Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Interfakultären Fachbereich für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

Modellierung vergleichbar machen anhand der Replikation eines hybriden agentenbasierten Ansatzes

vorgelegt von

MSc Katharina Rybnicek
u105066, UNIGIS MSC Jahrgang 2020

Betreuer/in:
Dr. Gudrun Wallentin

Zur Erlangung des Grades
„Master of Science – MSc”

St. Pölten, 28.8.2023

Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die vorliegende Arbeit wurde bisher in gleicher oder ähnlicher Form noch nicht als Masterarbeit eingereicht.

_____

Datum, Unterschrift

Danksagung

An dieser Stelle möchte ich mich bei denjenigen bedanken, die mich bei der Fertigstellung der Masterarbeit unterstützt haben. Frau Dr. Gudrun Wallentin danke ich für die Betreuung des Themas und für die Unterstützung bei etwaig aufkommenden Fragen.

Abstract

This work is dedicated to the topic of replicating a spatial, hybrid AB-SD model to test whether replication of the hybrid approach is possible. For this purpose, a hybrid, dynamic fish-plankton model was selected. The fish-plankton model is based on an alternation between individual agents and schools of fish in combination with spatially discretised plankton stocks controlled by cellular automata. Another structural element existing alongside the agents are superindividuals. They are used to measure the performance of the computing capacity in terms of simulation processing time.

The simulation results presented show emergence-based aggregation of individual agents into collectives and dynamic switching as an adaptive response of the model structure to the emergence of schools of fish. The replication results presented show a similar trend in model behaviour, but quantitative differences at the end of the 100-year simulation run. Replication of a hybrid fish-plankton model is possible. The results provide information about the influence of the software. Since not much is known to date about multiparadigmatic software development and its impact on results, future research should address this to provide a standardised framework that overcomes the traditional shortcomings in the replication process and results.

The initial model has been created in the NetLogo simulation platform version 5.3.1 and the replication in the GAMA simulation environment version 1.8.2. The programming language of the initial model is Logo and that of the replicated model is GAMAL. In each case, it is a Java-based programming language.

Zusammenfassung

Diese Arbeit widmet sich dem Thema der Replikation eines räumlichen, hybriden AB-SD-Modells, um zu testen, ob die Replikation des hybriden Ansatzes möglich ist. Hierzu wurde ein hybrides, dynamisches Fisch-Plankton-Modell ausgewählt. Das Fisch-Plankton-Modell basiert auf einem Wechsel zwischen einzelnen Agenten und Fischschwärme in Kombination mit räumlich diskretisierte Planktonbeständen, die durch zelluläre Automaten gesteuert werden. Ein weiteres neben den Agenten existierendes Strukturelement sind Superindividuen. Anhand ihnen wird die Leistung der Rechenkapazität in Bezug auf die Simulationsverarbeitungszeit gemessen.

Die vorgestellten Simulationsergebnisse zeigen die emergenzbasierte Aggregation einzelner Agenten zu Kollektiven und die dynamische Umschaltung als adaptive Reaktion der Modellstruktur auf das Auftauchen von Fischschwärmen. Die vorgestellten Replikationsergebnisse zeigen einen ähnlichen Trend im Modellverhalten, jedoch quantitative Unterschiede am Ende der 100-Jahre-Simulationslaufzeit. Die Replikation eines hybriden Fisch-Plankton-Modells ist möglich. Die Ergebnisse geben Aufschluss über den Einfluss der Software. Nachdem bis dato nicht viel über die multiparadigmatische Softwareentwicklung und ihre Auswirkungen auf die Ergebnisse bekannt ist, sollten sich zukünftige Forschungsarbeiten damit auseinandersetzen, um einen standardisierten Rahmen zu schaffen, der die traditionellen Unzulänglichkeiten im Replikationsprozess und in den Ergebnissen überwindet.

Das Ausgangsmodell ist in der Simulationsplattform NetLogo in der Version 5.3.1 und die Replikation in der Simulationsumgebung GAMA in der Version 1.8.2. erstellt worden. Die Programmiersprache des Ausgangsmodells ist Logo und die des replizierten Modells GAMAL. Es handelt sich jeweils um eine auf Java basierte Programmiersprache.

Inhaltsverzeichnis

Eidesstattliche Erklärung 2

Danksagung 3

Abstract 4

Zusammenfassung 5

Inhaltsverzeichnis 6

Abbildungsverzeichnis 8

Tabellenverzeichnis 8

1. Einleitung 9

1.1 Stand der Forschung 11

1.2 Modellierungsumgebungen 13

1.2.1 NetLogo 13

1.2.2 GAMA 13

1.2.3 Vergleich NetLogo und GAMA 13

1.3 Gewählter Ansatz 15

1.4 Forschungsfrage 17

2. Methodik 18

2.1 Modellkonzept 18

2.1.1 Ziel des Ausgangsmodells 19

2.1.2 Einheiten, Variablen, Maßstab 20

2.1.3 Modellablauf 21

2.1.4 Emergenz, Interaktionen, Stochastik und Beobachtung 24

2.2 Modellparametrisierung 25

3. Replikation des Modells 25

3.1 Replikationsumsetzung 26

3.1.1 Definition des Simulationszeitraums 26

3.1.2 Definition des Modellablaufs 27

3.1.3 Erstellen von Agenten 28

3.1.4 Bewegungsverhalten 29

3.1.5 Fressverhalten 32

3.1.6 Reproduktion und Sterblichkeit 33

3.1.7 Schulen 34

3.1.8 Plankton 34

3.2 Modellvergleich 35

3.2.1 Monte Carlo Simulation 36

3.2.2 Konfidenzintervall (KI) 37

4. Ergebnisse 38

5. Diskussion 61

6. Fazit 76

7. Literaturverzeichnis 81

8. Anhang 85

8.1 Code Ausgangsmodell (NetLogo) 85

8.2 Code repliziertes Modell (GAMA) 98

8.3 Flocking-Modell 115

Abbildungsverzeichnis

Abbildung 1: UML-Abbildung des Modellablaufs 23
Abbildung 2: In GAMA definierte Spezies inklusive Attribute, Reflexe und Actions 29
Abbildung 3: Start- und Endsituation nach 100 Simulationsjahren 39
Abbildung 4: Ergebnis der Realisierung des Schul-Agenten anhand des multi-level-Architekturansatzes 68

Tabellenverzeichnis

Tabelle 1: Metadaten zum Ausgangsmodell 10
Tabelle 2: Dimensionen Ausgangsmodell und repliziertes Modell 15
Tabelle 3: Definierte Variablen der Entitäten Fisch, Schule und Plankton in NetLogo 20
Tabelle 4: GAMA-Optimierungsmethoden zur Förderung der Rechenleistung 31
Tabelle 5: Ergebnisse der Dichtefunktion 41
Tabelle 6: Ergebnisse Welch-Test 43
Tabelle 7: Ergebnisse der Dichtefunktion weiterer Prüfgrößen 44
Tabelle 8: KI-Ergebnisse 46
Tabelle 9: Ergebnisse Monte-Carlo-Simulation (N=100) 48
Tabelle 10: Vergleich der Umsetzung in GAMA mit jener in NetLogo 57
Tabelle 11: Performancevergleich GAMA und NetLogo 61
Tabelle 12: Aufgetretene Herausforderungen im Replikationsprozess 78

1. Einleitung

Computermodelle werden entwickelt, um Phänomene der realen Welt zu erklären, zu beschreiben oder vorherzusagen (Legendi and Gulyas, 2012). Auch fiktive Phänomene lassen sich damit abbilden. Um zu beschreiben, ob und inwiefern die Modelle die Phänomene abbilden und nicht Resultate zufälliger Ereignisse sind, wird die Methode der Replikation angewendet. Unter Replikation wird laut Graeme (2013) die wiederholte Durchführung ein und desselben Experiments verstanden. Anschließend werden die Ergebnisse auf Ähnlichkeit untersucht. Sind die Ergebnisse nach wiederholtem Mal den abzubildenden Phänomenen hinreichend ähnlich, so ist das Computermodell gültig und die Replikation erfolgreich (Himme, 2007, Legendi and Gulyas, 2012). In der Praxis treten eine Reihe von Herausforderungen während der Replikation auf, zum Beispiel ist in der Wissenschaftsgemeinde nicht klar definiert, ob und wie ein Modell repliziert werden soll. Zusätzlich existieren unterschiedliche Meinungen darüber, wie Bewertungen von Replikationen und ihrer Ergebnisse vorzunehmen sind (Legendi and Gulyas, 2012). Der Lösungsansatz wäre eine einheitliche textuelle Richtlinie, wie zum Beispiel das ODD-Protokoll, welches jedoch in der Wissenschaftscommunity (noch) nicht allgemein akzeptiert und nachhaltig umgesetzt wird. Die Replikation als wissenschaftliche Methode ist somit laut Legendi und Gulyas (2012) bis zu ihrer allgegenwärtigen Akzeptanz und einheitlichen Anwendung als problematisch anzusehen und zum Teil in ihrer Umsetzung unbestimmt. Die Probleme und Herausforderungen, die während der Arbeit aufgetreten sind, werden in Kapitel 5 näher diskutiert.

Das generelle Ziel der Masterarbeit ist eine Neuimplementierung eines bestehenden hybriden ABM-SD Modells anhand seiner Modelldokumentation (ODD-Protokoll) und des originalen Source Codes in einer anderen Softwareumgebung (GAMA) mit anschließender Überprüfung, ob und inwieweit die Modelle die gleichen Ergebnisse liefern. Die Metadaten zum Ausgangsmodell sowie zum replizierten Modell sind der Tabelle 1 zu entnehmen. Der Zweck der Masterarbeit, der sich aus dem Ziel ergibt, ist zu testen, ob die Implementierung eines konzeptionellen Modells, das von einer Gruppe von WissenschaftlerInnen zu einem früheren Zeitpunkt beschrieben und implementiert wurde, in einer anderen Umgebung mit einer anderen Programmiersprache funktioniert. Die Fragen, ob und inwieweit komplexe Modelle replizierbar sind oder sein können bzw. sein sollen und in welchem Rahmen die Wissenschaftscommunity die wissenschaftlichen Ansprüche auf Transparenz, Objektivität und Nachvollziehbarkeit auf das Modellieren von komplexen Modellen anwenden können, sind hierbei zu berücksichtigen. Neben den genannten Gütekriterien ist auch die Validität und Verifizierung der Modelle zu klären (Popper, 2005, Wilensky, 2007). In der Arbeit wird nur die Modellverifizierung berücksichtigt, indem die Ergebnisse des replizierten Modells dem Verhalten des Ausgangsmodells gegenübergestellt werden (Rand and Wilensky, 2006). Treten die Modellergebnisse nach

wiederholter Durchführung im selben Wertebereich auf, so sind die ursprünglichen Ergebnisse erfolgreich repliziert (Rand and Wilensky, 2006). Von einer Modellvalidierung wird abgesehen, da keine Gegenüberstellung zwischen einem realen System und dem zu replizierenden Modell vorgesehen ist, da das zu replizierende Modell ein fiktives und kein reales System abbildet. Laut Wilensky (2007) und Popper (2005) sind valide Modellaussagen Ergebnisse nicht zufälliger Ereignisse und bilden die Grundlage für neue Erkenntnisse und Theorien (Graeme, 2013, Zhang and Robinson, 2021, Donkin et al., 2017). Die Ergebnisse des replizierten Modells werden zur Be- oder Widerlegung der Forschungsfragen, die in Kapitel 1.4 definiert sind, herangezogen.

Tabelle 1: Metadaten zum Ausgangsmodell

Modellname:	Design 4: Agent (fish) – spatial stock (plankton) ⟷ school-agent (fish) – spatial stock (plankton)
EntwicklerInnen:	Gudrun Wallentin und Christian Neuwirth
Software-Version:	NetLogo 5.3.1
Code-Verfügbarkeit:	https://www.comses.net/codebases/5254/releases/1.2.0/
Plattform-Verfügbarkeit:	https://ccl.northwestern.edu/netlogo/download.shtml
Szenario:	In dem zugrundeliegenden Szenario ist die Entwicklung einer planktonfressenden Fischpopulation in Abhängigkeit zur jeweiligen Nahrungsverfügbarkeit über einen Simulationszeitraum von 100 Jahren abgebildet. Beim Entwerfen und Implementieren der Replikationsstrategie wurde darauf geachtet, dass dieselben Mechanismen, konkreten Klassen und Phänomene genutzt werden, die das Fischverhalten, die Planktonbildung und die organisatorische Schulbildung steuern.

Insgesamt erstreckt sich die Masterarbeit auf über 6 Kapitel, wobei im ersten Kapitel eine kurze Einleitung zur Arbeit sowie zum Stand der Forschung gegeben wird. In Kapitel 2 werden die Methoden, welche im Rahmen der Replikation eingesetzt wurden, beschrieben. Im Unterkapitel 2.1 geht es in erster Linie um das Verstehen des zugrundeliegenden Modellkonzepts. In Kapitel 2.1.1 bis 2.1.4 sind die wesentlichsten Inhalte des ODD-Protokolls zum Ausgangsmodell wiedergegeben. In Kapitel 3 wird der Replikationsprozess sowie dessen Herausforderungen erläutert. Anschließend werden in Kapitel 4 die aus der Replikation gewonnenen Ergebnisse dargestellt und beschrieben. Die daraus gewonnenen Erkenntnisse werden in Kapitel 5 diskutiert. Abschließend wird in Kapitel 6 das Fazit, welches aus der

gesamten Erstellung der Masterarbeit gezogen wird, erläutert. In Kapitel 8 ist der Code des Ausgangsmodells sowie des implementierten Zielmodells angegeben.

1.1 Stand der Forschung

Im folgenden Kapitel werden die nach umfassender Literaturrecherche zum derzeitigen Kenntnisstand agentenbasierter Modelle gewonnenen Informationen zusammenfassend wiedergegeben: Die Replikation als wissenschaftliches Instrument fördert das gemeinsame Verständnis von Modellierungsentscheidungen innerhalb der Forschungsgemeinschaft (Rand and Wilensky, 2006) und liefert aufgrund der wiederholten Erstellung von Forschungsergebnissen und der systemischen Analyse von Äquivalenzen (Axtell et al., 1996) Auskünfte über die Glaubwürdigkeit und Zuverlässigkeit von Resultaten (Easley et al., 2000, Popper, 2005, Seagren, 2015). Trotz allem existieren kaum wissenschaftliche Studien zu Modellreplikationen. Zhang (2021) illustrieren die Tatsache anhand von 348 wissenschaftlichen Artikeln, wo nur 2,6 Prozent ein teilweise oder vollständig repliziertes Modell aufweisen. Generell werden vereinzelt Code-Repositorien und Modellbeschreibungen zur Verfügung gestellt (Donkin et al., 2017, Axtell et al., 1996, Wilensky, 2007). Aufgrund der geringen Anzahl an Replikationsstudien fehlt bisher ein vergleichbares Wissen zu ‚wie Replikation umgesetzt werden kann' und ‚wie die Ergebnisse zu interpretieren sind' (Wilensky, 2007, Raab et al., 2022, Rand and Wilensky, 2006). Die Wissenslücke gefährdet nicht nur den Fortschritt von Replikationsstudien sondern auch den gesamten Replikationsprozess, da laut Popper (2005) und Latour (1979) nur eine Replikation anhand einer vollumfassenden Dokumentation zum Modell zulässig ist und das nur so lange, wie die WissenschaftlerInnen dazu aufgefordert sind ihre Modelle zu beschreiben und alle Einzelheiten zu ihren Experimenten zu dokumentieren. Zur Unterstützung der ReplikatorInnen vertritt Wilensky (2007) daher die Auffassung, dass folgende Mindestanforderungen an die Publikationen gestellt werden müssen, um eine Replikation sinnvoll durchführen zu können:

- Zurverfügungstellung einer gut spezifizierten und formulierten konzeptionellen Modellbeschreibung,
- Zurverfügungstellung eines Pseudo-Codes,
- Zurverfügungstellung eines vollständigen Quellcodes des Modells.

Die von Wilensky (2007) geforderten Punkte geben zwar keine Garantie, dass bei deren Erfüllung die Replikation erfolgreich durchgeführt werden kann, jedoch mindern oder vermeiden sie bei ihrer Berücksichtigung potenzielle Fehlerquellen wie zum Beispiel die Missinterpretation einer rein textuellen Beschreibung des Modells. Durch den Quellcode ist die Modellreplikatorin oder der Modellreplikator im Stande, bei Missverständnissen und Fragen nach fehlenden Details zu suchen. Während oder nach der

Replikation kann der Code mit dem originalen Source Code verglichen werden (Wilensky, 2007). In der Wissenschaftsliteratur wird empfohlen, den Einsatzzeitpunkt des Quellcodes im Replikationsprozess bewusst zu wählen, da bei frühzeitigem Heranziehen die Gefahr besteht, dass der Modellreplikator, die Modellreplikatorin die spezifischen Methoden und Funktionen der jeweiligen Modellierungsplattform nicht mehr frei wählt, sondern sich zu sehr an dem Quellcode orientiert und dadurch eine Art Kopie des Ausgangsmodells anfertigt (Wilensky, 2007). Zusätzlich können sich ReplikatorInnen auch dazu hinreißen lassen, das replizierte Modell so lange anzupassen, bis die Ergebnisse mit jenen des Ausgangsmodells übereinstimmen. Die Anfertigung einer Modellkopie sowie die Überanpassung eines replizierten Modells lässt die Replikation überflüssig werden.

Neben der kaum existierenden und teilweise lückenhaft validierten Literatur ist auch die Wahl einer geeigneten Replikationsplattform nicht einheitlich geregelt. In einer idealisierten Welt würde jede Forscherin, jeder Forscher die gleiche Plattform und somit dieselbe Programmiersprache verwenden. Das ist eine Idealisierung falscher Praxis, da jede Wissenschaftlerin, jeder Wissenschaftler jene Methoden verwenden, die entweder zu ihren individuellen Stärken passen oder die dazu notwendigen Ressourcen zur Verfügung stehen (Donkin et al., 2017). Ein weiterer relevanter in diesem Zusammenhang stehender Aspekt ist, dass mit wachsendem technischen Fortschritt der Wechsel zwischen verschiedenen Simulationsplattformen und Methoden begünstigt und die Erstellung neuer Plattformen gefördert wird (Bajracharya and Duboz, 2013, Railsback et al., 2006). Somit wird die sprach- und plattformübergreifende Replikation zu einer immer präsenteren Methode, die die Wirkung von realistischeren Modellen unterstützt (Wallentin and Neuwirth, 2016).

Die wenigen Studien die zur sprach- und plattformübergreifenden Replikation gefunden wurden, haben das Fazit, dass die Umsetzung einer vollständigen Replikation eine Herausforderung darstellt (Donkin et al., 2017, Wilensky, 2007). Laut Donkin (2017) ist die Ergebnisvariation ein Resultat aus der Übersetzung eines theoretischen Modells in eine bestimmte Programmiersprache und deren Code-Komplexität. Sie ist auch der Auffassung, dass das Ziehen von Rückschlüssen aus den Abweichungen der Simulationsergebnisse auf die Codierungsentscheidungen eine fast unlösbare Aufgabe ist, da die einzelnen Auslöser aufgrund der unterschiedlichen Syntax, Systemabläufe, etc. nicht präzise genug spezifiziert werden können (Zhang and Robinson, 2021, Donkin et al., 2017). Bajracharya (2013) konnte des Weiteren nachweisen, dass das Design und die Implementierung des Ausgangsmodelsl von Natur aus verzerrt sein können, wodurch die ursprünglichen Ergebnisse durch den Einsatz einer anderen Simulationsplattform entkräftet werden. Ein weiteres Problem stellt die Replikation von der ursprünglichen Entwicklerin, von dem Entwickler dar (Donkin et al., 2017), da damit das Gütekriterium

der Objektivität nicht gewährleistet ist. Bajracharya (2013) beschreibt den Prozess der plattformübergreifenden Replikation allgemein als eine langwierige und komplexe Aufgabe mit abweichenden Ergebnissen. Wilensky (2007) stellt in seinem Artikel fest, dass die Kommunikation zwischen den HerstellerInnen des Ausgangsmodells und den ReplikationsentwicklerInnen gegen die genannten Risiken eingesetzt werden kann. Denn während seiner Studie fand ein reger Austausch mit dem Hersteller des Ausgangsmodells statt, wodurch die abweichenden Ergebnisse großteiles korrigiert und die Qualität des Reviewprozesses sowie der Replikation erhöht wurden.

Alle bisher genannten Herausforderungen unterstreichen die Notwendigkeit einer vollumfassenden Modelldokumentation, insbesondere für die sprach- und plattformübergreifende Replikation, da für die Umsetzung in eine andere spezifische Syntax eine Dokumentation bzw. Beschreibung der Modellstruktur und des -verhaltens essenziell ist.

1.2 Modellierungsumgebungen

1.2.1 NetLogo

Das Ausgangsmodell ist in NetLogo in der Version 5.3.1 erstellt. Bei NetLogo handelt es sich um eine Multi-Agenten-Programmiersprache mit integrierter Modellierungsumgebung (IDE), deren Sprache als ‚Logo' bezeichnet wird (https://ccl.northwestern.edu/netlogo/faq.html, 19.08.2022). Sie wurde 1999 von Uri Wilensky entwickelt und ist plattformunabhängig (https://ccl.northwestern.edu/netlogo/faq.html, 19.08.2022). NetLogo zeichnet sich durch seine Benutzerfreundlichkeit und sehr gute Dokumentation in Form von Modell-Bibliotheken die frei zur Verfügung stehen, aus. Systemtechnisch läuft NetLogo in einer virtuellen Java-Laufzeitumgebung auf einem lokalen Rechner (https://ccl.northwestern.edu/netlogo/faq.html, 19.08.2022).

1.2.2 GAMA

GAMA (GIS-Agent-based Modeling Architecture) ist eine Simulationssoftware mit einer integrierten Entwicklungsumgebung (IDE) zur Erstellung von räumlich expliziten agentenbasierten Simulationen (https://gama-platform.org, 25.7.2022). Systemtechnisch läuft GAMA wie NetLogo in einer virtuellen Java-Laufzeitumgebung auf einem lokalen Rechner.

1.2.3 Vergleich NetLogo und GAMA

GAMA ist wie NetLogo eine Modellierungssoftware. Die Stärke von GAMA liegt in der Implementierung von GIS-Daten und die von NetLogo in seiner leichten Erlernbarkeit als intuitive Programmiersprache (Railsback et al., 2006). NetLogo und GAMA sind Open Source und verfügen über frei zugängliche Software-Kits, Dokumentationen, Bibliotheken und Github-Repositories.

Die (Erklärungs-)Modelle, die in der GAMA-Bibliothek zur Verfügung stehen, wurzeln in einer objektorientierten Sprache. Die Objektorientierung ist ein Konzept der Softwaretechnik mit dem Ziel der Vereinfachung von komplexen Softwaresystemen (Lahres, 2009). Die Basis von objektorientierten Programmiersprachen bilden Objektklassen, die miteinander interagieren und deren Merkmale sich auf die Datenstruktur und das Verhalten beziehen (Eirund, 2013). Der Objektwert bestimmt den jeweiligen Zustand des Objekts (Instanzen) (Eirund, 2013). Objekte können sich aus mehreren Objekten zusammensetzen, sogenannten Sub-Objekten, deren Reaktion durch ausführbare Methoden bestimmt wird (Eirund, 2013). Das Konzept der objektorientierten Programmierung bedingt auch die Vererbung, das bedeutet, dass Klassen, die auf der Grundlage anderer Klassen existieren, deren Attribute und Methoden automatisch erhalten (Boles and Boles, 2004). Ein großer Vorteil der objektorientierten Programmierung liegt darin, dass sich der Sourcecode beliebig wiederverwenden und je nach Situation erweitern lässt. NetLogo folgt einem nicht objektorientierten Ansatz. Tabelle 2 zeigt die Dimensionen Erstellungsjahr, Hardware, Sprache, Toolkit, Algorithmen und Autoren des Ausgangsmodells und des replizierten Modells.

Ähnlich zu Raab (2022) können im direkten Vergleich zwischen NetLogo und GAMA folgende Erkenntnisse festgestellt werden: NetLogo und GAMA können unter Windows, Linux und McOS verwendet werden. Für die Masterarbeit wurden beide Plattformen unter Windows 10 installiert und es traten während der Installation keine Probleme auf. Die IDE von NetLogo wirkt etwas veraltet und weist nur eine begrenzte Anzahl an sichtbaren Funktionen (Schaltflächen) auf. Als Einschränkungen sind folgende Beispiele identifiziert worden: Der gesamte Code muss in eine Datei geschrieben werden, es gilt eine geringere Clean-Code Disziplin als in Java, es fehlt ein Debugger, etc. Die Oberfläche von GAMA hingegen ist moderner und bietet eine bessere Übersicht der zur Verfügung stehenden Werkzeuge und Bibliotheken. Der Editor in GAMA bietet die Möglichkeit einer automatischen Code-Farbgebung, sodass sich die Klassen und Methoden visuell besser abheben und sich sogar automatisch formatieren lassen. Des Weiteren bietet der Editor auch die Möglichkeit einer automatischen Kompilierung an. Auch das Debugging erfolgt automatisch, indem syntaktische und semantische Fehler inklusive einer Warnmeldung ausgewiesen werden. In NetLogo hingegen werden weder Klassen noch Methoden farblich ausgewiesen. Der Debugging Prozess wird auch erst durch das aktive Anklicken der Schaltfläche ‚Check' angestoßen und bietet generell eine begrenzte Anzahl an Debugging-Funktionen an. Auch die Warnhinweise sind in NetLogo weniger verständlich formuliert als in GAMA. Beide Sprachen verfügen über eine Graphen-Funktionalität, die das Erstellen und die Darstellung von Modellen unterstützen. Wenn es um den Aufbau von komplexen Modellen geht, wird für GAMA ein modularer Ansatz

empfohlen sowie die Nutzung der Simulationsausführungen auf mehreren Threads, die die Last der Bearbeitung einer größeren Datenmenge effizienter verteilen (Raab et al., 2022).

Tabelle 2: Dimensionen Ausgangsmodell und repliziertes Modell

Dimension	NetLogo (Ausgangsmodell)	GAMA (Repliziertes Modell)
Erstellungsjahr	2016	2022
Verwendete Hardware	Standard-Desktop-Computer (CPU=2.83 GHz und 8 GB RAM)	Standard-Desktop-Computer (CPU = 3.20 GHz und 8 GB RAM)
Sprache	Logo	GAMAL
Toolkit	NetLogo Programmbibliothek	GAMA-Programmbibliothek
Algorithmen	Nicht-Objektorientiert	Objektorientiert
Autoren	Gudrun Wallentin und Christian Neuwirth	Katharina Rybnicek

1.3 Gewählter Ansatz

In dem vorliegenden Kapitel geht es um die Auswahl des Modellierungsparadigmas, die normalerweise anhand der Forschungsfrage bzw. dem Forschungsziel stattfindet (Martin and Schlüter, 2015). Dieser Schritt fällt für das zu replizierende Modell weg, da bereits die AutorInnen die entsprechende Wahl für das Ausgangsmodell getroffen haben, weshalb dem zu replizierenden Modell derselbe hybride AB-SD des Ausgangsmodells zugrunde liegt. Laut Gray (2012) fallen folgende Modelltypen in diesen Modellierungsansatz hinein:

- individualbasierte Modelle, die mit einem systemdynamischen Modell interagieren oder
- systemdynamische Modelle, die in individualbasierte Modelle eingebettet sind oder
- individualbasierte Modelle, bei denen die Darstellung zwischen einer individualbasierten und einer gleichungsbasierten Form wechselt.

Dem hybriden AB-SD Ansatz unterliegt der von Gray definierte Modelltyp, wo ein individualbasiertes Modell (ABM) in einer systemdynamischen Umwelt (SD) (inter)agiert. Daraus entstehen bestimmte Komplexitäten, die sich zusätzlich aus dem dynamischen Wechsel der Darstellungen der Systemkomponenten zwischen einer individualbasierten und gleichungsbasierten Form ergeben. Der Wechsel (engl. Switch) wird durch das Über- oder Unterschreiten von Grenzwerten (engl. treshold) ausgelöst. Welche Systemkomponenten in ABM bzw. welche in SD abgebildet sind und welche Grenzwerte einen dynamischen Wechsel auslösen, ist im Konzept des Ausgangsmodells sowie in Kapitel 2.1 beschrieben.

Unter ABM (engl. Agent-Based Model) wird ein agentenbasierter bottom-up Modellierungsansatz und unter SD (engl. System Dynamics) ein systemdynamischer top-down Ansatz definiert (Wang et al., 2018, Vincenot et al., 2011, Swinerd, 2012). Sie zählen zu den gängigsten computergestützten Modellen zur Abbildung komplexer, dynamischer Systeme. Für die Analyse komplexer, ökologischer Systeme werden hybride Modelle eingesetzt, da sie mehrere Komplexitätsquellen zulassen (z.B. wechselseitige Interaktionen, Rückkopplungen, Verbindungen auf verschiedenen Aggregationsebenen, etc.) und die Stärken verschiedener Ansätze kombinieren (Martin and Schlüter, 2015, Wallentin and Neuwirth, 2016, Vincenot et al., 2011, Swinerd, 2012). Daraus entstehen neue konzeptionelle und technische Möglichkeiten. Neben den Möglichkeiten treten laut Railsback (2006) auch neue technische Herausforderungen auf, die neben der Softwareentwicklung selbst, auch die technischen Restriktionen in Form von hohen Rechenzeiten und hohem Speicherplatzbedarf in den Vordergrund stellen (Scheffer, 1995, Parry and Evans, 2008). Davon sind vor allem jene Simulationen betroffen, die eine hohe Anzahl an komplexen Individuen managen. Zur Lösung der Restriktionen sind Investitionen in leistungsfähige Rechner (Hardware-Investition) oder in Optimierungen des Softwaredesigns (z.B. Parallelisierung → Aufteilung der Verarbeitungs- und Datenlast) vorgesehen (Scheffer, 1995, Parry and Evans, 2008), wofür in der Praxis oftmals das Geld und/oder die Zeit fehlt (Parry and Evans, 2008). Ein kostengünstigerer Lösungsansatz ist jener der sogenannten Super-Individuen (Scheffer, 1995, Parry and Evans, 2008), bei dem eine zufällige Teilmenge von Individuen mit repräsentativen Merkmalen in ein Kollektiv zusammengefasst werden (Huston et al., 1988). Der Super-Individuen-Ansatz von Vincenot (2011) und Swinerd (2012) wurde in dem Ausgangsmodell um den emergenzbasierten Wechsel zwischen Individuen und Super-Individuen erweitert. Der Super-Individuen-Ansatz ist nicht nur ein nützliches Strukturelement zur Verbindung von hierarchischen Skalen (Wallentin and Neuwirth, 2016), sondern auch ein wichtiger Bestandteil rechenintensiver Simulationen, der sich positiv auf die verfügbare Rechenleistung und den Speicherplatz auswirkt (Parry and Evans, 2008). Bei der Umwandlung von Individuen zu Super-Individuen werden die Individuen anhand bestimmter Kriterien (z.B. Alter, Morphologie, etc.) zusammengefasst (Parry and Evans, 2008). Jedes Super-Individuum repräsentiert somit eine feste Anzahl an Individuen während des gesamten Simulationsverlaufs. Soll jedoch die individuelle Variabilität erhalten bleiben, ist der Ansatz der Super-Individuen ungeeignet, da diese zum Verlust an Variation, Dynamik und zu einer Überreaktion gegenüber stochastischen Werten führen (Parry and Evans, 2008). Der Super-Individuen Ansatz ist somit im AB-Kontext aufgrund des individuellen Informationsverlusts behutsam einzusetzen. Verkompliziert wird das Ganze durch die Rückumwandlung einzelner Individuen aus einem Super-Individuum. Der neu initialisierten Teilmenge werden dabei zufällige Variablenwerte zugewiesen, die nichts mehr mit ihren ursprünglichen Werten zu tun haben, die sie vor der Aufnahme in ein Super-Individuum aufwiesen. Laut Parry und Evans (2008)

besteht ein hohes Risiko, dass das Umschalten zwischen den Modellen sowie die Durchführung einer rückwirkenden Neuverteilung von Agenten zu erheblichen Fehlern führen und zusätzlich eine hohe Anforderung an den Prozessor und/oder Speicher stellen, deren Kapazitäten ohnehin schon begrenzt sind. Der Super-Individuen-Ansatz wird im Ausgangsmodell bei der Bildung sowie Auflösung von Schul-Agenten eingesetzt. Laut Wallentin und Neuwirth (2016) konnte für den Super-Individuen-Ansatz als Nachteil beobachtet werden, dass jeder Wechsel, egal auf welcher Detailebene, nur auf Kosten von Informationsverlusten möglich ist. Inwiefern sich der Informationsverlust auf das Systemverhalten auswirkt, ist nicht Teil der Untersuchungen. Der Fokus der Arbeit liegt auf der Replikation des Verhaltens auf Systemebene als emergentes Ergebnis.

