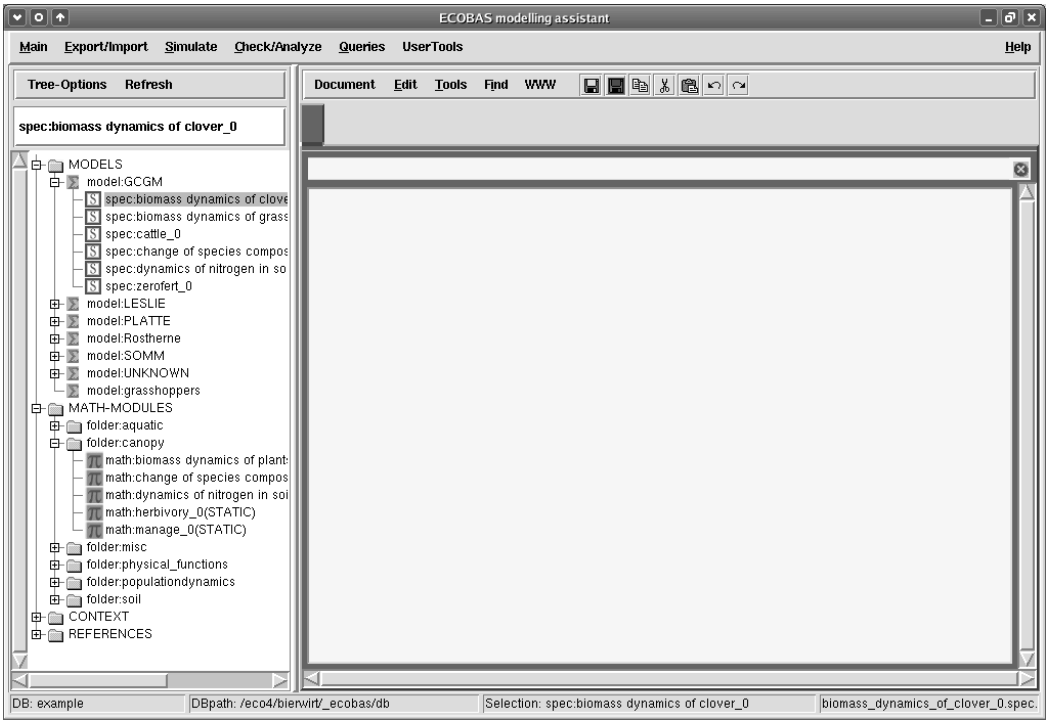
3.1.1 ECOBAS Modelling Assistant

Zur Erstellung, Spezifikation, Strukturierung und Verwaltung von Modellen steht eine Datenbank (ECOBAS_TWIXT) und der ECOBAS Modelling Assistant (EMA) zur Verfügung. Der *EMA* stellt die grafische Oberfläche zur Datenbankverwaltung und Modelleditierung dar. Wie in Abbildung 3.1 zu sehen ist, befindet sich im linken Bereich die Navigationsübersicht zur Auswahl von Modellen und Teilkomponenten. In das Hauptfeld können Module zur Modelleditierung aus der Datenbank geladen werden (Abbildung 3.2). Die Module der Modellkomponenten werden dabei im ECOBAS-XML-Format erstellt und editiert. Der *EMA* bietet hierzu im Hauptfeld eine leicht bedienbare graphische XML-Editoroberfläche.

3.1.2 Modellbeschreibung und -spezifizierung im ECOBAS Model Interchange Format

Die auswählbaren Strukturen und Elemente zur Modellbeschreibung und -spezifizierung sind im ECOBAS Model Interchange Format (ECOBAS-MIF) in einer XML Dokumenttyp-Definitionsdatei (mif.dtd) festgelegt. Die Module zur Modellspezifizierung werden unterteilt in einen ma-

Abbildung 3.1: ECOBAS Modelling Assistant

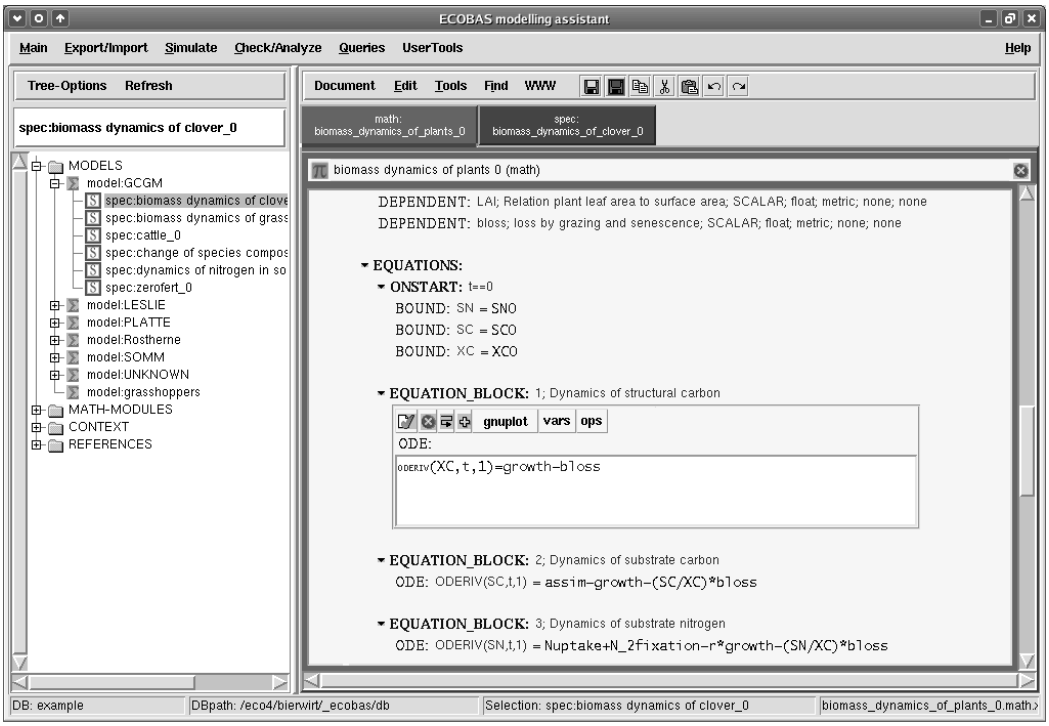


thematischen Teil zur Deklaration von Variablen, Gleichungen und Prozeduren (MATH-Modul) und einen Spezifizierungsteil (SPECIFICATION-Modul) zur Festlegung von Parameterwerten, physikalischen Einheiten, Gültigkeitsbereichen, der physikalischen Bedeutung von verwendeten Variablen und weiteren Kontextinformationen wie Messmethoden für Variablen, Literaturangaben und gültiger ökologischer Kontext. Für jede Modellkomponente werden jeweils beide Modultypen paarweise angelegt und auf Konsistenz, Vollständigkeit und syntaktische Korrektheit der Informationen geprüft.

3.2 Ergänzung agentenspezifischer ECOBAS-MIF Module

Es wurden die jeweils korrespondierenden Modultypenpaare '*MATH_AGENTRESOURCE*' und '*AGENTRESOURCE_SPECIFICATION*' zur Definition und Spezifikation der Raum-Ressourcen-Klassen sowie '*MATH_AGENT*' und '*AGENT_SPECIFICATION*' zur Definition und Spezifikation der Agentenklassen ergänzt. Diese sollen den in den Abschnitten 2.2.1 und 2.2.2 geforderten Eigenschaften zur Modellspezifikation in Form von *ECOBAS_MIF*-Modulen entsprechen. Die Grundstruktur zur Definition von Klassen von Raumzellen und Agenten kann, wie in Tabelle 3.1 dargestellt, identisch formuliert werden. Grundsätzlich wird dem, sowohl in *ECOBAS*, als auch in der objektorientierten Programmierung und UML typischen Aufbau für Objektklassen in ≪Klassen-Stereotyp≫ – Klassenname – Attribute – Operationen gefolgt. Die Operationen

Abbildung 3.2: ECOBAS Modelling Assistant - XML-Editor



werden hierbei allerdings aufgeteilt in den Bereich 'Methoden', welche Aktivitäten darstellen, die bei Bedarf und bei entsprechender Deklaration auch „von außen“, d.h. von anderen Modellobjekten aufgerufen werden können und in den Bereich 'sequenzielle Prozesse', in dem die Aktivitäten und Bedingungsabfragen definiert sind, die bei jedem Zeitschritt während der Simulation bei allen Instanzen in fest vorgegebener Reihenfolge ausgeführt werden.

Zur zentralen Definition des zugrundliegenden Raumes eines Modells wurden zudem die Module '*MATH_AGENTSPACE*' und '*AGENTSPACE_SPECIFICATION*' ergänzt.

Als Interface zur Verknüpfung von externen Datensätzen und der Initialisierung von Raumzellen und Agenten sowie zur Festlegung der Anzahl und Verteilung von Agenten zu Simulationsbeginn sollteein zusätzliches '*DATAMODULE*' angelegt werden.

3.3 Raumdeklaration

Die Raumeigenschaften können für ein Modell zentral in dem Modulpaar *MATH_AGENTSPACE / AGENTSPACE_SPECIFICATION* definiert werden. Es ist hierbei zunächst der am häufigsten verwendete (diskrete) Raumtyp der *regulären Gitter* vorgesehen. Neben der Dimension können in dem Spezifizierungsmodul die Ausdehnung, die Maßeinheit, die Diskretisierung und die Grenzbedingungen festgelegt werden. Für Letztere können die Typen des in sich geschlos-

senen Raumes und des beschränkten Raumes angegeben werden. Bei geschlossenen Räumen treten Agenten, die den Raum an einer Grenze verlassen automatisch auf der gegenüberliegenden Seite wieder in den Raum herein.

3.4 Agentenspezifische Erweiterungen zur Variablendeklaration

(a) Datentypen:
Zu den (mathematischen) Datentypen '*float*', '*integer*' und '*alpha*' wurden die Typen '*agent_id*' zur Adressierung von Agenteninstanzen und '*agentlist*' für Listen von Agenteninstanzen ergänzt. Agentenlisten können dabei alle vorhandenen Agenteninstanzen einer Klasse oder eine nach definierbaren Bedingungen selektierte Agentenauswahl einer Klasse umfassen.

(b) Positionsvariable:
Obwohl es sich bei der Raumposition bei Agenten und Raumzellen prinzipiell um „normale“ Zustandsvariablen bzw. Konstanten handelt, wird die Raumposition von Agenten bzw. Raumzellen mit einem eigenen Vaiablenbezeichner (*SPACE_POSITION*) deklariert. Die Raumeigenschaften Dimension, Grenzen und Diskretisierung können somit von der Raumdefinition aus dem *AGENTSPACE*-Modul für das Modell übernommen werden und bei Agenten können die Koordinatenbezeichnungen direkt in Funktionen zur Bewegung des Agenten im Raum sowie für weitere positionsabhängige Operationen verwendet werden. Es können sowohl die Einzelkoordinaten als Liste, als auch alternativ der Raumvektor als Einheit benannt werden. Die Möglichkeit der Bewegung des Agenten durch andere Modellobjekte kann durch Definition einer entsprechenden (öffentlichen verfügbaren) Methode erfolgen. Bei Raumzellen ist die Position definitionsgemäß fest.

(c) Initialisierungs-Variablen:
Insbesondere bei agentenbasierten Modellen müssen zur Initialisierung von Agenten und Raumzellen häufig Startwerte aus externen Parameterdateien bzw. Datenbanken oder von anderen Agenten übergeben werden. Diese Notwendigkeit resultiert zum Einen aus der Erzeugung relativ großer Anzahlen von Agenten mit der Möglichkeit zu individueller Initialisierung bei dem Simulationsstart und zum Anderen aus der Übergabe von Startwerten bei der Erzeugung von Agenten durch Agenten (natürlicherweise vor allem durch Reproduktion) während des Simulationslaufs. Zur übersichtlicheren Darstellung und besseren Kontrolle auf konsistente Formulierung der Operationen zur Erzeugung bzw. Reproduktion von Agenten, werden diese Übergabewerte bei Agenten als Variablen des Typs '*ONCREATION_INPUT*' gesondert deklariert (siehe Tabellen 3.5 und 3.6). Da bei Simulationsstart allerdings noch keine Agenten existieren, müssen den Übergabevariablen ebenso wie den Zustandsvariablen in der Initialisierungssektion 'ONCREATION' zusätzlich Startwerte für den Simulationsstart zugewiesen werden (Tab. 3.8). Dieses kann in Form von Konstanten, Defaultwerten, mathematischen Funktionen (z.B. Zufallsverteilungen) oder externen Parameterdateien erfolgen. Bei der Erzeugung von Agenten müssen die

'Eltern'-Agenten zudem auch zwingend die Raumkoordinaten übergeben, für die folglich auf gleiche Weise die Startwerte für den Simulationsbeginn festgelegt werden. Es ist vorgesehen, dass externe Parameterdateien, die Beispielsweise durch ein GIS-System erzeugt wurden, über ein zusätzliches Datenmodul zugeordnet werden. Diese Option könnte beispielsweise in der Initialisierungssektion erfolgen, indem als Startwert 'DATAMODULE' eingetragen wird.

(d) Input-Variablen:
Inputs treten für Agenten und Ressourcenzellen als Rückgabewerte von *CALLs* oder Inputs von *METHODs* auf. Zur Überprüfung der Konsistenz und Korrektheit der Interaktionsaufrufe zwischen Modellobjekten sollten die Inputvariablen vollständig deklariert werden. Zur Trennung des Deklarationteils für Variablen von der Darstellung operativer Prozesse (s.u.) sollten die Deklaration dabei nicht innerhalb der Prozeduren '*METHODs*', sondern im Variablen-Deklarationsteil und dem dazugehörigen Spezifikationsteil als *INPUT* erfolgen.

3.5 Agentenspezifische Operationen und Funktionen

3.5.1 Event-basierte, aufrufbare Operationen

Wie in 2.2.2 dargestellt, wird in dem vorliegenden Entwurf zur Darstellung von Interaktionen zwischen Modellobjekten dem Schema von Versendung und Verarbeitung von Nachrichten mit Übergabeparametern gefolgt. Operationen, die als Reaktion oder Antwort auf Nachrichten bereitstehen, werden als Methode (*METHOD*) deklariert und die Adressierung und Versendung einer Nachricht als Aufruf (*CALL*) (siehe Tabellen 3.8 und 3.10). Methoden, die „von außen“ von anderen Modellobjekten aufgerufen werden, werden mit der Option 'response' im Gegensatz zu 'internal' deklariert.

3.5.2 Sequenzielle Operationen bei jedem Zeitschritt

Neben der Definition von Methoden, die unter bestimmten Bedingungen von anderen Modellobjekten oder intern aufgerufen werden können, wird eine Sektion ('*SEQUENTIAL_PROCESSES*') festgelegt innerhalb welcher die Prozesse angelegt werden, die bei jedem Zeitschritt durchgeführt werden[1] (siehe Tab. 3.8). Die zeitliche Reihenfolge der abzuarbeitenden Operationen innerhalb dieser Sektion wird durch die Definitionsreihenfolge der Operationen festgelegt. Alternativ wäre eine Festlegung über die Angabe einer Ablaufpriorität möglich.
Es handelt sich bei der Festlegung dieser Sektion nicht um ein spezifisches Element agentenbasierter Modelle, allerdings erscheint bis dato die feste Vorgabe eines sequentiellen Bereichs für Operationen sinnvoll.

Zu Beginn des Bereichs *SEQUENTIAL_PROCESSES* wird zudem die Objektinitialisierung in der Sektion *ONCREATION*(Agenten) bzw.*ONSTART* für Ressourcenzellen festgelegt, die nur

[1] Agentenbasierte Modelle werden im Allg. zeitdiskret angelegt, um die Darstellung und Koordination der mannigfaltigen event-basierten Aktivitäten zu erleichtern.

bei der Objekterzeugung ausgeführt wird.