1.4 Forschungsfrage

Das Ziel der Masterarbeit ist auf die Replizierbarkeit eines hybriden Fisch-Plankton-Modells von Wallentin und Neuwirth (2016) zu testen. Das Fisch-Plankton-Modell erklärt eine Kategorisierung eines alternativen Modelldesigns zur dynamischen Schaltung zwischen System Dynamics (SD) und Agentenbasierten Modellen (ABM), welches eine enge Kopplung dieser beiden Modellierungsansätze erlaubt. Die Replikation basiert einerseits auf dem ODD-Protokoll der Publikation und andererseits auf dem als open-source zur Verfügung gestellten NetLogo Code des publizierten Modells.

Die Replizierbarkeit wird durch eine Neuimplementierung des Fisch-Plankton-Modells von der Simulationsumgebung NetLogo in die Plattform GAMA anhand folgender Kriterien getestet:

- Inwieweit lässt sich das Modell nur anhand der Publikation in einer anderen Modellierumgebung implementieren, inwieweit muss der Sourcecode gelesen werden?
- Welche quantitativen Unterschiede konnten in den Modellergebnissen zu Populationsdynamik der Fische, der Biomassenentwicklung des Planktons und der Schulbildung im originalen Modell-Design zum replizierten Modell festgestellt werden?
- Wie entsprechen die räumlichen Muster der Simulationsergebnisse nach einem simulierten Zeitraum von 100 Jahren den publizierten Ergebnissen?
- Inwiefern stimmen die Phasen, in denen sich die gekoppelten Modelle in der AB- bzw. der SD-Phase befinden, mit den publizierten Ergebnissen überein?
- Inwiefern konnten die Schlussfolgerungen mit dem replizierten Modell bestätigt werden?

Der erste Teil der Masterarbeit umfasst die Analyse des bereits existierenden Modellkonzepts und des NetLogo-Codes. Das Modellkonzept steht in Form eines ODD-Protokolls als Download unter dem Link https://www.comses.net/codebases/5254/releases/1.2.0/ zur Verfügung. Im nächsten Schritt wird mit der Übersetzung des NetLogo-Codes in GAMAL-spezifische Syntax begonnen. Nach erfolgreicher Code-

Transformation werden die Ergebnisse aus GAMA für 100 zufällig wiederholte Simulationsdurchläufe über einen Simulationszeitraum von 100 Jahren mittels Monte-Carlo-Simulation ausgeleitet. Dasselbe wird auch in NetLogo ausgeführt. Nach erfolgreicher Ausleitung wird das Konfidenzintervall für die insgesamt 200 Runs (100 GAMA, 100 NetLogo) berechnet, um die wahren Mittelwerte zu ermitteln. Da es sich bei dem Fisch-Plankton-Modell um ein probabilistisches Modell handelt, ist die Herangehensweise der Monte-Carlo-Simulation für 100 Simulationsdurchläufe (pro Simulationssoftware) und anschließender Berechnung des KI notwendig, um die Schwankungsbreite der Ergebnisse einzugrenzen. Die Ergebnisse des berechneten KI werden anschließend in einem Liniendiagramm abgebildet. Gleichen sich die Linienverläufe, so kann angenommen werden, dass keine Softwareabhängigkeit vorliegt und sich das Fisch-Plankton-System in NetLogo und GAMA ähnlich verhält. Treten Unterschiede in den Ergebnissen auf, so können diese aus der Logik der jeweiligen Sprache resultieren. Außerhalb der Arbeit können die Ergebnisse mit weiteren Simulationsplattformen getestet werden, um zu prüfen, ob ähnliche oder gleiche Ergebnisse wie mit GAMAL erzielt werden können.

2. Methodik

Das Methodenkapitel umfasst insgesamt zwei Abschnitte. Der erste Abschnitt 2.1 beinhaltet neben dem Verstehen des konzeptionellen Modells, das Sammeln und Analysieren von Informationen für die Planung der Replikation und Modellkonstruktion. In Unterkapitel 2.1.1 wird das Modellziel des Ausgangsmodells erläutert. In Unterkapitel 2.1.2 werden die verschiedenen Arten von Entitäten aufgelistet, die sowohl im Ausgangsmodell als auch im replizierten Modell entsprechend definiert sind. Unter dem Punkt Maßstab im selben Kapitel wird die zeitliche und räumliche Auflösung sowie die Ausdehnung des Modells kurz umrissen. In Unterkapitel 2.1.3 wird ein Überblick über die im Modell enthaltenen Prozesse und deren Abläufe gegeben. Im Abschnitt 2.2 werden jene Variablen angeführt, die beim jeweiligen Start einer Simulation mit einem Set an festgelegten Parameterwerten initialisiert werden. Durch die Analyse sowie anschließende textuelle Beschreibung des konzeptionellen Modells und der in Kapitel 3 genannten Vorgehensweise der Replikation sowie der Beschreibung der verwendeten Methoden ist die Nachvollziehbarkeit der Ergebnisse und die Antwortfindung auf die Forschungsfragen gewährleistet.

2.1 Modellkonzept

Den Inhalten der vorliegenden Kapitel 2.1 und 2.2. sowie den dazugehörigen Unterkapiteln liegt das ODD-Protokoll zugrunde. Der Zweck eines konzeptionellen Modells ist die Formalisierung eines übergeordneten Verständnisses des modellierten Systems (Grimm, 2019) und seiner Komponenten. Laut Zhang und Robinson (2021) zählen zum konzeptionellen Modell sämtliche schriftliche

Modellformulierungen wie Texte, Gleichungen, Abbildungen und andere Dokumentationsformen, außer der Quellcode und die Software. Der Erfolg einer Replikation wird von der inhaltlichen sowie sprachlichen Qualität des konzeptionellen Modells beeinflusst. Denn erst mit dem Verstehen des Konzepts ist die Definition von Komponenten und ihre codebasierte Umsetzung sowie Reihenfolge möglich. Das zu replizierende Fisch-Plankton-Modell beruht auf dem Modellierungskonzept ‚ABM embedded in SD' von Vincenot (2011) und Swinerd (2012), in dem festgelegt ist, welche Agenten in AB abgebildet sind und wie sie mit ihrer SD-Umgebung interagieren (Wallentin, 2017, Vincenot et al., 2011): Individuen (Fische) agieren in einem klassischen AB-Modell und interagieren mit einer SD-Umgebung (See), welche die dynamischen Eigenschaften (Planktonverfügbarkeit) der Umwelt repräsentiert. Es erfolgt laufend ein reversibler Wechsel zwischen AB und SD, welcher dynamisch mit folgenden Konfigurationen festgelegt ist:

- Konfiguration 1 - Fisch (ABM) und Plankton (CA): Fische bewegen sich in einem a-räumlichen Plankton-Raster. Sie schwärmen, nehmen Nahrung auf, reproduzieren sich (> als 4 Jahre) oder sterben, wenn sie älter als 6 Jahre sind oder zu wenig Nahrung aufnehmen. Das Plankton wächst logistisch und diffundiert mit 10% der lokalen Biomasse pro Monat in die jeweiligen acht Nachbarzellen (Moor-Nachbarschaft). Da die lokale Planktonknappheit keine Auswirkungen auf die populationsdynamische Fischentwicklung nimmt, richtet sich die Bewegungsrichtung der Fische nicht nach der planktonreichsten Zelle.
- Konfiguration 2 - Schule (ABM) und Plankton (SD): Schulagenten entwickeln sich aus Schwärmen, die mehr als 50 Fische umfassen. Aufgrund des verspäteten Feedbacks des ‚built-in Sytem Dynamics Modeler' in NetLogo (Wallentin, 2017), werden Schulen in zwei Fisch-Bestände (junge (< 4 Jahre) und adulte Fische (4-6 J.)) organsiert und in einem Array, chronologisch nach Alter absteigend sortiert, verwaltet. Die Reproduktion in den Schulen wird durch eine SD-Wachstumsgleichung gesteuert, die nur die erwachsenen, geschlechtsreifen Fische berücksichtigt. Zusätzlich ist die Fischentwicklung zur lokalen Planktonverfügbarkeit abhängig, weshalb sie ihre Bewegungsrichtung stets nach der planktonreichsten Zelle ausrichten. Zusätzlich korrigieren sie pro Zeitschritt ihre Bewegungsrichtung, um eine Kollision mit anderen Schulen zu vermeiden.

2.1.1 Ziel des Ausgangsmodells

Mit dem Ausgangsmodell wird das Ziel verfolgt, die raumzeitliche Entwicklung planktonfressender Fische in Abhängigkeit zur Planktonverfügbarkeit in einem See dynamisch darzustellen (Wallentin and Neuwirth, 2016, Wallentin, 2017).

2.1.2 Einheiten, Variablen, Maßstab

Einheiten:

- Fische (engl. agents),
- Schulen (engl. school agents),
- Plankton (engl. plankton).

Variablen: Die Fische werden durch die Attribute Alter (age), Geschlecht (sex) sowie Energieniveau (energy level) charakterisiert. Weitere definierte Variablen des Ausgangsmodells sind in Tabelle 3 enthalten.

Tabelle 3: Definierte Variablen der Entitäten Fisch, Schule und Plankton in NetLogo

Entität	Attribut	Beschreibung
Fisch	Alter (age)	Alter der Fische.
	Geschlecht (sex)	Geschlecht der Fische.
	Größe (size)	Größe der Fisch-Symbole.
	Farbe (color)	Farbe der Fisch-Symbole.
	Nahrungsaufnahme (E_ingested)	Energielevel der durch die Nahrungsaufnahme erreicht wurde.
	Überlebensenergie (E_survive)	Energielevel fürs Überleben.
	Wachstumsenergie (E_growth)	Energielevel fürs Wachsen.
	Reproduktionsenergie (E_repro)	Energielevel für die Reproduktion.
	Schwarm-Kollegen (flockmates)	Fische, die sich in einem bestimmten Radius zueinander befinden und somit zu schwärmen beginnen.
	Nächster Nachbar (nearest_neighbour)	Nächstgelegener Fisch zu einem anderen Fisch.
	Durchschnittliche Bewegung (avg_head)	Attribut fürs Flocken.
	Durchschnittliche Bewegung zu Schwarm-Kollegen (avd_twds_mates)	Attribut fürs Flocken.
Schule	Planktonreichste Zelle (richestCell)	Schulen schwimmen in Richtung der planktonreichsten Zelle.
	Nächstgelegene Schule (nearest_school)	Schulen die in der Nähe einer anderen Schule sind.
	Fertilität (fishesToReproduce)	Erwachsene weibliche Fische die sich in den Schulen reproduzieren.
	Mortalität (fishesToStarvate)	Fische die aufgrund der geringen Nahrungsaufnahme verhungern.

Plankton	See-Zelle (is_lake)	See-Zellen im Attersee.
	Küsten-Zelle (is_coast)	Küsten-Zellen außerhalb des Attersees.
	Plankton-Bestand (plankton)	Plankton-Bestand pro See-Zelle.
	Plankton-Biobestand (plankton_bio)	Initialer Plankton-Biomassebestand im Attersee.
	Maximale Tragfähigkeit von Plankton im Attersee (carrying_capital_total)	Maximale Tragfähigkeit von Plankton im Attersee.
	Maximale Tragfähigkeit von Plankton im Attersee pro See-Zelle (carrying_capacity_cell)	Maximale Tragfähigkeit von Plankton im Attersee pro See-Zelle.

Maßstab:

- Die geographische Ausdehnung des Simulationsgebietes umfasst den Attersee mit einer Gesamtfläche von 46 km^2.
- Der Attersee setzt sich aus einem Raster mit je 200m x 200m See- und Küstenzellen zusammen.
- Der Bewegungsradius der Fische und Schulen ist auf die See-Zellen beschränkt.
- Der Simulationszeitraum des Modells reicht von einem Tag bis zu einem Jahrhundert.

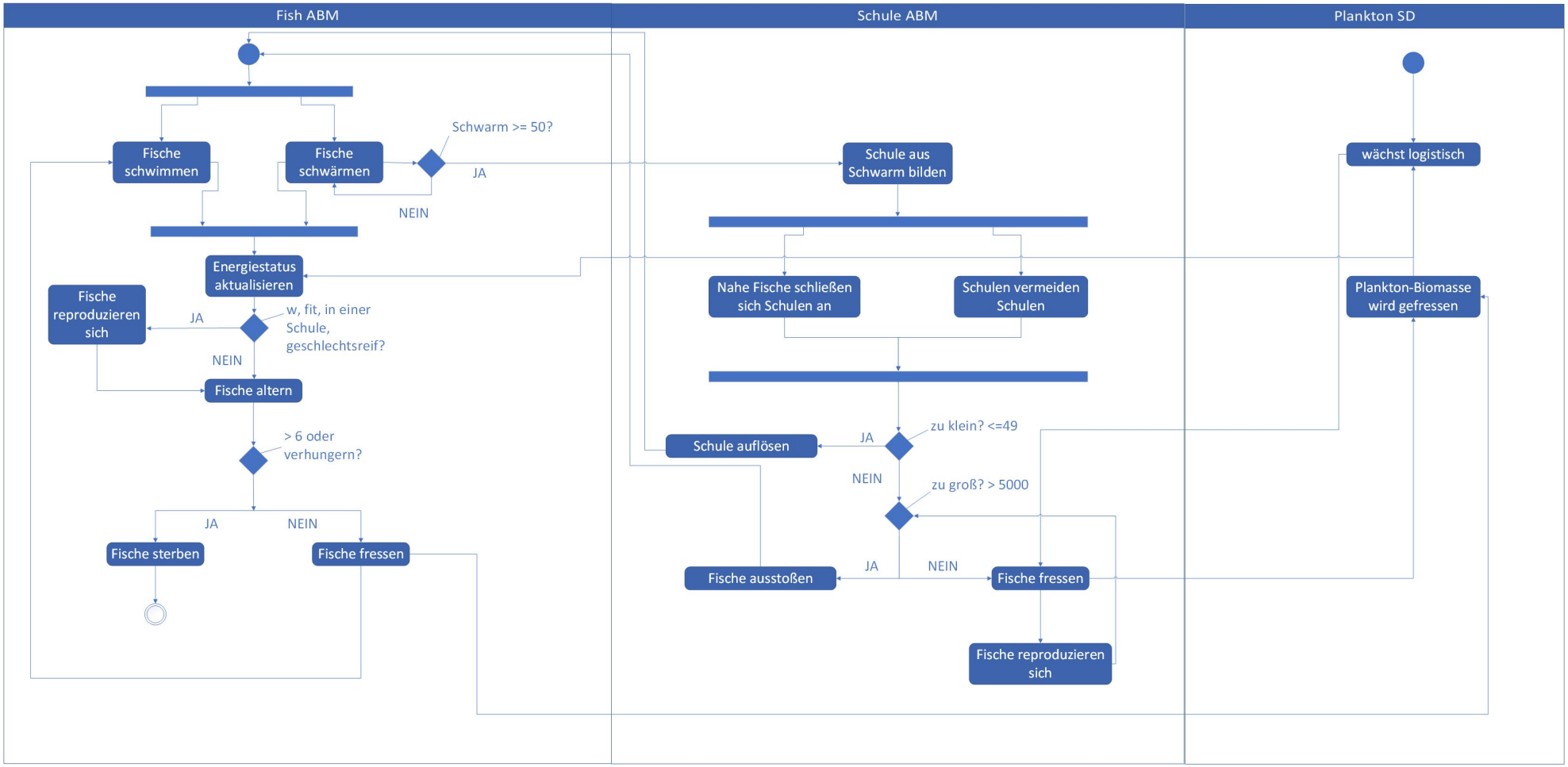
2.1.3 Modellablauf

Der Modellablauf ist in Abbildung 1 illustriert und basiert auf den UML-Grafiken aus dem ODD-Protokoll. Die Ablauffolge kann wie folgt beschrieben werden:

- Fische bewegen sich zufällig im See und orientieren sich in ihrer Bewegung nach den drei Grundregeln (Separation, Angleichung, Zusammenhalt) des Boids-Modells von Reynolds (1987).
- Unter optimalen Konditionen gebärt ein geschlechtsreifer, weiblicher Fisch fünf Jungfische pro Jahr.
- Das maximale Alter von Fischen beträgt sechs Jahre, außer sie nehmen zu wenig Nahrung auf und verhungern.
- Fische schwärmen. Ab einer Schwarmgröße von 50 Einzelfischen bilden sich Schulen und ersetzen den Schwarm.
- Fällt die Fischanzahl in Schwärme unter 50 Stück, so löst sich der Schwarm in Einzelfische auf.
- Ein Schwarm korrigiert seinen Bewegungsradius um 15 Grad zur planktonreichsten Zelle in der Moore-Nachbarschaft.
- Ein Schwarm korrigiert seinen Bewegungsradius um 25 Grad, um eine Kollision mit anderer Schule zu vermeiden.
- Ein Fisch frisst pro Tag 2,7 Gramm Plankton unter maximaler Planktonverfügbarkeit im See.

- Plankton wächst logistisch bis zu einer maximalen Tragfähigkeit des Sees von 2.300 Tonnen.
- Plankton diffundiert 10% seiner Biomasse in die nächstgelegenen acht Nachbarzellen (Moore-Nachbarschaft) pro Monat.

Abbildung 1: UML-Abbildung des Modellablaufs



2.1.4 Emergenz, Interaktionen, Stochastik und Beobachtung

Emergenz: Schulen sind eine aggregierte Form eines Schwarms und werden als sogenannte Super-Agenten bezeichnet. Super-Agenten haben die Aufgabe, die Anzahl an Einzel-Fischen zu reduzieren, um damit die Rechenleistung (engl. Performance) positiv zu beeinflussen. Ein Wechsel zwischen den Fischen und aggregierten Schulen sollte mit Bedacht ausgeführt werden, da jeder Wechsel mit einem Informationsverlust verbunden ist. Denn wenn die Einzel-Fische in die Schule aufgenommen werden, werden sie in die numerische Bestandsliste der Schul-Agenten überführt und anschließend gelöscht. Die Sortierung der Reihenfolge in der Bestandsliste (Array) erfolgt chronologisch nach Alter. Die Meta-Informationen (wie z.B. Geschlecht, etc.) gehen verloren. Wenn der Schul-Agent unter 50 Fische fällt, wird er wieder aufgelöst und die Einzel-Agenten aus den Schulen herausgelöst. Die Meta-Informationen der herausgelösten Fische werden dabei randomisiert vergeben, da die Informationen zuvor beim Wechsel in die Schulen verloren gegangen sind. Es ist daher abzuwägen, welche relevanten Informationen erhalten bleiben sollen und welche zugunsten der Performance verloren gehen können.

Interaktionen:

- Trifft ein Fisch in einem Sichtradius von 700m auf einen anderen Fisch, beginnen diese zu schwärmen. Ein Schwarm bildet sich ohne Anführer.
- Steigt die Anzahl der Fische in einem Schwarm auf > 50 so bildet sich eine Schule.
- Schulen setzen sich aus einem Bestand junger Fische (*engl. juvenile*) und einem Bestand erwachsener Fische (*engl. adult*) zusammen.
- Steigt die Anzahl der Fische in einer Schule auf > 5.000 so werden einzelne Fische aus dem Bestand ausgestoßen.

Stochastik: Die initiale Positionierung, das Alter und Geschlecht der Fische im Attersee sind stochastisch. Auch der Zeitpunkt, ab wann Schulen gebildet werden, unterliegt einer Wahrscheinlichkeit.

Beobachtung: In GAMA können verschiedene Experimente erstellt werden, die jeweils unterschiedliche Inhalte grafisch abbilden. Für das Monitoring der Simulation wurden der Typ ‚Chart - Time Series' (https://gama-platform.org/wiki/IncrementalModel_step2, 16.8.2022) zur visuellen Darstellung der Fisch-Populationsdynamik, Plankton-Dynamik Grimm (2006), Performance und Schulbildung über einen Simulationszeitraum von 100 Jahren ausgewählt. Während des Simulationsablaufs werden die Echtzeit-Daten zu ‚Anzahl an Fische', ‚Plankton-Biomasse', und ‚Anzahl an Schulen' im Monitoring-Bereich abgebildet. Dadurch kann die Simulationsentwicklung zeitgleich zur Simulation beobachtet werden. Daneben wurden auch sogenannte ‚Batch-Experiments' erstellt, die es erlauben, mehrere Simulationsdurchläufe zeitgleich durchzuführen. Dadurch können die Auswirkungen der

Modellparameter und ihre Werte analysiert und gegebenenfalls optimiert werden (https://gama-platform.org/wiki/BatchExperiments, 16.8.2022). Am Ende der Simulation sollen die Daten in einem CSV-File ausgegeben werden, um anhand deren statische Auswertungen durchführen zu können. Vor allem Daten zu den maximalen Werten, minimalen Werten und den Durchschnittswerten sind relevant.

2.2 Modellparametrisierung

Als Voraussetzung für den Vergleich der Simulationsergebnisse sind die Simulationen mit denselben Parametern zu versehen. Die Modelle sind mit folgenden Einstellungen initialisiert (Wallentin and Neuwirth, 2016):

- Seegröße: 46 km^2
- Simulationslaufzeit: 100 Jahre
- ABM Step Betrag: 1 Tag
- CA-Zellgröße: 200mx200m
- Maximales Fischalter: 6 Jahre
- Geschlechtsreife: 4 Jahre
- Max. Nachwuchs pro Jahr und pro weiblichen Fisch: 5
- Futtermenge pro Fisch: 2,7 Gramm
- Max. Geschwindigkeit von Fischen und Schulen: 20 Meter pro Tag
- Mindestanzahl an Fischen pro Schule: 50
- Maximale Anzahl an Fischen pro Schule: 5.000
- Max. Richtungskorrektur zur planktonreichsten Zelle: 15 Grad
- Max. Richtungskorrektur zur Kollisionsvermeidung mit anderen Schulen: 25 Grad
- Plankton Diffusionsrate: 10 % pro Monat
- Max. Tragfähigkeit der Plankton-Biomasse im See: 2.300 Tonnen
- Plankton-Wachstumsrate: 0,01
- Fisch Wachstumsrate: 0,002
- Initiale Plankton-Biomasse im See: 2150 Tonnen
- Initiale Fischanzahl: 25
- Initiales Fischalter: Randomisiert, 0-6 Jahre
- Initiale Geschlechtsverteilung der Fische: Randomisiert, 50% weiblich, 50% männlich

3. Replikation des Modells

In diesem Kapitel erfolgt die Beschreibung der plattformübergreifenden Konzeptumsetzung des Ausgangsmodells. Da eine detaillierte Beschreibung der Umsetzungsschritte den textlichen Rahmen der

Arbeit überschreiten würde, werden jene Schritte beschrieben, wo Unterschiede vom GAMA- zum NetLogo-Toolkit festgestellt wurden. Eine detaillierte Betrachtung des Codes ist im Kapitel 8 möglich.

Der replizierte Code wurde testgetrieben entwickelt, um zu überprüfen, wie sich der replizierte Programmcode in seiner Funktionalität verhält. Somit konnten nach jedem Entwicklungsschritt valide Rückschlüsse vom Agentenverhalten auf den Code abgeleitet und laufend iterative Verbesserungen am replizierten Modell vorgenommen werden. Laut Wilensky (2007) sollte man sich bei der Replikation bewusst gegen die Strategie des Ausgangsmodells und für ein anderes Paradigma entscheiden, welches dem ursprünglichen Modell zuwiderläuft, um die Unterschiede zwischen dem konzeptionellen und dem implementierten Modell leichter zu erkennen. Dagegen spricht der Exkurs, der während der Entwicklung des replizierten Modells gemacht wurde, indem die Schul-Agenten mittels einer advanced-Spezies anstelle eines Arrays umgesetzt wurden und daraus Unterschiede in den Ergebnissen resultierten. Näheres dazu siehe Kapitel 5.

3.1 Replikationsumsetzung

3.1.1 Definition des Simulationszeitraums

Der Simulationszeitraum des hybriden Fisch-Plankton Modells sind 100 Jahre und das ‚ABM step increment' ist mit 1 Tag festgelegt. GAMA basiert auf einer abstrakten Zeiteinheit, die durch die Anzahl an Zyklen (engl. cycle) bestimmt wird (https://gama-platform.org/wiki/ManipulateDates, 16.08.2022). Zyklen sind Integer-Werte, die nicht durch den Modellierer angepasst werden können (https://gama-platform.org/wiki/ManipulateDates, 16.08.2022). Sie werden durch eine sogenannte step-Variable (float) definiert, deren Dauer wiederum modifiziert werden kann. Der default-Wert für einen step entspricht einer realen Sekunde (https://gama-platform.org/wiki/ManipulateDates, 16.08.2022). Eine weitere GAMA-Zeitvariable ist ‚time' (float), die den Zeitraum ab Beginn des Simulationsstarts misst (time = cycle * step) (https://gama-platform.org/wiki/ManipulateDates, 16.08.2022). Die Werte für den Zyklus werden in GAMA während der Simulation im linken oberen Maskenbereich angezeigt. Die Anzeige ist in Sekunden, Tagen, Monaten und Jahren möglich. Die Notation von Zeit ist in GAMA wie

folgt festgelegt und bildet die Basis für die zeitliche Manipulation der globalen step-Variablen (https://gama-platform.org/wiki/ManipulateDates, 16.08.2022):

- #s : second - 1 second
- #mn : minute - 60 seconds
- #hour : hour - 60 minutes - 3600 seconds
- #day : day - 24 hours - 86400 seconds
- #week: week - 7 days - 604800 seconds
- #month : month - 30 days - 2592000 seconds
- #year : year - 12 month - 31104000 seconds

NetLogo basiert auf sogenannten **ticks**, die eine abstrakte Zeiteinheit darstellen. Ticks sind numerische Werte des Typs Integer oder Float (https://ccl.northwestern.edu/netlogo/docs/dictionary.html#tick, 16.8.2022), die die Simulation bei jedem Tick aktualisiert. Die Anzahl der Zeitschritte wird am oberen Rand des Beobachtungsfensters angezeigt. 1 Tick entspricht in der Simulation 1 Tag.

3.1.2 Definition des Modellablaufs

Die Prozesse und Abläufe werden in GAMA durch sogenannte **Aktionen** oder **Reflexe** gesteuert. Unter Aktion (engl. actions) werden Funktionen oder Prozeduren einer Spezies verstanden (https://gama-platform.org/wiki/DefiningActionsAndBehaviors, 16.8.2022), die entweder einen Wert als Aktion rückliefern oder als Prozedur nicht. Die Ausführung der Aktionen sind mit einem sogenannten Do-Befehl dezidiert aufzurufen oder bei einer Werterückgabe mit dem Befehl **any_agent action(arguments)** (https://gama-platform.org/wiki/DefiningActionsAndBehaviors, 16.8.2022). Reflexe werden automatisch bei jedem step ausgeführt (https://gama-platform.org/wiki/DefiningActionsAndBehaviors, 16.8.2022). Die Reihenfolge, in der die Reflexe aufgerufen werden, sind als solche zu definieren. Sie ergibt sich aus der chronologischen Aneinanderreihung der Reflexe-Befehle im Code. Die Verhaltensarchitektur ist eine built-in Architektur und ist unter folgendem Link näher spezifiziert: https://gama-platform.org/wiki/BuiltInArchitectures.

Die Prozesse und Abläufe werden in NetLogo durch sogenannte Commander und Reporters gesteuert, die den Agenten sagen, was zu tun ist (https://ccl.northwestern.edu/netlogo/docs/programming.html, 16.8.2022). Ein **command** ist eine Aktion für einen Agenten, resultierend mit einem bestimmten Effekt (https://ccl.northwestern.edu/netlogo/docs/programming.html, 16.8.2022). **Reporter** sind Instruktionen für einen Agenten, die er reportet, wenn er danach gefragt wird (https://ccl.northwestern.edu/netlogo/docs/programming.html, 16.8.2022). Kommandos und Reporter sind in NetLogo built-in Variable und setzen sich aus Verben zusammen wie zum Beispiel **create**, **die**,

inspect, etc. Kommandos und Reporter, die der Modellierer selbst erstellt, werden als Prozeduren bezeichnet (https://ccl.northwestern.edu/netlogo/docs/programming.html, 16.8.2022). Die Reihenfolge der Prozesse und der Abläufe werden in NetLogo mit der sogenannten Go-Prozedur festgelegt (https://ccl.northwestern.edu/netlogo/2.0.0/docs/tutorial3.html, 16.8.2022).

3.1.3 Erstellen von Agenten

In NetLogo werden Agenten anfänglich als **breed** definiert, also eine Art Gattung von Agenten (https://ccl.northwestern.edu/netlogo/docs/dict/breed.html, 29.4.2023): **breed [name-in-mehrzahl name-in-einzahl]**. Sobald ein Breed definiert ist, können neue Agenten mit dem Befehl **create-fish anzahl []** daraus erstellt werden. Die Anzahl der Fische wird zu Simulationsbeginn mit 25 initialisiert. Der Standardagent Fisch ist mit einem orangefarbenen Fisch-Icon symbolisiert.

In GAMA erfolgt die Initialisierung der Agenten ähnlich zu NetLogo. Die Fische werden mit dem Befehl **action init_fish** samt seinen Attributen im globalen Bereich erstellt. Die grafische Ausprägung wird in **aspect default** festgelegt. Es wurde bewusst zu einer Liniendarstellung aus Optimierungsgründen (vgl. https://gama-platform.org/wiki/OptimizingModels#shape, 29.4.2023) entschieden. Denn aufwendige grafische Formen oder Mix-Darstellungen wirken sich negativ auf die Rechengeschwindigkeit aus. Die Anzahl der Fische wird zu Simulationsbeginn mit 25 initialisiert. Da sich die Schulen aus den Schwärmen bilden, müssen sie nicht wie die Fisch-Spezies im globalen Bereich initialisiert werden. Der Standardagent Schule ist mit einem gelben Kreis symbolisiert. In Abbildung 2 sind sämtliche in GAMA definierten Spezies angeführt.