3.5.3 Erweiterung vorgegebener Operationen und Funktionen

Spezielle Operationen und Funktionen, die nicht in Form üblicher mathematischer Formulierungen dargestellt werden können, aber die Arbeit des Modellierers erheblich vereinfachen, müssen zusätzlich als Sprachelemente zur Modellspezifikation explizit vorgegeben werden. Damit eine Spezifikationssprache dem Wissenschaftler möglichst intuitiv erfassbar bleibt, sollten diese zusätzlichen Ausdrücke allerdings soweit wie möglich auf ein Minimum an elementaren Bausteinen reduziert bleiben. Eine Übersicht über spezifische Operationen und Funktionen dieser Form ist in den Tabellen 3.9 und 3.11 dargestellt. Als Basis werden im vorliegenden Entwurf die Operationen zur Agentenerzeugung während der Simulation ('REPRODUCE' und 'CREATE'), zum Sterben von Agenten, bzw. Übergang in eine andere Klasse ('DIE') sowie die Funktionen zur Selektion von Agenten während des Simulationsablaufs definiert (siehe Tab. 3.9 und 3.11).

Tabelle 3.1: Grundstruktur des *MATH*-Moduls zur Definition von Agenten-Klassen und Raumzellenklassen.

Feld-Bezeichner/Schlüsselbegriff	*Bedeutung/Erläuterungen*
ECOBASMIF: (Version)	
MATH_AGENT|MATH_AGENTRESOURCE: (name, version, folder)	MATH-Modul zur Klassen-Definition von Agenten oder Raumzellen
GENERAL AUTHOR|MODEL|DOCUMENTED_BY|KEYWORDS| REFERENCES	Allgemeine Hintergrundinformationen zu Modellentwicklern, Acronym und Veröffentlichungen
VARIABLE_DECLARATION TIME | SPACE_POSITION | ONCREATION_INPUT[1] | CONSTANT | STATE | DEPENDENT | INPUT	Deklaration (mathematisch-formale Charakterisierung) von Konstanten und Variablen (Attribute). Details siehe in Tabelle 3.5
METHODS METHOD (internal | response)	Methoden sind Operationen, die vom Indiviuum selbst (internal), oder von anderen Objekten (response) aufgerufen werden können. Methoden können mehrere Input-Werte übergeben bekommen und mehrere Output-Werte zurückgeben.
SEQUENTIAL_PROCESSES ON_CREATION[1]|ONSTART | PROCESS_BLOCK	Operationen, die bei der Initialisierung (ONCREATION[1] | ONSTART) bzw. bei jedem Zeitschritt ausgeführt werden. Die Operationen werden in der gelisteten Reihenfolge abgearbeitet.
DESCRIPTION:	Freie Textbeschreibung; Erläuterungen zur Modelldokumentation
FIGURE|TECHNICAL :	Angabe von Abbildungen und technischen Informationen
(End of MATH_AGENT / MATH_AGENTRESOURCE)	

[1] bei Agentenklassen

Tabelle 3.2: Grundstruktur des *SPECIFICATION*-Moduls zur Definition von Agenten-Klassen und Raumzellenklassen.

Feld-Bezeichner/Schlüsselbegriff	*Bedeutung/Erläuterungen*
ECOBASMIF: (Version)	
AGENT_SPECIFICATION | AGENTRESOURCE_SPECIFICATION: (name,version,math_modul)	*SPECIFICATION*-Modul zur Klassen-Definition von Agenten oder Raumzellen
GENERAL AUTHOR | MODEL | DOCUMENTED_BY | KEYWORDS | REFERENCES	Allgemeine Hintergrundinformationen zu Modellentwicklern, Acronym und Veröffentlichungen
VARIABLE_PROPERTIES TIME | SPACE_POSITION[1] | ONCREATION_INPUT[1] | CONSTANT | STATE | DEPENDENT | INPUT	Spezifizierung von Konstanten und Variablen (Attribute). Details siehe in Tabelle 3.6 und 3.7
DESCRIPTION:	Freie Textbeschreibung; Erläuterungen für die Modelldokumentation
FIGURE | TECHNICAL:	Angabe von Abbildungen und technische Informationen
(End of AGENT_SPECIFICATION / AGENTRESOURCE_SPECIFICATION)	

[1] bei Agentenklassen

Tabelle 3.3: *MATH*-Modul zur Raumdeklaration eines Modells

Feld-Bezeichner/Schlüsselbegriff	*Bedeutung/Erläuterungen*
ECOBASMIF: (Version)	
MATH_AGENTSPACE: (name, version, folder)	
GENERAL AUTHOR | MODEL | DOCUMENTED_BY | KEYWORDS | REFERENCES	Allgemeine Hintergrundinformationen zu Modellentwicklern, Acronym und Veröffentlichungen
SPACE_DECLARATION	
SPACE_TYPE (grid1d | grid2d | grid3d)	Auswahloptionen zum Raumtyp
SPACE_RANGE name: '$AGENTSPACE$'	Bezeichnung der Raumausdehnung (Variablenformat: Vektor-Konstante mit der Dimension entsprechend dem Raumtyp). $AGENTSPACE$ ist als Name fest vorgegeben.
DESCRIPTION:	Freie Textbeschreibung; Erläuterungen
(End of MATH_AGENTSPACE)	

Tabelle 3.4: *SPECIFICATION*-Modul zur Raumdeklaration eines Modells

Feld-Bezeichner/Schlüsselbegriff		*Bedeutung/Erläuterungen*
ECOBASMIF: (Version)		
AGENTSPACE_SPECIFICATION: (name, version, math_module)		
GENERAL		
AUTHOR | MODEL | DOCUMENTED_BY | KEYWORDS | REFERENCES		Allgemeine Hintergrundinformationen zu Modellentwicklern, Acronym und Veröffentlichungen
SPACE_PROPERTIES		
SPACE_TYPE :	boundary_type : (bounded | periodic) ghostcells :< $Integer$ >	Verhalten an den Raumgrenzen: als beschränkter Raum (bounded) oder als geschlossener Raum (periodic - die Agenten treten bei Überschreiten der Grenze auf der gegenüberliegenden Seite wieder in den Raum ein)
SPACE_RANGE :	name: '$AGENTSPACE$' [1] values :< $Liste\ v.\ Zahlen$ > unit :< $Zeichenkette$ > resolution :< $Integer$ >	Raumausdehnung Ausdehnung der Raumrichtungen [2] physikal. Maßeinheit (cm,m,km ...) Rasterauflösung (Diskretisierung)
DESCRIPTION:		Freie Textbeschreibung; Erläuterungen
(End of AGENTSPACE_SPECIFICATION)		

[1] Die Werte der Raumausdehnung '*AGENTSPACE*' können als Vektorkomponenten in der Form AGENTSPACE[n] in jedem Modul des zugehörigen Modells verwendet werden (im Allg. also: AGENTSPACE[1]: Ausdehnung des Modellraumes in x-Richtung, AGENTSPACE[2]: Ausdehnung in y-Richtung).

[2] Komma-getrennte Liste der Raumausdehnungen in die einzelnen Koordinatenrichtungen.

Tabelle 3.5: Übersicht über die Deklaration von Attributen (Konstanten und Variablen) im MATH-Modul von Agentklassen und Raumzellenklassen.

Feld-Bezeichner/Schlüsselbegriff		*Bedeutung/Erläuterungen*
MATH_AGENT|MATH_AGENTRESOURCE		
VARIABLE_DECLARATION		
TIME :	name : t	Bezeichnung der fortlaufenden Simulationszeit.
SPACE_POSITION :	coord_list : $< Liste\ d.\ Koordinatennamen > \mid < Vektorname >$	Bezeichnung der Raumkoordinaten des Agenten bzw. der Raumzelle. Beispiel: 'posX, posY' als Benennung der Einzelkoordinaten oder 'posXY' für die Verwendung als Vektor. Die Eigenschaften werden von der Raumdeklaration in *AGENTSPACE* übernommen.
ONCREATION_INPUT[2] :	name :$< Zeichenkette >$ text :$< Text >$ vtyp : (float|integer|alpha|agent_id|agentlist) scale : (metric|ordinal|nominal) dim :$< Liste\ v.\ Integerwerten >$	Startwerte, die bei der Erzeugung der Agenten von dem erzeugenden Objekt übergeben werden.[1]
CONSTANT|STATE| DEPENDENT :	name :$< Zeichenkette >$ text :$< Text >$ vtyp : (float|integer|alpha|agent_id|agentlist) scale : (metric|ordinal|nominal) dim :$< Liste\ v.\ Integerwerten >$ iface : (private|public)	Deklaration der Konstanten, Zustandsvariablen und abhängigen Variablen [1]
INPUT :	name :$< Zeichenkette >$ text :$< Text >$ vtyp : (float|integer|alpha|agent_id|agentlist) scale : (metric|ordinal|nominal) dim :$< Text >$	Variablen, die als Inputs von anderen Objekten bei Response-Methoden (Inputs) und Objekt-Calls (Returns, siehe Tab. 3.10) auftreten. [1]

[1] name: Variablenname
text: Kurzbeschreibung/Bedeutung
vtyp: Variablentyp
scale: Art der Wertskala
dim: Dimension: Skalar/Vektor/Matrix: (*Reihen,Spalten*) z.B. (1,2) für 2D-Variablen)
iface: Wert der Variablen ist intern oder öffentlich verfügbar

[2] nur bei Agentenklassen

Tabelle 3.6: Variablenspezifizierung im *SPECIFICATION*-Modul für Agentenklassen

Feld-Bezeichner/Schlüsselbegriff		*Bedeutung/Erläuterungen*
AGENT_SPECIFICATION : VARIABLE_PROPERTIES		
TIME :	name :< $Zeichenkette$ > unit :< $Zeichenkette$ >	Variablenname Zeiteinheit (SI)
SPACE_POSITION :	coord_list :< $Liste\ d.\ Koord.$ > default :< $Liste$ > space_range :< $Liste\ v.\ (Zahl, Zahl)$ >	Variablenname Defaultwert gültiger Raumbereich[1]
ONCREATION_INPUT:[2]	name :< $Zeichenkette$ > unit :< $Zeichenkette$ > meaning :< $Text$ > default :< $Zahl\ od.\ Zeichenkette$ > range :< $Zahl, Zahl$ >	Variablenname physikal. Maßeinheit (SI) Bedeutung Defaultwert Gültigkeitsbereich (Min., Max.)
STATE :	name :< $Zeichenkette$ > unit :< $Zeichenkette$ > meaning :< $Text$ > default :< $Zahl\ od.\ Zeichenkette$ > range :< $Zahl, Zahl$ >	Variablenname physikal. Maßeinheit (SI) Bedeutung Defaultwert Gültigkeitsbereich (Min., Max.)
CONSTANT :	name :< $Zeichenkette$ > unit :< $Zeichenkette$ > meaning :< $Text$ > value :< $Zahl\ od.\ Zeichenkette$ >	Variablenname physikal. Maßeinheit (SI) Bedeutung Wert
DEPENDENT :	name :< $Zeichenkette$ > unit :< $Zeichenkette$ > meaning :< $Text$ > range :< $Zahl, Zahl$ >	Variablenname physikal. Maßeinheit Bedeutung Gültigkeitsbereich (Min., Max.)
INPUT :[3]	name :< $Zeichenkette$ > unit :< $Zeichenkette$ > meaning :< $Text$ > range :< $Zahl, Zahl$ >	Variablenname physikal. Maßeinheit Bedeutung Gültigkeitsbereich (Min., Max.)

[1] Gültige Bereiche (Min,Max) für die Raumkoordinaten des Agenten. Default ist '$AGENTSPACE$': die für das Gesamtmodell definierte Raumgröße.

[2] Die Startwerte zu Beginn der Simulation (es sind noch keine Agenten vorhanden) müssen in der Sektion *ONCREATION* im *MATH*-Modul festgelegt werden (siehe Tab. 3.8).

[3] Zur Kontrolle der Kompatibilität müssen die Inputvariablen für „Response"-Methoden (werden von anderen Objekten aufgerufen) und Returnvariablen von Calls an andere Objekte spezifiziert werden.

Tabelle 3.7: Variablenspezifizierung im *SPECIFICATION*-Modul für Raum-Ressourcen-Klassen

Feld-Bezeichner/Schlüsselbegriff		*Bedeutung/Erläuterungen*
AGENTRESOURCE_SPECIFICATION		
VARIABLE_PROPERTIES		
TIME :	name :< $Zeichenkette$ > unit :< $Zeichenkette$ >	Variablenname Zeiteinheit (SI)
STATE :	name :< $Zeichenkette$ > unit :< $Zeichenkette$ > meaning :< $Text$ > default :< $Zahl\ od.\ Zeichenkette$ > range :< $Zahl, Zahl$ >	Variablenname physikal. Maßeinheit (SI) Bedeutung Defaultwert Gültigkeitsbereich (Min., Max.)
CONSTANT :	name :< $Zeichenkette$ > unit :< $Zeichenkette$ > meaning :< $Text$ > value :< $Zahl\ od.\ Zeichenkette$ >	Variablenname physikal. Maßeinheit (SI) Bedeutung Wert
DEPENDENT :	name :< $Zeichenkette$ > unit :< $Zeichenkette$ > meaning :< $Text$ > range :< $Zahl, Zahl$ >	Variablenname physikal. Maßeinheit Bedeutung Gültigkeitsbereich (Min., Max.)
INPUT :[1]	name :< $Zeichenkette$ > unit :< $Zeichenkette$ > meaning :< $Text$ > range :< $Zahl, Zahl$ >	Methodenname Variablenname physikal. Maßeinheit Bedeutung Gültigkeitsbereich (Min., Max.)

[1] Zur Kontrolle der Kompatibilität müssen die Inputvariablen für „*Response*"-Methoden (Methoden, die von anderen Objekten aufgerufen werden) und Returnvariablen von *Calls* an andere Objekte spezifiziert werden.

Tabelle 3.8: Deklaration von Methoden und sequenziellen Prozessen im *MATH*-Modul für Agenten- und Raum-Ressourcen-Klassen.

Feld-Bezeichner/Schlüsselbegriff		*Bedeutung/Erläuterungen*
MATH_AGENT|MATH_AGENTRESOURCE		
METHODS		
METHOD :	name :$< Zeichenkette >$ iface : $(internal\|response)$ inputs : $(Liste)$ returns : $(Liste)$ text :$< Text >$	Methodenname Aufruf Intern od. von Außen Input-Variablen[1] Return-Variablen freier Textkommentar
(INT | FLOAT | ALPHA | AGENT_ID | AGENTLIST)$*$		temporäre lokale Variablen
AEQ | SST | IF | LOOP | CALL[2] | ... ⋮		Mathematische Formulierung von Prozessen als algebraische Gleichungen, State-Transitions, If-Bedingungen, Schleifen etc.
METHOD :		
⋮		
SEQUENTIAL_PROCESSES		
ONCREATION :		
AEQ | BOUND | ONSTART_BOUND | IF | LOOP | CALL[2] | ... ⋮		Zuweisung von Startwerten für Zustandsvariablen (BOUND) und für ONCREATION_INPUTS (ONSTART_BOUND) [3].
PROCESS_BLOCK :	id :$< Zeichenkette >$ title :$< Text >$	ID-Bezeichner Titel
AEQ | SST | IF | LOOP | CALL[2] | ... ⋮		Mathematische Formulierung von Prozessen als algebraische Gleichungen, State-Transitions, If-Bedingungen, Schleifen etc.
PROCESS_BLOCK :		
⋮		

[1] Inputs von Response-Methoden müssen im Deklarationsteil als INPUT-Variablen angegeben werden.

[2] Die verschiedenen Call-Routinen zum Aufrufen von Methoden sowie weitere Operationen sind in den Tabellen 3.9 und 3.10 aufgelistet.

[3] Für die erste Generation von Agenten bei Simulationsstart müssen für die Übergabewerte (*ONCREATION_INPUT*s) und die Raumposition (SPACE_POSITION) Startwerte definiert werden. Startwerte können durch Konstanten, mathemat. Funktionen (z.B. normalverteilte Zufallswerte) oder durch externe Parameterdateien festgelegt werden.

Tabelle 3.9: Vorgegebene agentenspezifische Operationen im *MATH*-Modul für Agenten-Klassen

Feld-Bezeichner/Schlüsselbegriff		*Bedeutung/Erläuterungen*
CREATE :	class :< $Zeichenkette$ > coordinates :< $Zeichenkette$ > transfer_values :< $Liste$ > oncreation_inputs :< $Liste$ >	Erzeugung eines Agenten der Klasse '*class*' und Übergabe der Liste von Startwerten zur Initialisierung (die Variablen müssen in der entsprechenden Klasse als *ONCREATION_INPUT* deklariert sein)
REPRODUCE :	transfer_values :< $Liste$ > oncreation_inputs :< $Liste$ >	Erzeugung eines Agenten der eigenen Klasse an der eigenen Raumposition und Übergabe der Liste von Startwerten zur Initialisierung (die Variablen müssen als *ONCREATION_INPUT* deklariert sein)
DIE :	new_class :< $Zeichenkette$ > transfer_values :< $Liste$ > oncreation_inputs :< $Liste$ >	Tod des Agenten oder optional: Tod und Übergang in eine neue Klasse mit den aufgelisteten Übergabeparametern (die Agenten-ID wird dabei erhalten)

Tabelle 3.10: Vorgegebene *CALL*-Routinen in den *MATH*-Modulen für Agenten-Klassen und Raum-Ressourcen-Klassen.

Feld-Bezeichner/Schlüsselbegriff		*Bedeutung/Erläuterungen*
CALL :	method :< $Zeichenkette$ > inputs :< $Liste$ > returns :< $Liste$ > text :< $Text$ >	Aufruf einer eigenen Methode oder Funktion. Es können Listen von Übergabe- und Rückgabeparametern angegeben werden.
CALL_AGENT :	agent_id :< $Zeichenkette$ > method :< $Zeichenkette$ > inputs :< $Liste$ > returns :< $Liste$ > text :< $Text$ >	Aufruf einer Methode in einem Agenten. Es können Listen von Übergabe- und Rückgabeparametern angegeben werden.
CALL_RESOURCE :	resource_name :< $Zeichenkette$ > coordinates :< $Liste$ > method :< $Zeichenkette$ > inputs :< $Liste$ > returns :< $Liste$ > text :< $Text$ >	Aufruf einer Methode in einer Raum-Ressourcen-Zelle. Es können Listen von Übergabe- und Rückgabeparametern angegeben werden.
BROADCAST_AGENTLIST :	agentlist :< $Zeichenkette$ > method :< $Zeichenkette$ > inputs :< $Liste$ > text :< $Text$ >	Aufruf einer Methode in einer Liste von Agenten bzw. einer Agentenklasse. Es kann optional eine Liste von Übergabeparametern angegeben werden.
BROADCAST_RESOURCE :	resource_name :< $Zeichenkette$ > method :< $Zeichenkette$ > inputs :< $Liste$ > text :< $Text$ >	Aufruf einer Methode in einer Resourcenklasse. Es kann optional eine Liste von Übergabeparametern angegeben werden.
CALL_AGENTVALUE :	agent_id :< $Zeichenkette$ > variable :< $Zeichenkette$ > return :< $Zeichenkette$ > text :< $Text$ >	Abfrage eines öffentlichen Variablenwertes eines Agenten.
CALL_AGENTVALUE_SUM :	agentlist :< $Zeichenkette$ > variable :< $Zeichenkette$ > return :< $Zeichenkette$ > text :< $Text$ >	Abfrage der Summe der Werte einer öffentlichen Variable einer Agentenliste.
CALL_RESOURCEVALUE :	resource_name :< $Zeichenkette$ > coordinates :< $Liste$ > variable :< $Zeichenkette$ > return :< $Zeichenkette$ > text :< $Text$ >	Abfrage eines öffentlichen Variablenwertes einer Raum-Ressourcen-Zelle.

Tabelle 3.11: Vorgegebene Funktionstypen zur Auswahl von Agenten:

selectAgent($agentlist; condition$[1])	Es wird die Agenten-ID des Agenten zurückgegeben, der durch die genannten Bedingung selektiert wird. Die Bedingung muss eindeutig sein (grösster/kleinster Wert einer Zustandsvariablen, Zufallsauswahl etc.).
selectAgentlist($agentlist; condition$[2])	Es wird die Liste von Agenten-ID's zurückgegeben, die der genannten Bedingung entsprechen.
selectAgentlist_atPosition($agentlist; list\ of\ coordinates; radius$)	Es wird die Liste von Agenten-ID's einer Liste oder Klasse zurückgegeben, die sich zum Zeitpunkt der Abfrage an der genannten Position (Raumzelle) befinden. Optional kann ein Radius von Raumzellen angegeben werden.

[1] Bedingungen für eine eindeutige Agentauswahl können mit Minimum-/Maximum-Operatoren sowie üblichen mathemat. und logischen Operatoren ausgedrückt werden. Falls mehrere Agenten der Bedingung entsprechen, so erfolgt eine Zufallsauswahl aus diesen Agenten: min/max($variable$), min/max($distance$), random, etc.

[2] Bedingungen für eine Agentauswahl können mit üblichen mathemat. und logischen Operatoren ausgedrückt werden: $variable <>= value$, $distance <>= value$, etc.

4 Fallstudie: Das Grasshopper-Modell von D.J. Fielding

Mit dem Ziel einer ersten Überprüfung auf Verwendbarkeit und Vollständigkeit der Spezifizierungsmöglichkeiten der entworfenen ECOBAS-Module wurde das in FIELDING (2004) dokumentierte, agentenbasierte Heuschreckenmodell für gemäßigte Breiten von D.J. Fielding mit den Definitionselementen dieser Module neu formuliert. In Ermangelung eines anderen Akronyms wird das Modell im Folgenden '*Grasshopper*-Modell' genannt.

Das *Grasshopper*-Modell enthält mehrere typische Elemente agentenbasierter ökologischer Modelle. Dieses sind insbesondere die Modellierung von individuellen Unterschieden innerhalb einer Population zusammen mit direkter und indirekter Konkurrenz in Bezug auf eine gemeinsam genutzte Ressource, die räumlich heterogene Verteilung der Resourcen sowie eine genetisch fixierte Eigenschaft mit einer festgelegten Mutationswahrscheinlichkeit. Anhand der durch Vererbung fixierten, und nur durch spontane Mutation geringfügig veränderlichen Eigenschaft des Ausgangsgewichts der Nachkommen (Eigröße) wird in dem Grasshopper-Modell in einer einfachen Form der Einfluß des Selektionsdrucks durch die Umwelt auf die Charakteristik der am besten angepassten Eigenschaften der Individuen über mehrere Generationen simuliert.

4.1 Modellübersicht

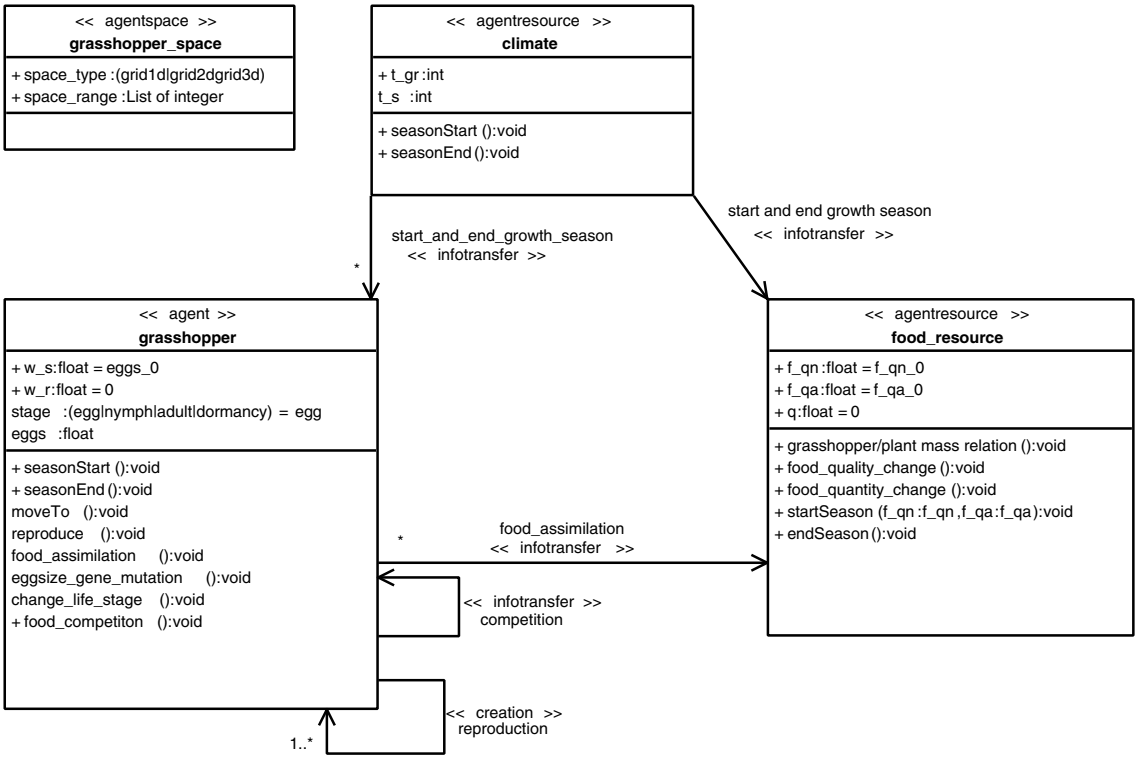
Das Modell besteht anschaulich aus den Komponenten einer zweidimensionalen Landschaft mit einem Nahrungsmittelangebot, der sich darin bewegenden Heuschrecken, die das Nahrungsangebot fressen, um die Nahrung konkurrieren und sich vermehren und einer rudimentären Klimakomponente, welche für den gesamten Raum und die Heuschrecken einheitlich den Beginn und das Ende der Wachstumsperiode vorgibt. Eine Übersicht über die hieraus erstellten drei Klassen ist in Abbildung 4.1 gegeben [1]. Die Definition und Zuordnung der Systemeinheiten (1) Heuschrecke als Stereotyp 'Agent' sowie (2) der Nahrungsressourcen und (3) des Klimas als Stereotyp 'Ressource' entspricht in diesem vergleichsweise wenig komplexen Modell der naheliegenden „natürlichen“ Anschauung.

In der Studie von Fielding wurde mit verschiedenen Simulationen von variierenden Modellkonfigurationen vor allem der Einfluß des Konkurrenzkampfs und der heterogenen räumlichen Strukturierung von Ressourcen auf die Selektion und die Entwicklung 'optimaler' Eigenschaften für den Lebenszyklus der Heuschrecken über mehrere Generationen untersucht [2]. Hierbei

[1] Das Klassendiagramm dient hier lediglich als Kurzübersicht und entspricht nicht vollständig der UML-Standarddarstellung

[2] Es werden bei der Re-Formulierung des Grasshopper-Modells mit dem hier angestrebten Ziel nicht alle in

Abbildung 4.1: Klassendiagramm des Heuschrecken-Modells.



werden in dem Modell vor allem die Parameterentwicklung der Größe der Eier, der Dauer des Nymphenstadiums bis zum Beginn der Reproduktion und des Körpergewichts bei Erreichen des Reproduktionsstadiums über mehrere Entwicklungsgenerationen von Heuschrecken beobachtet.

Der Konkurrenzkampf um die Nahrungsressourcen zwischen den Heuschrecken wird dargestellt als Verdrängung bzw. Reduzierung der Nahrungsaufnahme, die abhängig von der Körpergröße von Kontrahenten und der lokal vorhandenen relativen Nahrungsknappheit (Verhältnis von Heuschrecken-Biomasse zu Pflanzen-Biomasse innerhalb einer Zelle).

Die Größe der Eier der Heuschrecken wird als vererbte Eigenschaft mit einer festgelegten Mutationswahrscheinlichkeit und -breite modelliert. Für das Erreichen des Reproduktionsstadiums der Heuschrecken innerhalb einer Wachstumsperiode wird in der hier dargestellten ECOBAS-Formulierung die Modellvariante eines festgelegten Schwellenwertes des Körpergwichts verwendet.

---

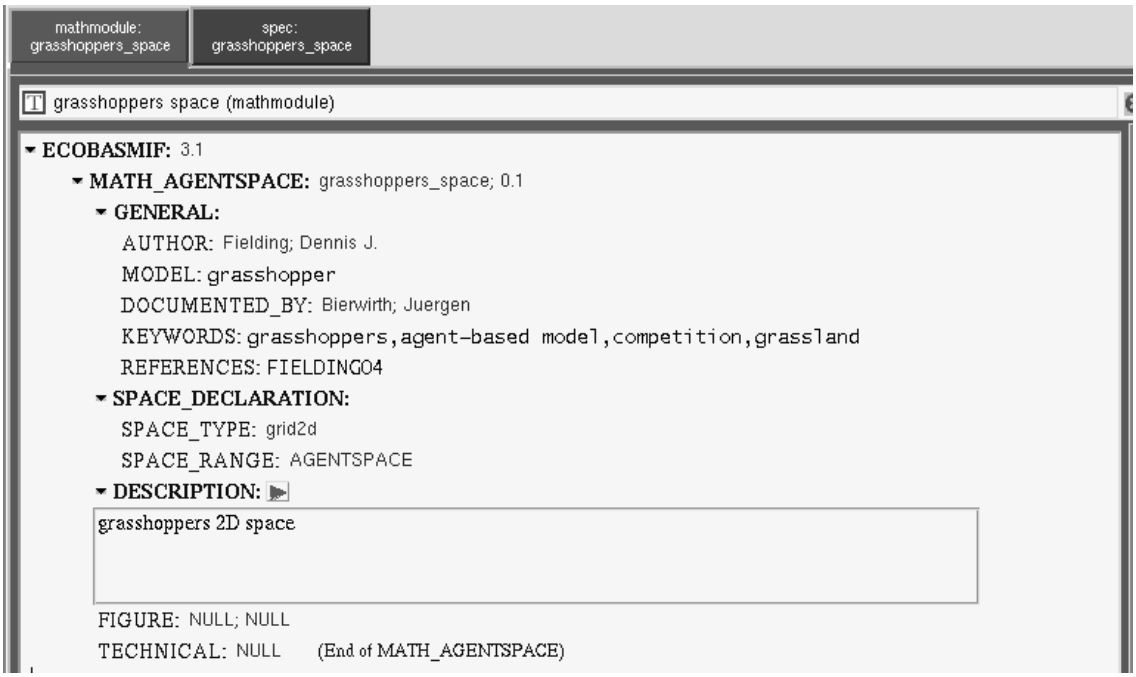
der Literaturquelle behandelten Modellvarianten dargestellt. Ebenso sei hier nur kurz erwähnt, dass in dem zugrunde liegenden Artikel von Fielding umfangreiche Ergebnisse insbesondere zum Aspekt der Auswirkungen der räumlichen Heterogenität und der Konkurrenzbeeinflussungen innerhalb einer Population dargestellt werden.