Abbildung 2: In GAMA definierte Spezies inklusive Attribute, Reflexe und Actions

3.1.4 Bewegungsverhalten

Der Befehl, dass ein Fisch einen Patch vorwärts schwimmt, lautet sowohl in NetLogo als auch in GAMA **move**. Initial werden die Fische randomisiert im See gesetzt, ab dieser Position schwimmen sie immer geradeaus bis zu 20 Meter pro Tag. Stoßen sie in NetLogo- oder GAMA-Welt an den Rand der Welt, so wird ihre heading-Variable um 180 Grad korrigiert. Für die realistische Gestaltung der Bewegungsmuster

unterliegt dem Bewegungsverhalten der Fische die Besonderheit der Schwarmintelligenz (Flocking) als synchronisiertes Gruppenverhalten (Wallentin and Neuwirth, 2016, Reynolds, 1987). Flocking wurde als fertiges Skript in GAMA (siehe Kapitel 8.3) und in NetLogo (https://ccl.northwestern.edu/netlogo/models/Flocking, 21.4.2023) mit dem Befehl **flock** implementiert. *Hinweis: Im GAMA Flocking-Skript wurde die Notation zur ‚0-heading direction in GAMA is east instead of north! -> thus +90' mittels dem Befehl* ***atan2(y_comp, x_comp)*** *ausgebessert, da die Angaben für die x- und y-Komponenten vertauscht waren.*

Jeder simulierte Fisch ist in GAMA als auch in NetLogo als unabhängiges Individuum implementiert, der entsprechend seiner lokalen Wahrnehmung und seiner dynamischen Umgebung nach den drei Gesetzen des Boids-Modells von Reynolds (1987) navigiert:

1. Kollisionsvermeidung: Vermeidung von Kollisionen mit nahen Schwarmkameraden.
2. Geschwindigkeitsanpassung: Versuch der Geschwindigkeitsanpassung an naheliegenden SchwarmkollegInnen (Flockmates).
3. Zentrierung: Versuch, nahe an nahe gelegenen Flockmates zu bleiben.

Die Zahl, die vor der jeweiligen Verhaltensregel steht, gibt die Ausführungsreihenfolge (Priorisierung) an. Da jedes Individuum ab einer bestimmten Schwarmgröße mit seiner Navigationsaufgabe überfordert wäre, ist die Größe eines Schwarms zu begrenzen (Reynolds, 1987). Die Begrenzung des Schwarms wird in NetLogo mit der Methode **limit-school-size** und in GAMA mit dem Reflex **emit_fish** auf 5.000 Stück umgesetzt. Das Navigationsmodul sammelt alle relevanten Beschleunigungswünsche der Individuen und mittelt sie, um daraus eine einzige verhaltensmäßig gewünschte Beschleunigung zu bestimmen (Reynolds, 1987). Für das Fisch-Plankton-Modell bedeutet die Umsetzung der kollektiven Schwarmintelligenz folgendes: Fische, die sich in einem Radius von 700m begegnen, beginnen einander zu folgen. Ab einer Schwarmgröße von mehr als 49 Einzelfischen bildet sich ein Super-Individuum in Form eines Schul-Agenten. Ab diesem Moment folgt der Schul-Agent nicht mehr der ABM-Verhaltenswelt sondern einem SD-Regelwerk. Die Idee hinter diesem Konzept ist, dass aus einer Menge ‚einfacher' Individuen ein intelligentes Kollektiv entsteht, welches die Computer-Rechenleistung über den gesamten Simulationszeitraum performant hält, da sich ABM und SD jeweils unterschiedlich auf die CPU-Rechenleistung auswirken (Wallentin and Neuwirth, 2016, Swinerd, 2012). In GAMA tritt im Vergleich zu NetLogo das Problem auf, dass ab einer gewissen Individuenzahl in Schwärme die CPU-Auslastung bis hin zu einer OutofMemoryException steigt. Zusätzlich erhöhte sich die Dauer eines Zyklus, wodurch die Ausführung eines Simulationsschrittes im Vergleich zu NetLogo ab einer Schwarmgröße von 80.000 Fischen um mehr als das Dreifache langsamer wird. Das Problem in GAMA dürfte an der Flocking-Methode liegen. Laut Reynolds ist das Flocking nicht auf eine hohe Effizienz der

Rechenleistung ausgelegt, sondern auf die Darstellung realer Verhaltensweisen (Reynolds, 1987). Weshalb das Flocking keinen signifikanten Einfluss auf die Rechenleistung in NetLogo nimmt, ist nicht bekannt. Um den negativen Auswirkungen des Flocking entgegenzuwirken, wurden in GAMA-Optimierungsmethoden (siehe Tabelle 4) umgesetzt. Leider konnte damit keine effizientere Rechenleistung festgestellt werden.

Tabelle 4: GAMA-Optimierungsmethoden zur Förderung der Rechenleistung (https://gama-platform.org/wiki/OptimizingModels, 15.7.2022)

Benchmarking	Überwachung wieviel Zeit welche Modellparts in Anspruch nehmen. Anhand der Ergebnisse können gezielt Optimierungsschritte an den jeweiligen Parts vorgenommen werden.
Scheduling	Auffinden von Agenten, die keiner Funktion nachgehen, um sie zu löschen. Dadurch wird vermieden, dass unnötig hohe Datenmengen produziert werden.
Grid	Es stehen verschiedene Befehle zur Optimierung von Grid-Agenten zur Entlastung des Computerspeichers zur Verfügung, z.B.: • use_regular_agents: ist der Wert auf ‚false' gesetzt, so werden spezielle Klassen von Agenten verwendet, die zur Entlastung des Speichers führen. • use_individuale_shapes: ist der Wert auf 'false' gesetzt, so wird für alle Agenten derselbe Geometrietyp (z.B. Punkt, Linie, etc.) verwendet, der zur Entlastung des Speichers führt. • etc.
List-Operatoren	Verwendung von List-Container.
Spatial-Operator	Definition von räumlichen Operatoren, um das Wirkungsfeld der Agenten zu begrenzen, um die Rechenleistung zu entlasten.
Darstellung	Zur Darstellung der Agenten sollen einfache Geometrieformen wie z.B. Punkte, Linien, etc. verwendet werden, um das Rendering zu vereinfachen.

Die Geschwindigkeit von Schulen und Einzelfischen unterscheiden sich visuell in GAMA zu jenen in NetLogo, trotz des gemeinsamen festgelegten Bewegungsradius von 20 Meter pro Tag. Das Phänomen tritt verstärkt auf, je höher die Individuenzahl. Die tatsächliche Geschwindigkeit wird weder in GAMA noch in NetLogo gemessen. Schulen vermeiden die Kollision mit anderen Schulen, indem sie in einem Sichtbarkeitsbereich von 200 m ihre Richtung um 25 Grad korrigieren. Die Kollisionsvermeidung wird in GAMA mit der Methode **collision_detect** und in NetLogo **seperate_from_other_schools** umgesetzt. Ein weiteres einflussnehmendes Kriterium auf das Bewegungsverhalten von Schulen ist die Korrektur der Schwimmrichtung um 15 Grad pro tick/step nach der planktonreichsten Zelle. Die Korrektur wird in GAMA mit dem do **turn_towards_richest_cell** als Bestandteil der **move**-Bewegung und in NetLogo mit dem Befehl **turn-towards (towards max-one-of neighbors (plankton-bio)) 15** als Bestandteil der Methode **separate-from-other-schools** umgesetzt.

Zusammenfassend kann festgehalten werden, dass sich das Bewegungsverhalten von Fischen und Schwärmen/Schulen in GAMA zu jenen in NetLogo mit steigender Individuenzahl unterscheidet. Bajracharya (2013) vertritt in diesem Zusammenhang den Nachweis, dass die Unterschiede im Bewegungsverhalten der Agenten in den verschiedenen Plattformen Einfluss auf die Ähnlichkeit der Simulationsergebnisse der jeweiligen Plattformen nehmen. Inwiefern sich das definierte Bewegungsverhalten im replizierten Modell auf die Simulationsergebnisse auswirkt, wird in Kapitel 5 diskutiert.

3.1.5 Fressverhalten

Fische als einzelnes Individuum als auch akkumuliert in Schulen fressen pro Tag 2,7 Gramm Plankton. Das Plankton wird proportional zum gesamten Planktonbedarf pro Tag von jeder See-Zelle entnommen, wo sich der Agent gerade darauf befindet.

Die Nahrungsaufnahme der Einzelfische und Schulen erfolgt in GAMA mit der Methode **plankton_feed**, die wie folgt umgesetzt ist: **Plankton <- plankton – ((length(fish_species overlapping self) + sum_of(school_species overlapping self, each.fish_stock))* 0.0000027** In NetLogo heißt die Methode **bioass-removed** und ist wie folgt umgesetzt: **report ((count fish-here + sum (fish_stock) of schools-here) * 0.0000027** Die Nahrungsaufnahme wirkt sich in beiden Modellen direkt auf den Energiehaushalt der Fische aus, der wiederum Einfluss auf die Fortpflanzung (E_repro > 15) sowie Sterblichkeit (E_repro = 0) nimmt.

Die Berechnung des Energiehaushalts der Fische erfolgt nach den Formeln von Sibly (2013):

$$Energy\ ingested\ (Ei) = 2.7 * \frac{planktondensity}{planktondensity + 2}$$

$$Energy\ maintenance\ (Em) = \ 0.005\ * (age * 100)\hat{}0.75$$

$$\text{Energy growth (Eg)} = 0.5 * (\text{maxEi} - \text{Em})$$

$$Energy\ reproduction\ Er(t+1) = Er\ (t)+Ei\ (t)-Em\ (t)-Eg\ (t)$$

Die Formeln des Energiehaushaltes werden in GAMA sowie in NetLogo in die Methoden **update_energy** implementiert. Die individuelle Energiebilanz ist abhängig von der Planktonverfügbarkeit. Fische sterben in GAMA wie auch in NetLogo mit dem Befehl **do die**.

3.1.6 Reproduktion und Sterblichkeit

Die Reproduktion und Sterblichkeit von Fisch-Agenten zu Schul-Agenten unterscheiden sich wie folgt: Wie in Kapitel 3.1.5 erwähnt, hängt die Reproduktion der Fisch-Spezies einerseits von der jeweiligen Nahrungsaufnahme ab, die einen direkten Einfluss auf den jeweiligen Energiehaushalt nimmt (E_repro > 15) und andererseits davon, ob mindestens ein männlicher und mindestens ein weiblicher geschlechtsreifer (> 4 Jahre) Fisch aufeinandertreffen. Die Reproduktion ist in GAMA mit dem Reflex **reproduce** und in NetLogo mit einer gleichnamigen Methode umgesetzt.

Einzelne Fisch-Agenten sterben, entweder durch zu wenig Nahrungsaufnahme (E_repro < 0), oder wenn sie älter als sechs Jahre sind. Die Sterblichkeit ist in GAMA mit dem Reflex **age_and _die** und in NetLogo im **to go**-Abschnitt mit der if-Anweisung **if age > 6 or E-repro < 0 (die)** implementiert.

Die Reproduktion und Sterblichkeit von Schul-Agenten sind in der Literatur mittels folgenden Formeln definiert (Wallentin and Neuwirth, 2016):

$$Fish\ growth = \ r\text{–}fish * fish * (\frac{plankton}{K - plankton})$$

$$Fish\ mortality = \ \text{fish} * (1 - ((\text{plankton}/\text{K} - \text{plankton})0\hat{}.015))$$

Die Variablen sowie ihre Größen sind nicht näher spezifiziert. Ohne Originalcode ist es daher schwierig zu erahnen, was sich genau hinter den jeweiligen Größen (z.B. r-fish) befindet. Zudem weicht die Formel zur Sterblichkeit (Fish mortality) im Originalcode zu jener in der Literatur ab:

$$(1 - ((\text{CA(location).plankton /carrying\_capactiy\_cell})\hat{}0.33)) \ * \ 0.24$$

Weshalb im Quellcode die Sterblichkeit mit einer zur Literatur abweichenden Formel umgesetzt wurde, ist nicht bekannt. Im Rahmen der Replikation wurde dieselbe Formel wie im Originalcode übernommen. Die Sterblichkeit sowie die Reproduktion sind in NetLogo und in GAMA wie folgt implementiert:

NetLogo:

```
fish-growth-increment ((adult-fish-stock/2)*(5/365)*(plankton-bio) of patch-here/carrying-capacity
starvation-rate (1-((plankton-bio/carrying-capacity) ^ 0.033))*0.24
```

GAMA:

```
fishGrowthIncrement <- ((adultfish_stock/2)*(5/365)*CA(location.plankton/ carrying_capacity_cell)
starvation_rate <- (1-((CA(location).plankton /carrying_capactiy_cell)^0.33)) * 0.24
```

Die Entwicklung der Fische in den Schulen hängt gemäß den Formeln stark von der Planktonverfügbarkeit pro See-Zelle ab. Die Größe der Reproduktion und der Sterblichkeit unterliegen Zufälligkeiten, da ihr berechneter Wert davon abhängt, wie viele Schulen sich wie oft über eine See-Zelle bewegen und wie hoch dadurch ihre Nahrungsaufnahme ist. Die Höhe der Nahrungsaufnahme hängt zudem von der jeweiligen zur Verfügung stehenden Plankton-Biomasse pro See-Zelle ab, die zu Beginn der Simulation randomisiert im See verteilt wird. Inwiefern die äußeren Zufälligkeiten sowie jene in den Formeln die Ergebnisabweichungen beeinflussen, ist nicht messbar.

3.1.7 Schulen

Agentenbasierte Plattformen bieten die Möglichkeit eine Liste von Agenten zu sortieren. In NetLogo und in GAMA sind die Fische in Schulen in Listen (Arrays) definiert, deren Anzahl chronologisch sortiert nach Alter absteigend verwaltet wird. Die Liste ist in zwei Rubriken unterteilt:

- junge Fische (0-4 Jahre): Listenindex NetLogo: 1.460-2.190, Listenindex GAMA: 0-1.460
- adulte Fische (4-6 Jahre): Listenindex NetLogo: 0-(2.190-1.461), Listenindex GAMA: 1.461-2.190.

Die Aufteilung der Schulen in juvenile und adulte Fischbestände ist dem vierjährigen Reproduktionsversatz geschuldet sowie der fehlenden Berücksichtigung in der NetLogo-spezifischen Methode ‚built-in Sytem Dynamics Modeler' (Wallentin, 2017). Die Sortierung der jungen und adulten Fische ist in GAMA zu NetLogo umgekehrt. Die umgekehrte Sortierung wurde aus Gründen der Einfachheit gewählt. Schwimmen Einzelfische im Radius von Schulen (Vision=300m) werden diese in die Schulen automatisch aufgenommen und anhand ihres Alters dem jeweiligen Listenindex (i+1) hinzugefügt.

3.1.8 Plankton

Plankton wird initial mit einer Gesamtmasse von 2.300 Tonnen im See randomisiert verteilt. Das Plankton wächst logistisch und ist in der Literatur (Wallentin and Neuwirth, 2016) wie folgt definiert:

$$Plankton\ growth = \text{plankton} * \text{r-plankton} * (1 - \text{plankton/K-plankton})$$

Die größte konzeptionelle Herausforderung in GAMA ist das Setzen eines entsprechen **integration step** für die rk4-Methode zur Berechnung der logistischen Wachstumsfunktion. Bei der rk4-Methode handelt es sich um das klassische Runge-Kutta-Verfahren zur numerischen Lösung von Differentialgleichungen (https://gama-platform.org/wiki/DifferentialEquations#integration-method-with-the-method-facet, 12.7.2022). Es wurde versucht, den idealen Wert für den integration step zu finden, da er einen großen Einfluss auf die Präzision der Ergebnisse nimmt. Hierzu wurde die globale Variable **step** (= Dauer eines Simulationsstep in Sekunden) auf die Einheit Tag festgelegt (https://gama-platform.org/wiki/ManipulateDates, 14.7.2022). Jedoch beeinflusst der step-Wert den integration step, weshalb des Öfteren als Ergebnis der logistischen Wachstumsfunktion der Wert NaN retour gegeben wurde. NaN steht für not a number und ist eine Fehlermeldung für jene Werte, die außerhalb eines Intervalls liegen bzw. zu groß sind, um entsprechend dargestellt werden zu können. Als Lösung wurden die Facets **t0** und **tf** definiert, die den ersten sowie den zweiten Rahmen des Integrationsintervalls festlegen (https://gama-platform.org/wiki/DifferentialEquations#integration-steps, 30.4.2023).

Die Diffusion des Planktons (10% pro Monat) in die nächstgelegenen acht Nachbarzellen wird in NetLogo mit dem Befehl **diffuse plankton-bio (0.1/30)** und in GAMA mit **diffuse var:diff_p on:lake_cells avoid_mask:true** umgesetzt. Ob das in den beiden Gleichungen hinterlegte built-in Skript **diffuse** dieselbe Funktionsweise aufweist, kann nicht festgestellt werden, da keine Einsicht in das Skript möglich ist. Der Planktonwert pro Zelle wird durch die Nahrungsaufnahme pro Fisch wie folgt reduziert (Wallentin and Neuwirth, 2016):

$$Plankton\ decay = fish * 0.0000027$$

3.2 Modellvergleich

Um die Modelle miteinander vergleichen zu können, werden im ersten Schritt die Daten im CSV-Format aus GAMA und NetLogo exportiert. Anschließend werden die Daten mit dem Werkzeug R zur statistischen Datenanalyse und zur grafischen Darstellung der Ergebnisse aufbereitet und analysiert. R ist ein beliebtes Tool für alle Arten der Datenaufbereitung und -analyse (https://www.r-project.org, 24.07.2023). Laut McCullough (2009) ist ein direkter Vergleich der Ergebnisse zwischen dem Ausgangs- und dem Zielmodell nicht sinnvoll, da verwendete Zufallswerte, wie beispielsweise für die Startposition, das Alter, etc., keine zuverlässigen Ergebnisse hervorbringen. Bekanntlich neigen die Ergebnisse von Wahrscheinlichkeitsverteilungen grundsätzlich aufgrund ihrer innewohnenden Stochastik zur starken Differenz (Wilensky, 2015). Sie zeigen somit keine eindeutige Präferenz zugunsten einer der Simulationsplattformen. Wie stark die Ergebnisse differieren hängt von mehreren Faktoren ab, wie der

instabilen Dynamik (welche zu unpräzisen Ergebnissen führt) oder der Spiegelung etlicher unterschiedlicher Trajektorien, die das Einschlagen des Systems in unterschiedliche Verlaufspfade ermöglichen, je nachdem welche Zufallsgrößen erzeugt werden (McCullough, 2009). Nicht nur die Zufälligkeiten führen zu Unterschieden in den Ergebnissen, sondern auch die spezifische Syntax der jeweiligen IDE kann sich auf die Resultate auswirken (Viehof, 2017). Die Vorgänge des Replizierens und des kritischen Reflektierens über Modelle sind essenzielle Elemente der Erkenntnisgewinnung. Eine fehlende Replizierbarkeit ist ein kritischer Umstand, weshalb man als ReplizererIn oftmals versucht ist, ein Modell so lange abzuändern, bis die Ergebnisse übereinstimmen (Richmond, 2001, Wilensky, 2007). Dieses Vorgehen verfälscht jedoch das ursprüngliche Modellverhalten. Es ist daher wichtig, bei der Ergebnisinterpretation sich ins Bewusstsein zu rufen, dass aufgrund der zuvor genannten Gründe Unterschiede in den Ergebnissen auftreten können und allein schon wegen der spezifischen Übersetzungen der Modelleigenschaften in die jeweilige Programmiersprache verursacht werden.

Um auf fehlende Replizierbarkeit zu testen, werden die Simulationen mehrfach wiederholt (siehe Kapitel 3.2.1). Für die statistische Datenauswertung wird das Konfidenzintervall (siehe Kapitel 3.2.2) anstelle der Prüfung auf Einhaltung eines Replikationsstandards (RS) eingesetzt (Schmid, 2006), da die Replikationsstandards (Wilensky, 2007, Axtell et al., 1996) für stochastische Modelle ungeeignet sind (Axtell et al., 1996, Wilensky, 2007), siehe Kapitel 4. Die Ergebnisse sind in Kapitel 4 dargestellt, wo die Screenshots als beispielhafte Momentaufnahmen zu verstehen sind (Füllsack, 2010).

3.2.1 Monte Carlo Simulation

Das Verfahren der Monte-Carlo Simulation ist ein Hilfsmittel aus der Wahrscheinlichkeitstheorie (Frey and Nießen, 2001) und wird vorzugsweise deshalb angewendet, um eine hohe Anzahl an brauchbaren Resultaten in einer vernünftigen Rechenzeit zu erhalten (Theis C., 2002). Die Monte-Carlo-Simulation besteht aus Algorithmen, die wiederholte Zufallsstichproben verwenden, um die Wahrscheinlichkeit des Eintretens von Ergebnisbereichen zu ermitteln (Müller-Gronbach et al., 2012, Theis C., 2002). Es werden daher für jede Simulation die Attribute (z.B. Alter, Geschlecht, etc.) sowie die räumliche Distribution der Individuen zufällig verteilt. Nach jedem Simulationslauf (Run) wird eine Kopie des Systems erstellt, anhand derer ein Mittelwert gebildet wird (Theis C., 2002). Anschließend können die Verteilungen der unterschiedlichen Plattformen gegenübergestellt werden, da durch die mehrmaligen Simulationswiederholungen mit denselben Parametern ein direkter Vergleich zulässig ist (Wallentin, 2020).

Die Monte-Carlo-Simulation des Fisch-Plankton-Modells wurde in GAMA als auch in NetLogo mit je 100 Runs derselben Eingangsparameter (siehe Kapitel 2.2) durchgeführt. Der Analysezeitraum ist mit 100 Jahren festgelegt. Als Prüfgrößen sind die ‚Anzahl der Fische', ‚Anzahl adulter Fische außerhalb von Schulen', ‚Anzahl der Schulen' und ‚Plankton-Biomasse' definiert. Zur effizienteren Datenausleitung wurden zeitgleich mehrere Simulationen (Runs) in NetLogo mittels der Behavior-Space-Methode und in GAMA mittels dem batch-Experiment ausgeführt. Jeder Run gibt somit Werte für eine Zeitreihe zurück, anhand der die probabilistischen Variationen der Runs untersucht werden. Es geht um die Frage, wie stark die Simulationsergebnisse in NetLogo bzw. in GAMA streuen, wenn die Simulationen mehrfach ausgeführt werden. Die Ergebnisse aus GAMA und NetLogo, die jeweils eine gewisse Schwankung aufweisen, können ebenfalls miteinander verglichen werden. Die graphische Gegenüberstellung der Ergebnisse erfolgt mittels Liniendiagrammen.

3.2.2 Konfidenzintervall (KI)

Das Konfidenzintervall (KI) auch Vertrauensintervall genannt, gibt jenes Intervall an, in dem sich ein Populationsparameter (z.B. arithmetisches Mittel) mit einer Wahrscheinlichkeit des Konfidenzniveaus (1- α) befindet (Rudolf, 2008). Anders formuliert das KI gehört zu den Methoden der Inferenzstatistik, wo aus gezogenen zufälligen Stichproben die entsprechenden Parameter geschätzt und anhand dessen ein Bereich (Intervall) definiert wird, in dem der gesuchte Parameter mit einer festgelegten Wahrscheinlichkeit liegt (Planing, 2021, Du Prel et al., 2009). Das KI gibt somit die Genauigkeit der Schätzung eines untersuchten Populationsparameters wieder (Du Prel et al., 2009), anhand deren die Unterschiede in den Simulationsergebnissen bewertet werden (Viehof, 2017). Die Größe des KI wurde in der Arbeit auf 99% festgelegt und als z-Wert 2,58. Ein Konfidenzintervall von 99% beschreibt, dass 99% der aus einer Grundgesamtheit gezogenen Stichproben den Erwartungswert μ der Grundgesamtheit überdecken (Rudolf, 2008). Es wird daher erwartet, dass lediglich 1% der Stichproben nicht im ‚wahren' Mittelwertbereich der Gesamtpopulation liegt (Rudolf, 2008, Rasch et al., 2021). Das KI wird im Rahmen der Arbeit anhand der Stichprobenkennwerteverteilung der Mittelwerte aller Monte-Carlo-Simulationsergebnisse (N=100 pro Plattform) mittels folgender Formel berechnet (https://datatab.de/tutorial/konfidenzintervall, 18.1.2023):

$$KI = x \pm z * \frac{s}{\sqrt{n}}$$

KI … Konfidenzintervall
x … Mittelwert
z … z-Wert für das Konfidenzintervall
s … Standardabweichung
n … Stichprobengröße

Zunächst wird ein Stichprobenmittelwert, der als Ausgangspunkt der Schätzung dient, markiert (Planing, 2021). Als nächstes wird ein Intervall (Bereich) rund um diesen Wert bestimmt, mit einer 99%-igen Überdeckungswahrscheinlichkeit (auch Konfidenzkoeffizient) (Bender and Lange, 2001b). Bei 99%-KI ist sichergestellt, dass der gesuchte Mittelwert mit einer Wahrscheinlichkeit von 99% in diesem Intervall liegt (Planing, 2021).

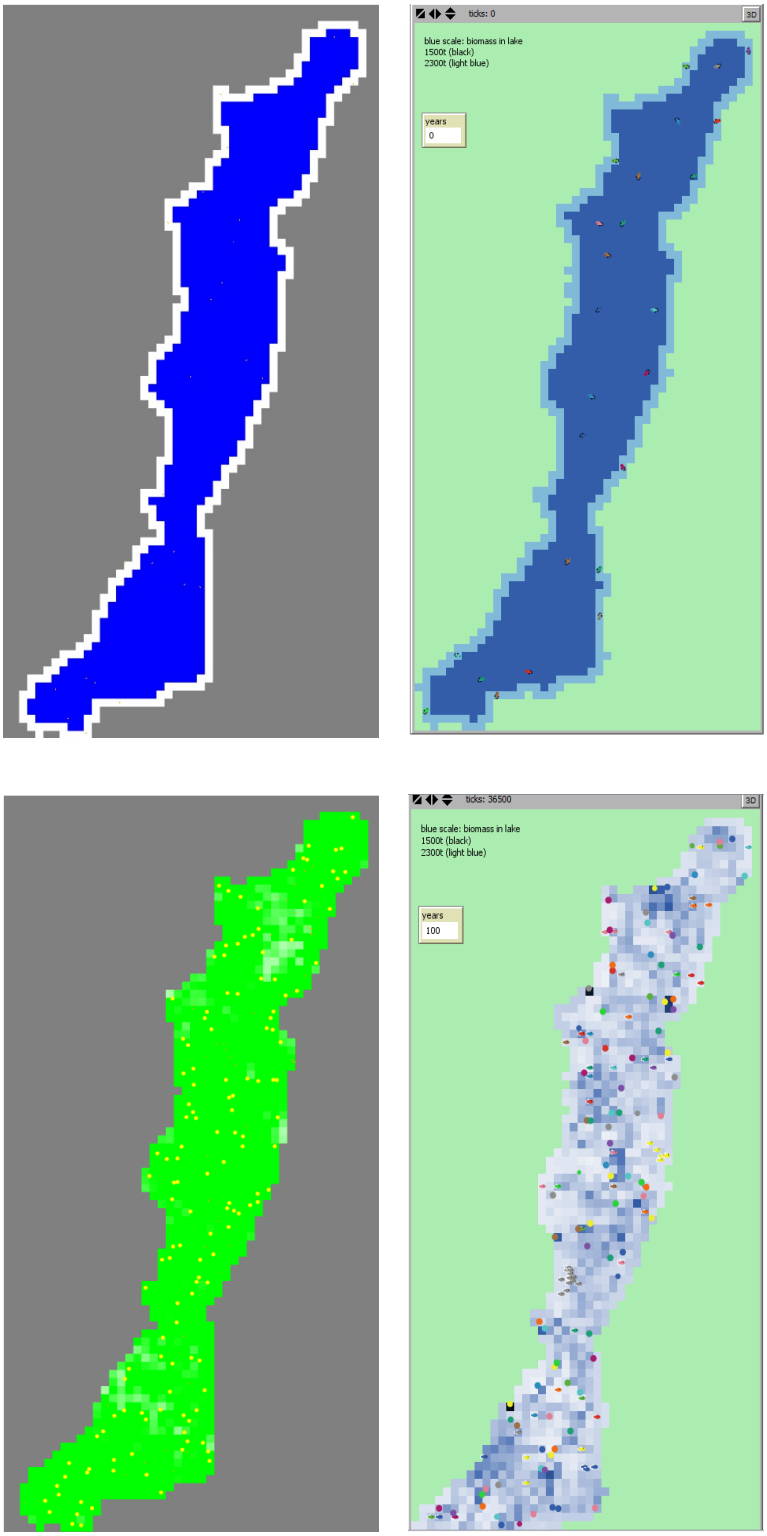
Die Ergebnisse des KI werden mittels einem Liniendiagramm visualisiert und anhand eines augenscheinlichen Vergleichs in Bezug auf Periodizität, Symmetrie, Lage und Höhe von Extrema, etc. über deren Validität entschieden (Viehof, 2017).

4. Ergebnisse

Der Grad der Ähnlichkeit bemisst sich nach dem Verhältnis der Replikationsergebnisse mit den Modellergebnissen. Laut Axelrod (1997) gibt es drei Kriterien anhand derer die unterscheidenden Ereigniseigenschaften bewertet werden können: numerische Identität, Verteilungsäquivalenz und relationale Übereinstimmung. Die numerische Identität wird in der Masterarbeit ausgeschlossen, da es erfordert, dass das Ausgangsmodell und das replizierte Modell exakt dieselben Ergebnisse liefern. Es hat sich bereits im Vorfeld gezeigt, dass nicht einmal dasselbe Fisch-Plankton-Modell pro Simulationsdurchlauf am selben Rechner ein und dasselbe Ergebnis liefert. Somit ist das Ähnlichkeitsmaß für die Feststellung auf volle Replikation ungeeignet. Auch das Kriterium der Verteilungsäquivalenz ist als Bewertungsmaß nachteilig, da aufgrund der stochastischen Natur des Fisch-Plankton-Modells der Nachweis einer statistisch signifikanten äquivalenten Verteilung der Ergebnisse schwer zu legen ist. Auch das Kriterium der relationalen Übereinstimmung ist aufgrund der innewohnenden Stochastik des Modells ungeeignet, da die Ergebnisse der beiden implementierten Modelle für das Maß eine ähnliche Beziehung zwischen Input- und Outputvariablen aufweisen müssen. Nachdem die Outputvariablen trotz derselben Inputvariablen schwanken, sind sie für den Zweck ungeeignet. Somit hat sich schlussendlich das statistische Maß des Konfidenzintervalls am meisten für die Bestimmung der Ähnlichkeit der Ergebnisse als fähig erwiesen. Für die Berechnung des Konfidenzintervalls muss zuvor die Simulation 100-mal pro Plattform über einen Simulationszeitraum von 100 Jahren mittels Monte-Carlo-Simulation ausgeleitet werden. Die Stichproben müssen aus dem gleichen Satz an experimentellen Szenarien, die mit denselben Werten initialisiert wurden, stammen (siehe Kapitel 2.2). Am Ende der Datenausleitung werden die Ergebnisse auf Ähnlichkeit analysiert. In

Abbildung 3 ist die jeweilige Start- und Endsituation der Simulation in NetLogo (rechte Grafiken) und GAMA (linke Grafiken) dargestellt.