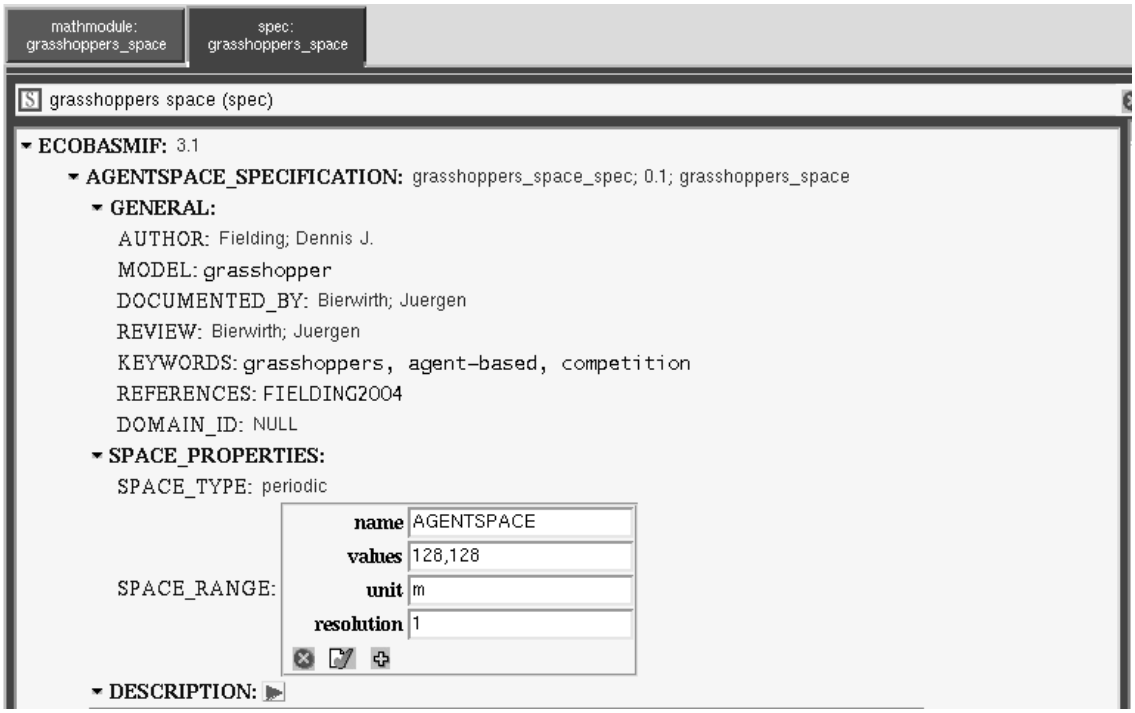
4.2 Raumdeklaration

Die Raumdeklaration in Form des *AGENTSPACE MATH*- und *SPECIFICATION*-Moduls ist in Abbildung 4.2 gezeigt. Der dem Modell zugrunde liegende Raum ist ein reguläres 2-D Gitter (siehe SPACE_TYPE - Deklaration). Die vorgegebene Variablenbezeichnung der Raumgröße ist *AGENTSPACE* (siehe SPACE_RANGE - Deklaration). Im *SPECIFICATION*-Modul ist entsprechend der Literaturvorgabe ein geschlossener periodischer Raumtyp (SPACE_TYPE 'periodic') ausgewählt (siehe auch Erläuterungen in Tab. 3.4). Exemplarisch für die Verwendung des XML-Editors sind die Optionen zur Spezifikation der SPACE_RANGE-Variable AGENTSPACE angezeigt (Die Eingabefelder werden per Mausklick geöffnet). Es wurde entsprechend der Vorlage eine Raumgröße von 128×128m mit einer Diskretisierung von 1 m angelegt. Bei der Modellsimulation müssten also für jede Ressourcen-Klasse jeweils $128^2 = 16384$ Zellen angelegt werden (gedacht als übereinander gelagerte Ebenen von räumlich angeordneten Zellen – siehe Abschnitt 2.2.1) welches alleine bereits verdeutlicht, wie schnell der Bedarf an Hardwareressourcen bei Modellformulierungen dieser Form bei Verwendung größerer Anzahlen von Agenten und Variablenparametern ernorm ansteigen kann.

Anmerkung: Die Sektion '`GENERAL`' wird für alle Komponenten des Modells in identischer Weise angelegt und ist daher in den Abbildungen der weiteren Module ausgeblendet.

Abbildung 4.2: Das *MATH*- und das *SPECIFICATION*-Modul zur Raumdeklaration des Heuschreckenmodells.





4.3 Klimadarstellung

Das Klima ist als Einflußgröße auf die Heuschreckenentwicklung in den beschriebenen Modellvarianten (FIELDING (2004, S.173)) kein Gegenstand der Untersuchung und reduziert sich damit lediglich auf die Vorgabe der konstanten Dauer der Wachstumsperiode. Diese könnte natürlich problemlos auch als Eigenschaft innerhalb der Heuschrecken- und der Nahrungsklasse formuliert werden. Zur Darstellung eines modularen Modellaufbaus und der prinzipiellen Ermöglichung einer leichten Erweiterbarkeit des Modells z.B. auf die Untersuchung des Einflusses des Temperaturverlaufs während der Vegetationsperiode oder regional unterschiedlicher Mikroklimate, wird in der hier dargestellten Beispielformulierung dennoch eine eigenständige Ressourcenklasse angelegt.

Zu Beginn und zum Ende der der Wachstumsperiode wird jeweils eine Start- und Beendungs-Nachricht an die Food-Ressourcenzelle und an die Liste aller vorhandenen Agenten in der gleichen Raumzelle gesendet. An die Foodressourcenzellen wird dabei die Information der Dauer der Wachstumsperiode entsprechend der Literaturvorgabe bereits beim Start weitergegeben, da diese zur Berechnung der zeitabhängigen Komponente des Abfalls der Verdaulichkeit (Futterqualität) der Pflanzenbiomasse verwendet wird und für alle Jahresdurchläufe konstant gehalten wird.

Abbildung 4.3: Das *MATH*-Modul der Ressourcenklasse Klima.

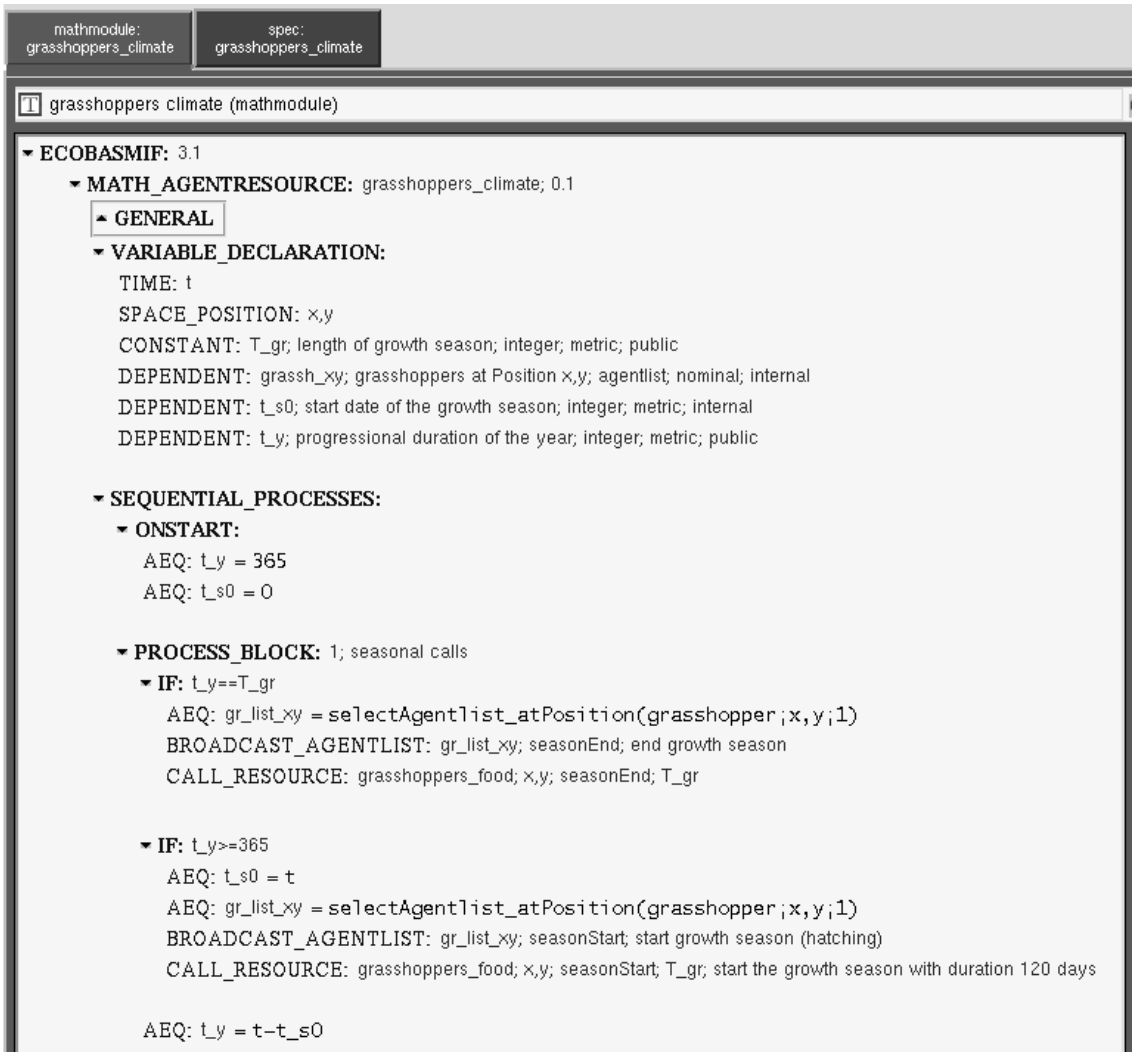
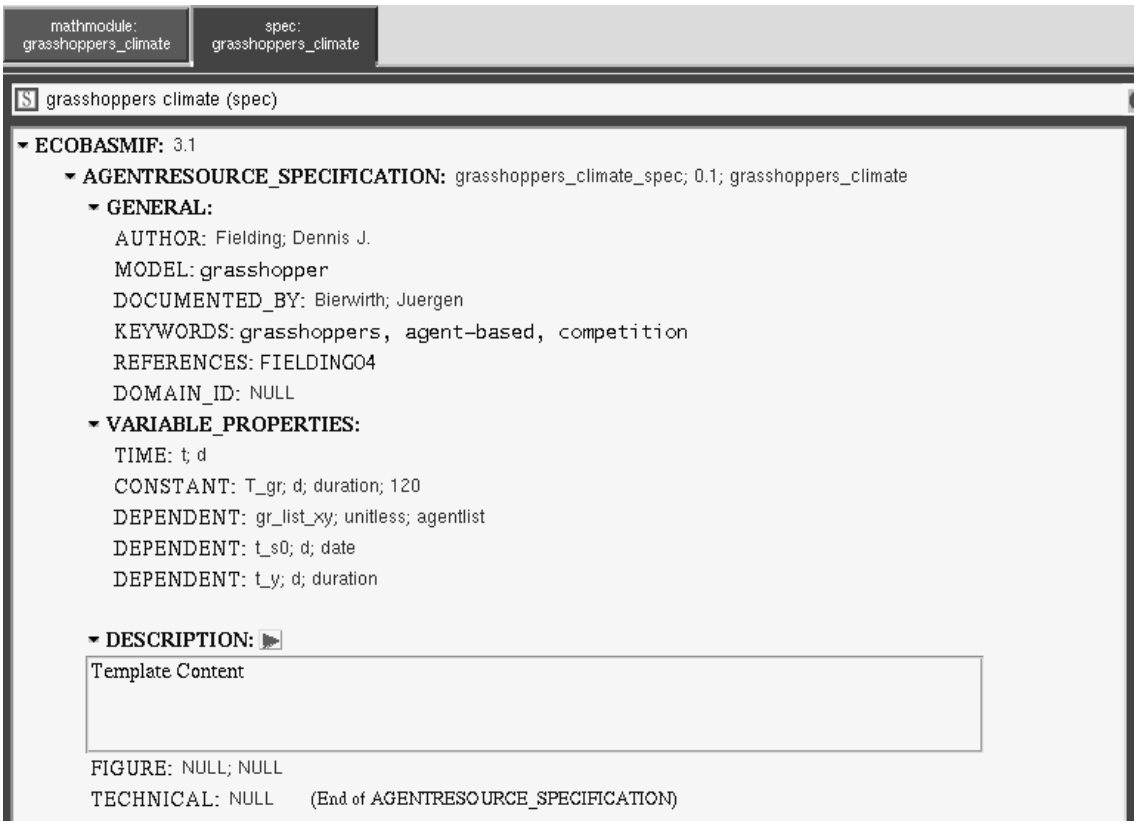


Abbildung 4.4: Das *SPECIFICATION*-Modul zur Klimadarstellung.



4.4 Darstellung der Nahrungsressourcen-Klasse

Zu Beginn der Vegetationsperioden (Aufruf der Methode 'startSeason') wird in allen Zellen die Pflanzenbiomasse auf den konstanten Startwert (f_{qn0}; 4000mg Trockengewicht) festgelegt. Der Startwert für die Futterqualität (Verdaulichkeit) zu Beginn der Saison (f_{qa0}) wird für jede Zelle individuell als Zufallswert einer Normalverteilung mit festem Mittelwert ($f_{qd} = 0.215$) und fester Standardabweichung gesetzt (siehe Abbildungen 4.5 und 4.8). In den Untersuchungen von Fielding wurden zur Überprüfung des Einflusses der heterogenen Ressourcenverteilung zudem auch die Varianten einer gleichmäßigen Verteilung und einer Verteilung entsprechend einer Fraktal-Funtion verwendet.

Der Abfall der Futterqualität (f_{qa}) während der Vegetationsperiode stellt sich als Funktion der Ausgangsqualität zu Beginn der Periode (f_{qa0}), der fortschreitenden saisonalen Zeitdauer (t_s) und der Länge Vegetationsperiode (T_{gr}) und des proportionalen Anteils an (selektiv) abgefressenener Pflanzenbiomasse (q) dar (Abb. 4.6: PROCESS_BLOCK 2):

$$f_{qa} = f(f_{qa0}, t, T_{gr}, q) = f_{qa0}\left(1 - 0.8\left(\frac{t_s}{T_{gr}}\right)^{1.5}\right)(1 - q^2) \tag{4.1}$$

In Abbildung 4.7 wird exemplarisch die Definition der 'Response'-Methode *f_cons* gezeigt, durch deren Aufruf Heuschrecken Nahrung verzehren. Übergabewerte der Heuschrecke sind die Obergrenze der aufnehmbaren Pflanzenbiomasse pro Zeitschritt (f_{m_max}: der Inputwert ist abhängig von der momentanen Körpermasse der Heuschrecke) und die „gewünschte“ Menge an Nahrungäquivalenten (f_w: assimilierbare Nährstoffmenge: Pflanzenmasse $*$ Verdaulichkeit). Bei der Methodenverarbeitung wird anhand des augenblicklichen Nahrungsvorrats und der Qualität berechnet, wieviel Pflanzenmasse demzufolge gefressen wird (Rückgabewert ist die daraus berechnete Menge an Nahrungsäquivalenten) und der Nahrungsvorrat in der Zelle wird entsprechend verringert.

Abbildung 4.5: Die Variablendeklaration und die Methoden der Nahrungsressourcen-Klasse.

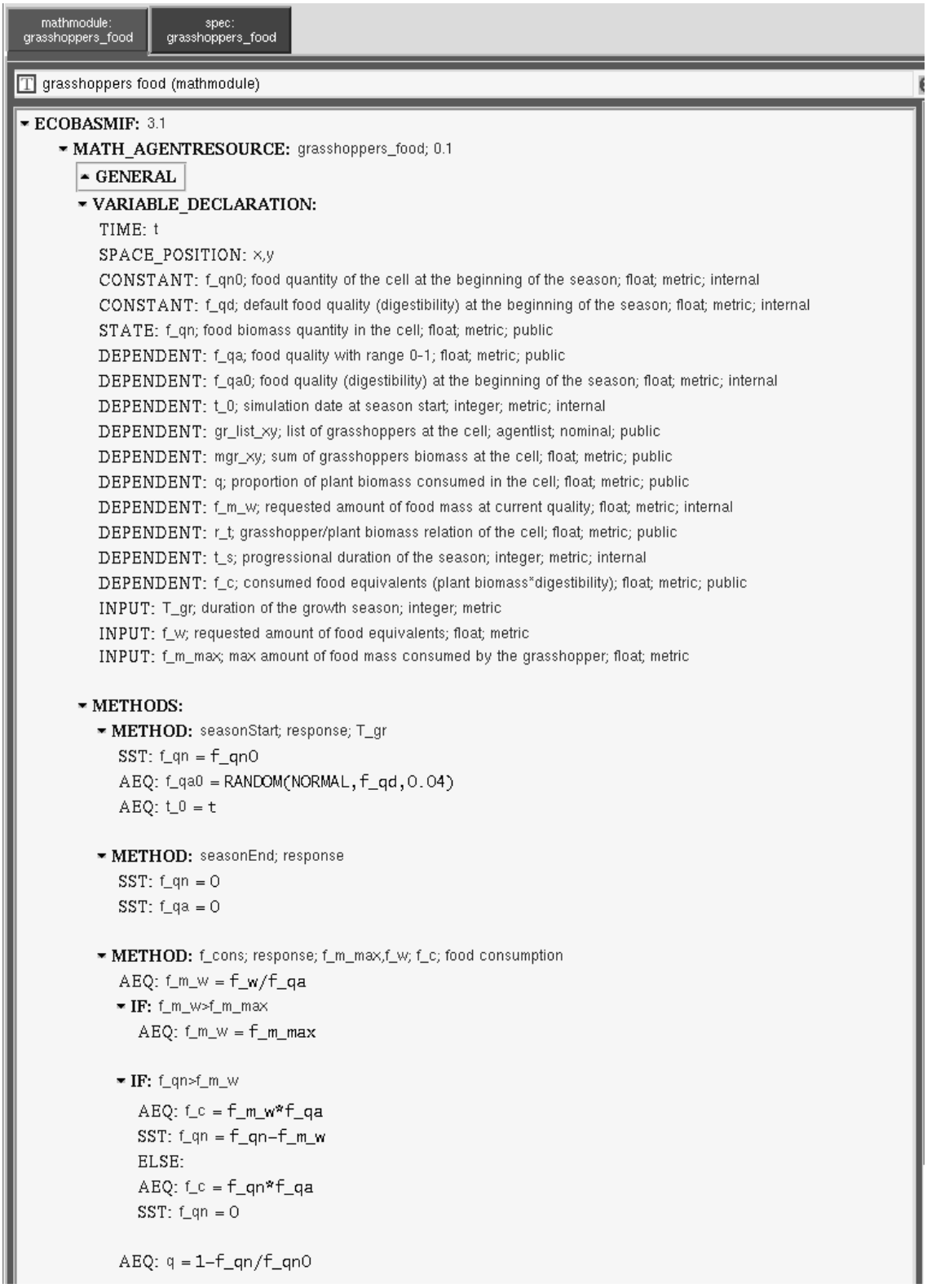


Abbildung 4.6: Die sequenziellen Prozesse der Nahrungsressourcen-Klasse.

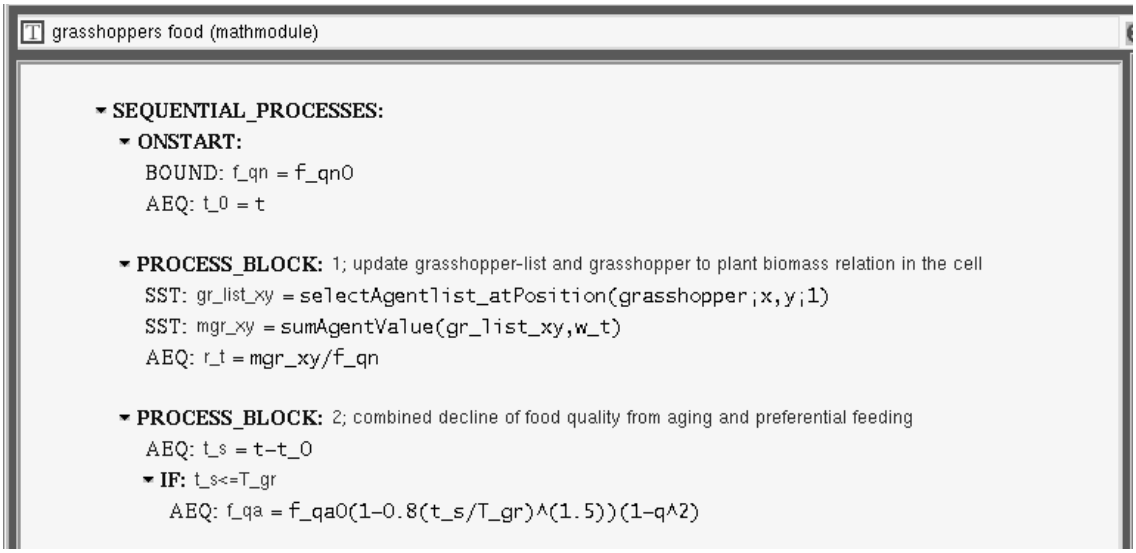


Abbildung 4.7: Methodenformulierung am Beispiel der Funktion zum Verzehr von Pflanzenbiomasse im Nahrungsressourcen-Modul.

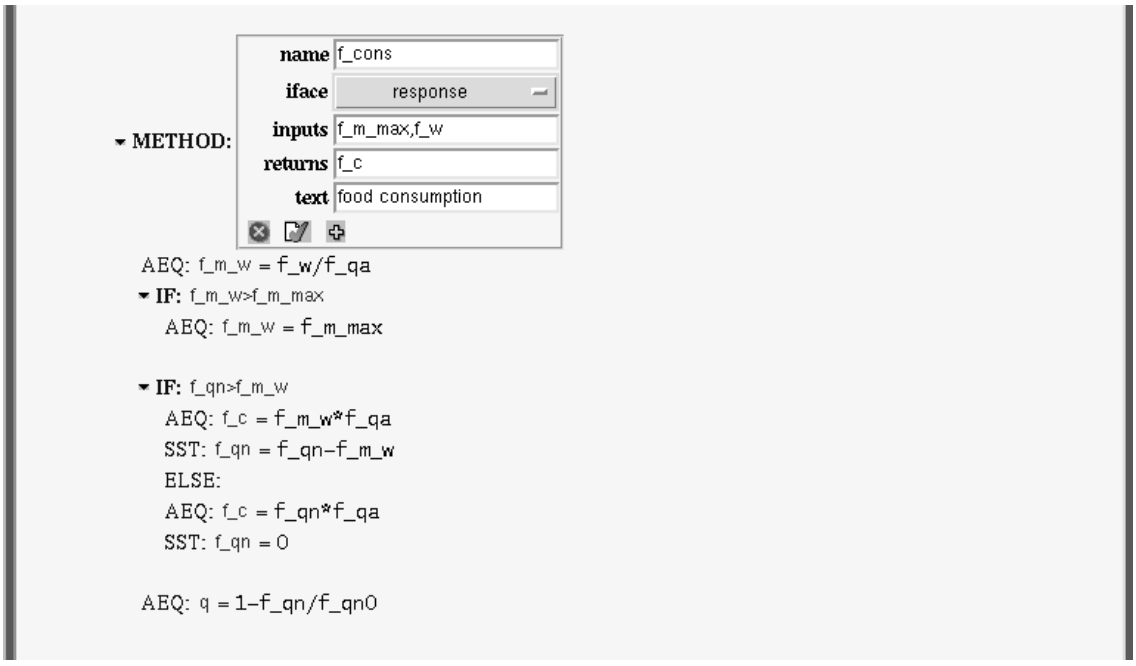
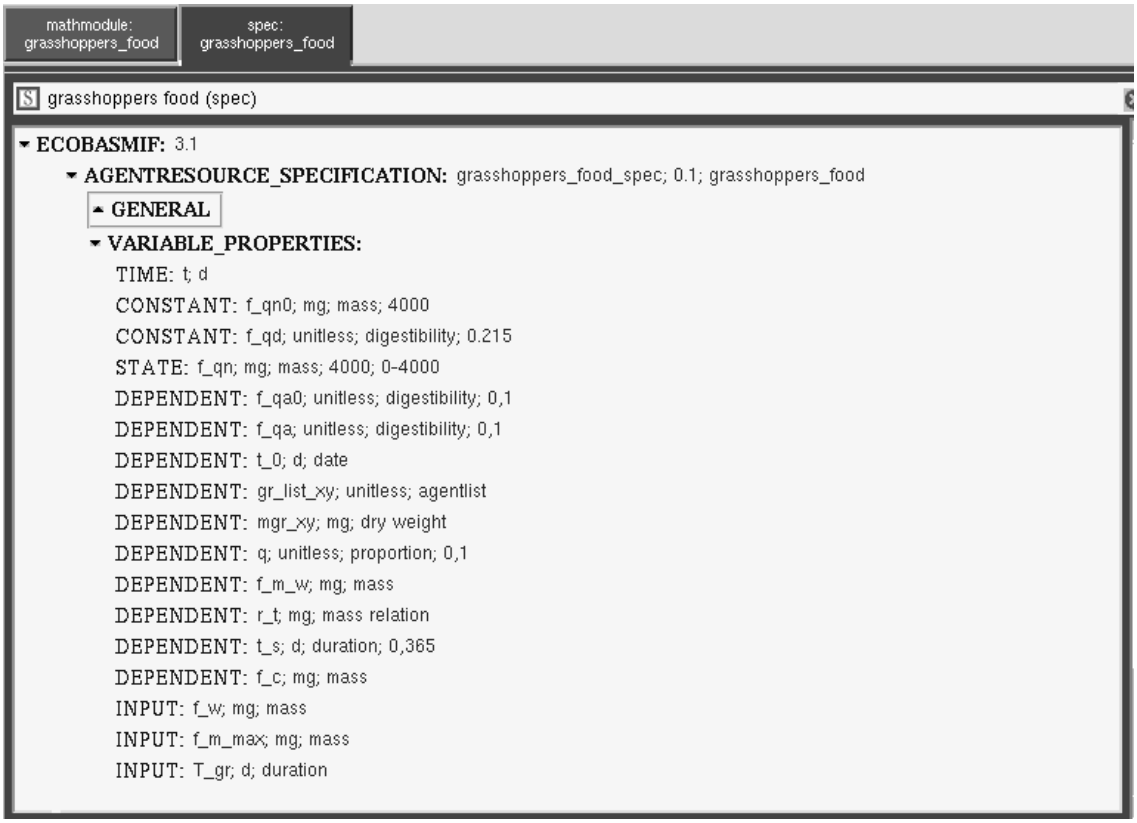


Abbildung 4.8: Die Variablenspezifizierung der Nahrungsressourcen-Klasse im *SPECIFICATION*-Modul.



4.5 Darstellung der Heuschrecken-Klasse

Die wesentlichen Merkmale der Heuschrecken in der hier re-formulierten Variante des Grasshopper-Modells lassen sich repräsentieren durch folgende Eigenschaften:

(a) Entwicklungsstadien (Variablenname: $stage$)
Heuschrecken durchlaufen die drei Hauptentwicklungsphasen (i) Ei, (ii) Nymphe und (iii) ausgewachsenes Insekt (Geschlechtsreife). Während der Winterruhe können Eier zudem im dormanten Zustand überdauern. Nymphen und adulte Heuschrecken sterben mit dem Ende der Vegetationsperiode(*seasonEnd*-Methode) und Nymphen schlüpfen gleichzeitig mit dem Beginn der Vegetationsperiode (*seasonStart*-Methode). Nymphen können im vorgegegebenen Modell nicht in der gleichen Vegetationsperiode aus den Eiern schlüpfen in der sie abgelegt wurden.

(b) Körpermasse (Variablenname: w_s)
Die Körpermasse nimmt ausschließlich im Nymphenstadium mit der Nahrungsaufnahme zu, wenn mehr Nahrung aufgenommen wurde, als Körperenergie verbraucht wurde (PROCESS_BLOCK: 3).

(c) Reproduktivgewicht: das Gewicht der noch nicht abgelegten Eier (Variablenname: w_r)
Ab dem Zeitpunkt der Geschlechtsreife wird im Modell die Nahrungsassimilation abzüglich der verbrauchten Körperenergie vollständig in Reproduktivgewicht, d.h. die Erzeugung von Eiern umgesetzt (PROCESS_BLOCK: 4).

(d) mittlere Wachstumsrate über drei Tage (Variablenname: k_3)
Diese ist Grundlage der Beurteilung des Hungerzustands der Heuschrecke und bestimmt die Motivation, sich von der momentanen Position wegzubewegen. Im Modell wird dieses dargestellt durch die direkte Umrechnung in eine Wahrscheinlichkeit zum Verlassen der Raumzelle (PROCESS_BLOCK: 5). Adulte Heuschrecken haben dabei einen relativ großen Bewegungsradius (die halbe Ausdehnung des Modellraumes), Nymphen können sich bei einem Zeitschritt jeweils nur in die Nachbarzelle fortbewegen.
Unterschreitet die mittlere Wachstumsrate von 3 Tagen einen spezifischen Wert, so stirbt die Heuschrecke.

(e) Schwellen-Gewicht zum Erreichen der Geschlechtsreife des adulten Stadiums (Variablenname: w_a)

(f) das „Eigewichts-Gen“ zur Vererbung des fixierten Eigewichts (Variablenname: $eggs$)
In der dargestellten Modellvariante ist das Eigewicht fixiert und wird ausschliesslich bei der Reproduktion mit einer festen Wahrscheinlickeit verändert („Mutation des Eigewichts-Gens“). Dieses wird durch Aufruf der Methode '*eggs_m*' im Initialisierungsabschnitt '*ON-CREATION*' umgesetzt (siehe Abb. 4.11 und 4.10).

(g) die Raumpostition (Raumkoodinatenbezeichnung: $xPos, yPos$)

4.5.1 Konkurrenz-Interaktionen

Einer der wesentlichen Untersuchungsgegenstände des Modells ist die Frage, ob die Modellierung der Nahrungskonkurrenz innerhalb einer Heuschreckenpopulation mit einer räumlich heterogen Nahrungsstruktur und individuellen Unterschieden der Heuschrecken auf die Simulationsergebnisse signifikante Auswirkungen hat im Vergleich zu einem Modellansatz mit homogener Ressourcenverteilung und der Optimierung von „Durchschnitts"-Individuen der Population.

Die Konkurrenzbeziehung zwischen den Heuschrecken wird im Grasshopper-Modell durch die folgenden zwei Komponenten formuliert: (a) Die indirekte Beeinträchtigung untereinander über das Wegfressen begrenzter Nahrungsressourcen („wer zu spät kommt, den bestraft das Leben").
(b) die direkte Störung der Individuen untereinander, die sowohl von der Bestandesdichte und Nahrungsknappheit, als auch von der Körpergröße der Individuen beim direkten Aufeinandertreffen abhängig ist („der Stärkere gewinnt").

Die Modellierung der ersten Konkurrenzkomponente ist dabei offensichtlich und besteht in der Formulierung des Freßvorgangs an sich. Einige Aspekte wurden hierzu bereits in Abschnitt 4.4 mit der Nahrungsverzehrs-Methode *f_cons* der Nahrungsressourcen-Klasse erläutert, die von den Heuschreckenindividuen aufgerufen wird (Abb. 4.11: PROCESS_BLOCK: 1). Der Bedarf an Nahrungsaufnahme pro Zeitschritt ist dabei abhängig von der Körpergröße des Individuums, durch welche die maximal aufnehmbare Pflanzenmasse begrenzt wird und der Menge an Nährstoffeinheiten bestimmt wird. Durch den Maximalwert der aufnehmbaren Pflanzenmasse pro mg Körpermasse (c_max) wird dabei begrenzt, in wie weit eine schlechtere Futterqualität (Verdaulichkeit) durch höhere Pflanzenmasse ausgeglichen werden kann.

Zur Modellierung einer direkten Störung von Individuen untereinander (Abb.4.10: Methode *f_comp*) wird ein Indiviuum aus der Liste der sich momentan in der Raumzelle befindlichen Heuschrecken ausgewählt. In Abhängigkeit von dem Verhältnis der eigenen Körpergröße zu der des zufällig ausgewählten Individuums wird die Menge der von der Nahrungsressourcenzelle angeforderten Nahrung auf 50% reduziert. Hierbei spielt zusätzlich das in der Ressourcenzelle vorhandene Gesamtverhältnis von Heuschreckenmasse zu Pflanzenmasse eine Rolle. Je geringer die Knappheit an Nahrung in der Zelle ist, desto geringer ist die Wahrscheinlichkeit eines Zusammentreffens von Individuen mit der Folge der Behinderung der Nahrungsaufnahme.