Abbildung 3: Start- und Endsituation nach 100 Simulationsjahren (links: GAMA, rechts: NetLogo)



Die Ergebnisanalyse von plattform- und sprachübergreifenden Replikationen zeigt gegenüber Replikationen gängiger AB-Plattformen aufgrund ihrer höheren Komplexität eine intensivere Tätigkeit (Donkin et al., 2017). Um festzustellen, inwiefern die Simulationsergebnisse aus NetLogo und GAMA

variieren, werden im ersten Analyseschritt ihre numerischen Daten sowie distributive Verteilung untersucht. Hierzu wird der Mittelwert als statistischer Lageparameter pro Simulation (Run) bestimmt, um festzustellen, wo sich im Durschnitt die Datenpunkte der Simulationsmenge befinden (Benninghaus, 2007). Durch den Vergleich der zentralen Lagen der Stichproben kann ermittelt werden, wie stark sich die Ergebnisse hinsichtlich ihrer Verteilung ähneln oder unterscheiden (Benninghaus, 2007). Vor dem eigentlichen Analyseschritt sind jedoch die Daten auf Normalverteilung zu testen (Bleymüller et al., 2020):

H0 = Die NetLogo/GAMA Simulationsergebnisse für 100 Runs sind normalverteilt.

HA = Die NetLogo/GAMA Simulationsergebnisse für 100 Runs sind NICHT normalverteilt.

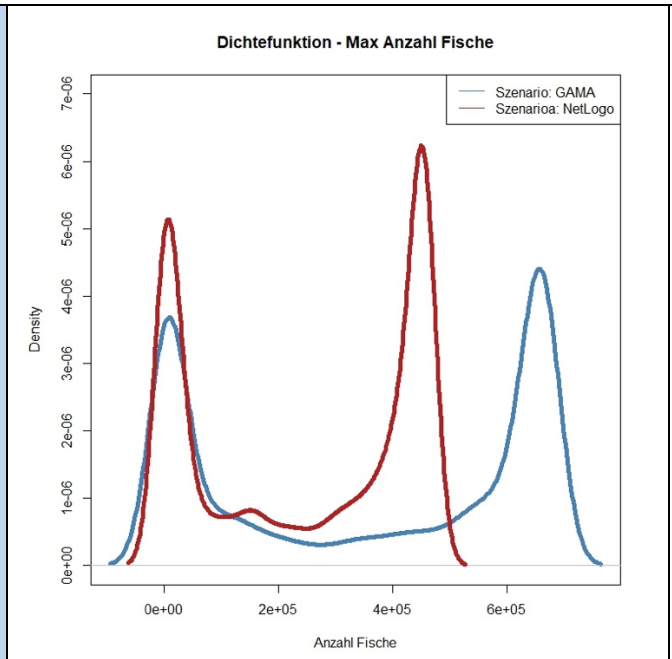
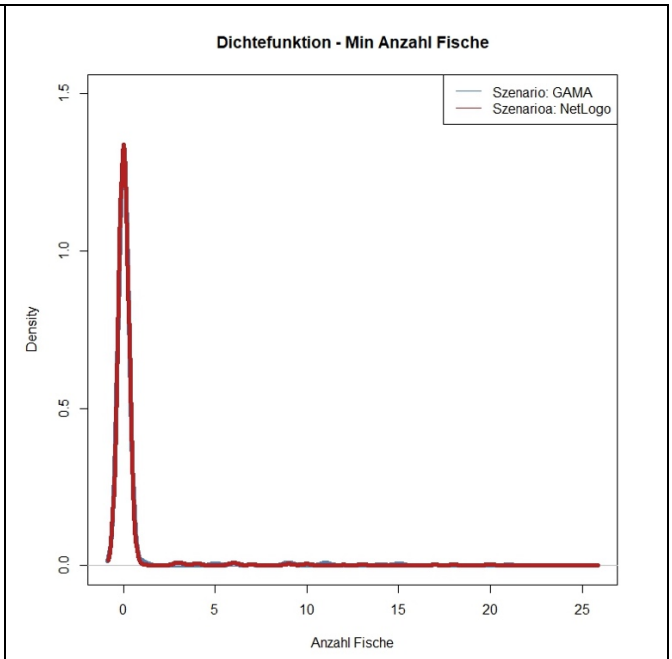
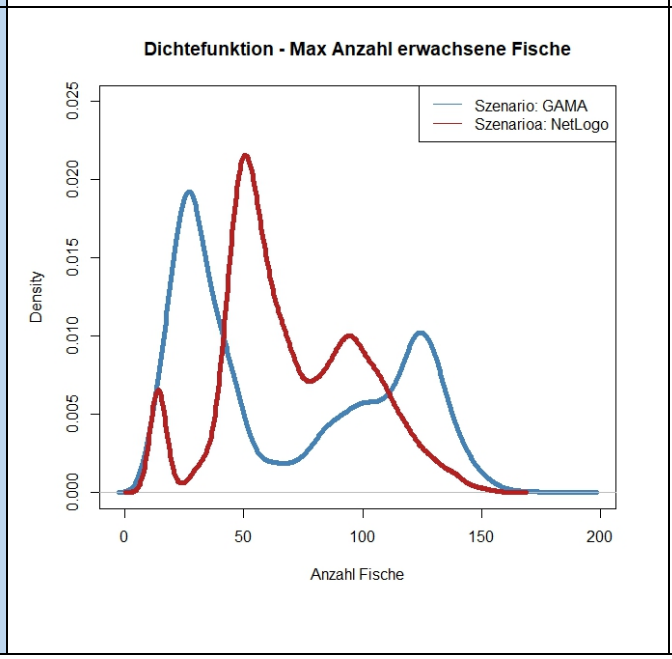
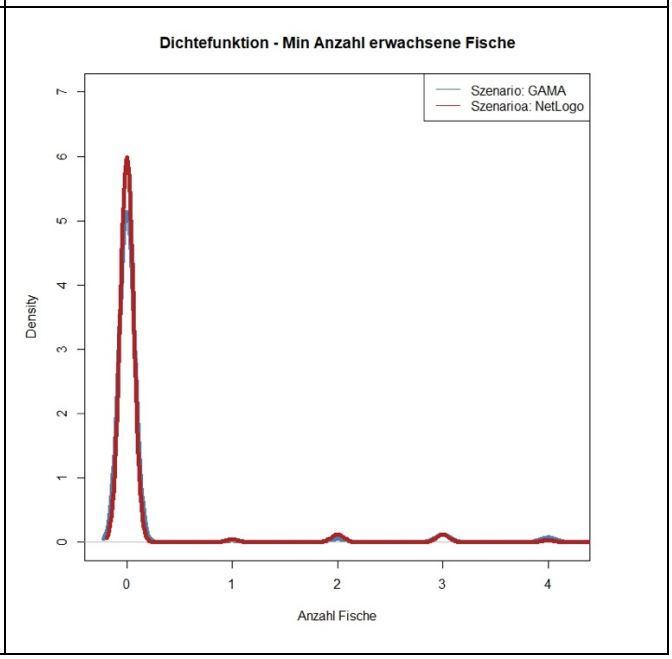
Die Dichtefunktion ist eine Wahrscheinlichkeitsverteilung, die mittels einer Kurve anzeigt, wo sich die Werte (Mittelwerte) der Zufallsvariablen am häufigsten sammeln (Planing, 2021). Liegt der Verlauf der Messwerte der Dichtefunktion annähernd zu einer theoretischen Normalverteilung (= Verteilung weist einen symmetrischen Verlauf auf), so spricht man von einem normalverteilten Merkmal, über dessen Intervalle [-1; +1], [-2, 2] und [-3, 3] jeweils ungefähr 68%, 95% und 99,7% der gesamten Verteilungsfläche liegen (Kähler, 2013, Seistock et al., 2020). Große Abweichungen des Linienverlaufs zur Normalverteilungskurve sind ein Indikator dafür, dass keine Normalverteilung vorliegt. In Tabelle 5 sind die Ergebnisse der Dichtefunktion für alle acht Prüfgrößen angeführt. Die Ergebnisse der Messwerte der Dichtefunktion für Netlogo sind in der Linienfarbe Rot und jene für GAMA in der Linienfarbe Blau abgebildet. Die Dichtefunktion und die Ermittlung des p-Wertes (Du Prel et al., 2009) bestimmt je nach Merkmalsausprägung, ob die Nullhypothese angenommen wird (Bender and Lange, 2001a). Der p-Wert ist ein statistisches Maß, der das wahrscheinliche Ausmaß der Evidenz gegen die Nullhypothese wiedergibt (Du Prel et al., 2009). Kleine p-Werte stellen eine starke Evidenz dar, wo die beobachtbaren Unterschiede statistisch signifikant (p-Wert < Signifikanzniveau α = 5%) sind (Du Prel et al., 2009, Lange and Bender, 2007).

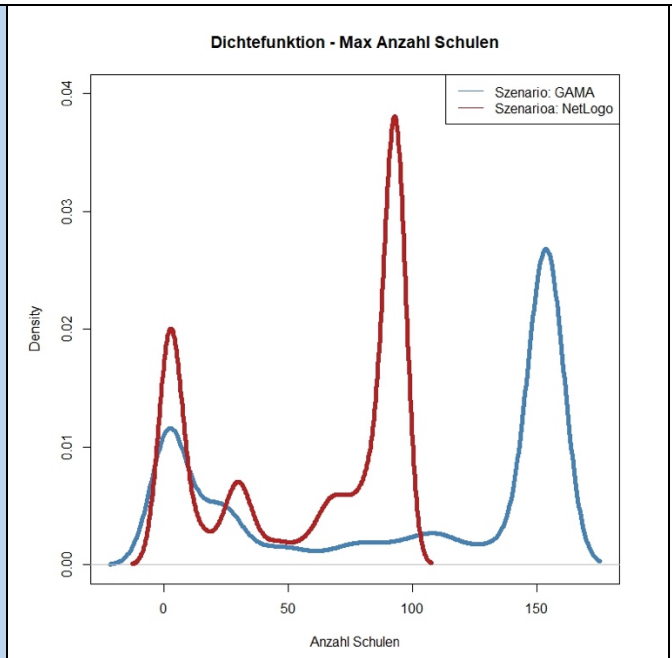
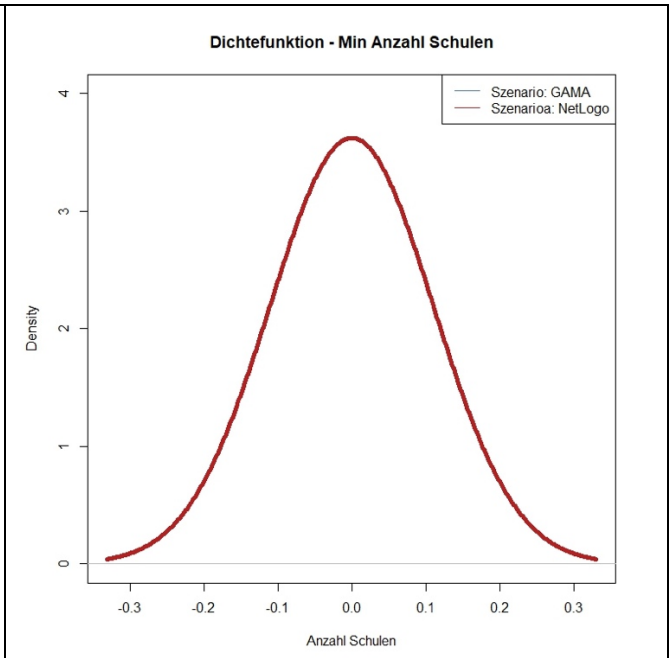
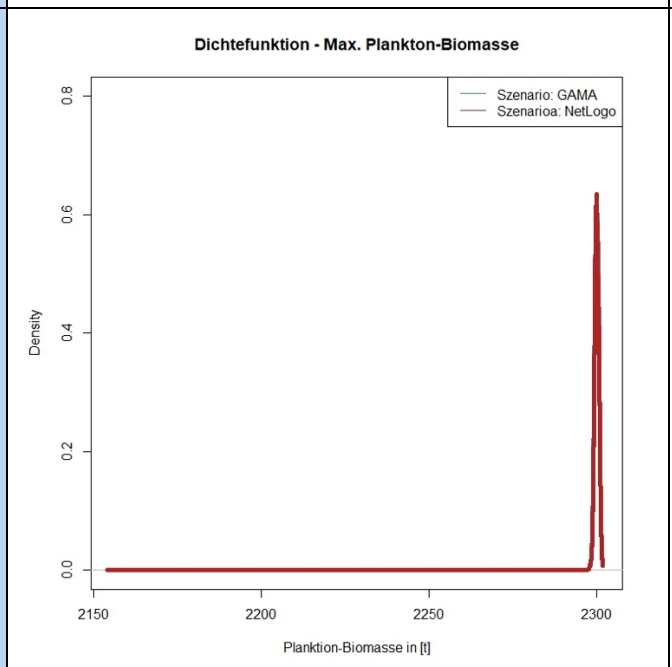
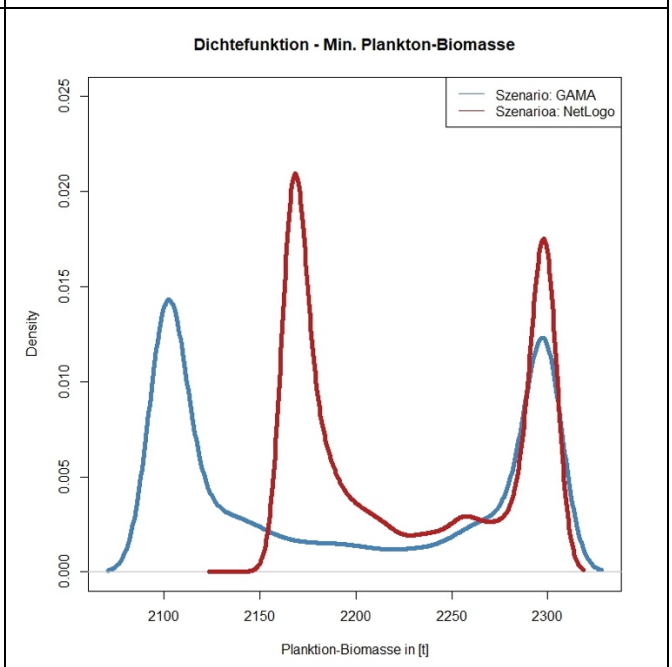
Da das Fisch-Plankton-Modell eine große Datenmenge produziert (Rand and Wilensky, 2006) ist es erforderlich, Schwerpunkte anhand denen der Erfolg der Replikation festgestellt wird, festzulegen. Als empirische Prüfgrößen werden folgende Parameter der Grundgesamtheit festgelegt, anhand derer die Dichtefunktion und visuelle Überprüfung auf Normalverteilung neben anderen statistischen Tests (Müller and Denecke, 2013) durchgeführt wird:

1. Prüfgröße: Maximale Anzahl an Fische im Attersee.
2. Prüfgröße: Minimale Anzahl an Fische im Attersee.
3. Prüfgröße: Maximale Anzahl adulter Fische (> 4 Jahre) im Attersee.

4. Prüfgröße: Minimale Anzahl adulter Fische (> 4 Jahre) im Attersee.
5. Prüfgröße: Maximale Anzahl an Schulen im Attersee.
6. Prüfgröße: Minimale Anzahl an Schulen im Attersee.
7. Prüfgröße: Maximum an Plankton-Biomasse (in Tonnen) im Attersee.
8. Prüfgröße: Minimum an Plankton-Biomasse (in Tonnen) im Attersee.

Tabelle 5: Ergebnisse der Dichtefunktion

Prüfgröße 1+2:		
Prüfgröße 3+4:		

Prüfgröße 5+6:		
Prüfgröße 7+8:		

Die Ergebnisse der Dichtefunktionen aller acht Prüfgrößen zeigen deutlich, dass die Daten nicht normalverteilt sind. Die Dichtefunktion der maximalen Anzahl des gesamten Fischbestandes (Prüfgröße 1) illustriert, dass die Messwerte aus NetLogo und GAMA einen ähnlichen Verlauf aufweisen, jedoch in ihrer Größe versetzt sind. Die Linienverläufe der zweiten Prüfgröße sind hingegen deckungsgleich. Es kann daher die Annahme getroffen werden, dass die Mittelwerte der Prüfgröße 1 und 2 der beiden Plattformen NetLogo und GAMA in einem ähnlichen Bereich verdichtet auftreten. Dasselbe gilt für die Prüfgrößen 3 und 4. Anders sieht es für die Prüfgröße 5 aus, die die Entwicklung der maximalen Anzahl an Schulen im Attersee über einen Zeitraum von 100 Jahre abbildet. Hier oszilliert der Linienverlauf in NetLogo stärker gegenüber jenem in GAMA und erreicht einen maximalen Dichtewert von fast 0,04. Die

Linienverläufe der Prüfgröße zur minimalen Anzahl an Schulen unterscheiden sich in NetLogo komplett anders zu GAMA. Denn die Dichte in NetLogo liegt bei rund 3,6 und in GAMA ist keine Linie vorhanden, da der Durchschnittswert null ergibt. Die Linienverläufe für die Prüfgröße 7 sind in NetLogo und GAMA wieder deckungsgleich. Die Prüfgröße 8 zeigt wie die Prüfgröße 1 einen ähnlichen Linienverlauf, jedoch einen Versatz in der Höhe der erreichten Dichtewerte. Neben den Dichtefunktionen werden zusätzlich die p-Werte zur rechnerischen Bestimmung einer Normalverteilung mittels Welch-Test ermittelt. Der Welch-Test ist ein statistisches Verfahren zur Feststellung von signifikanten Unterschieden bei ungleichen Varianzen unabhängiger Stichproben (Seistock et al., 2020). Ergibt sich ein Signifikanzwert (p-Wert) von 5% (p=.050) oder geringer, handelt es sich um einen statistisch signifikanten Unterschied (Seistock et al., 2020). Die Ergebnisse des Welch-Test sind in Tabelle 6 angeführt.

Tabelle 6: Ergebnisse Welch-Test

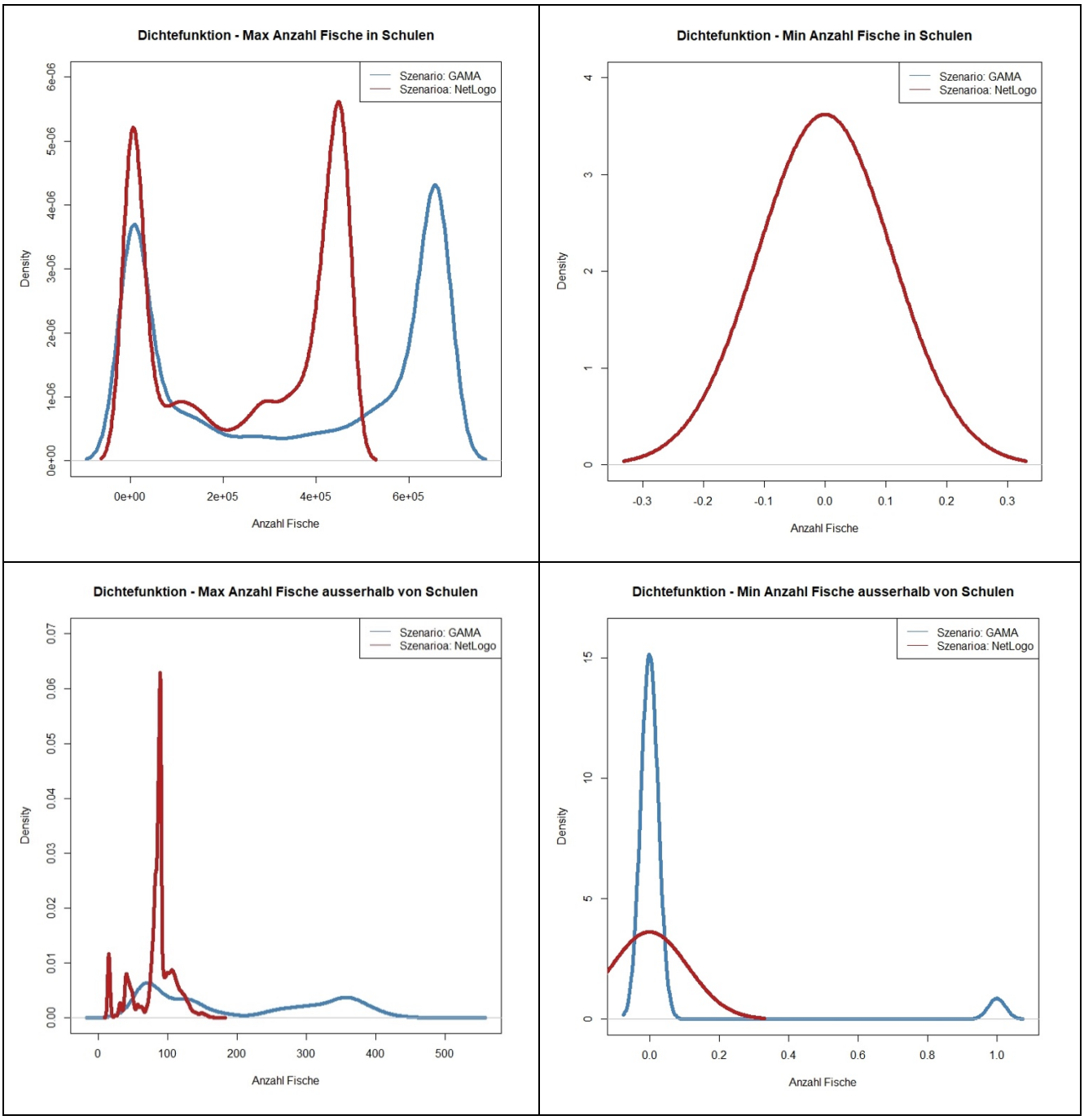
	Repliziertes Modell (GAMA)	Ausgangsmodell (NetLogo)		
Prüfgrößen	**MW**	**MW**	**Unterschied in %**	**p-Wert**
Max. Anzahl Fische	361236,20	251915,00	30,26	<2.2e-16
Min. Anzahl Fische	0,50	0,49	6,55	0,066
Max. adulter Fische	68,70	70,34	2,28	1.193e-08
Min. adulter Fische	0,15	0,12	15,75	5.966e-07
Max. Schulen	96,09	57,87	39,76	<2.2e-16
Min. Schulen	0,00	0,00	0,00	NA
Max. Plankton	2299,59	2299,59	0,40	0,926
Min. Plankton	2193,47	2226,96	1,52	<2.2e-16

Da alle p-Werte in Tabelle 5 < 0,05 sind, kann H0 verworfen und HA angenommen werden. Die Ergebnisse zeigen dasselbe Resultat wie die Dichtefunktionen, nämlich dass die Daten aller Prüfgrößen keine Normalverteilung aufweisen, außer für die Größen ‚Min. Anzahl Fische' und ‚Max. Plankton'. Der Trend, dass die Mittelwerte in GAMA höher sind als in NetLogo zieht sich über alle Prüfgrößen, wobei den höchsten prozentualen Unterschied die maximale Anzahl an Schulen mit 39% und den geringsten die maximale Planktonverfügbarkeit mit 0,2% aufweisen.

Da bei der Analyse der Dichtefunktionen die größten Abweichungen bei den Prüfgrößen 5 und 6 festgestellt wurden, sind Dichtefunktionen zu weiteren Prüfgrößen (maximale und minimale Anzahl an

Fischen in Schulen, maximale und minimale Anzahl an Fischen außerhalb von Schulen) ermittelt worden. Es handelt sich hierbei um Prüfgrößen, die die Fischpopulation in und außerhalb von Schulen abbilden. Die Ergebnisse sind in Tabelle 7 angeführt. Da auch hier die visuelle Betrachtung der Ergebnisse auf nicht normalverteilte Daten geschlossen werden kann, wurde auf die zusätzliche Berechnung des p-Wertes mittels Welch-Test abgesehen.

Tabelle 7: Ergebnisse der Dichtefunktion weiterer Prüfgrößen



Die Linienverläufe der Entwicklung der maximalen Anzahl an Fischen in Schulen ist in NetLogo und GAMA ähnlich, jedoch unterscheiden sie sich in ihrer Größe. Die Werte für die minimale Anzahl an Fische

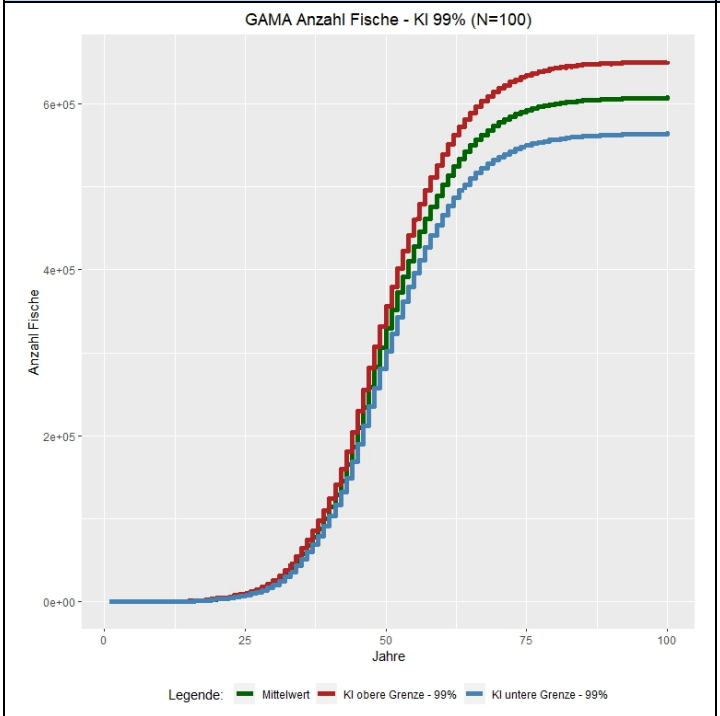
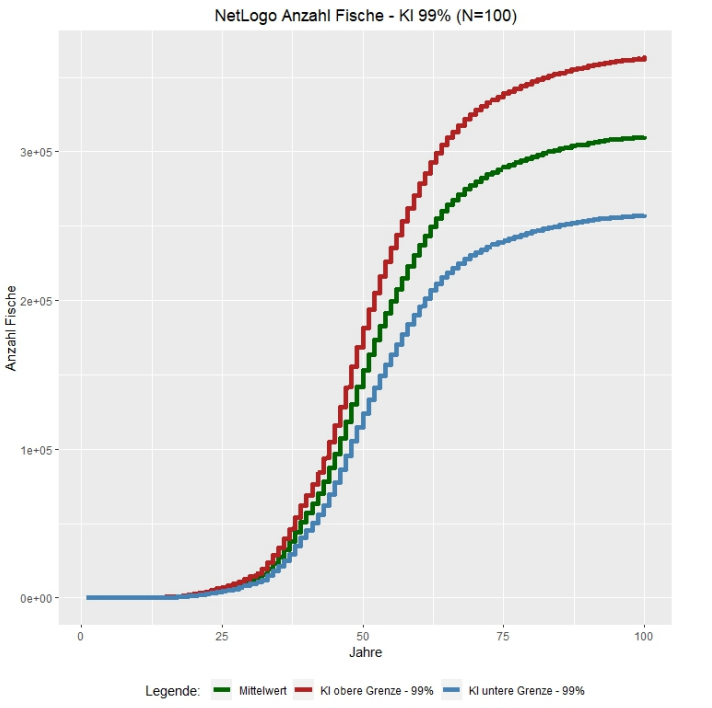
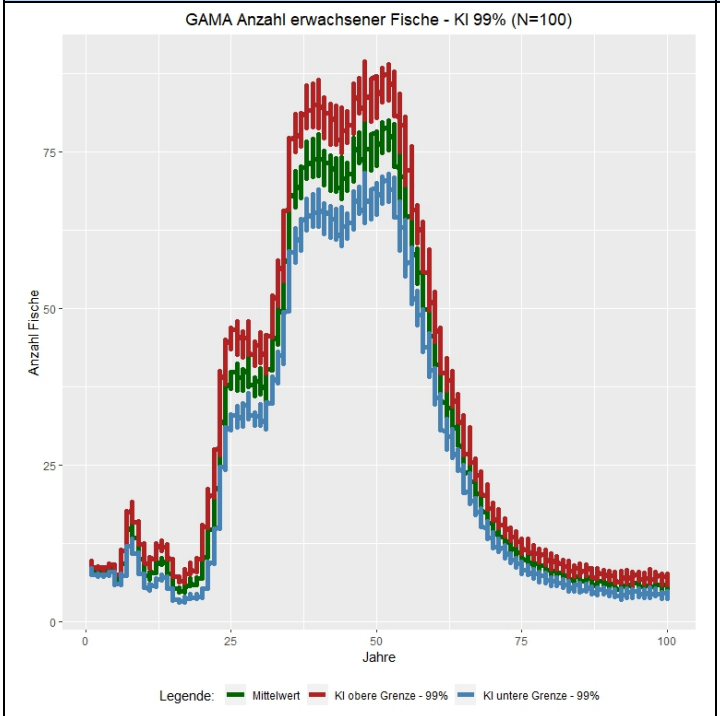
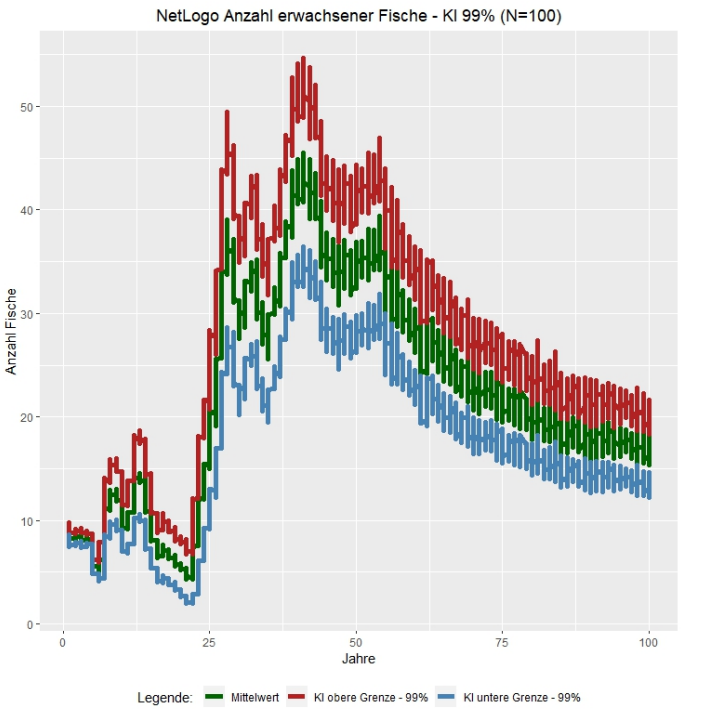
in Schulen zeigen hingegen einen komplett anderen Verlauf, denn die Linie für GAMA (Mittelwert = 0) ist im Gegensatz zu jener in NetLogo nicht vorhanden. Die signifikant größten grafischen Unterschiede treten in der Prüfgröße der maximalen und minimalen Anzahl der Fische außerhalb von Schulen auf. Sie unterscheiden sich hier nicht nur in ihrem Linienverlauf, sondern auch in ihren Dichtewerten. Es kann daher die Aussage getätigt werden, dass die Entwicklung der Fische außerhalb von Schulen ein anderes Verhalten in NetLogo aufweisen als in GAMA. Auf mögliche Ursachen, die die signifikanten Unterschiede hervorrufen, wird am Ende des Kapitels sowie im Kapitel 5 näher eingegangen.