Abbildung 4.9: Die Variablendeklaration der Heuschrecken-Klasse im *MATH*-Modul.

Abbildung 4.10: Die Methodendeklaration der Heuschrecken-Klasse im *MATH*-Modul.

Abbildung 4.11: Definition der sequenziellen Prozesse (i) der Heuschrecken-Klasse im *MATH*-Modul.

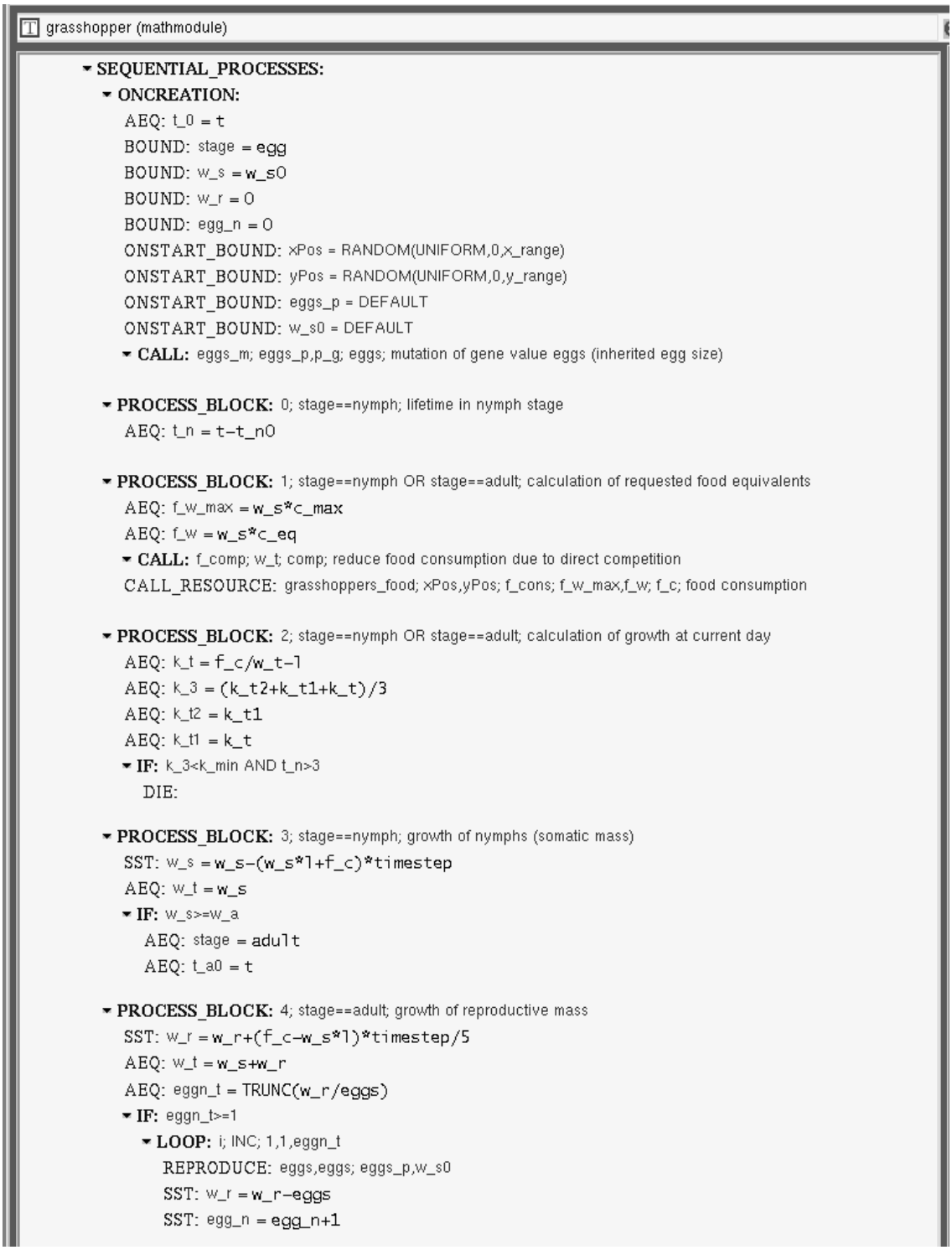


Abbildung 4.12: Definition der sequenziellen Prozesse (ii) der Heuschrecken-Klasse im *MATH*-Modul.

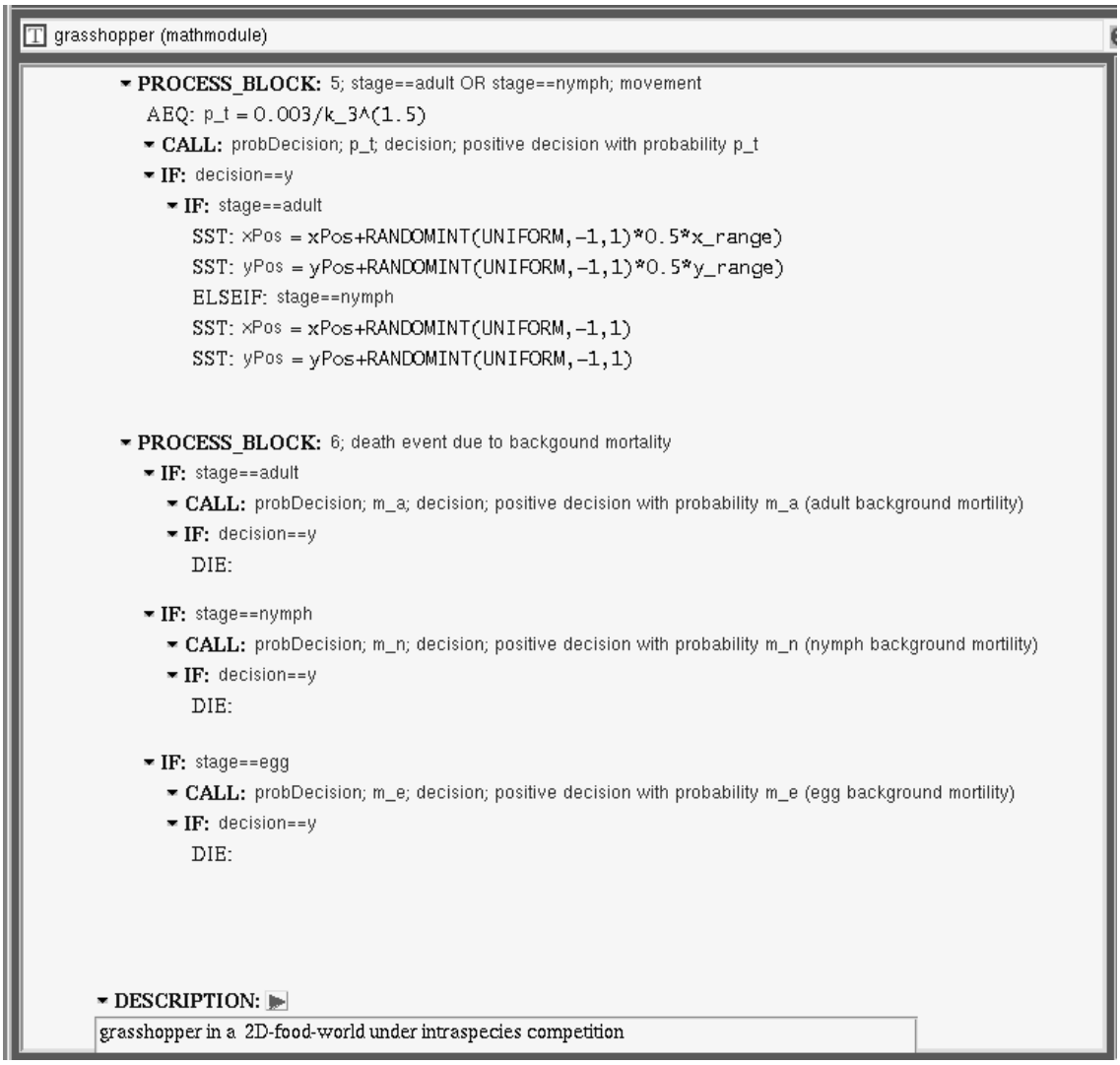
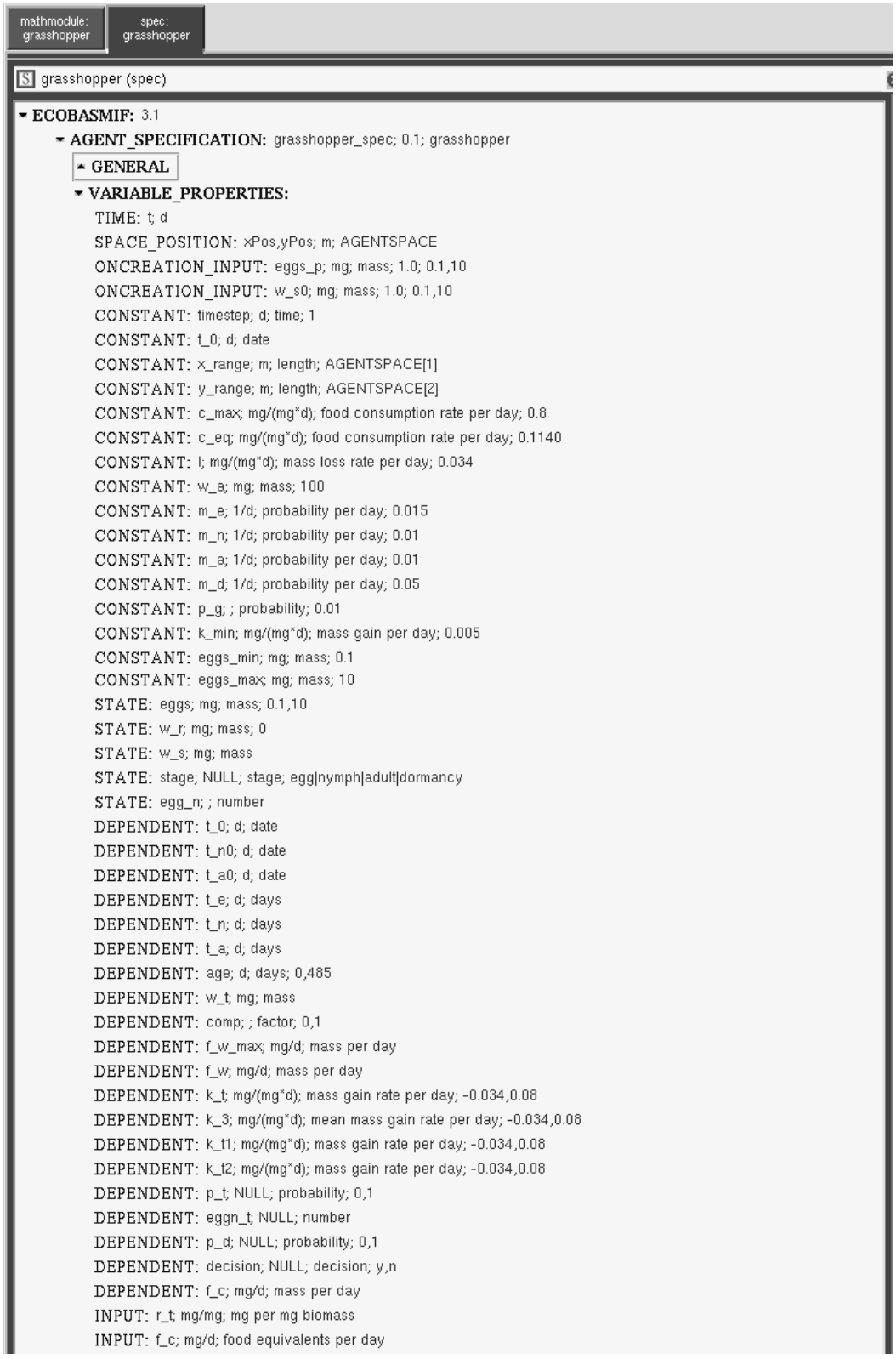


Abbildung 4.13: Die Variablenspezifikation der Heuschrecken-Klasse.



5 Zusammenfassung und Ausblick

Es wurden die Hintergründe, Ziele und Problematiken individuenbasierter Modellierungsansätze aufgegriffen sowie konzeptionelle Anforderungen und Elemente eines Systems zur Erstellung und Dokumentation individuenbasierter Modelle aufgezeigt. Obwohl der Forschungsbereich der individuenbasierten ökologischen Modellierung inzwischen nicht mehr ganz jung ist (wesentliche Aktivitäten haben sich seit Anfang der 80er Jahre entwickelt), ist es bis dato nur unzureichend gelungen, befriedigende Standards für die Unterstützung der Modellentwicklung, der Modellüberprüfung und der Dokumentation zu entwickeln. Wie in der Literatur zahlreich diskutiert wird, ist es für eine erfolgreiche Weiterentwicklung und Verbesserung des wissenschaftlichen Nutzens individuenbasierter Modellierung für ökologische Fragestellungen essentiell, dass die Kommunizierbarkeit und die Überprüfbarkeit der Modelle verbessert und vereinfacht wird. Der in dieser Arbeit verfolgte Ansatz basiert darauf, als Ziel möglichst nicht eine neue Meta-Sprache als Erweiterung einer bestehenden Programmierumgebung zu entwickeln, sondern, entsprechend dem Ansatz von ECOBAS, die Modellformulierung so weit wie möglich auf die allgemein bekannten mathematischen Sprachelemente zu konzentrieren und zusätzlich einen hiermit konsistenten Rahmen für die Deklaration der physikalischen und ökologischen Eigenschaften der Modellparameter zur Verfügung zu stellen. Außerhalb dieses Rahmens wurde versucht, die Anzahl zusätzlicher Sprach- und Deklarationselemente zur Unterstützung typischer individuenbasierter Modellstrukturen so gering wie möglich zu halten. Die zusätzlichen Sprachelemente resultieren im Wesentlichen aus der Eigenschaft individuenbasierter Modelle, dass während der Simulationslaufzeit Individuen erzeugt werden können und sterben. Neben den direkt hierzu notwendigen Routinen wurden in diesem Zusammenhang die Sprachelemente zur Objektinteraktion in Form von „Messaging“- und „Broadcasting“-Routinen erweitert und einige Funktionen zur Objektselektion vorgesehen.

Als Grundstruktur zur Modellerstellung wurde eine Aufteilung in die Raumdeklaration, die Ressourcenobjekte und die Agentenobjekte des zu modellierenden Gesamtsystems festgelegt. Hierauf aufbauend wurden XML-Dokumenttyp-Definitionen (XML-DTDs) zur Festlegung der Modellelemente und Eingabe aller Modellinformationen im XML-Editor des *ECOBAS Modelling Assistant* erstellt.

Zu einer ersten Überprüfung auf Konsistenz und Vollständigkeit der erstellten Definitionsschemata wurde als Fallstudie ein Heuschreckenmodell mit mehreren typischen Elementen individuenbasierter Modelle wie räumlich heterogener Ressourcenverteilung, individueller Konkurrenz-Interaktionen und Vererbbarkeit individueller Eigenschaften in den angelegten XML-Formaten re-formuliert. Die hierbei aufgetretenen Schwierigkeiten wurden iterativ in den Entwurf des De-

finitionsschemas eingebracht, sodass im Rahmen der Fallstudie eine erste weitgehend vollständige Modellformulierung erstellt werden konnte.

Ausblick

Im Rahmen der vorliegenden Arbeit kann der gewählte Ansatz zur Entwicklung eines geeigneten Modelldefinitionsformats, mit dem individuenbasierte ökologische Modelle in einer lesbaren und zugleich genügend exakten Form erstellt und dokumentiert werden können bestätigt werden. Darüber hinaus kann jedoch noch nicht abgeschätzt werden, welche Schwierigkeiten bei dem Versuch der Erstellung eines Konverters zur weitgehend automatisierten Umsetzung des Modelldefinitionsformates in ein Simulationssystem auftreten würden und welches Simulationssytem ein geeignetes Zielsystem darstellt. In diesem Zusammenhang ist für eine Weiterentwicklung des Ansatzes zu prüfen, welche Spezifikationen ein geeignetes Simulationssystem als Zielsystem für die Erstellung eines Konverters auf der einen Seite und zur Modellanalyse und Darstellung der Simulationsergebnisse auf der anderen Seite ausmachen. Als erste Möglichkeit wäre zu analysieren, in wie weit sich die Implementierungsebenen von SeSAm oder von SWARM/RePAST als vermutlich am weitesten entwickelte Simulationstools im Bereich agentenbasierter Modellierung mit dem jeweils angeschlossenen Simulationssystem als Exportformat für einen Konverter eignen. Dieses würde eine bedeutende Reihe von Tools zur Analyse und zur Darstellung der Simulationsergebnisse zugänglich machen. Darüber hinaus ist auch auf der Seite der Modellkonzeption und Erstellung weiter zu beobachten, wie sich mögliche Weiterentwicklungen von speziellen, auf UML aufbauende Systeme wie SeSAm-UML zur standardisierten Erstellung komplexer Modelle kombinieren oder als Startoberfläche zur Modellkonstruktion nutzen lassen.

A Agentenspezifische Erweiterungen in den ECOBAS-MIF DTDs

```
<!--======================== Main Document Type ==============================-->
<!--==========================================================================-->
<!ELEMENT ECOBASMIF ((MATH_AGENT|MATH_AGENTRESOURCE|MATH_AGENTSPACE|
                     AGENT_SPECIFICATION|AGENTRESOURCE_SPECIFICATION|
                     AGENTSPACE_SPECIFICATION)+)>

<!--============================ Math AGENT Modules ============================-->
<!--==========================================================================-->

<!ELEMENT MATH_AGENT   (GENERAL, AGENT.VARIABLE_DECLARATION,
                        AGENT.METHODS, AGENT.SEQUENTIAL_PROCESSES,
                        DESCRIPTION?,((FIGUREA|FIGURE)|(TECHNICAL|TECHNICALA))*)
>
<!ATTLIST MATH_AGENT
                        name       CDATA    #REQUIRED
                        version    NMTOKEN  #REQUIRED
                        folder     CDATA    #IMPLIED
>
<!ELEMENT MATH_AGENTRESOURCE (GENERAL, AGENTR.VARIABLE_DECLARATION,
                              AGENTR.METHODS,
                              AGENTR.SEQUENTIAL_PROCESSES,
                              DESCRIPTION?,
                              ((FIGUREA|FIGURE)|(TECHNICAL|TECHNICALA))*)
>
<!ATTLIST MATH_AGENTRESOURCE   name       CDATA    #REQUIRED
                               version    NMTOKEN  #REQUIRED
                               folder     CDATA    #IMPLIED
>
<!ELEMENT MATH_AGENTSPACE   (GENERAL, SPACE_DECLARATION, DESCRIPTION?)
>
<!ATTLIST MATH_AGENTSPACE
                            name       CDATA    #REQUIRED
                            version    NMTOKEN  #REQUIRED
                            folder     CDATA    #IMPLIED
>

<!--========================== Specification Modules ==========================-->
<!--==========================================================================-->

<!ELEMENT AGENT_SPECIFICATION  (SPECIFICATION.GENERAL,
                                AGENT.VARIABLE_PROPERTIES, DESCRIPTION?,
                                ((FIGUREA|FIGURE)|(TECHNICAL|TECHNICALA))*)
>
<!ATTLIST AGENT_SPECIFICATION
                                name        CDATA    #REQUIRED
                                version     NMTOKEN  #REQUIRED
                                math_modul  CDATA    #REQUIRED
>
<!ELEMENT AGENTRESOURCE_SPECIFICATION (SPECIFICATION.GENERAL,
                                       AGENTR.VARIABLE_PROPERTIES,
                                       DESCRIPTION?,
                                      ((FIGUREA|FIGURE)|(TECHNICAL|TECHNICALA))*)
>
<!ATTLIST AGENTRESOURCE_SPECIFICATION
                                         name        CDATA    #REQUIRED
                                         version     NMTOKEN  #REQUIRED
                                         math_modul  CDATA    #REQUIRED
>
<!ELEMENT AGENTSPACE_SPECIFICATION  (SPECIFICATION.GENERAL,
                                     SPACE_PROPERTIES,
                                     DESCRIPTION?) >
<!ATTLIST AGENTSPACE_SPECIFICATION
                                     name        CDATA    #REQUIRED
                                     version     NMTOKEN  #REQUIRED
                                     math_modul  CDATA    #REQUIRED >
```

```
<!--========== MATH AGENT/AGENTRESOURCE/AGENTSPACE CHILDREN ELEMENTS ==========-->
<!--============================================================================-->

<!ELEMENT GENERAL ((AUTHOR|MODEL|DOCUMENTED_BY|REVIEW|KEYWORDS|REFERENCES)+) >
<!ELEMENT SPECIFICATION.GENERAL    ((AUTHOR|MODEL|DOCUMENTED_BY|REVIEW|
                                    KEYWORDS|REFERENCES|DOMAIN_ID)+) >

<!ELEMENT AGENT.VARIABLE_DECLARATION (AGENT.TIME, AGENT.SPACE_POSITION,
                                     (AGENT.ONCREATION_INPUT|AGENT.CONSTANT|
                                      AGENT.STATE|AGENT.DEPENDENT|AGENT.INPUT)+ )>
<!ELEMENT AGENTR.VARIABLE_DECLARATION (AGENT.TIME, AGENT.SPACE_POSITION,
                                      (AGENT.CONSTANT|AGENT.STATE|
                                       AGENT.DEPENDENT|AGENT.INPUT)+ )>

<!ELEMENT SPACE_DECLARATION (SPACE_TYPE, SPACE_RANGE)>

<!ELEMENT AGENT.METHODS (AGENT.METHOD+) >

<!ELEMENT AGENTR.METHODS (AGENTR.METHOD+) >

<!ELEMENT AGENT.SEQUENTIAL_PROCESSES (AGENT.ONCREATION,
                         (AGENT.PROCESS_BLOCK|AGENT.IF|AGENT.LOOP|
                          AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                          AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                          AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                          AGENT.CALL_RESOURCEVALUE|AGENT.REPRODUCE|AGENT.DIE|
                          AGENT.CREATE)+, CONSTRAINT*)
>

<!ELEMENT AGENTR.SEQUENTIAL_PROCESSES (AGENTR.ONSTART,
                         (AGENTR.PROCESS_BLOCK|AGENTR.IF|AGENTR.LOOP|
                          AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                          AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                          AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                          AGENT.CALL_RESOURCEVALUE)+,
                          CONSTRAINT*)
>

<!--==== SPECIFICATION AGENT/AGENTRESOURCE/AGENTSPACE CHILDREN ELEMENTS =====-->
<!--============================================================================-->

<!ELEMENT AGENT.VARIABLE_PROPERTIES  (AGENT_PROP.TIME,
                            AGENT_PROP.SPACE_POSITION,
                            (AGENT_PROP.ONCREATION_INPUT|AGENT_PROP.CONSTANT|
                             AGENT_PROP.STATE|AGENT_PROP.DEPENDENT|
                             AGENT_PROP.INPUT)+)
>
<!ELEMENT AGENTR.VARIABLE_PROPERTIES  (AGENT_PROP.TIME,
                            (AGENT_PROP.CONSTANT|AGENT_PROP.STATE|
                             AGENT_PROP.DEPENDENT|AGENT_PROP.INPUT)+)
>
<!ELEMENT SPACE_PROPERTIES  (PROP.SPACE_TYPE, PROP.SPACE_RANGE)>

<!--================= AGENT/AGENTRESOURCE VARIABLE_DECLARATION ==============-->
<!--============================================================================-->

<!ELEMENT AGENT.TIME  EMPTY >
<!ATTLIST AGENT.TIME  name  CDATA #REQUIRED
>
<!ELEMENT AGENT.SPACE_POSITION  EMPTY >
<!ATTLIST AGENT.SPACE_POSITION  coord_list  CDATA #REQUIRED
>

<!ELEMENT AGENT.ONCREATION_INPUT  EMPTY >
<!ATTLIST AGENT.ONCREATION_INPUT  name  CDATA #REQUIRED
                        text   CDATA #IMPLIED
                        vtyp   (float|integer|alpha|agent_id|agentlist)  "float"
                        scale  (metric|ordinal|nominal|logical)  "metric"
                        dim    CDATA #IMPLIED
>

<!ELEMENT AGENT.STATE   EMPTY >
<!ATTLIST AGENT.STATE   name  CDATA #REQUIRED
                        text  CDATA #IMPLIED
                        vtyp  (float|integer|alpha|agent_id|agentlist)  "float"
```

```
                      scale (metric|ordinal|nominal|logical)  "metric"
                      iface (internal|public) #REQUIRED
                      dim   CDATA #IMPLIED
>

<!ELEMENT AGENT.CONSTANT   EMPTY >
<!ATTLIST AGENT.CONSTANT   name  CDATA #REQUIRED
                           text  CDATA #IMPLIED
                           vtyp (float|integer|alpha|agent_id|agentlist) "float"
                           scale (metric|ordinal|nominal|logical)  "metric"
                           iface (internal|public) #REQUIRED
                           dim   CDATA #IMPLIED
>

<!ELEMENT AGENT.DEPENDENT   EMPTY >
<!ATTLIST AGENT.DEPENDENT   name  CDATA #REQUIRED
                            text  CDATA #IMPLIED
                            vtyp (float|integer|alpha|agent_id|agentlist) "float"
                            scale (metric|ordinal|nominal|logical)  "metric"
                            iface (internal|public) #REQUIRED
                            dim   CDATA #IMPLIED
>
<!ELEMENT AGENT.INPUT  EMPTY >
<!ATTLIST AGENT.INPUT  name   NMTOKEN #REQUIRED
                       text   CDATA #IMPLIED
                       vtyp   (float|integer|alpha|agent_id|agentlist) "float"
                       scale  (metric|ordinal|nominal|logical)  "metric"
                       dim    CDATA #IMPLIED
>

<!--====================== AGENT VARIABLE_SPECIFICATION ======================-->

<!ELEMENT AGENT_PROP.TIME  EMPTY >
<!ATTLIST AGENT_PROP.TIME  name    NMTOKEN #REQUIRED
                               unit    NMTOKEN #REQUIRED
>

<!ELEMENT AGENT_PROP.SPACE_POSITION  EMPTY >
<!ATTLIST AGENT_PROP.SPACE_POSITION
                         coord_list   CDATA    #REQUIRED
                         unit         NMTOKEN  #REQUIRED
                         default      CDATA    #IMPLIED
                         space_range  CDATA    #IMPLIED "AGENTSPACE"
>
<!ELEMENT AGENT_PROP.ONCREATION_INPUT  EMPTY >
<!ATTLIST AGENT_PROP.ONCREATION_INPUT  name  NMTOKEN #REQUIRED
                       unit                  NMTOKEN #REQUIRED
                       meaning               CDATA   #REQUIRED
                       default               CDATA   #IMPLIED
                       oncreation_source    (agent|datamodule) "agent"
                       onstart_source        (default|datamodule) "default"
                       range                 CDATA #IMPLIED
                       method                CDATA #IMPLIED
>
<!ELEMENT AGENT_PROP.STATE  EMPTY >
<!ATTLIST AGENT_PROP.STATE  name    NMTOKEN #REQUIRED
                            unit    NMTOKEN #REQUIRED
                            meaning CDATA   #REQUIRED
                            default CDATA   #IMPLIED
                            range   CDATA   #IMPLIED
                            method  CDATA   #IMPLIED
>
<!ELEMENT AGENT_PROP.CONSTANT  EMPTY >
<!ATTLIST AGENT_PROP.CONSTANT  name         NMTOKEN #REQUIRED
                               unit         NMTOKEN #REQUIRED
                               meaning      CDATA   #REQUIRED
                               value        CDATA   #IMPLIED
                               method       CDATA   #IMPLIED
                               outdomain    CDATA   #IMPLIED
>
<!ELEMENT AGENT_PROP.DEPENDENT  EMPTY >
<!ATTLIST AGENT_PROP.DEPENDENT  name         NMTOKEN #REQUIRED
                                unit         NMTOKEN #REQUIRED
                                meaning      CDATA   #REQUIRED
                                range        CDATA   #IMPLIED
```

```
                              method       CDATA   #IMPLIED
                              outdomain    CDATA   #IMPLIED
>
<!ELEMENT AGENT_PROP.INPUT  EMPTY >
<!ATTLIST AGENT_PROP.INPUT  name         NMTOKEN #REQUIRED
                            unit         NMTOKEN #REQUIRED
                            meaning      CDATA   #REQUIRED
                            range        CDATA   #IMPLIED
                            method       CDATA   #IMPLIED
                            outdomain    CDATA   #IMPLIED
>

<!--==================== MATH_AGENTSPACE SPACE DECLARATION ==================-->

<!ELEMENT SPACE_TYPE  EMPTY >
<!ATTLIST SPACE_TYPE  type (grid1d|grid2d|grid3d) "grid2d">

<!ELEMENT SPACE_RANGE  EMPTY >
<!ATTLIST SPACE_RANGE  name CDATA #FIXED "AGENTSPACE" >

<!--==================== SPEC_AGENTSPACE SPACE PROPERTIES  ==================-->
<!ELEMENT PROP.SPACE_TYPE  EMPTY >
<!ATTLIST PROP.SPACE_TYPE  boundary_type (bounded|periodic) #REQUIRED "periodic"
                           ghostcells    NMTOKEN #IMPLIED
>
<!ELEMENT PROP.SPACE_RANGE  EMPTY >
<!ATTLIST PROP.SPACE_RANGE  name          CDATA #FIXED "AGENTSPACE"
                            values        CDATA #REQUIRED
                            unit          NMTOKEN #REQUIRED
                            resolution    NMTOKEN #REQUIRED
>

<!--========================================================================-->
<!--============== AGENT/AGENTRESOURCE METHODS AND FUNCTIONS ===============-->
<!--========================================================================-->

<!ELEMENT AGENT.METHOD  (((INT|FLOAT|ALPHA|AGENT_ID|AGENTLIST)*,
                         (AEQ|SST|AGENT.LOOP|AGENT.IF|
                          AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                          AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                          AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                          AGENT.CALL_RESOURCEVALUE|AGENT.REPRODUCE|AGENT.DIE|
                          AGENT.CREATE)+)
>
<!ATTLIST AGENT.METHOD    name   CDATA #REQUIRED
                          iface  (internal|response) #REQUIRED
                          inputs   CDATA #IMPLIED
                          returns  CDATA #IMPLIED
                          text   CDATA #IMPLIED
>

<!ELEMENT AGENTR.METHOD  ((INT|FLOAT|ALPHA|AGENT_ID|AGENTLIST)*,
                         (AEQ|SST|AGENTR.LOOP|AGENTR.IF|
                          AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                          AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                          AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                          AGENT.CALL_RESOURCEVALUE)+)
>
<!ATTLIST AGENTR.METHOD   name     CDATA  #REQUIRED
                          iface    (internal|response) #REQUIRED
                          inputs   CDATA  #IMPLIED
                          returns  CDATA  #IMPLIED
                          text     CDATA  #IMPLIED
>

<!ELEMENT AGENT.FUNCTION  ((INT|FLOAT|ALPHA|AGENT_ID|AGENTLIST)+,
                           (AEQ|SST|AGENT.IF|AGENT.LOOP|
                            AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                            AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                            AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                            AGENT.CALL_RESOURCEVALUE|AGENT.REPRODUCE|AGENT.DIE|
                            AGENT.CREATE)+)
>
<!ATTLIST AGENT.FUNCTION  name    NMTOKEN #REQUIRED
                          inputs  CDATA #IMPLIED
```

```
                        type    (FLOAT|INT|ALPHA|AGENT_ID|AGENTLIST) "FLOAT"
                        title   CDATA #IMPLIED
>
<!ELEMENT AGENTR.FUNCTION  ((INT|FLOAT|ALPHA|AGENT_ID|AGENTLIST)+,
                       (AEQ|SST|AGENTR.IF|AGENTR.LOOP|
                        AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                        AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                        AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                        AGENT.CALL_RESOURCEVALUE)+)
>
<!ATTLIST AGENTR.FUNCTION  name    NMTOKEN #REQUIRED
                        inputs   CDATA #IMPLIED
                        type    (FLOAT|INT|ALPHA|AGENT_ID|AGENTLIST) "FLOAT"
                        title   CDATA #IMPLIED
>

<!--=========================================================================-->
<!--================ AGENT/AGENTRESOURCE SEQUENTIAL_PROCESSES children ======-->
<!--=========================================================================-->

<!ELEMENT AGENT.ONCREATION  ((BOUND|ONSTART_BOUND|TEST_BOUND|AEQ|AGENTINI.IF|AGENTINI.LOOP|
                        AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                        AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                        AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                        AGENT.CALL_RESOURCEVALUE|AGENT.CREATE|NOTICE)+)
>
<!ELEMENT AGENTR.ONSTART ((BOUND|AEQ|AGENTINI.IF|AGENTINI.LOOP|
                        AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                        AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                        AGENT.CALL_RESOURCEVALUE|AGENT.BROADCAST_AGENTLIST|
                        AGENT.BROADCAST_RESOURCE|NOTICE)+)
>

<!ELEMENT AGENT.PROCESS_BLOCK  ((AGENT.IF|AGENT.LOOP|
                            AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                            AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                            AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                            AGENT.CALL_RESOURCEVALUE|AGENT.REPRODUCE|
                            AGENT.DIE|AGENT.CREATE|AEQ|SST)+) >
<!ATTLIST AGENT.PROCESS_BLOCK  id         NMTOKEN #REQUIRED
                            condition  CDATA   #IMPLIED
                            title      CDATA   #IMPLIED >
<!ELEMENT AGENTR.PROCESS_BLOCK ((AGENT.IF|AGENT.LOOP|
                           AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                           AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                           AGENT.CALL_RESOURCEVALUE|AGENT.BROADCAST_AGENTLIST|
                           AGENT.BROADCAST_RESOURCE|AEQ|SST)+) >
<!ATTLIST AGENTR.PROCESS_BLOCK  id         NMTOKEN #REQUIRED
                            condition  CDATA   #IMPLIED
                            title      CDATA   #IMPLIED >
<!ELEMENT AGENTINI.IF ((AGENTINI.IF|AGENTINI.LOOP|AEQ|BOUND|
                      AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                      AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                      AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                      AGENT.CALL_RESOURCEVALUE)+,
                     (ELSEIF, (AGENTINI.IF|AGENTINI.LOOP|AEQ|BOUND|
                      AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                      AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                      AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                      AGENT.CALL_RESOURCEVALUE)+)?,
                     (ELSE,  (AGENTINI.IF|AGENTINI.LOOP|AEQ|BOUND|
                      AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                      AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                      AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                      AGENT.CALL_RESOURCEVALUE)+)?)
>
<!ATTLIST AGENTINI.IF  condition CDATA #REQUIRED
>

<!ELEMENT AGENT.IF  ((AGENT.IF|AGENT.LOOP|AEQ|SST|
                      AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                      AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                      AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                      AGENT.CALL_RESOURCEVALUE|AGENT.REPRODUCE|AGENT.DIE|
                      AGENT.CREATE|ODE|(PDE,UBOUND,LBOUND))+,
```

```
                    (ELSEIF, (AGENT.IF|AGENT.LOOP|AEQ|SST|
                     AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                     AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                     AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                     AGENT.CALL_RESOURCEVALUE|
                     AGENT.REPRODUCE|AGENT.DIE|AGENT.CREATE|
                     ODE|(PDE,UBOUND,LBOUND))+)?,
                    (ELSE,(AGENT.IF|AGENT.LOOP|AEQ|SST|
                     AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                     AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                     AGENT.REPRODUCE|AGENT.DIE|AGENT.CREATE|
                     ODE|(PDE,UBOUND,LBOUND))?))>
<!ATTLIST AGENT.IF  condition CDATA #REQUIRED >

<!ELEMENT AGENTR.IF  ((AGENTR.IF|AGENTR.LOOP|AEQ|SST|
                       AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                       AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                       AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                       AGENT.CALL_RESOURCEVALUE|ODE|(PDE,UBOUND,LBOUND))+,
                      (ELSEIF, (AGENTR.IF|AGENTR.LOOP|AEQ|SST|
                       AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                       AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                       AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                       AGENT.CALL_RESOURCEVALUE|ODE|(PDE,UBOUND,LBOUND))+)?,
                      (ELSE,(AGENTR.IF|AGENTR.LOOP|AEQ|SST|
                       AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                       AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                       AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                       AGENT.CALL_RESOURCEVALUE|
                       ODE|(PDE,UBOUND,LBOUND))?))>
<!ATTLIST AGENTR.IF  condition CDATA #REQUIRED >

<!ELEMENT AGENTINI.LOOP  ((AGENTINI.IF|AGENTINI.LOOP|AEQ|BOUND
                           AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                           AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                           AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                           AGENT.CALL_RESOURCEVALUE)+)

>
<!ATTLIST AGENTINI.LOOP  index CDATA #REQUIRED
                    inc   CDATA #REQUIRED
                    set   CDATA #REQUIRED
>

<!ELEMENT AGENT.LOOP ((AGENT.IF|AGENT.LOOP|AEQ|SST|
                       AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                       AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                       AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                       AGENT.CALL_RESOURCEVALUE|AGENT.REPRODUCE|AGENT.DIE|
                       AGENT.CREATE|ODE|(PDE,UBOUND,LBOUND))+) >
<!ATTLIST AGENT.LOOP   index CDATA  #REQUIRED
                       inc   (INC|SET) "INC"
                       set   CDATA  #REQUIRED
>
<!ELEMENT AGENTR.LOOP ((AGENT.IF|AGENT.LOOP|AEQ|SST|
                        AGENT.CALL|AGENT.CALL_AGENT|AGENT.CALL_RESOURCE|
                        AGENT.BROADCAST_AGENTLIST|AGENT.BROADCAST_RESOURCE|
                        AGENT.CALL_AGENTVALUE|AGENT.CALL_AGENTVALUE_SUM|
                        AGENT.CALL_RESOURCEVALUE|ODE|(PDE,UBOUND,LBOUND))+) >
<!ATTLIST AGENTR.LOOP   index CDATA  #REQUIRED
                        inc   (INC|SET)     "INC"
                        set   CDATA  #REQUIRED
>

<!--====================== AGENT/AGENTRESOURCE METHOD CALLS ==================-->

<!ELEMENT AGENT.CALL  EMPTY >
<!ATTLIST AGENT.CALL  method   NMTOKEN #REQUIRED
                      inputs   CDATA #IMPLIED
                      returns  CDATA #IMPLIED
                      text     CDATA #IMPLIED
>
<!ELEMENT AGENT.CALL_AGENT  EMPTY >
<!ATTLIST AGENT.CALL_AGENT  agent_id    NMTOKEN #REQUIRED
                            method      NMTOKEN #REQUIRED
```

```
                          inputs      CDATA #IMPLIED
                          returns     CDATA #IMPLIED
                          text        CDATA #IMPLIED
>
<!ELEMENT AGENT.CALL_RESOURCE  EMPTY >
<!ATTLIST AGENT.CALL_RESOURCE  resource_name NMTOKEN #REQUIRED
                               coord_list    CDATA   #REQUIRED
                               method        NMTOKEN #REQUIRED
                               inputs        CDATA   #IMPLIED
                               returns       CDATA   #IMPLIED
                               text          CDATA   #IMPLIED
>
<!ELEMENT AGENT.BROADCAST_AGENTLIST  EMPTY >
<!ATTLIST AGENT.BROADCAST_AGENTLIST  agentlist   NMTOKEN #REQUIRED
                                     method      NMTOKEN #REQUIRED
                                     inputs      CDATA #IMPLIED
                                     text        CDATA #IMPLIED
>
<!ELEMENT AGENT.BROADCAST_RESOURCE  EMPTY >
<!ATTLIST AGENT.BROADCAST_RESOURCE  resource_name NMTOKEN #REQUIRED
                                    method        NMTOKEN #REQUIRED
                                    inputs        CDATA   #IMPLIED
                                    text          CDATA   #IMPLIED
>
<!ELEMENT AGENT.CALL_AGENTVALUE  EMPTY >
<!ATTLIST AGENT.CALL_AGENTVALUE  agent_id    NMTOKEN #REQUIRED
                                 variable    NMTOKEN #REQUIRED
                                 return      CDATA #IMPLIED
                                 text        CDATA #IMPLIED
>
<!ELEMENT AGENT.CALL_AGENTVALUE_SUM  EMPTY >
<!ATTLIST AGENT.CALL_AGENTVALUE_SUM  agentlist   NMTOKEN #REQUIRED
                                     variable    NMTOKEN #REQUIRED
                                     return      CDATA #IMPLIED
                                     text        CDATA #IMPLIED
>
<!ELEMENT AGENT.CALL_RESOURCEVALUE  EMPTY >
<!ATTLIST AGENT.CALL_RESOURCEVALUE  resource_name NMTOKEN #REQUIRED
                                    coord_list    CDATA   #REQUIRED
                                    variable      CDATA   #IMPLIED
                                    text          CDATA   #IMPLIED
>

<!------------------------ AGENT CREATION PROCEDURES ----------------------->

<!ELEMENT AGENT.REPRODUCE  EMPTY >
<!ATTLIST AGENT.REPRODUCE  transfer_values    CDATA  #REQUIRED
                           oncreation_inputs  CDATA  #REQUIRED
>
<!ELEMENT AGENT.DIE  EMPTY >
<!ATTLIST AGENT.DIE  new_class          NMTOKEN #IMPLIED
                     transfer_values    CDATA #IMPLIED
                     oncreation_inputs  CDATA #IMPLIED
>
<!ELEMENT AGENT.CREATE  EMPTY >
<!ATTLIST AGENT.CREATE  class              NMTOKEN #REQUIRED
                        coordinates        CDATA   #REQUIRED
                        transfer_values    CDATA   #REQUIRED
                        oncreation_inputs  CDATA   #REQUIRED
>

<!--== set inital values of a oncreation_input variable on simulation start ==-->

<!ELEMENT ONSTART_BOUND  (RC*,R) >
<!ATTLIST ONSTART_BOUND  d0  CDATA  #REQUIRED
                         rc  CDATA  #IMPLIED
>
```