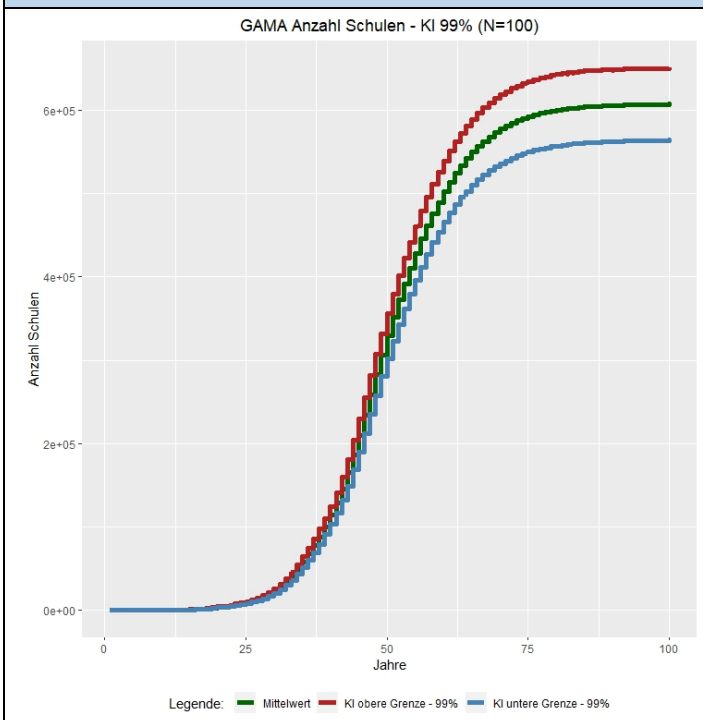
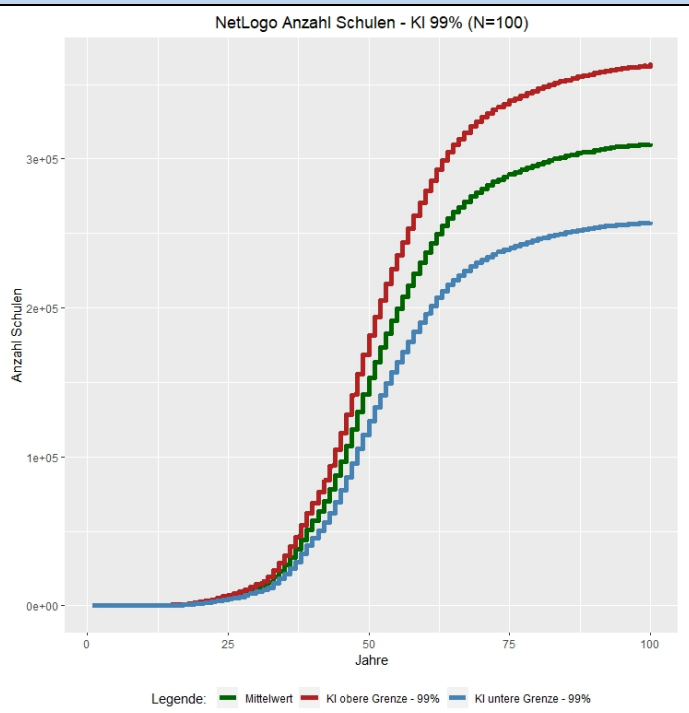
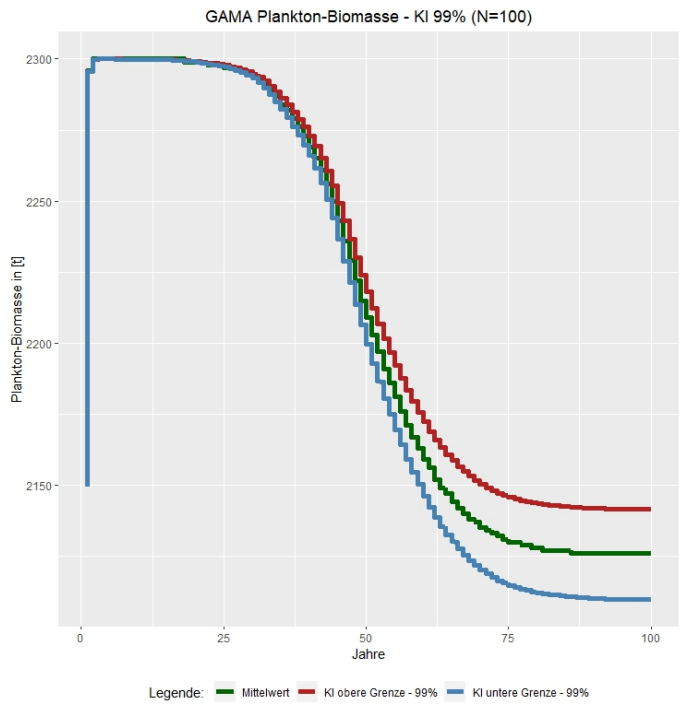
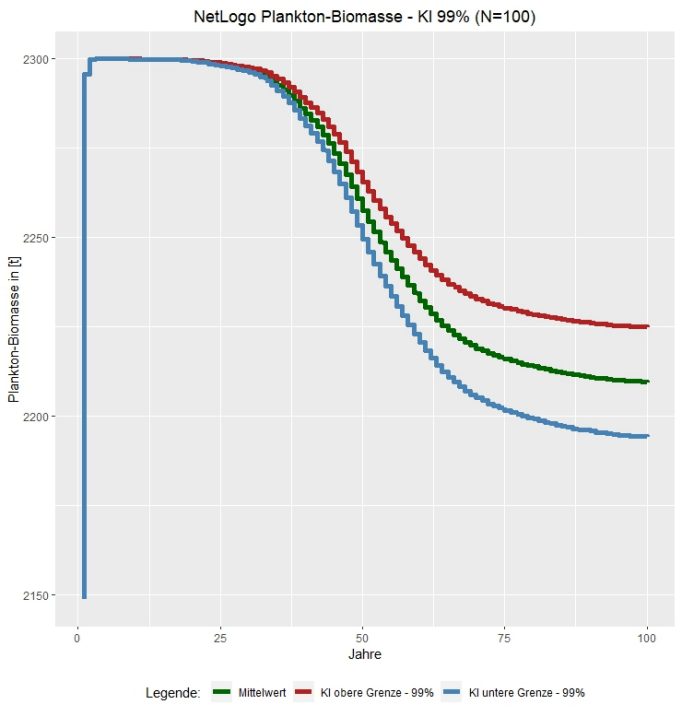
Nach abgeschlossener Prüfung auf Normalverteilung der Ergebnisdaten wird im nächsten Schritt das Konfidenzintervall bestimmt. Obwohl die Resultate aller acht Prüfgrößen gezeigt haben, dass keine Normalverteilung vorliegt, ist die Ermittlung des Konfidenzintervalls aufgrund des sogenannten Grenzwerttheorem zulässig. Das zentrale Grenzwerttheorem besagt, „(…) dass unabhängig von der Verteilung der Daten in der Grundgesamtheit die Stichprobenkennwerteverteilung der Mittelwerte mit wachsendem Stichprobenumfang in eine Normalverteilung übergeht.“ (Planing, 2021) Eine genügend große Stichprobe bedeutet in der Regel N > 30 (Planing, 2021). Nachdem die Stichprobengröße in der Masterarbeit mit N=100 festgelegt ist, ist die Annahme des zentralen Grenzwerttheorem zulässig. Das KI wurde für folgende Prüfgrößen konstruiert:

- gesamter Fischbestand im Attersee,
- gesamter adulter Fischbestsand (> 4 Jahre) im Attersee,
- gesamte Anzahl an Schulen im Attersee,
- gesamter Plankton-Biomassebestand in Tonnen im Attersee

Auch hier ist wie bei den anderen Prüfgrößen zuvor der Simulationszeitraum mit 100 Jahre festgelegt. Der Ausgangspunkt des 99%-KI ist eine erwartungstreue Punktschätzung der Stichprobenmittelwerte für die Kennzahlen der Prüfgrößen (Planing, 2021). Die Aufgabe des 99%-KI ist, den Bereich der sogenannten wahren Mittelwerte in der Normalverteilung zu bestimmen, in dem 99% der Mittelwerte liegen. Der z-Wert wird mit z=2,58 festgelegt (Planing, 2021), weshalb der 99%-KI einen Bereich von -2,58 und +2,58 abdeckt. Mit einer Wahrscheinlichkeit von 99% liegen somit die Mittelwerte der Grundgesamtheit zwischen den KI definierten oberen und unteren Grenzen (Planing, 2021). In Tabelle 8 sind die Ergebnisse der KI-Berechnungen für GAMA und NetLogo dargestellt.

Tabelle 8: KI-Ergebnisse

Prüfgröße: Gesamter Fischbestand (innerhalb und außerhalb von Schulen)	
GAMA	**NetLogo**
	
Prüfgröße: Gesamter erwachsener Fischbestand (> 4 Jahre) ausserhalb von Schulen	
GAMA	**NetLogo**
	

Prüfgröße: Anzahl von Schulen	
GAMA	NetLogo
	
Prüfgröße: Plankton	
GAMA	**NetLogo**
	

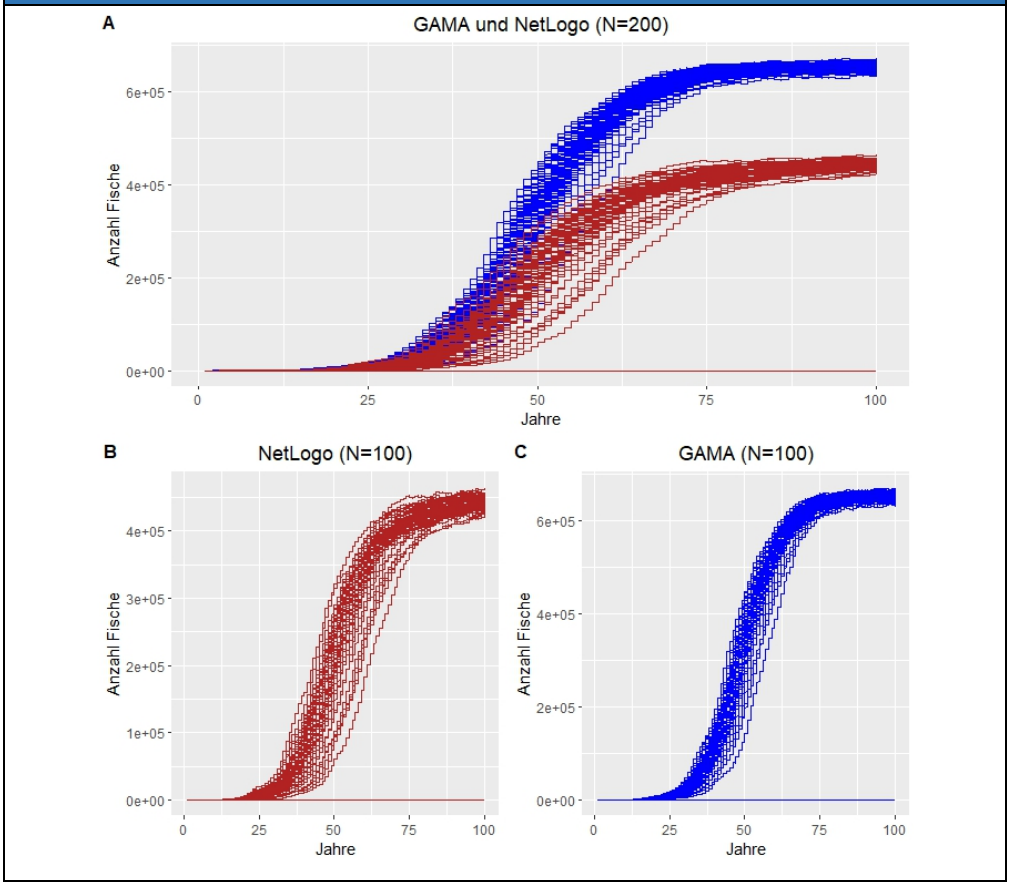
Die KI-Ergebnisse zeigen, dass alle Mittelwerte der jeweiligen Prüfgrößen sich innerhalb der ermittelten KI oberen und unteren Grenze befinden. Zusammengefasst kann somit festgestellt werden, dass die

Mittelwerte der jeweiligen Prüfgrößen in NetLogo sowie in GAMA hinsichtlich ihrer probabilistischen Variation der Runs innerhalb ihres wahren Mittelwertbereichs liegen.

Im nächsten Analyseschritt werden die Ergebnisse der Monte-Carlo-Simulation visuell aufbereitet, um festzustellen, wie stark die Simulationsergebnisse in NetLogo bzw. GAMA streuen, wenn die Simulationen jeweils mehrfach ausgeführt (mehrere Runs) werden. Die Simulationsergebnisse aus NetLogo sind in der Farbe Rot und jene aus GAMA in der Farbe Blau dargestellt.

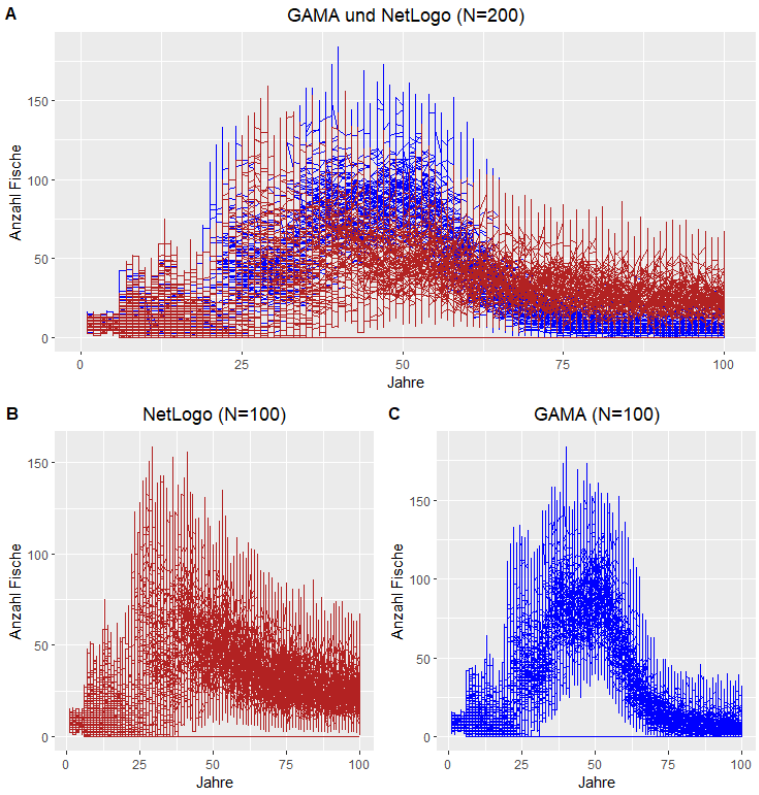
Sowohl das ursprüngliche wie auch das replizierte Modell zeigen eine Reihe von emergenten Eigenschaften, die identifiziert werden können. Die Fische zeigen eine Tendenz zur Selbstorganisation, d.h. es bildet sich eine spontane Ordnung („Gleichgewicht“) heraus. Die Ordnung ist durch ein anhaltendes Sterben und geboren werden von Fischen sowie der zur Verfügung stehenden Plankton-Biomasse gekennzeichnet. Die Ordnung zeigt gelegentlich bemerkenswerte und unregelmäßige Auf- und Abschwünge. Eine Depression setzt aufgrund der Limitierung der Plankton-Biomasse ab einem Simulationszeitraum von > 50 Jahren ein. Die wichtigsten Ergebnisse sind in Tabelle 9 zusammengefasst.

Tabelle 9: Ergebnisse Monte-Carlo-Simulation (N=100)

Prüfgröße: Gesamter Fischbestand (innerhalb und außerhalb von Schulen)


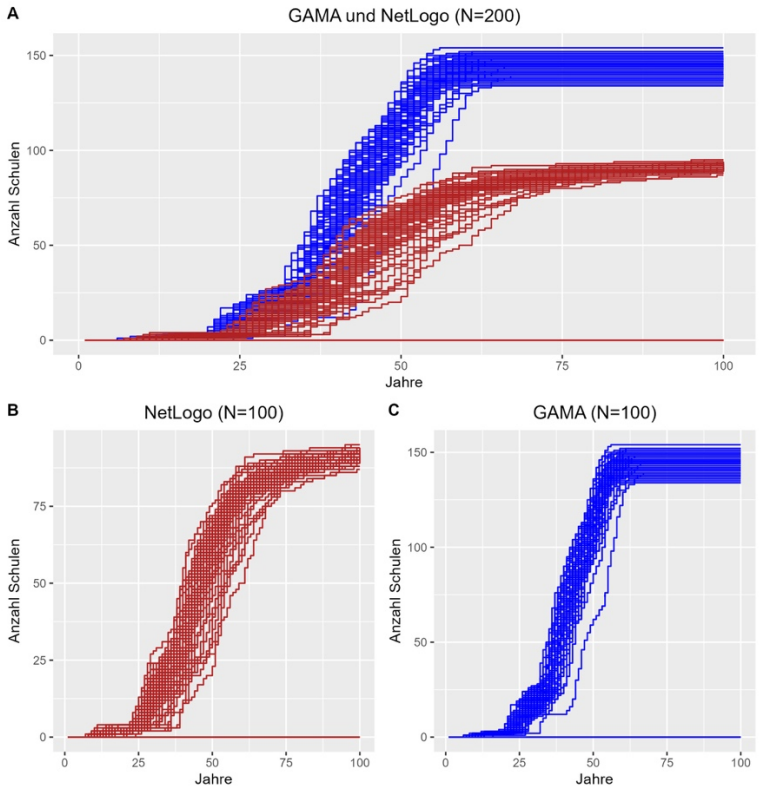
Die Ergebnistrajektorien aus NetLogo und GAMA sind in ihrem Verlauf gleich, jedoch unterscheiden sie sich in ihrer Größe. Während in NetLogo nach 100 Jahren der Fischbestand bei rund 450.000 liegt so sind es in GAMA rund 650.000.

Prüfgröße: Gesamter erwachsener Fischbestand (> 4 Jahre) außerhalb von Schulen



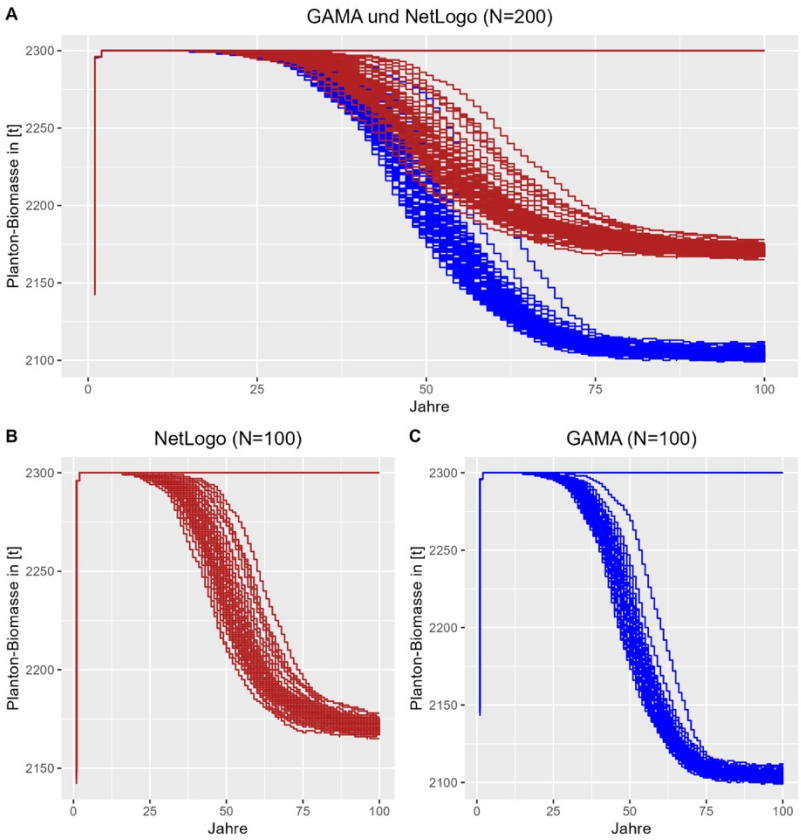
Die Ergebnistrajektorien unterscheiden sich stark. Die probabilistische Streuung der Modelle zeigt, dass die Entwicklung der Fische außerhalb von Schulen einen unterschiedlichen Einfluss auf die Modelle nimmt.

Prüfgröße: Anzahl Schulen



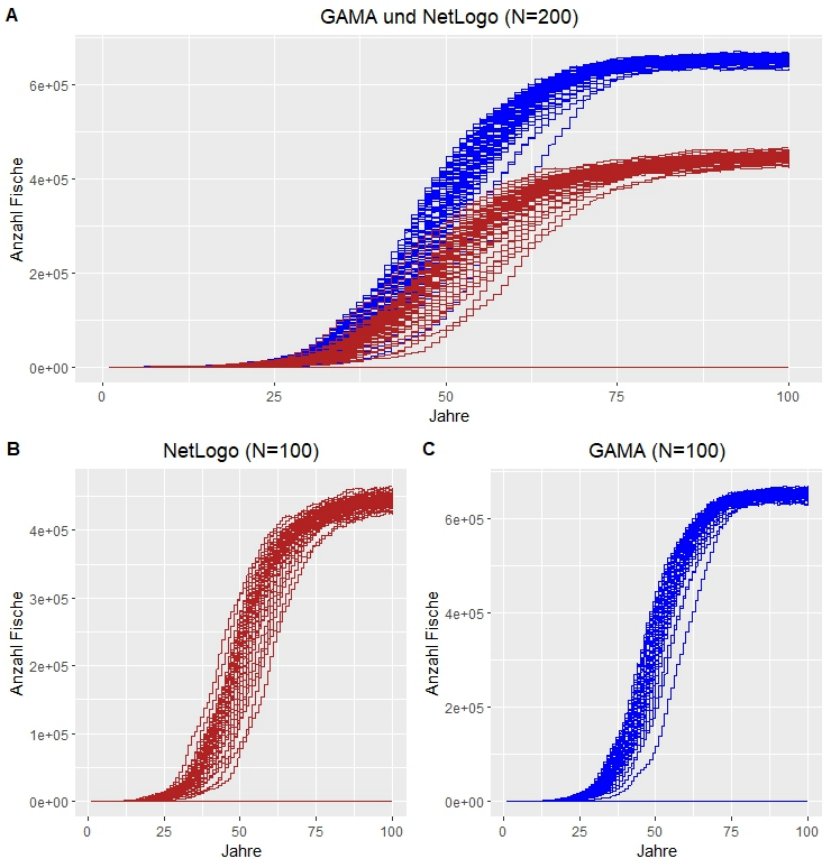
Die Ergebnistrajektorien zeigen einen leichten Unterschied zwischen den Runs. In GAMA bilden sich Schulen zu einem früheren Simulationszeitpunkt als in NetLogo. Ab ca. 25 Jahren pendelt sich die Schulbildung in beiden Plattformen auf ein ähnliches Maß, mit dem Unterschied, dass nach 100 Jahren, in NetLogo rund 100 Schulen und in GAMA rund 150 Schulen sind. Das ergibt eine Differenz von rund 50 Schulen.

Prüfgröße: Plankton-Biomasse



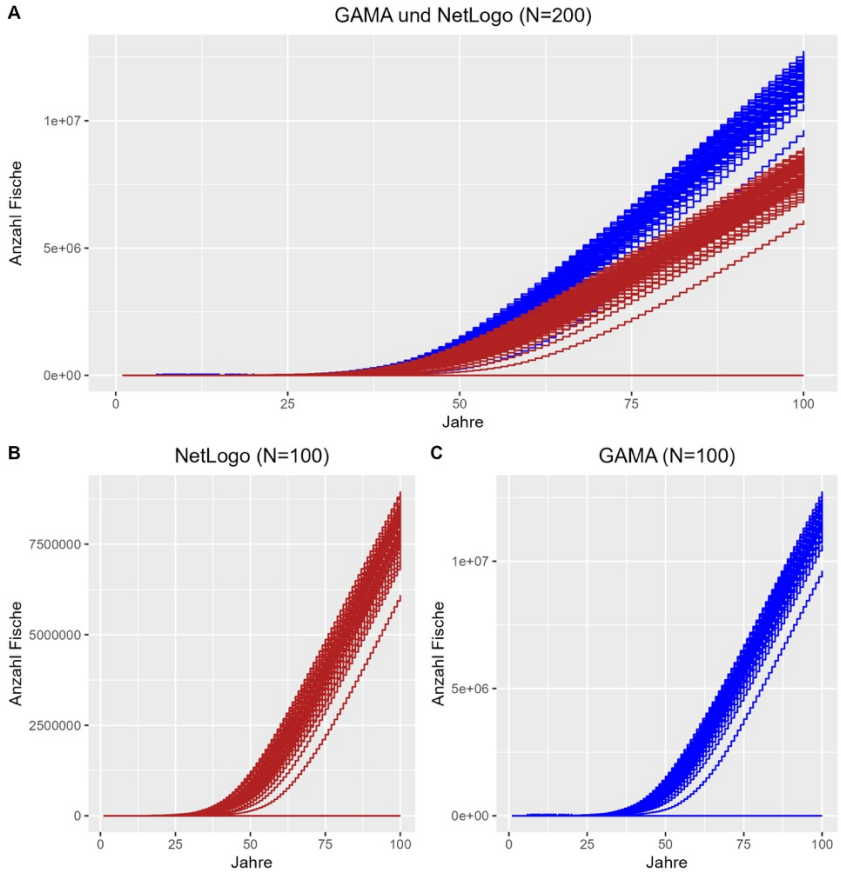
Der Verlauf der Plankton-Biomasse ist in beiden Plattformen ähnlich. In GAMA nimmt ab ca. 60 Jahren die Plankton-Biomasse im Vergleich zu NetLogo stärker ab, resultierend aus der höheren Anzahl an Fischen, die die Plankton-Biomasse als Nahrung aufnehmen.

Prüfgröße: Fische innerhalb von Schulen



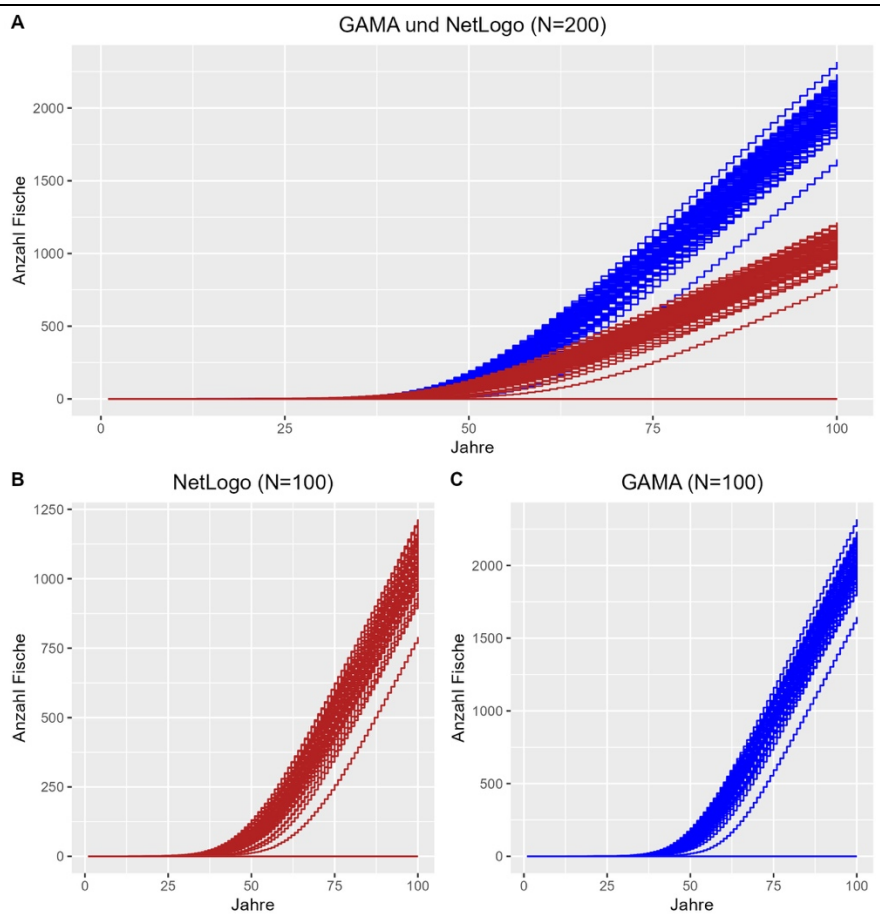
Die Entwicklung der Fische in Schulen weisen in NetLogo und GAMA einen ähnlichen Verlauf auf. Sie unterscheiden sich lediglich in ihrer Endgröße mit einer Differenz von rund 200.000.

Prüfgröße: Reproduktion in Schulen



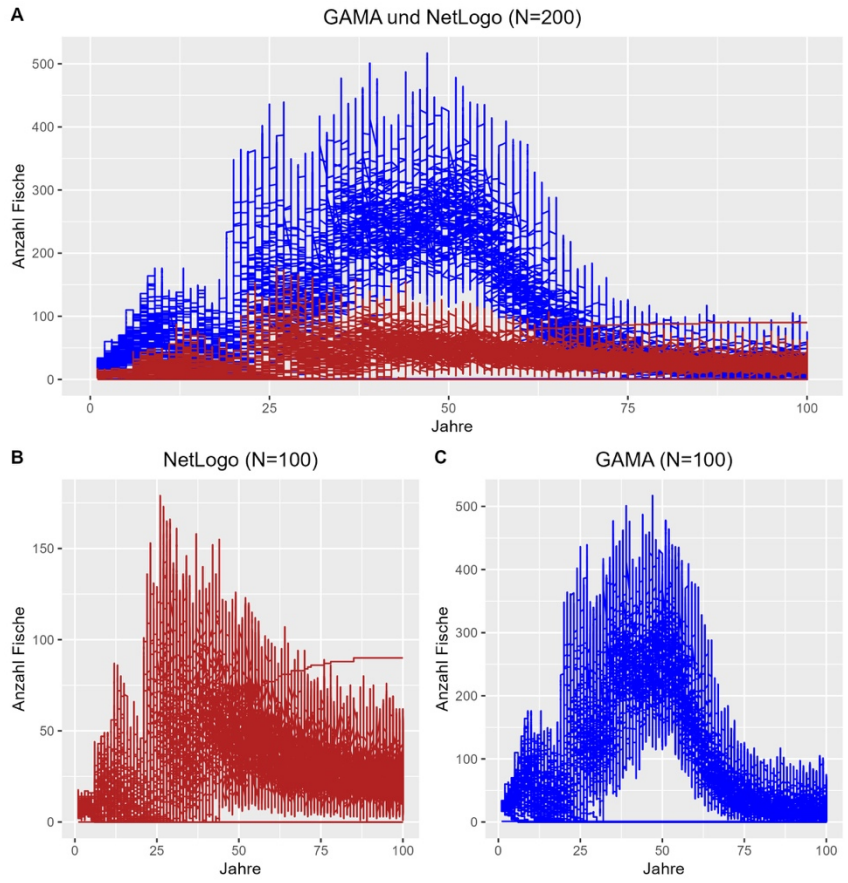
Die Reproduktion innerhalb von Schulen ist in ihrem Verlauf in NetLogo zu jenen in GAMA sehr ähnlich. Sie unterscheiden sich lediglich in ihrer Anzahl. Über die gesamte Simulationslaufzeit von 100 Jahren sind in NetLogo rund 80.000.000 und in GAMA 10.000.000 Fische in Schulen geboren worden.