Literaturverzeichnis

Benz, J., R. Hoch und T. Legovic; ECOBAS – modelling and documentation; *Ecological Modelling*; 138:3–15; 2001.

Bousquet, F. und C. Le Page; Multi-agent simulations and ecosystem management: a review; *Ecological Modelling*; 176:313–332; 2004.

Collier, N.; RePast: An Extensible Framework for Agent Simulation; Social Science Research Computing, University of Chicago; 2002.

Coveney, P. und R. Highfield; *The Arrow of Time*; W. H. Allen, London; 1990.

DeAngelis, D. L. und L. J. Gross; *Individual-Based Models and Approaches in Ecology*; Chapman and Hall; 1992.

Epstein, J. und R. Axtell; *Growing Artificial Societies. Social Science from the Bottom Up*; Brookings Institution Press, Washington D. C.; 1996.

Ferber, J; *Multi-Agent Systems – An Introduction to Distributed Artificial Intelligence*; Addison-Wesley; 1999.

Fielding, D. J.; Intraspecific competition and spatial heterogeneity alter life history traits in an individual-based model of grasshoppers; *Ecological Modelling*; 175:169–187; 2004.

Ginot, V.; A multi-agents architecture to enhance end-user individual modelling; *Ecological Modelling*; 157:23–41; 2002.

Grimm, V.; Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the future?; *Ecological Modelling*; 115:129–148; 1999.

Grimm, V.; Visual Debugging: A Way of Analyzing, Understanding and Communicating Bottom-Up Simulation Models in Ecology; *Natural Resource Modeling*; 15:23–38; 2002.

Grimm, V. und F. Railsback; *Individual-based Modeling and Ecology*; Princeton University Press, New Jersey, USA; 2005.

Hoch, R., T. Gabele und J. Benz; Towards a standard for documentation of mathematical models in ecology; *Ecological Modelling*; 113:3–12; 1998.

Huston, M., D. L. DeAngelis und W. Post; New computer models unify ecological thoery; *Bioscience*; 38:682–691; 1988.

Janssen, M. A. (Herausgeber); *Complexity and Ecosystem Management - The Theory and Practice of Multi-Agent Systems*; Edited papers from the meeting of the International Society

for Ecological Economics; Edward Elgar Publishing, Northampton; 2002.

KLOPFER, E.; Technologies to support the creation of complex systems models – using StarLogo software with students; *BioSystems*; 71:111–122; 2003.

KLÜGL, F., C. OECHSLEIN, F. PUPPE und A. DORNHAUS; Multi-Agent Modelling in Comparison to Standard Modelling; in: BARROS, F. J. und N. GIAMBIASI (Herausgeber), *AIS'2002 (Artificial Intelligence, Simulation and Planning in High Autonomy Systems)*; S. 105–110; SCS Publishing House; 2002.

ŁOMNICKI, A.; Individual-based models and the individual-based approach to population ecology; *Ecological Modelling*; 115:191–198; 1999.

LOREK, H. und M. SONNENSCHEIN; Modeling and simulation software to support individual-based ecological modelling; *Ecological Modelling*; 115:199–216; 1999.

LUHMANN, N.; *Ökologische Kommunikation: kann d. moderne Gesellschaft sich auf ökolog. Gefährdungen einstellen ?*; Opladen: Westdt. Verlag; 3. Auflage; 1990.

MANSON, S. M.; Validation and verification of multi-agent systems; in: JANSSEN, M. A. (Herausgeber), *Complexity and Ecosystem Management - The Theory and Practice of Multi-Agent Systems*; S. 63–74; Edward Elgar Publishing; 2002.

MINAR, N., R. BURKHART, C. LANGTON und M. ASKENAZI; The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations; Working Paper 96-06-042, Santa Fe Institute, Santa Fe, NM; 1996.

OECHSLEIN, CHR.; *Vorgehensmodell mit integrierter Spezifikations- und Implementierungssprache für Multiagentensimulationen*; Dissertation; Julius-Maximilians-Universität Würzburg; Shaker Verlag Aachen; 2004.

OECHSLEIN, CHR., F. KLÜGL, R. HERRLER und F. PUPPE; UML for Behaviour-Oriented Multi-Agent Simulations; in: DUNIN-KEPLICZ, B. und E. NAWARECKI (Herausgeber), *From Theorie to Practice in Multi-Agent Systems, Proceedings of the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems (CEEMAS 2001)*; S. 217 ff.; Springer Verlag; 2001.

RAILSBACK, S. F.; Concepts from complex adaptive systems as a framework for individual-based modelling; *Ecological Modelling*; 139:47–62; 2001.

RESNICK, M.; *Turtles, Termites and Traffic Jams: explorations in massively parallel microworlds*; Complex Adaptive Systems; MIT Press, Cambridge, MA; 1995.

ROPELLA, G. E. P., S. F. RAILSBACK und S. K. JACKSON; Software Engineering Considerations for Individual-Based Models; *Natural Resource Modeling*; 15:5–22; 2002.

ROSÀ, R., A. PUGLIESE, A. VILLANI und A. RIZZOLI; Individual-based vs. deterministic models for macroparasites: host cicles and extinction; *Theoretical Population Biology*;

63:295–307; 2003.

Solé, R. V., J. G. Gamarra, M. Ginovart und D. López; Controlling Chaos in Ecology: From Deterministic to Individual-based Models; *Bulletin of mathematical Biology*; 61:1187–1207; 1999.

Topping, Chr. J., T. S. Hansen, Th. S. Jensen, J. U. Jepsen, F. Nikolajsen und P. Odderskær; ALMaSS, an agent-based model for animals in temperate European landscapes; *Ecological Modelling*; 167:65–82; 2003.