Prüfgröße: Sterblichkeit in Schulen



Die Sterblichkeit von Fischen in Schulen ist in NeteLogo geringer als in GAMA. Nach 100 Jahren sind in NetLogo rund 1.250 Fische gestorben, in GAMA rund 2.500. Obowhl in NetLogo mehr Fische in Schulen geboren werden, sterben weniger im Vergleich zu GAMA.

Prüfgröße: Fische außerhalb von Schulen



Die Entwicklung der Fische außerhalb von Schulen zeigt einen deutlichen Unterschied in den Simulationstrajektorien von NetLogo und GAMA. Sie unterscheiden sich nicht nur in ihrer Anzahl, sondern auch in ihrem Verlauf. Der Linienverlauf ist von deutlichen Zu- und Abnahmen der Fische gekennzeichnet. Die Schwankungen ergeben sich durch die Reproduktion und die Sterblichkeit.

Zusammenfassend für das Ergebniskapitel kann festgehalten werden, dass bis auf die Entwicklung von adulten Fischen außerhalb von Schulen, die Simulationstrajektorien auf eine entsprechende Replikation des Ausgangsmodells mit dem Zielmodell schließen lassen. Ein ähnliches Bild ist bei der Studie von Wilensky (2007) zu beobachten, wo das Zielmodell dem Ausgangsmodell strukturell ähnlich ist, jedoch Unterschiede in den Ergebnissen auftreten, die aufgrund der Beschreibung des konzeptionellen Modells nicht erwartet und mit dem ursprünglichen Modell nicht erzielt wurden. Die Abweichungen in den Replikations- zu den Modellergebnissen des Fisch-Plankton-Modells werden der zufallsbedingten Verteilung der Agenten im Raum sowie ihrer Zusammensetzung hinsichtlich Geschlecht und Alter sowie der spezifischen Syntax zugeschrieben.

Um festzustellen, weshalb sich die adulten Fische außerhalb von Schulen in GAMA anders entwickeln, wurde auf potenzielle Ursachen geprüft. Eine dieser Prüfmethoden waren die Plausibilitätsprüfungen. Es wurden dieselben Plausibilitätsprüfungen in GAMA sowie in NetLogo absolviert. Mittels der

Plausibilitätsprüfungen soll festgestellt werden, ob und wie plausibel (glaubwürdig) die Daten zustande gekommen sind. Um bei beiden Modellen zu gewährleisten, dass keine zufälligen Ereignisse die Ergebnisse verfälschen, wurden im ersten Schritt alle zufälligen Faktoren deaktiviert und durch absolute Werte ersetzt:

- Die initiale Aufteilung der Plankton-Biomasse von 2.300 Tonnen erfolgt ursprünglich für 2.000 Tonnen gleichmäßig und 300 Tonnen randomisiert auf alle See-Zellen. Das wurde für die Plausibilitätsprüfung geändert, indem die 2.300 Tonnen auf alle See-Zellen gleichmäßig verteilt wurden.
- Die Planktondiffusion wurde deaktiviert.
- Die initiale geschlechtsspezifische Verteilung der Fische wurde von der zufälligen Distribution an weiblichen und männlichen Fischen auf eine Gleichverteilung von 50% weibliche und 50% männliche geändert.
- Die initiale altersspezifische Verteilung der Fische wurde vom zufälligen Altersverhältnis auf ein gleiches Verhältnis aller Altersgruppen (age <- 0.24*1) gesetzt.
- Die errechneten Werte der Starvation- und Reproduktions-Rate, welche von den jeweiligen fisch- und planktonspezifischen Gegebenheiten abhängig sind, wurden durch fixe Werte ersetzt.

Im nächsten Schritt wurde die Länge des Schul-Arrays sowie dessen Einträge auf Korrektheit und Performanz ermittelt. Dazu wurde die length Eigenschaft des Arrays bei jedem Step/Tick mit dem write-Befehl ausgegeben und kontrolliert, ob der Index nicht unter 2.190 (=365*6) liegt. Um zu prüfen, ob die Array-Inhalte sich in einem bestimmten Wertebereich entwickeln, werden die Inhalte mittels Konsole ausgegeben. Es konnte somit festgestellt werden, dass sich die Elemente des Arrays hinsichtlich ihrer Reproduktions- und Sterblichkeitsraten in GAMA und in NetLogo über den gesamten Simulationszeitraum sich in einem ähnlichen Wertebereich entwickeln. Für den nächsten Analyseschritt wurden die Schulen mittels Label beschriftet. Dadurch kann schnell festgestellt werden, wie viel Fische eine Schule enthält. Auch das Emittieren kann damit überprüft werden, indem, wenn eine Schule eine maximale Größe von 5.000 Fischen erreicht, dessen Label-Angabe in ihrem angezeigten Wert mit dem darauffolgenden Step sinkt. Anschließend wird die jeweilige Anzahl der emittierten Fische mittels write-Befehl ausgegeben. Stimmt die emittierte Anzahl mit der neuen Label-Angabe überein, ist das Emittieren erfolgreich durchgeführt worden. Als nächstes wird überprüft, ob die Schulen ihr Schwimmverhalten nach der planktonreichsten Zelle ausrichten und das Zusammentreffen mit anderen Schulen vermeiden. Die Funktionsprüfung wurde mittels Sichtprüfung durchgeführt. Des Weiteren wurde überprüft, ob auch die Fische, die in Schulen aufgenommen wurden, aus dem tatsächlichen Fischbestand außerhalb von Schulen entnommen wurden. Liniendiagramme als visuelle

Informationsdarstellungen wurden für die Anzahl an Fische außerhalb von Schulen und für die Anzahl an Schulen realisiert. Anhand der visuell festgestellten Zunahme bei der Anzahl an Schulen konnte eine entsprechende Abnahme bei der Anzahl an Fischen außerhalb von Schulen festgestellt werden. Liniendiagramme eigenen sich für den Vergleich von Datensätzen und zeigen ein kombiniertes Verhalten von Schulentwicklung und der Bestandsreduzierung der Fische.

Zuletzt wird das Plankton hinsichtlich seiner Entwicklung und Reduktion analysiert. Die Planktonentwicklung wird mittels Liniendiagramm ausgegeben. Zusätzlich werden die Werte der Planktonvariable (ist Planktonwert pro Zelle) in der Konsolenausgabe ausgegeben. Die Werte der Planktonvariable aus GAMA werden mit jenen aus NetLogo verglichen. Sie zeigen denselben Wertebereich an. Die Summen der Ergebnisse pro Zelle ergeben auch den Wert 2.300 kg gesamt in beiden Plattformen. Damit kann sichergestellt werden, dass die Entwicklung des Planktons hinsichtlich des logistischen Wachstums und der Reduktion durch die Nahrungsaufnahme der Fische ähnlich bzw. gleich jenem im ODD-Protokoll beschriebenen Umstand ist. Die für die Plausibilitätsprüfung deaktivierte Plankton-Diffusion von 10% des Planktonbestandes in die nächsten acht Nachbarzellen wurde wieder aktiviert. Trotzdem zeigt die Verteilung des Planktons einen ähnlichen Verlauf, wie bei der Deaktivierung der Diffusion. Es wird daher angenommen, dass die Diffusion keinen signifikanten Einfluss auf die Entwicklung der Fischpopulation nimmt. Am Ende der Plausibilitätsprüfungen werden die Fisch-Plankton-Modelle jeweils auf ihren ursprünglichen Zustand zurückgestellt. Die Erkenntnisse aus den Plausibilitätsprüfungen sind in der Tabelle 10 zusammengefasst.

Tabelle 10: Vergleich der Umsetzung in GAMA mit jener in NetLogo

Funktion	NetLogo	GAMA	Unterschied
Fische erstellen	• Initial 25 Stück. • Geschlecht und Alter randomisiert. • Fische altern pro step (Tag). • Fische bewegen sich 20 m / step (Tag).	• Initial 25 Stück. • Geschlecht und Alter randomisiert. • Fische altern pro step (Tag). • Fische bewegen sich 20 m / step (Tag).	Keine.
Fische schwärmen (flocking)	Flocking Methode von Gudrun Wallentin.	Flocking Methode aus der NetLogo-Library.	Ab einer bestimmten Anzahl an Fische in Schwärme nimmt die Rechengeschwindigkeit in GAMA ab, bis hin zum Programmabsturz.
Fische schwimmen	20 m pro Tag.	20 m pro Tag.	Keine.

Energiehaushalt Fische	Update-energy erfolgt nach den Formeln von Sibly (2013).	Update-energy erfolgt nach den Formeln von Sibly (2013).	Keine.
Fische reproduzieren sich	Jährliche Reproduktion pro weiblichem Fisch von 5 Nachkommen → 5/365. Voraussetzung: • 2 Flockmates (weiblich, männlich). • Älter als 4 Jahre.	Jährliche Reproduktion pro weiblichem Fisch von 5 Nachkommen → 5/365. Voraussetzung: • 2 Flockmates (weiblich, männlich). • Älter als 4 Jahre.	Keine.
Schulen - Aufnahme und Identifikation	• Aufnahme von Fischen in eine Schule ab einem Sichtbarkeitsradius von 700m. • Ersetzen von einzelnen Fischen durch einen Schul-Agenten. • Aufnahme der einzelnen Fische in eine Fischbestandsliste, geteilt in juvenile und erwachsene Fische. • Fischbestandsliste ist ein list-Attribut. • Eingliederung der einzelnen Fische als Position in der Liste → chronologische Reihung nach Alter; Position 0 = ältester Fisch, Position n = jüngster Fisch. • Kompletter Fischbestand in der Schule = juvenile Fische + erwachsene Fische. • Nach erfolgreicher Aufnahme eines einzelnen Fisches in die Schule wird der Fisch-Agent gelöscht.	• Aufnahme von Fischen in eine Schule ab einem Sichtbarkeitsradius von 700m. • Ersetzen von einzelnen Fischen durch einen Schul-Agenten. • Aufnahme der einzelnen Fische in eine Fischbestandsliste, geteilt in juvenile und erwachsene Fische. • Fischbestandsliste ist ein list-Attribut. • Eingliederung der einzelnen Fische als Position in der Liste → chronologische Reihung nach Alter; Position 0 = jüngster Fisch, Position n = ältester Fisch. • Kompletter Fischbestand in der Schule = juvenile Fische + erwachsene Fische. • Nach erfolgreicher Aufnahme eines einzelnen Fisches in die Schule wird der Fisch-Agent gelöscht.	Sortierung der juvenilen und adulten Fische ist in NetLogo in umgekehrter Reihenfolge zu GAMA.

See – Skalierung	Skalierungsfaktor in NetLogo relevant, da das Original .asc Attersee File eine Dimensionierung von 10x10 Grid-Size aufweist und die Zellen aber 200x200 lt. Modellparameter aufweisen sollen.	Skalierungsfaktor in GAMA nicht relevant, da das .asc File aus 200x200 m Zellen besteht.	Skalierungsvariable (scale-factor) in GAMA vernachlässigbar, da das importierte Grid bereits eine Auflösung von 200x200 m aufweist. Das Originalfile aus NetLogo ist in seiner Ausdehnung zu groß, weshalb es nicht problemlos in GAMA importiert werden kann, → Resultat: Absturz von GAMA. Das originale .asc File wurde daher mittels ArcGIS in ein Raster mit einer Zellauflösung von 200x200m transformiert. Resultat: abweichende Anzahl der Zellen zu jenen in NetLogo. See-Zellen gesamt NetLogo: 1162 See-Zellen gesamt GAMA: 1175 → Abweichung von rund 1,2 %. Die Verteilung des Planktons erfolgt mit 1.797 kg pro Zelle in GAMA gleich wie in NetLogo.
Schulen vermeiden andere Schulen	Treffen Schulen in einer Sichtbarkeit von 200 m aufeinander, drehen sie sich um 25 Grad in ihrer Bewegung weg.	Treffen Schulen in einer Sichtbarkeit von 200 m aufeinander, drehen sie sich um 25 Grad in ihrer Bewegung weg.	Keine.
Schulen schwimmen	Bewegen sich pro Tag 20 m.	Bewegen sich pro Tag 20 m.	Geschwindigkeitsabweichung

			en von Schulen zu Fischen können sowohl in NetLogo als auch in GAMA stattfinden → Überprüfung ist schwierig.
Schulen limitieren Größe	Ab 5.000 Fischen in einer Schule, werden keine neuen aufgenommen und jeder weitere Fisch wird abgegeben, bis die Fische in den Schulen unter dem Grenzwert von 5.000 liegen.	Ab 5.000 Fischen in einer Schule, werden keine neuen aufgenommen und jeder weitere Fisch wird abgegeben, bis die Fische in den Schulen unter dem Grenzwert von 5.000 liegen.	Keine
Fische reproduzieren sich in Schulen	Anhand einer SD-Formel.	Anhand einer SD-Formel.	Keine.
Fische sterben in Schulen	Sterben entweder weil sie älter als 6 Jahre sind oder weil sie verhungern.	Sterben entweder weil sie älter als 6 Jahre sind oder weil sie verhungern.	Keine.
Plankton	Plankton-Biomasse wird pro See-Zelle für den gesamten Attersee errechnet (logistisches Wachstum). Diffusion pro Zelle (8) 10% pro Monat.	Plankton-Biomasse wird pro See-Zelle für den gesamten Attersee errechnet (logistisches Wachstum). Diffusion pro Zelle (8) 10% pro Monat.	Keine.
Plankton – Reduktion	Fische + Schulen fressen pro Tag 2,7 Gramm pro Fisch.	Fische + Schulen fressen pro Tag 2,7 Gramm pro Fisch.	Keine.
Schulen – Plankton	Schulen schwimmen zur planktonreichsten Zelle.	Schulen schwimmen zur planktonreichsten Zelle.	Keine.

Laut Raab (2022) sind neben der beschriebenen Modellkomplexität auch die Ausführungszeit ein entscheidendes Merkmal. Die Zeit wird in den Systemen als Modellzeit angegeben. Die Abläufe - deren Zustandsänderungen voneinander abhängen - sind synchronisiert. Um den Synchronisationsaufwand gering zu halten, sollte die Abarbeitung aller paralleler Prozesse sequenziell erfolgen (Blunk, 2019). Im Fisch-Plankton-Modell findet die Abarbeitung aller Abläufe jedoch parallel zum jeweiligen Zeitpunkt statt. Die Zeit, die während der Abarbeitung vergeht, ist die Ausführungszeit am Computer (Blunk, 2019). Eine Simulation ist umso effizienter je geringer die Ausführungszeit ist (Blunk, 2019). Zur

Bewertung der Ausführungszeit wurde die Dauer einer Simulation in GAMA und in NetLogo gemessen. Die Ergebnisse sind in Tabelle 11 wiedergegeben.

Tabelle 11: Performancevergleich GAMA und NetLogo

Tool	Fish (Anzahl)	Plankton (in Tonnen)	Dauer
GAMA	646.222	2.106,09	3 h 08 min
NetLogo	443.949	2.171,00	28 min

In der Ausführungsgeschwindigkeit einer Simulation übertrifft NetLogo GAMA um das 6-fache. GAMA verwendet für die parallele Ausführung einer definierten Anzahl an Simulationen die Funktion batch-Experiment. Diese Funktion führt in der Simulation mit einer hohen Anzahl von Prozessen zu einem Problem bezüglich des Gesamtspeichers durch eine geringe Speicherausnutzung (Blunk, 2019). Die Implementierung zeigt, dass ab einer Anzahl von 5 parallelen Experimenten das System in ein OutofMemory-Exception läuft, trotz verteilter Last auf mehrere Rechenkerne (threads). In NetLogo hingegen stellt die parallele Ausleitung von 100 Runs keine Probleme dar. Da die Prüfung auf das Erreichen des jeweiligen Stack-Endes einen negativen Einfluss auf die Laufzeit nehmen würde, wurde dessen Einbau in der Arbeit nicht vorgesehen (Blunk, 2019). Ein weiterer signifikanter Unterschied zwischen GAMA und NetLogo ist die Größe ihrer jeweiligen Ausgabefiles. GAMA produziert für denselben Inhalt um ein 133 mal größeres File, was den Leistungsfluss durch höhere Daten negativ beeinflusst. Welche zusätzlichen Datenmengen bei der Simulationsauswertung in GAMA gespeichert werden, ist nicht klar nachvollziehbar.

5. Diskussion

Bevor die Ergebnisse zusammengefasst, die Ausarbeitung beleuchtet, die Hürden in der Forschung genannt und die Empfehlung für weiterführende Forschung gegeben wird, möchte ich darauf hinweisen, dass sich die für die Ergebnisinterpretation hinzugezogenen Vergleichsstudien sowie die daraus entnommenen Referenzen rein auf AB-Modelle beschränken, da der aktuelle Forschungsstand zu hybriden AB-SD Replikationsmodellen eine Forschungslücke aufweist und daher keine themengleichen Studien gefunden werden konnten (Wallentin and Neuwirth, 2016).

Einer der bedeutsamsten Voraussetzung für eine erfolgreiche Replikation ist die Planung (Bajracharya and Duboz, 2013). Eine gute Planung liefert eine fundierte Grundlage anhand der Replikationsentscheidungen getroffen werden können, die weiters den Replikationsprozess steuern und dessen Erfolg bestimmen. Aus diesem Grund findet zuallererst in der Arbeit eine Ist-Analyse statt, die den derzeitigen Zustand des zu replizierenden Modells abbildet und mögliche Umsetzungspotenziale

aufzeigt. Denn der Ausgangspunkt einer Replikation ist eine Prognose künftiger Ergebnisse in Abhängigkeit zu den Handlungsmöglichkeiten. In der Arbeit wurde die Replikation des hybriden Fisch-Plankton-Modells anhand des ODD-Protokolls und Source Codes geplant. Das ODD-Protokoll und der Source Code sind geeignete Instrumente, um die Planung zu unterstützen und die Transparenz, Aussagefähigkeit und Qualität der Replikationen zu erhöhen. Zusätzlich bilden das ODD-Protokoll und der Source Code ein geeignetes Rahmenwerk, das die Definition bestimmter Anforderungen, die innerhalb der Replikation erfüllt werden müssen, an das replizierte Modell ermöglicht. Eine der bedeutsamsten Anforderung an die Planung ist die Einhaltung der Transparenz. Mit fortschreitender Replikation wird die Einhaltung der Transparenz immer herausfordernder, da Aufwände und Forderungen auftreten, die zu Beginn des Vorhabens nicht (vollständig) absehbar waren und dadurch nicht transparent erfasst werden können. Zur Aufdeckung von Replikationsrisiken sind Vergleichsstudien hilfreich, da zu jedem Ereignis eine kausale Erklärung existiert. Durch sie können frühzeitige Bedrohungen erkannt und bei wesentlichen Entscheidungen die zu erwarteten Ergebnisse und Risiken gegeneinander abgewogen werden. Trotz der Bedeutung von Modellreplikationen für die empirische Forschung als Nachweis der Replizierbarkeit eines bestimmen Forschungsergebnisses unter unabhängigen Bedingungen (Erdfelder, 2018) existieren kaum ausreichend dokumentierte Studien. Es fehlen vor allem Publikationen zu plattformübergreifenden, hybriden Modellen, weshalb das Auffinden eines in der Wissenschaft geforderten Bewertungsverfahrens für das replizierte Fisch-Plankton-Modell aufgrund fehlender empirischer Evidenz nur schwer möglich ist. Vor allem die Bewertung mit konventionellen Evidenzkonzepten ist nicht geeignet, da es sich bei dem hybriden AB-SD Modellierungsansatz um eine junge Entwicklung handelt, die sich von den traditionellen Simulationsparadigmen abkehrt (Wallentin and Neuwirth, 2016). Wann immer ein neues Werkzeug oder eine neue Technik auftaucht, braucht es Zeit, um die Details ihrer Anwendung, Fähigkeiten und Grenzen herauszufinden (Heath et al., 2009). Für die geeignete Replikation des hybriden ABM-SD Modells musste somit erst herausgefunden werden, welche Simulationstechniken sich am besten eignen. Die Herausforderungen liegen dabei in der großen Anzahl und Bandbreite möglicher Störfälle sowie Ableitung von Lösungsstrategien zu diversen Toolkits der einzelnen Programmiersprachen. Obwohl das Ausgangsmodell sowie das replizierte Modell auf einer Java-basierten Sprache beruhen, weisen sie starke Unterschiede in ihrer Sprachprimitive und Struktur auf. Das beeinflusst die Art und Weise wie Agenten dargestellt, wie Aktionen geplant und durchgeführt werden und wie sich Agenten bewegen (Art und Geschwindigkeit). Die verschiedenen Ansätze der Erfassung wirken sich negativ auf die Ergebnisse aus und führen zu Unterschieden in der Umsetzung hybrider Modelle (Railsback et al., 2006, Bajracharya and Duboz, 2013). Im schlimmsten Fall behindern sie sogar den Replikationsprozess und machen den Vergleich der Ergebnisse unmöglich. Heath (2009) stellt in ihrem Artikel fest, dass speziell für ABM erst geeignete statistische und nichtstatistische Validierungstechniken

entwickelt werden müssen (Heath et al., 2009). Der Mangel an Vergleichsstudien erweckt den Eindruck, dass WissenschaftlerInnen eher neue Modelle entwickeln, als bestehende zu replizieren (Seagren, 2015). Ohne direkte Replikation kann in der Wissenschaft jedoch kaum ein allgemein akzeptierter, empirischer Tatbestand definiert werden (Erdfelder, 2018). Olaru (2009) zeigt anhand seiner Studie, die 279 wissenschaftliche Artikel zu agentenbasierter Modellierung aus 92 verschiedenen Publikationen umfasst, ob und wie viele davon repliziert sind. Daraus ergibt sich folgendes:

- in 104 Artikel (37,3 % von 279) befinden sich keine Angaben zum Paket oder zur Programmiersprache,
- nur 44 Artikel (15,8 % von 279) ermöglichen den Zugriff auf das Modell,
- 29 % der Modelle sind nicht validiert, bei 17 % ist das konzeptionelle Modell validiert, 19 % sind operativ validiert und 35 % sind sowohl konzeptionell als auch operativ validiert.

  Anmerkung: Der wissenschaftliche Gebrauch des Begriffs ‚Validierung' ist in seiner Verwendung nicht eindeutig, genauso wenig, was der Begriffsinhalt umfasst (Louie and Carley, 2008). Es ist daher in der unterschiedlichen Literatur darauf zu achten, was unter dem Begriff verstanden wird.

Neben Olaru (2009) skizziert auch Heath (2009) die mangelnde Validierungstendenz von Studien. Das ist erstaunlich, da sie gegen die Standards guter wissenschaftlicher Praxis verstößt. In der ABM-Community sollte es daher nicht zur Regel werden, dass Modelle nur teilweise oder gar nicht validiert werden. Die Validierung ist das einzige Instrument, um einen dokumentierten Beweis zu liefern, dass ein Modell die spezifizierten Anforderungen (Ergebnisse) reproduzierbar erfüllt (Heath et al., 2009). Dasselbe gilt auch für hybride AB-SD Modelle, weshalb ein Mangel an akkordierten Modellen oder Modellkomponenten zur Wiederverwendung vorliegt (Bell et al., 2015). In der Arbeit wurden die Replikationsbemühungen vollständig validiert und dokumentiert, um einen Beitrag zu replizierten plattformübergreifenden hybriden Simulationsstudien in der Fachliteratur zu leisten.

Die folgenden Ausführungen stützen sich auf die empirische Untersuchung, inwiefern die Replizierbarkeit der Ergebnisse möglich ist, welche Erkenntnisse aus der Replikation gewonnen werden und anhand welcher Kriterien (Prüfgrößen) die Replizierbarkeit gemessen wird. Auf die Forschungsfrage, ob die wissenschaftlichen Ergebnisse des Fisch-Plankton-Modells replizierbar sind, kann folgende Antwort gegeben werden: Die Ergebnisse des Fisch-Plankton-Modells konnten hinsichtlich ihres Trends, jedoch nicht in ihrer Größe repliziert werden.

Inwieweit das Modell nur anhand der Publikation in einer anderen Modellierumgebung implementiert werden konnte und inwieweit der Sourcecode gelesen werden musste, kann wie folgt beantwortet werden: Die Replikation wurde anhand des ODD-Protokolls und Source Codes des Ausgangmodells durchgeführt. Durch das ODD-Protokoll und den Source Code wurde die Logik des Modells zur Verfügung gestellt, wodurch eine schnellere Überprüfung der Sachlage sowie eine einfachere Replikation in einem Softwarepaket oder einer Programmiersprache nach Wahl ermöglicht werden soll (Heath et al., 2009). Das gewählte Softwarepaket ist GAMA in der Version 1.8.2 und nicht wie ursprünglich vorgesehen 1.8.1. In diesem Zusammenhang möchte ich darauf hinweisen, dass die Angabe der Versionsnummer relevant ist, da sie einen spezifischen Entwicklungsstand des Quellcodes angibt (Kunkel, 2013). Von der Version 1.8.1 wurde abgesehen, da unter dieser Version die Simulation nicht über eine Laufzeit von 40 Jahren hinausgeht, ehe sie in eine OutOfMemoryException läuft und schlussendlich wegen des Speichermangels abbricht. Dieses Problem tritt in der Version 1.8.2 nicht auf. Bei der OutOfMemoryException-Fehlermeldung handelt es sich um ein Java-Performance-Problem. Aus der Beschreibung zum Release (siehe https://gama-platform.org/wiki/OlderVersions, 18.7.2023) in Bezug auf das Java-Performance-Problem wurde die Annahme entnommen, dass das Problem an der niedrigeren Version der JDK-Umgebung in der Version 1.8.1 liegt.

Zurückkommend auf den Artikel von Wallentin und Neuwirth (2016) ist bereits eine detaillierte Prozessbeschreibung zum Fisch-Plankton-Modell vorhanden. Zum Teil weist der Artikel redundante Informationen (z.B. Zweck, Entitäten, Prozessübersichten, etc.) zum ODD-Protokoll auf. Das ODD-Protokoll, als detaillierte textuelle Beschreibung des Modells, ist für die Replikation und ihrer Planung (z.B. wie können Agenten und ihr Verhalten mit welchen Klassen und Methoden abgebildet werden, etc.) eine wichtige Informationsquelle. Es schafft ein leichteres Modellverständnis, ohne dabei die Sachverhalte übermäßig technisch zu beschreiben. Die textlichen Ausführungen erfolgen unabhängig zur verwendeten Hard- und Software und haben neben den textlichen Erläuterungen auch Angaben zu den Algorithmen und Gleichungen, die für den SD-Teil des Fisch-Plankton-Modells relevant sind. Es kann generell bestätigt werden, dass neben der Qualität und Vollständigkeit der Modelldokumentation der Replikationserfolg auch von dessen Interpretation abhängt (Grimm et al., 2006, Grimm et al., 2014, Schmolke et al., 2010, Donkin et al., 2017). Daher sind für das klare Verständnis des Modellverhaltens und der Abläufe weitere Instrumente erforderlich. Als Instrument zur Vermeidung von Fehlinterpretationen wird in der Arbeit der Source Code verwendet. Das folgende Beispiel kann das belegen: Das ODD-Protokoll enthält keine Information zur zeitlichen Abfolge des Agentenverhaltens. Aus der Kontrollstruktur, die die Reihenfolge des Ablaufs der Reflex- und Action-Methoden vorgibt, ergeben sich neue Agenten-Zustände. Der Sequenz-Zustand mit dem Wert des bisherigen Verhaltenszustands dient als Ausgangsbasis. Der Sequenz-Zustand wird in Abhängigkeit vom jeweiligen

Ereignis-Zustand aktualisiert. Sind alle Sequenz-Bedingungen abgearbeitet, ergibt sich am Ende der Simulationslaufzeit das Modellverhalten. Wie und in welchem Ausmaß sich die Änderungen der Ablauffolge auf das Modellverhalten auswirken, kann in den Simulationsexperimenten als virtuelle Labore getestet werden. Die Informationen zur Ablauffolge ist nur im Source Code sichtbar und nicht im ODD-Protokoll. Grimm (2020) vertritt hingegen die Auffassung, dass das ODD-Protokoll keine Beschreibungen zu den Simulationsexperimenten enthalten soll. Er schlägt die Erweiterung der textuellen Beschreibung um Checklisten vor, mit denen die wichtigsten Informationen zur Modellkalibrierung festgehalten werden können. Ich bin der Meinung, dass eine Kombination aus ODD und der Beschreibung zu den Simulationsexperimenten bzw. dem Source Code die bessere Grundlage für die Replikation darstellt als das Protokoll allein. Wie das Beispiel zeigt, gehen Informationen die für die Ausführung relevant sind im ODD-Protokoll nicht hervor (wie die Reihenfolge der Abläufe) oder sind nicht enthalten.

ODD wird momentan vorwiegend in der Ökologie und in den Sozialwissenschaften eingesetzt (Grimm et al., 2020). Andere Wissenschaftsdisziplinen hingegen dokumentieren ihre Publikationen und Modelle anhand ihrer eigenen Terminologie und Techniken (Heath et al., 2009). Während Grimm die Auffassung einer gemeinsamen ABM-Sprache vertritt, ist Heath (2009) der Ansicht, dass erst dann von einer gemeinsamen Sprache die Rede ist, wenn sich eine bestimmte ABM-Technik, -Praktik und -Methode aus den verschiedenen ABM-Theorien zum Standard entwickelt hat. Bis dahin entwickeln die verschiedenen Disziplinen weiterhin ihre Modelle anhand bewährter und akzeptierter Ansätze (Heath et al., 2009). Darüber hinaus vertritt Grimm (2020) die Ansicht, dass alle, die nicht mit der ABM-Sprache vertraut sind, trotzdem Algorithmen und Codepassagen verstehen können und in der Lage sind, diese zu überprüfen. Gegen die Position spricht, dass beim Erlernen einer Programmiersprache wenigstens die Grundelemente vollständig verstanden werden müssen, um ihre Funktionalität entsprechend zu verstehen und auf Richtigkeit prüfen zu können. Es können noch mehrere Argumente gegen die Auffassung von Grimm entgegengehalten werden, denn selbst wenn man Programmierkenntnisse aufweist, ist GAMAL und Logo eine eigene Sprache. Sie basieren zwar auf der Programmiersprache Java, lehnen sich aber nur an ihrer Syntax an. Es ist daher notwendig, sich mit beiden Programmiersprachen auseinanderzusetzen, um zu verstehen, dass sie tun, was sie tun. Generell ist eine 1:1 Übersetzung von Logo in GAMAL wegen ihrer unterschiedlichen Sprache nicht möglich. Die dritte Programmiersprache, die in der Arbeit verwendet wurde, ist R und ist sowohl eine Statistiksoftware als auch eine objektorientierte, interaktive Programmiersprache, mit der neben statistischen Auswertungen, vielfältige Grafiken und Simulationen durchgeführt werden können. R ist im Handling für Anfänger gewöhnungsbedürftig. Dasselbe gilt auch für Logo. Ich möchte verdeutlichen, dass ohne Erfahrung im Coden, das einfache Verstehen und Überprüfen von Code-Passagen nicht

möglich ist. Hinzu kommt, je komplexer ein Modell, desto schwieriger ist das Verstehen seines Codes. Die meisten Plattformen bieten für die Einarbeitung eine Reihe an Standardkonzepten, Standard-Bibliotheken sowie Code Templates, die als Rahmenwerk (Framework) für die Implementierung/Übersetzung von der Quell- in die Zielsprache verwendet werden können. Zusätzlich stellen die unterschiedlichen Plattformen jeweils ihre eigenen Simulationswerkezuge bereit, ohne die jeweilige Modellkomplexität dabei einzuschränken (Railsback et al., 2006). Laut Wilensky (2007) können unter der Ignoranz der Verzerrungen, die dem ursprünglichen Modell durch die gewählte Sprache auferlegt wurde, die Unterschiede zwischen dem konzeptionellen und dem implementierten Modell in einer neuen Umgebung leichter beobachtet werden. Daraus kann abgeleitet werden, dass die Unterschiede in den Ergebnissen aus den Verzerrungen durch die gewählte Sprache je Plattform resultieren. Während der Replikation des Fisch-Plankton-Modells wurden laufend Plausibilitätsprüfungen durchgeführt, um einerseits die offensichtlichen Implementierungsfehler zu eliminieren und andererseits die Ursachen für die festgestellten Verzerrungen zu finden. Nach Abschluss der Prüfungen komme ich zu dem Schluss, dass sich die stochastischen Ereignisse und das Bewegungsverhalten, insbesondere das Flocking, in GAMA anders verhalten und das Modellverhalten beeinflussen. Bevor näher auf die Simulationsergebnisse eingegangen wird, möchte ich noch kurz den Faktor Einsatzzeitpunkt des Quellcodes im Replikationsprozess erwähnen. Laut Zhang und Robinson (2021) und Wilensky (2007) soll der Einsatzzeitpunkt nicht zu früh im Prozess gewählt werden, da sich der Quellcode sonst negativ auf den Replikationserfolg auswirkt. Das ist dann der Fall, wenn die Entwicklerin, der Entwickler den Code so früh einsetzt, dass sie/er ihn nur noch als Vorlage verwendet und nicht mehr frei in der Wahl seiner/ihrer Umsetzungsmethode ist. Als Ergebnis erhält man eine Art Kopie des Ausgangsmodells. Man spricht dann auch von einem Refactoring. Unter Refactoring wird die Strukturverbesserung eines Quellcodes unter der Beibehaltung des beobachtbaren Programmverhaltens verstanden (Fowler, 2018). Als EntwicklerIn ist man eher dazu verleitet, eine Art Kopie des Ausgangsmodells oder ein Refactoring auszuführen, als auf potenzielle neue Umsetzungsmittel oder andere Plattformen zuzugreifen, da die Einarbeitung die meiste Zeit in Anspruch nimmt. Für den Modellvergleich eignet sich der Quellcode, um direkt (Funktions-) Codeblöcke zu vergleichen. Ein Vergleich der GAMA-Codeblöcke mit den NetLogo- Codeblöcken wurde in der Arbeit durchgeführt. Zusätzlich wurden die Outputwerte der einzelnen Funktionsblöcke gegenübergestellt, um auf Ähnlichkeit zu prüfen. Die Inputwerte der einzelnen Simulationen sind die initialen Größen, die im Kapitel 2.2. angeführt sind. Zu Beginn einer jeden Simulation ist darauf zu achten, dass die Eingangsgrößen dieselben sind, da die Ausgangsgrößen von den Eingangsgrößen abhängen (Viehof, 2017). Normalerweise wird der Einfluss der Input- auf die Outputwerte mittels einer Sensitivitätsanalyse untersucht (Viehof, 2017). Aufgrund des begrenzten Umfangs der Arbeit wurde keine Sensitivitätsanalyse durchgeführt. Als abhängige Größe ist das Verhalten der Agenten definiert.

Die Zusammenfassung der Antwort auf die erste Forschungsfrage ist: Die Replikation des Fisch-Plankton-Modells konnte nur anhand des ODD-Protokolls gemeinsam mit dem Source Code durchgeführt werden. Obwohl die Verwendung des Source Codes an dem eigentlichen Sinn des ODD-Protokolls vorbeigeht, nämlich, dass der vollständige Text das Lesen und Verstehen des Source Code ersetzen soll, haben sich neben mir auch andere Modellbauerinnen, Modellbauer (Grimm et al., 2020) für das Einsehen des Source Codes als effizientere Methode ausgesprochen. Falls ein Quellcode nicht zur Verfügung gestellt werden kann, sollte für die Förderung des Modellverständnisses zumindest ein Pseudocode mit in die schriftliche Beschreibung des Modelldesigns genommen werden. Als zusätzlich fördernde Maßnahme kann die Kommunikation mit der Modellerstellerin, dem Modellersteller angedacht werden. Denn die Modellerstellerinnen und Modellersteller können ihre Anwendungen hinreichend erläutern und damit die Replikatorin, der Replikator bei der Umsetzung unterstützen. Zusätzlich kann dadurch die Mehrdeutigkeit, die zu unterschiedlichen Interpretationen und Umsetzungsarten führt, durch die Abstimmung zwischen den EntwicklerInnen und den Replikatorinnen und Replikatoren reduziert werden (Zhang and Robinson, 2021, Wilensky, 2007). Ich sehe die Kommunikation zwischen den Entwicklern als einen nicht unwesentlichen Faktor für die korrekte Durchführung einer Replikation. Denn allein der Gedanken- und Erfahrungsaustausch bei zum Beispiel der Übersetzung der Syntax in eine andere Sprache, kann das Ergebnis der Replikation positiv beeinflussen. Wie Replikatorinnen, Replikatoren von der Unterstützung der EntwicklerInnen profitieren, zeigt Wilensky (2007) in seiner Studie. Das Problem in der Praxis, welches auch in der Arbeit festgestellt wurde, ist, dass der Austausch mit den EntwicklerInnen des Ausgangsmodells aufgrund von Zeit- und Ressourcenmangel nicht möglich ist. Als Alternative könnte das Szenario der Verlinkung des ODD-Protokolls mit dem Source Code angedacht werden. Die Transparenz wird gefördert, die Mehrdeutigkeit von Modellen gemindert und der Replikationsprozess erleichtert (Grimm et al., 2020). Dadurch könnte ein Anreiz für viele WissenschaftlerInnen geschaffen werden, aktiver Modelle zu replizieren, da die Hürden wie beispielsweise das Verstehen des Modellcodes durch die Vorteile der Verlinkung reduziert werden. Grimm (2020) sieht in diesem Anreiz sogar eine Lösung der vorherrschenden Replikationskrise, deren Ursprung die zum Teil unvollständige Modellbeschreibung darstellt.

Welche Plattform für eine Replikation verwendet wird, hängt neben dem Hintergrund (Programmiererinnen, Programmierer oder Nicht-Programmiererinnen und Nicht-Programmierer) vom Simulationsexperten auch von dessen Präferenz ab (Heath et al., 2009). Auch Faktoren, wie zum Beispiel Kosten (Open Source ja/nein, Abhängigkeiten, Installations- und Wartungsaufwand, etc.) wirken sich auf die entsprechende Auswahl aus. Derzeit existiert kein standardisiertes Verfahren in der Wissenschaftscommunity für die plattform- und sprachübergreifende Replikation. Es ist auch nicht definiert, wie bei einer Modellübersetzung in eine andere Plattformen vorzugehen ist (Zhang and

Robinson, 2021, Wilensky, 2007). Daher entstehen Fragen wie: Ist die Übersetzung eines Modells auf die gleiche Weise wie für andere Plattform durchzuführen, oder frei von der Befangenheit der ursprünglichen Sprache? Ist bei einer digitalen Transformation mit Veränderungen in den Ergebnissen zu rechnen? In der Fachliteratur wurden keine Antworten auf diese Fragen gefunden. Zhang und Robinson (2021) sind der Auffassung, dass egal wie eine plattformübergreifende Replikation stattfindet, neue Software und Bibliotheken genutzt werden sollen. Die Gefahr liegt darin, dass durch die Verwendung neuer Bibliotheken ein anderes Modellverhalten produziert wird, das anhand eines Beispiels illustriert wird: Die Schul-Agenten wurden initial mit dem GAMA-spezifischen multi-level Architekturansatz (https://gama-platform.org/wiki/MultiLevelArchitecture, 4.5.2023) umgesetzt. Der multi-level Architekturansatz erlaubt es die Schulen außerhalb von Listen (Array) zu verwalten. Die Schulen werden als macro-Spezies definiert. Macro-Spezies setzen sich aus sogenannten micro-species-in-group (= Fische in einer Schule) zusammen. Die Fische in Schulen sind somit Teil der macro_species und können aufgrund des Vererbungsprinzips die Funktionalitäten und Eigenschaften ihrer Eltern-Spezies (Fisch-Agenten) erhalten. Die vererbten Funktionen und Eigenschaften können jederzeit in den micro-species-in-group überschrieben werden, sodass eine entsprechende Verhaltensanpassung (z.B. Überschreibung von ABM-Verhaltensregeln durch SD-Formeln) möglich ist. Der multi-level Architektur Ansatz wird standardmäßig dazu verwendet, wenn eine Entität (hier Fisch) in verschiedenen Typen von Agenten (hier als einzelnes Individuum, als Teil eines Schwarms oder als Teil einer Schule) repräsentiert werden soll. Es stehen auch sogenannte built-in-Methoden zur Verfügung um die Aufnahme der Fische in Schulen (capture) sowie deren Ausstoß (release) steuern zu können. Nach erfolgreicher Durchführung der Replikation des NetLogo-Modells mittels dem multi-level Architekturansatz wurde folgendes Ergebnisbild für die Prüfgröße der gesamten Fischanzahl erstellt:

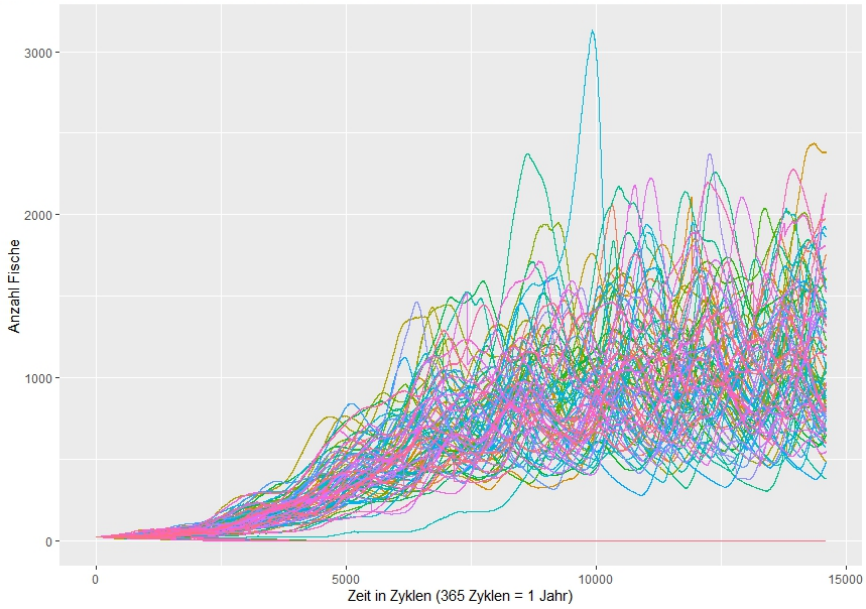


Abbildung 4: Ergebnis der Realisierung des Schul-Agenten anhand des multi-level-Architekturansatzes

Vergleicht man die Ergebnisse in Abbildung 4 mit jenem Resultat in Kapitel 4 ist auf dem ersten Blick erkennbar, dass die Ergebnisse einen ganz anderen Verlauf der Trajektorien zeigen. Es wurde daher entschieden, den multi-level-Ansatz zu verwerfen und die Realisierung der Schulen auf Listen umzubauen. Die Argumentation von Zhang und Robinson (2021) überzeugt an dieser Stelle, nämlich, nur wenn man mit dem konzeptionellen Modell als auch mit dem Code vertraut ist, man auch in der Lage ist, Fehler zu erkennen sowie die ursprünglichen Modellergebnisse zu replizieren und Verbesserungen am replizierten Modell vorzunehmen. Für das ‚Wie' eine Replikation durchgeführt werden soll, existiert bislang in der Wissenschaft noch keine einheitliche Definition. Zum einen wird geraten, frei von der ursprünglichen Sprache zu replizieren. Zum anderen sieht die Replikation eine deckungsgleiche Umsetzung des Ausgangsmodells vor. Es besteht somit Forschungsbedarf von plattformeigenen Modellierungsverfahren und wie mit verschiedenen Entwicklungsplattformen deckungsgleiche Replikationsergebnisse erzielt werden können. In dieser Arbeit wird der Auffassung gefolgt, dass eine deckungsgleiche Umsetzung des Ausgangsmodells vorgesehen wird. Darüber hinaus sollten die wissenschaftlichen Ergebnisse verifiziert werden, um neue Methoden auf der Grundlage verschiedener Perspektiven zu entwickeln (Zhang and Robinson, 2021).

Welche quantitativen Unterschiede in den Modellergebnissen zu Populationsdynamik der Fische, der Biomasseentwicklung des Planktons und der Schulbildung für das originale Modell-Design zum replizierten Modell festgestellt werden konnten, wird folgendermaßen beantwortet: Die Ergebnisse des replizierten Fisch-Plankton-Modells zeigen dasselbe Resultat wie von Donkin (2017), wo ein komplexes, agentenbasiertes Modell unter der Verwendung der Plattformen NetLogo und Repeast repliziert wurde, nämlich, dass es Unterschiede in ihrer Größe gibt, jedoch nicht in ihrer Entwicklung. Die replizierten Ergebnisse weisen somit zu ihren ursprünglichen Ergebnissen eine strukturelle Ähnlichkeit auf, jedoch keine Übereinstimmung in ihren Größen. Die Variabilität in den Ergebnissen wird hauptsächlich der Codekomplexität und der Übersetzung einzelner Funktionen in eine andere Programmiersprache zugeschrieben (Wilensky, 2007). Um festzustellen, ob die Abweichungen in den Ergebnissen erklärbar sind, wurden eine Reihe an Plausibilitätsprüfungen durchgeführt. GAMA bietet dazu eine Reihe an systemeigenen Methoden an. Aus den Tests kann das Fazit gezogen werden, dass trotz der unterschiedlichen Ergebnisse der vergleichenden Analyse die Replikation als erfolgreich angesehen werden kann. Für die vergleichenden Analysen wurden systematisch anhand von Kriterien (Prüfgrößen) die beiden Modelle als Vergleichsobjekte zueinander in Beziehung gesetzt. Im Zentrum der vergleichenden Analyse stehen die Simulationen, die statistischen Methoden und die vergleichende visuelle Methode, die sich den am ähnlichsten definierten Verhalten bedient. Neben dem fehlenden einheitlichen Replikationsverfahren existieren auch keine standardisierten Verfahren und Methoden zur Datenerhebung, -auswertung und -interpretation. Aus der Recherche von Olaru (2009) liegt vor, dass

0,5% der untersuchten Artikel statistische und 95% nichtstatistische Validierungstechniken verwenden. 4,5% setzen eine Kombination aus statistischen und nichtstatistischen Validierungstechniken ein. Nicht-statistische Validierungstechniken überprüfen, ob das konzeptionelle Modell korrekt in das gewählte Simulationsprogramm überführt wurde. Die Aktivität des Modellvergleichs und ihre Verifizierung bietet den Rahmen für die Feststellung der externen und operativen Gültigkeit eines Modells (Olaru et al., 2009). Der geringe Einsatz von statistischen Methoden zur Validierung von ABM lässt die Vermutung aufkommen, dass in diesem Bereich vorwiegend Expertenmeinungen und qualitative Untersuchungen dazu eingesetzt werden. Die untersuchten Artikel führen zu dem Schluss, dass viel weniger statistische als nicht-statistische Validierungsverfahren - obwohl statistische Validierungsverfahren in anderen Simulationsparadigmen weit verbreitet sind - für das ganzheitliche Verständnis von Qualität und Güte angebracht sind. Heath (2009) führt als Hauptgrund die Nichteignung der Ergebnisse von ABM-Simulationen für statistische Analysen an. Denn für agentenbasierte Modelle werden oft komplexe Systeme eingesetzt, deren unstrukturierten Daten für statistische Analysen nicht aussagekräftig genug sind (Heath et al., 2009). Agentenbasierte Modelle unterliegen weiters kausalen Unschärfen (Emergenz) über einen längeren Zeitraum. Louie (2008) vertritt ganz allgemein die Auffassung, dass Multiagenten-Systeme kompliziert zu validieren sind, da sie einen neuen Simulationsansatz darstellen, wo traditionelle Validierungsmethoden nicht immer anwendbar sind. Es besteht somit künftig ein Bedarf an neuen (statistischen) Validierungs- und Datenerfassungstechniken speziell für ABM und hybride Modelle. Hinsichtlich der Validierung der replizierten Ergebnisse ist die Angabe des 99%-igen Konfidenzintervalls genannt. Das Konzept des KI unterstützt den Abgleich der Ergebnisse und stellt sicher, dass das Modell stabil ist und die Ergebnisse nicht durch ungewöhnliche Umstände hervorgerufen wurden (Wilensky, 2007). Um einen Überblick über den Datensatz des replizierten Modells zu erhalten, ist in erster Instanz versucht worden, die zentrale Tendenz, Streuung und Verteilung der Daten je Simulation zu beschreiben. Die empirischen Daten zeigen wie bei Rand (2006) keine identen Ergebnisse aufgrund der fehlenden statistischen Ähnlichkeit. Die Trajektorien des Ausgangsmodells für die Prüfgrößen Anzahl der Fische in- und außerhalb von Schulen, die Anzahl von Schulen und die Planktonverfügbarkeit zeigen zu jenen des replizierten Modells einen ähnlichen Verlauf. Nachdem beide Modellsimulationen mit denselben Parametern initialisiert wurden, deuten die Trajektorien darauf hin, dass die Ergebnisunterschiede nicht aus der strukturellen Fehlimplementierung resultieren, sondern sie durch die plattformspezifische Syntax und Semantik erklärt werden können. Weitere ergebnisbeeinflussende Faktoren sind stochastische Prozesse, die für eine gezielte Analyse und weitere Bewertung von Systemzuständen relevant sind. Bei quantitativen Merkmalen lassen sich die Merkmalsausprägungen durch Zahlen oder Größenwerte ausdrücken. Je nach Systemzustand lassen sich vorkommende Zahlen oder Größenwerte unterscheiden. Im Folgenden wird die Bedeutung der Stochastik und ihre Tätigkeit auf die Größenwerte herausgearbeitet: Je nachdem wie viele Einzelfische

aufeinandertreffen, bilden sich Schwärme und daraus wiederum Schulen. Die Anzahl der Fische in den Schulen wird durch die Reproduktions- und Sterblichkeitsrate in seiner Größe gesteuert und variiert je nach Altersstruktur. Das Plankton und dessen Verfügbarkeit pro Zelle wirkt sich auf die Populationsentwicklung aus, die sich wiederum vom Plankton ernährt. Je nach Systemzustand, der durch stochastische Ereignisse bestimmt wird, entwickelt sich eine verschieden hohe Anzahl an Fischen, die in Abhängigkeit zu ihrer Reproduktions- und Sterblichkeitsrate variieren. Rand (, 2006 #517) konnte in seiner Studie erst deckungsgleiche replizierte Ergebnisse erzielen, indem er das replizierte Modell solange angepasst hat, bis die gewünschten Resultate geliefert wurden. Gegen diese Vorgehensweise spricht, dass eher von einer Überanpassung als von einer Replikation des Ausgangsmodells gesprochen wird. In der Arbeit wurde von einer Überanpassung des replizierten Fisch-Plankton-Modells abgesehen. In diesem Zusammenhang vertritt Railsback (2006) die Auffassung, dass die spezifische Modellstruktur eines der Probleme beim Vergleich von Plattformen darstellt. Zusätzlich können laut Bajracharya (2013 #479) die Zeitplanungsmechanismen in den unterschiedlichen Plattformen nicht vollständig kontrolliert werden. Die zeitliche Verlaufsform von Entscheidungsprozessen unterliegt einer Vielzahl an einfachen Mechanismen, deren Vielfalt unterschiedliche Verlaufsformen auslösen können. Diese Unterschiede spiegeln sich in den Ergebnissen wider. Zusätzlich kann durch die Nichteinhaltung der Ausführungsreihenfolge ein nichtreproduzierbares Verhalten ausgelöst werden (Augsten, 2020), welches schwierig ist zu begründen. In NetLogo ergibt sich die Ausführungsreihenfolge durch den to-Befehl und dem Namen der auszuführenden Prozedur (z.B. to-go) (siehe https://ccl.northwestern.edu/netlogo/docs/programming.html#procedures, 5.5.23). In GAMA wird die Ausführungsreihenfolge durch die chronologische Auflistung der Reflexe im Programmcode definiert.

Zusammenfassend für die Fragestellung nach den quantitativen Unterschieden in den Simulationsergebnissen lässt sich folgendes festhalten: Die quantitativen Angaben der Prüfgrößen Anzahl der Fische in- und außerhalb von Schulen, die Anzahl von Schulen und die Planktonverfügbarkeit sind nicht ident. Das Delta der Anzahl der Fische am Ende der Simulation beträgt in GAMA zu NetLogo rund 200.000. Das Modellverhalten wird durch die vorliegende Simulation, in der sich die Fische randomisiert bewegen, auf der Suche nach Nahrung und anderen Fischen sind, um zu schwärmen oder wenn die Fortpflanzungskriterien stimmen sich entsprechend zu vermehren, bestimmt. Die Geburtenrate sowie Sterblichkeit werden in Schulen mit einem definierten Algorithmus berechnet. Anhand des Outputs kann bestimmt werden, ob die Kennzahlen mit den erwarteten Leistungskennzahlen des Ausgangsmodells übereinstimmen. Sowohl hinsichtlich der Kennzahlen als auch der beobachteten Interaktionen stimmt das Modell in GAMA mit jenem in NetLogo überein. Auch das Plankton entwickelt sich in beiden Plattformen gleich. Für die Prüfgröße der Entwicklung von Fischen außerhalb von Schulen existiert die größte Abweichung. Die einzigen beiden erklärbaren Einflussgrößen

ist das Flocking und die Schulbildung. Beim Flocking handelt es sich um eine zufällige Einflussgröße auf die Geschwindigkeit und das Bewegungsmuster der Fische. Je schneller oder langsamer die Fische schwimmen, desto zeitlich früher oder später bilden sich Schwärme. Ab einer bestimmten Schwarmgröße bilden sich stattdessen Schulen. Wiederum ab einer bestimmten Schulgröße werden Fische aus den Schulen emittiert. Ab diesem Moment folgen die emittierten Fische als Individuen wieder anderen Regeln. Aufgrund dieser Aneinanderreihung von zufälligen Ereignissen wird eine mögliche Ursache für die andersartige Entwicklung der Einzelfische in GAMA ab einem bestimmten Simulationszeitpunkt vermutet. Es konnte aber auch beobachtet werden, dass die unterschiedliche Entwicklung der Fische außerhalb von Schulen in GAMA erst nach ca. 15 Jahren eintritt. Davor ist die Entwicklung der Fische außerhalb von Schulen ident zu jener in NetLogo. In den ersten 15 Jahren existieren jedoch noch keine Schulen. Es wird daher vermutet, dass in NetLogo die Fische schneller emittieren, wodurch sich die Fische außerhalb von Schulen anders entwickeln. Ich möchte abschließend noch einmal darauf hinweisen, dass nicht allein das replizierte Fisch-Plankton-Modell keine mit den ursprünglichen Ergebnissen deckungsgleichen Resultate liefert, sondern andere ReplikatorInnen dieselben Erkenntnisse haben. So auch Bajracharya (2013), die die Auffassung vertritt, dass Simulationen (Experimente) verschiedener Plattformen nicht dieselben Ergebnisse liefern. Inwiefern nun die räumlichen Muster der Simulationsergebnisse nach einem simulierten Zeitraum von 100 Jahren den publizierten Ergebnissen entsprechen, wird im Folgenden beantwortet: Die räumlichen Muster wurden anhand der Dimensionen Raum, Zeit und Dynamik festgestellt. Die grafische Gegenüberstellung des Start- und Endpunktes der Simulation (siehe Kapitel 4, Abbildung 3) ermöglicht das Erkennen von Unterschieden. Trotz der Übernahme der Logik aus sämtlichen Methoden und Klassen aus NetLogo und der entsprechenden Übersetzung in GAMA, konnten Unterschiede in den räumlichen Mustern festgestellt werden. Nicht in der räumlichen Verteilung von Schulen und Fischen, sondern in der zellenbasierten Farbsättigung, die sich aus dem Planktongehalt pro Zelle ergibt. Mögliche Gründe für die Unterschiede sind:

- Geringere Zellanzahl in GAMA durch das Resampeln des originalen ASC-File. Resample ist eine ArcGIS Datenmanagement Methode zur Änderung der räumlichen Auflösung eines Rasters. Der Grund für das Resampeln ist, dass das originale ASC-File zu groß ist, um in GAMA geladen werden zu können. Die Zellgröße wurde in ArcGIS auf 200x200 eingestellt. Durch den automatischen Resampling Vorgang, wurden Zellen zusammengelegt. Daraus resultiert die geringere Zellanzahl in GAMA.
- Mit fortschreitender Simulationslaufzeit nimmt zwar die Fisch- und Schulanzahl in GAMA um ein Vielfaches zu, jedoch verlangsamt sich ihre Bewegung. Daher wird pro Zeitschritt in GAMA weniger ‚schnell' Nahrung aufgenommen als in NetLogo, wo sich die Agenten mit einer viel

höheren Dynamik pro Zeitschritt fortbewegen. In NetLogo wird daher mehr Plankton pro Zelle pro Zeitschritt entnommen.

Die Wachstumsrate für das Plankton sowie dessen Diffusion ist in beiden Plattformen gleich. Anschließend möchte ich die Frage ‚Inwiefern stimmen die Phasen, in denen sich die gekoppelten Modelle in der AB- bzw. SD-Phase befinden, mit den publizierten Ergebnissen überein?' beantworten. Hinsichtlich der Phasenübergänge ermöglichen beide Modelle die Darstellung eines komplexen dynamischen Systems. Das dahinterliegende Konzept der Phasenübergänge ist wie folgt definiert: Nachdem die Fischpopulation auf eine lebensfähige Größe angewachsen ist, die sich in Fischschwärmen organisieren, veranlasst die Überschreitung des Grenzwertes von 50 Einzelfische die Umwandlung in die aggregierte Darstellung Schule. Der Wechsel von Agenten zu Schul-Agenten wird durch die automatische Erkennung einer bestehenden Schule operationalisiert und durch die automatische Erkennung eines einfachen Schwellenwertes von 50 Fischen optimiert. Durch den Wandel in einen Schul-Agenten geht ein Schwarm einzelner Fische von AB in SD über, während einzelne Fische und kleinere Schwärme weiterhin im AB-Modus bleiben. Diese Umsetzung ermöglicht einen schrittweisen Phasenübergang von AB in SD. Fällt die Fischpopulation in einem Schwarm unter 50 Fische, schaltete das Modell auf seine Ausgangskonfiguration zurück. Das Fisch-AB wird analog zur Initialisierungsphase des Modells parametrisiert. Jede Schule kann bis zu 5.000 Fische aufnehmen, alle darüber hinaus, werden abgestoßen. Die überschüssigen Fische werden dabei in einzelne Fisch-Agenten umgewandelt. Der Ausstoß repräsentiert den Phasenübergang von SD in AB. Die Verteilung der Fischpopulation zum Zeitpunkt des Wechsels wird durch eine stochastische Auswahl der verbleibenden Fische nach dem Abfischen der vorhandenen Schwärme beibehalten. Für das Plankton existieren SD-Strukturen. Die Modelle erforschen die Beziehungen zwischen dem Verhalten der Akteure und dem emergenten Verhalten des Systems unter Verwendung flexibler Strukturen (Phasenübergänge). Das replizierte Modell wurde schrittweise transparent entwickelt, mit ad-hoc Anpassungen, die sich aus den intensiven Analysen ergaben, um eine bessere Replikation zu erreichen und die Annahmen zu verstehen. Die Modelle weisen ähnliche Übergänge in den Phasen AB und SD auf, womit die Phasen mit den publizierten Ergebnissen übereinstimmen. Für die Prüfung auf Übereinstimmung wurden Modellgütekriterien definiert (siehe Kapitel 4). Die Überprüfung der Kriterien erfolgt in dem Set, indem untersucht wurde, ob eine Ausprägung einer Eigenschaft bei einer anderen Gruppe in der anderen Plattform ebenso differenziert auftritt. Durch die Verknüpfung der Elemente der beiden Modelle wird auf Datenpassung getestet, nämlich ob die Agenten und ihre Beziehungen zwischen den Konstrukten übereinstimmen. In Anbetracht einiger inhärenten Schwierigkeiten beim Replizieren, kann durch die Übereinstimmung der Phasen und ihrer Übergänge mehr Vertrauen in die Replikation und in ihre Implementierung geschaffen werden. Die Ergebnisse gewinnen zusätzlich an Plausibilität. Innerhalb

ihres Anwendungsbereichs weisen die Modelle eine zufriedenstellende Genauigkeit auf und unterstützen die Hypothesen, dass für die empirische Validität komplexer hybrider AB-SD Modelle innovativere Ansätze geschaffen werden sollen und dass die klassische empirische Validität nicht immer primäre Grundlage für die Annahme oder Ablehnung von Modellen sein sollten (Olaru et al., 2009). Darüber hinaus möchte ich anmerken, dass eine ineffiziente Umsetzung beim Emittieren der Fische in NetLogo bei der Überschreitung des Grenzwertes von 5.000 Fischen in Schulen festgestellt wurde. Diese Ineffizienz nimmt einen potenziellen Einfluss auf die Rechengeschwindigkeit. Denn bevor ein Fisch emittiert werden kann, muss bei jedem Zeitschritt (Raab et al., 2022) die gesamte Fischliste von 5.000 Einzelexemplaren durchgegangen werden, bis ein zufälliger Index ausgewählt wird, aus dem sich die Anzahl der emittierten Fische ergibt. Das Durchsuchen der Liste nach einem zufällig gewählten Index, kostet bei der Menge an Fischen eine bestimmte Rechenzeit, die mit steigender Anzahl an Schulen exponentiell zunimmt. Ich würde als Lösung vorschlagen, dass entweder jene Anzahl an Fischen emittiert wird, die entweder an erster oder letzter Stelle in der Liste stehen.

Als nächste Frage soll geklärt werden ‚Inwiefern konnten die Schlussfolgerungen mit dem replizierten Modell bestätigt werden?'. Um die Schlussfolgerungen mit dem replizierten Modell bestätigen zu können, wurde eine konzeptionelle als auch operationelle Validierung durchgeführt. Die räumliche Darstellung der Entitäten Fische und Plankton ist im replizierten Modell gleich wie im Ausgangsmodell. Die Fische werden sowohl im Ausgangsmodell wie im replizierten Modell als Agenten oder Schwarmagenten dargestellt. Das Plankton wird in beiden Modellen für die eine Konfiguration in einem räumlichen Bestand und für die andere Konfiguration in einem zellulären Automaten abgebildet. Die Zustandsvariablen werden in beiden Modellen gleich definiert. Die zeitliche Skala der beiden Modelle hat eine Auflösung von einem Tag und erstreckt sich über einen Simulationszeitraum von 100 Jahren. Der See ist grob rechteckig mit 20 x 20 Zellen über eine Gesamtfläche von 46 km^2 abgebildet. In der räumlichen Modellkonfigurationsvariante der Modelle diffundierte das Plankton 10 % seiner Biomasse pro Monat in benachbarte Zellen.

Die Schlussfolgerung für die Bewertung der langfristigen Populationsdynamik in einem 100-jährigen Simulationsexperiment, wofür am Ende einer jeden Simulation die Anzahl von Fischen und Plankton exportiert wurden, sieht wie folgt aus: Die Populationsdynamik als auch das Plankton weisen in beiden Modellentwürfen die erwartete logistische Wachstumsdynamik auf. Anfangs entwickelte sich die Fischpopulation langsam mit konstantem Wachstum bis die Planktonmenge das weitere Wachstum begrenzte und sich die Population auf ein Gleichgewicht einpendelte. Die Dynamik der Populationsentwicklung ist in beiden Modellen ähnlich, aber die Grenze des Gleichgewichts unterschiedlich. Die Schlussfolgerung, die von Wallentin und Neuwirth (2016) für die

Modellkonfiguration Agent/Bestand - Schul-Agent/Bestand gezogen wurde, lautet, dass das Ausmaß des Populationswachstums durch die lokale Planktonverfügbarkeit festgelegt wird, weshalb die endgültige Populationsgröße auf ca. 450.000 Fische begrenzt ist. Dieselbe Schlussfolgerung gilt für das replizierte Modell, jedoch liegt die endgültige Populationsgröße bei 650.000 Fische.

Eine weitere Schlussfolgerung von Wallentin und Neuwirth (2016) bezieht sich auf das von Scheffer (1995) vorgestellte Konzept der Superindividuen. Die Superindividuen sind gemittelte Attribute und werden als rechnerisch praktikablere Alternative zur expliziten Darstellung von Individuen vorgeschlagen (Scheffer, 1995). Der Ansatz wurde von Vincenot (2011), und Swinerd (2012) um die interne Struktur eines Superindividuums mittels SD-Modell erweitert. Das Konzept der hybriden Superindividuen wurde um einen reversiblen Wechsel zwischen Individuen und Superindividuen angepasst. Die Schlussfolgerung besagt, dass eine höhere Aggregationsstufe zwangsweise nicht zu einer höheren Rechenleistung und Bearbeitungszeit führen. Gegen diese Position spricht das replizierte Modell, da hier eine höhere Rechenleistung und Bearbeitungszeit in Anspruch genommen wird als in NetLogo. Es wurden GAMA-Systemoptimierungen (siehe https://gama-platform.org/wiki/OptimizingModelsSection, 19.7.2023) zur Steigerung der Recheneffizienz und um eine höhere Performance zu erreichen, vorgenommen. Die Optimierungen brachten keine gewünschten Effekte (z.B. mehr verfügbarer Speicher, erhöhte Rechenleistung, etc.). Einer der Gründe für die Systemverzögerungen, der einen Einfluss auf die Rechenleistung nimmt, ist die rechenintensivere listenbasierte Umsetzung von Schulen in GAMA anstelle einer objektorientierten Programmierung.

Abschließend möchte ich folgendes zusammenfassen: Die in Kapitel 1.4 definierten Forschungsfragen (inkl. operativen Teilziele) fordern die Bestimmungen der Objektivität und Nachvollziehbarkeit der aggregierten Simulationsergebnisse sowie die Beurteilung des Grads der Reproduzierbarkeit des komplexen Modells und seiner Ergebnisse und was es dazu alles gebraucht hat (z.B. ODD ja/nein, Ergänzungen im ODD, Source Code ja/nein, oder etwas ganz anderes). Der Grad der Reproduzierbarkeit ist abhängig von der Planung und von den Risiken, die Abweichungen in den Ergebnissen auslösen. In den replizierten Simulationsergebnissen des hybriden Fisch-Plankton-Modells wurden Unterschiede festgestellt, die aufgrund der Beschreibung des konzeptionellen Modells nicht erwartet und mit dem ursprünglichen Modell nicht erzielt wurden. Die Ergebnisse der Masterarbeit decken somit Risiken in der plattformübergreifenden Replikation eines hybriden ABM-SD Modells auf. Ich bin derselben Meinung wie Zhang und Robinson (2021), dass ein wirksames Mittel gegen die Replikationsrisiken ein systemischer Rahmen für komplexe Modellreplikationsverfahren ist. Eines der schwerwiegendsten Replikationsprobleme ist das Fehlen von Kenntnis darüber, wie die Planung einer Replikation stattfinden soll, welche Risiken bei der Replikation auftreten können, ob das Erreichen von deckungsgleichen

Ergebnissen möglich ist, etc. Denn durch die Nichtexistenz geeigneter Literatur ist bereits zu Beginn des Replikationsprozesses das Finden eines geeigneten Ansatzes bzw. ein systemischer Rahmen (= in Form eines allgemeinen, wissenschaftlichen Programms, der den Umgang mit Komplexität regelt) problematisch. Ich schließe mich der Meinung von Kleindorfer und Ganeshan (1994) an, dass das Problem nicht die Validierung darstellt, sondern dessen Operationalisierung. Die Modellvalidierung sollte daher nicht aus einer Reihe von vorgeschriebenen Validierungsansätzen bestehen, sondern eher einer Verteidigung gleichen, in der berufen wird, dass das Modell einer allgemeinen akzeptierten Sichtweise nach Kriterien, wie z.B. Einfachheit, Konsistenz, etc., gerecht wird (Olaru et al., 2009, Kleindorfer, 1994 #522, Kleindorfer and Ganeshan, 1994). Unabhängig davon, dass die erwarteten Ergebnisse nach dem Replikationsprozess nicht erzielt wurden, bin ich derselben Meinung wie Zhang und Robinson (2021), nämlich dass eine erfolgreiche Replikation zwar die Zuverlässigkeit und Reproduzierbarkeit eines Modells kennzeichnet, jedoch ihr Misserfolg Lehren vermittelt, die mitunter das wertvollere Ergebnis einer Replikation sein können. In weiteren Forschungsarbeiten könnte, angelehnt an Richardie (2006), an einem Standard gearbeitet werden, der vorgibt, wie Modelle/Replikationen präsentiert aber auch wie die Ergebnisanalysen durchgeführt werden sollen, als eine Art Best Practice für räumliche, hybride ABM-SD Modelle. In einer sich anschließenden Studie könnte untersucht werden, wieso sich das Gleichgewicht der Fischpopulation nicht bei 450.000 einpendelt und sich die Fische außerhalb von Schulen in GAMA anders entwickeln.

6. Fazit

Es existieren zwar eine Vielzahl an verschiedenen Modellierungsparadigmen und -umgebungen und die Kombination verschiedener Modellierungsansätze ist im Fokus diverser Studien, jedoch zur Beantwortung der Frage, ob und wie ein hybrider ABM-SD Ansatz replizierbar ist, bleibt in der Wissenschaftsliteratur unbeantwortet. Generell konnten keine Vergleichsstudien zum Thema Replikation hybrider Modellierung gefunden werden. Ich komme daher zu dem Schluss, dass die vorgestellte Masterarbeit ein wertvolles Beispiel für die Replikation von hybriden Modellen darstellt. Sie verdeutlich auch die Notwendigkeit weiterer Replikationen, die auch Unterschiede in den Ergebnissen zulassen. Denn für die Bewertung des Erfolgs oder auch Misserfolgs einer Replikation sind mehrere vergleichbare Studien notwendig, um festzustellen, inwiefern sich die methodische Vorgehensweise sowie die Erfahrung, die anhand der Datenanalyse und -interpretation gesammelt wurde, decken und sogar ähnliche Ergebnismuster aufzeigen.

Zur Relevanz des Quellcodes für die Replikation wurde in der Wissenschaftsliteratur keine eindeutige Aussage gefunden. Einerseits kann bestätigt werden, dass der Quellcode maßgeblich zum Verständnis

des ursprünglichen Modells beiträgt und andererseits, wenn er zu früh im Replikationsprozess eingesetzt wird, den Replikator nachteilig beeinflusst, indem dieser die Unabhängigkeit zum ursprünglichen Modell in der Realisierung des replizierten Modells verliert. Dieses Paradoxon veranschaulicht den ersten Replikationsversuch des Fisch-Plankton-Modells mit dem multi-level Architekturansatz, auf den folglich auf den listenbasierten Ansatz gewechselt wurde, da die Ergebnisse sich von den ursprünglichen Modellen komplett unterschieden. Zur Anpassung des Modells an die Ergebnisse wurde somit der multi-level Ansatz verworfen und der listenbasierte Ansatz gewählt.

Darüber hinaus lässt sich folgendes bestätigen, nämlich, dass die Replikation als wissenschaftliche Methode, die auf Nachbildung eines bestehenden Modells anhand einer Modelldokumentation basiert, so lange als wissenschaftliche Methode zulässig ist, so lange WissenschaftlerInnen dazu aufgefordert sind, ihre Modelle zu beschreiben und die Einzelheiten zu ihren Experimenten zu dokumentieren. Denn ohne eine vernünftige Dokumentation des Ausgangsmodells wäre eine Replikation des Fisch-Plankton-Modells nicht möglich gewesen. Es ist sicherlich zulässig, die Inhalte auf unterschiedliche Art zu lösen, jedoch nicht die Artefakte inhaltlich verschieden umzusetzen. Denn wenn man durch Fehlinterpretation des Modells die Verhaltensweisen auf fälschliche Weise definiert, kann nicht gewährleistet werden, dass die Replikation gültig ist. Es ist daher klar die Grenze zu ziehen zwischen Fehlinterpretation und den der Softwareabhängigkeit zuzuschreibenden Variabilität. Denn gerade die Replikation von Computermodellen als ‚junges' Forschungsfeld und deren Erkenntnisse sind enorm wichtig für den Beweis, dass die Experimente und nicht das Resultat zufällige Ereignisse sind.

Ich komme zu dem Schluss, dass die Replikation von hybriden Modellen ein komplizierter und einer mit unvorhersehbaren Problemen behafteter Prozess ist. Die durch die Masterarbeit gesammelten Erfahrungen sollen einen Beitrag für die Modellierungsgemeinschaft leisten, um künftig die Replikation von hybriden Modellen auf eine solide Grundlage zu stellen. Zusätzlich sollen die Erkenntnisse und Ergebnisse der Masterarbeit einen Beitrag zur Diskussion zu plattformübergreifender Replikation hybrider Modelle leisten. Ich habe auch einige Probleme erörtert, die im Bereich der Modellreplikation noch gelöst werden müssen (z.B. Flocking, Einflussnahme von Softwareversionen, etc.) und hoffe damit, die Diskussion und vor allem die Motivation von Replikationsstudien voranzutreiben, um Antworten auf die Probleme zu bekommen und Lösungen zu finden. Allerdings wird nur eine kontinuierliche Auseinandersetzung mit den mit den Ergebnisunterschieden verbundenen Fragen sowohl bei den Modellreplikatorinnen und Modellreplikatoren als auch bei den Modellautorinnen und Modellautoren zu einer Verbesserung der Replikation in der Praxis führen. Ich hoffe, dass ich mit den Erfolgen und Misserfolgen der Replikationsstudie zu neuen Erkenntnissen und nachfolgend zu Verbesserungen in der Modellstruktur, der Methodik, den Datenanforderungen, etc. beitragen kann. Meine Empfehlung ist

definitiv das Vorantreiben der Replikation von hybriden Modellen. Ich würde vorschlagen, dass das Fisch-Plankton-Modell in einer NetLogo ähnlicheren Sprache repliziert wird, um festzustellen, inwiefern sich die Replikationsergebnisse von jenen zu GAMA unterscheiden. Eventuell kann somit eine weitere Beantwortung zum Thema Replikation, Softwareabhängigkeit und Variabilität der Ergebnisse gegeben werden.

Zum Schluss habe ich noch einmal alle Herausforderungen zusammengefasst, die während der Replikation des Fisch-Plankton-Modells aufgetreten sind und den daraus gewonnenen Erkenntnissen, siehe Tabelle 12.

Tabelle 12: Aufgetretene Herausforderungen im Replikationsprozess

Herausforderung	Erkenntnis
Die Modellbeschreibung passt nicht mit der Modellbeschriftung überein.	Ohne der direkten Kommunikation mit den Modellherausgeberinnen und Modellherausgebern wäre eine Sicherstellung der falschen Beschriftung nicht möglich gewesen, was dazu geführt hätte, dass das Modellverhalten des replizierten Modells aufgrund der textuellen anderen Beschreibung des System, ein anderes als das Ausgangsmodell gewesen wäre. **Lehre:** Die Kommunikation mit den Herausgeberinnen und Herausgebern des Ausgangsmodells ist wertvoll und trägt zur Klärung von Missverständnissen sowie zur effizienteren Umsetzung bei.
Die Reproduktion von Schulen ist im Paper anders formuliert als im Quellcode.	**Lehre:** Ohne dem Quellcode zum Ausgangsmodell wäre nicht festgestellt worden, dass die Formel eine andere ist. Wenn möglich, sollte zusätzlich zur konzeptionellen Dokumentation der Quellcode angefordert werden, da er ein nützliches Instrument für den Abgleich in der Umsetzung darstellt.
Die Anzahl der See-Zellen differiert in GAMA zu NetLogo. Die Ursache dafür ist, dass das zur Verfügung gestellte .asc File zu groß ist, um in GAMA geladen werden zu können. Durch das Resamplen des originalen .asc File auf die vorgegebene 200x200 Zellengröße, differiert die Anzahl der See-Zellen um 1,2%.	**Lehre:** Es sollten, wenn möglich, die Rasterfiles in einer geringen Filegröße zur Verfügung gestellt werden, um Schwankungen in der Auflösung, die durch den Resample-Prozess aufgrund der Abhängigkeit zum jeweiligen Bearbeitungs-Tool verursacht werden, verhindert werden. Besonders wie in diesem Fall, wenn mittels der im Raster stattfindenden Prozesse Einfluss auf das

	Modellverhalten (Entwicklung der Fische in Abhängigkeit zur Planktonverfügbarkeit pro See-Zelle) nimmt.
Aufgrund der built-in Architektur in GAMA und dem Pendant dazu in NetLogo ist es oftmals schwierig nachzuvollziehen, wie die genauen Funktionsweisen und Auswirkungen der jeweiligen Statements sind. Dasselbe gilt für die Übernahme von Methoden aus der jeweiligen Plattform-Bibliothek. Als Beispiel kann die Diffusion als fertig definiertes Statement sowie das Flocking in NetLogo angeführt werden.	Lehre: Ein 1:1 Vergleich aller plattformeigener Statements ist nicht möglich. Es kann mittels Plausibilitätsprüfungen festgestellt werden, ob sie in ihrer Funktionsweise ähnlich oder gleich sind, indem die jeweiligen Resultate in ihren Wertebereichen verglichen werden.
Im ersten Anlauf des Replikationsprozesses wurde die Schul-Spezies in GAMA mittels der multi-level Architektur umgesetzt, bis als Resultat ein komplett unterschiedliches Modellverhalten zu jenem in NetLogo festgestellt wurde. Im zweiten Anlauf des Replikationsprozesses wurde die multi-level Architektur durch denselben listenbasierten Ansatz wie in NetLogo ersetzt.	Lehre: Die Implementierung des multi-level Ansatzes hatte mehr als 3 Monate Entwicklungszeit in Anspruch genommen. Es ist noch immer unklar, ob es richtig oder falsch ist, den Schul-Agenten mittels multi-level Architektur umzusetzen oder mittels Array. Der multi-level Architekturansatz würde sogar mehr dem objektorientierten Paradigma GAMAs entsprechen. In der Wissenschaftsliteratur wird die Umsetzung mithilfe von plattformspezifischen Tools empfohlen. Aufgrund des Fehlens einer Vorgabe, wie eine Replikation zu erfolgen hat, gibt dieses Thema noch Raum für Diskussionen in weiterführenden Studien.
Die Zeitschritte einer Simulation werden in NetLogo via ticks gemessen, was eine computerlesbare Zeiteinheit darstellt. Hingegen in GAMA wird die Zeit pro Zyklus mit Steps gemessen, was der Realzeit von 1 Sekunde entspricht. Die Dauer wie lange eine Simulation in einer Plattform in Anspruch nimmt ist daher unterschiedlich. Die reale Simulationszeit ist in GAMA viel länger als in NetLogo.	Lehre: Bevor mit einer Replikation begonnen wird, ist es ratsam sich mit dem plattformspezifischen Toolkit auseinanderzusetzen, um potenzielle Abweichungen in den Ergebnissen vorab einschätzen zu können.

Die Aufteilung der Datenausleitung mittels batch-Experiment in GAMA ist nur mit max. 5 Runs möglich. Die Datenmenge ist viel zu groß, weshalb die Ausleitung mit mehr als 5 Runs nicht möglich ist.	**Lehre:** Die Simulation mit einer höheren Anzahl an komplexen Agenten ist in GAMA weit langsamer als in NetLogo. Inwiefern es an der ineffizienten Umsetzung der NetLogo-Modellstruktur in GAMA liegt, ist unklar. Im weiteren wäre eine Replikation des Fisch-Plankton-Modells losgelöst von den Vorgaben im Quellcode ratsam, um festzustellen, ob eine Umsetzung in GAMA-spezifischer Sprache eine effizientere Datenausleitung ermöglicht. Die Befehle, die als Optimierungsmechanismen auf der GAMA-Homepage angeführt wurden, wurden im Code eingesetzt. Es können trotzdem nicht mehr als 5 Runs parallel ausgeleitet werden.
Flocking macht das Modellverhalten ab einer gewissen Individuenzahl sehr langsam.	**Lehre:** Bevor ein fertiges Skript eingebunden wird, ist darauf prüfen, welche Auswirkungen es auf die Geschwindigkeit des Systems nimmt.
Simulation reagiert bei unterschiedlichen Softwareversionen anders. Die Simulation konnte bei der GAMA-Vorversion nur bis zu 40 Jahre durchgeführt werden.	**Lehre:** Es sollte im Hinterkopf behalten werden, dass die Systemversionen einen Einfluss auf das Modellverhalten nehmen. Modelle sollten daher in verschiedenen Versionen der Software ausgeführt werden, um zu prüfen, inwiefern die Versionen einen Einfluss auf das Modell nehmen und wie dadurch die Replikation sichergestellt werden kann.

7. Literaturverzeichnis

AUGSTEN, S. 2020. *Was ist eine Dependency?* [Online]. Available: https://www.dev-insider.de/was-ist-eine-dependency-a-899057/ [Accessed].

AXELROD, R. Advancing the Art of Simulation in the Social Sciences. *In:* CONTE, R., HEGSELMANN, R. & TERNA, P., eds. Simulating Social Phenomena, 1997// 1997 Berlin, Heidelberg. Springer Berlin Heidelberg, 21-40.

AXTELL, R., AXELROD, R., EPSTEIN, J. M. & COHEN, M. D. 1996. Aligning simulation models: A case study and results. *Computational & mathematical organization theory,* 1**,** 123-141.

BAJRACHARYA, K. & DUBOZ, R. 2013. *Comparison of three agent-based platforms with a simple epidemiological model (WIP)*.

BELL, A. R., ROBINSON, D. T., MALIK, A. & DEWAL, S. 2015. Modular ABM development for improved dissemination and training. *Environmental Modelling & Software,* 73**,** 189-200.

BENDER, R. & LANGE, S. D. 2001a. Was ist der p-Wert? *Deutsche Medizinische Wochenschrift,* 126.

BENDER, R. & LANGE, S. D. 2001b. Was ist ein Konfidenzintervall. *Deutsche Medizinische Wochenschrift,* 132.

BENNINGHAUS, H. 2007. *Deskriptive Statistik*, Springer-Verlag.

BLEYMÜLLER, J., WEISSBAC, R. & DÖRRE, A. 2020. *statistik für Wirtschaftswissenschaftler*, Vahlen.

BLUNK, A. 2019. Simulationssprachen-Effiziente Entwicklung und Ausführung.

BOLES, D. & BOLES, C. 2004. *Objektorientierte Programmierung spielend gelernt mit dem Java-Hamster-Modell*, Springer.

DONKIN, E., DENNIS, P., USTALAKOV, A., WARREN, J. & CLARE, A. 2017. Replicating complex agent based models, a formidable task. *Environmental Modelling & Software,* 92**,** 142-151.

DU PREL, J.-B., HOMMEL, G., RÖHRIG, B. & BLETTNER, M. 2009. Konfidenzintervall oder p-Wert. *Deutsches Aerzteblatt,* 106**,** 335-339.

EASLEY, R. W., MADDEN, C. S. & DUNN, M. G. 2000. Conducting Marketing Science: The Role of Replication in the Research Process. *Journal of Business Research,* 48**,** 83-92.

EIRUND, H. 2013. *Objektorientierte programmierung*, Springer-Verlag.

ERDFELDER, E. U., ROLF 2018. Zur Methodologie von Replikationsstudien. *Psychologische Rundschau.,* 69 (2018) 1**,** 3-21

FOWLER, M. 2018. *Refactoring*, Addison-Wesley Professional.

FREY, H. C. & NIESSE, G. 2001. *Monte-carlo-simulation: quantitative risikoanalyse für die versicherungsindustrie*, Herbert C. Frey.

FÜLLSACK, M. 2010. Die Simulation komplexer Systeme Forschen in der Von-Neumann-Galaxie. *MITTEILUNGEN DES INSTITUTS FÜR WISSENSCHAFT UND KUNST,* 65.

GRAEME, P. 2013. Who Needs Replication? *CALICO Journal,* 30, No. 1**,** 10-15.

GRAY, R. & WOTHERSPOON, S. 2012. Increasing model efficiency by dynamically changing model representations. *Environmental Modelling & Software,* 30**,** 115-122.

GRIMM, V. 2019. Ecological Models: Individual-Based Models. *Encyclopedia of Ecology.*

GRIMM, V., AUGUSIAK, J., FOCKS, A., FRANK, B. M., GABSI, F., JOHNSTON, A. S. A., LIU, C., MARTIN, B. T., MELI, M., RADCHUK, V., THORBEK, P. & RAILSBACK, S. F. 2014. Towards better modelling and decision support: Documenting model development, testing, and analysis using TRACE. *Ecological Modelling,* 280**,** 129-139.

GRIMM, V., BERGER, U., BASTIANSEN, F., ELIASSEN, S., GINOT, V., GISKE, J., GOSS-CUSTARD, J., GRAND, T., HEINZ, S. K., HUSE, G., HUTH, A., JEPSEN, J. U., JØRGENSEN, C., MOOIJ, W. M., MÜLLER, B., PE'ER, G., PIOU, C., RAILSBACK, S. F., ROBBINS, A. M., ROBBINS, M. M., ROSSMANITH, E., RÜGER, N., STRAND, E., SOUISSI, S., STILLMAN, R. A., VABØ, R., VISSER, U. & DEANGELIS, D. L. 2006. A standard protocol for describing individual-based and agent-based models. *Ecological Modelling,* 198**,** 115-126.

GRIMM, V., RAILSBACK, S. F., VINCENOT, C. E., BERGER, U., GALLAGHER, C., DEANGELIS, D. L., EDMONDS, B., GE, J., GISKE, J. & GROENEVELD, J. 2020. The ODD protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism. *Journal of Artificial Societies and Social Simulation,* 23.

HEATH, B., HILL, R. & CIARALLO, F. 2009. A survey of agent-based modeling practices (January 1998 to July 2008). *Journal of Artificial Societies and Social Simulation,* 12**,** 9.

HIMME, A. 2007. *Gütekriterien der Messung: Reliabilität, Validität und Generalisierbarkeit*, Springer.

HUSTON, M., DEANGELIS, D. & POST, W. 1988. New Computer Models Unify Ecological Theory. *BioScience,* 38**,** 682-691.

KÄHLER, W.-M. 2013. *Statistische Datenanalyse: Verfahren verstehen und mit SPSS gekonnt einsetzen*, Springer-Verlag.

KLEINDORFER, G. B. & GANESHAN, R. 1994. *the Philosophy of Science and Validation in Simulation*.

KUNKEL, M. 2013. *Semantische Versionsnumerierung 2.0.0* [Online]. Available: https://blog.selfhtml.org/2013/nov/08/semantische-versionsnumerierung-2-0-0 [Accessed].

LAHRES, B. R. M., G. 2009. *Objektorientierte Programmierung: Einstieg und Praxis*, Rheinwerk Computing.

LANGE, S. & BENDER, R. 2007. Was ist ein Signifikanztest? Allgemeine Aspekte. *DMW-Deutsche Medizinische Wochenschrift,* 132**,** e19-e21.

LATOUR, B. 1979. Steve W oolgar. *Laboratory Life: the Construction of Scientific Facts*.

LEGENDI, R. & GULYAS, L. 2012. Replication of the Macro ABM Model. *CRISIS, working paper*.

LOUIE, M. A. & CARLEY, K. M. 2008. Balancing the criticisms: Validating multi-agent models of social systems. *Simulation Modelling Practice and Theory,* 16**,** 242-256.

MARTIN, R. & SCHLÜTER, M. 2015. Combining system dynamics and agent-based modeling to analyze social-ecological interactions—an example from modeling restoration of a shallow lake. *Frontiers in Environmental Science,* 3**,** 66.

MCCULLOUGH, B. D. 2009. Open Access Economics Journals and the Market for Reproducible Economic Research. *Economic Analysis and Policy,* 39**,** 117-126.

MÜLLER, C. & DENECKE, L. 2013. *Stochastik in den Ingenieurwissenschaften: Eine Einführung mit R*, Springer Berlin Heidelberg.

MÜLLER-GRONBACH, T., NOVAK, E. & RITTER, K. 2012. *Monte Carlo-Algorithmen*, Springer-Verlag.

OLARU, D., PURCHASE, S. & DENIZE, S. Using docking/replication to verify and validate computational models. 18th World IMACS/MODSIM Congress, 2009. Citeseer, 4432-4438.

PARRY, H. R. & EVANS, A. J. 2008. A comparative analysis of parallel processing and super-individual methods for improving the computational performance of a large individual-based model. *Ecological Modelling,* 214**,** 141-152.

PLANING, P. 2021. *Statistik Grundlagen.*, Patrick Planing.

POPPER, K. 2005. *The logic of scientific discovery*, Routledge.

RAAB, R., LENGER, K., STICKLER, D., GRANIGG, W. & LICHTENEGGER, K. 2022. An Initial Comparison of Selected Agent-Based Simulation Tools in the Context of Industrial Health and Safety Management. *Proceedings of the 2022 8th International Conference on Computer Technology Applications.* Vienna, Austria: Association for Computing Machinery.

RAILSBACK, S. F., LYTINEN, S. L. & JACKSON, S. K. 2006. Agent-based Simulation Platforms: Review and Development Recommendations. *Simulation,* 82**,** 609 - 623.

RAND, W. & WILENSKY, U. 2006. Verification and validation through replication: A case study using Axelrod and Hammond's ethnocentrism model. *North American Association for Computational Social and Organization Sciences (NAACSOS)***,** 1-6.

RASCH, B., FRIESE, M., HOFMANN, W. & NAUMANN, E. 2021. Inferenzstatistik. *Quantitative Methoden 1.* Springer.

REYNOLDS, C. W. 1987. Flocks, Herds, and Schools- A Distributed Behavioral Model. *ComputerGraphics,* 21.

RICHIARDI, M., LEOMBRUNI, R., SAAM, N. & SONNESSA, M. 2006. A Common Protocol for Agent-Based Social Simulation. *Journal of Artificial Societies and Social Simulation,* 9.

RICHMOND, B. 2001. *An Introduction to*

Systems Thinking.

RUDOLF, M., & KUHLISCH, W. 2008. *Biostatistik: Eine Einführung für Biowissenschaftler* München, Pearson Studium.

SCHEFFER, M. B., J. M.; DEANGELIS, D. L.; ROSE, K. A.; VAN NES, E. H. 1995. Super-individuals a simple solution for modelling large

populations on an individual basis. *Ecological Modelling,* 80**,** 161-170.

SCHMID, S., ZINGG, A., BIBER, P., & BUGMANN, H. 2006. Evaluation of the forest growth model SILVA along an elevational gradient in Switzerland. *European Journal of Forest Research,* 125(1)**,** 43-55.

SCHMOLKE, A., THORBEK, P., DEANGELIS, D. L. & GRIMM, V. 2010. Ecological models supporting environmental decision making: a strategy for the future. *Trends in Ecology & Evolution,* 25**,** 479-486.

SEAGREN, C. W. 2015. A Replication and Analysis of Tiebout Competition Using an Agent-Based Computational Model. *Social Science Computer Review,* 33**,** 198-216.

SEISTOCK, D., BUNINA, A. & ADEN, J. 2020. Der t-, Welch-und U-Test im psychotherapiewissenschaftlichen Forschungskontext. *Empfehlungen für Anwendung und Interpretation. SFU Forschungsbulletin,* 8**,** 87-105.

SIBLY, R., GRIMM, V., MARTIN, B., JOHNSTON, A., KUŁAKOWSKA, K., TOPPING, C., CALOW, P., NABE-NIELSEN, J., THORBEK, P. & DEANGELIS, D. 2013. Representing the acquisition and use of energy by individuals in agent-based models of animal populations. *Methods in Ecology and Evolution,* 4**,** 151-161.

SWINERD, C. M., K. R. 2012. Design classes for hybrid simulations involving agent-based and system dynamics models. *Simulation Modelling Practice and Theory,* 25**,** 118-133.

THEIS C., K., W. 2002. *Grundlagen der Monte Carlo Methode* [Online]. Technische Universität Graz. Available: https://itp.tugraz.at/MML/MonteCarlo/MCIntro.pdf [Accessed 24.04.2023 2023].

VIEHOF, M. W., H. 2017. Modellvalidierung im Anwendungsbereich der Fahrdynamiksimulation.

VINCENOT, C. E., GIANNINO, F., RIETKERK, M., MORIYA, K. & MAZZOLENI, S. 2011. Theoretical considerations on the combined use of System Dynamics and individual-based modeling in ecology. *Ecological Modelling,* 222**,** 210-218.

WALLENTIN, G. 2017. Hybrid Fish-plankton model ODD protocol.

WALLENTIN, G. 2020. *UNIGIS module: Spatial Simulation* [Online]. https://unigis-salzburg.github.io/Opt_Spatial-Simulation/. [Accessed].

WALLENTIN, G. & NEUWIRTH, C. 2016. Dynamic hybrid modelling: Switching between AB and SD designs of a predator-prey model. *Ecological Modelling,* 345**,** 165-175.

WANG, H., ZHANG, J. & ZENG, W. 2018. Intelligent simulation of aquatic environment economic policy coupled ABM and SD models. *Science of The Total Environment,* 618**,** 1160-1172.

WILENSKY, U. R., W. 2007. Making Models Match: Replicating an Agent-Based Model. *Journal of Artificial Societies and Social Simulation,* 10.

WILENSKY, U. R., W. 2015. *An Introduction to Agent-Based-Modeling. ,* Cambridge, The MIT Press.

ZHANG, J. & ROBINSON, D. T. 2021. Replication of an agent-based model using the Replication Standard. *Environmental Modelling & Software,* 139**,** 105016.

8. Anhang

8.1 Code Ausgangsmodell (NetLogo)

```
extensions [gis]
breed [fish a-fish ]
breed [schools school ]
fish-own [
  age
  sex
  nearest-neighbor
  flockmates
  school-ID
  E-ingested
  E-survive
  E-growth
  E-repro
]
schools-own [
  school-ID
  fish-stock
  juvenile-fish-stock
  adult-fish-stock
  fish-growth-lst
]
patches-own [
  lake              ; values of lake raster (attersee.asc dataset): 0 = no lake; 1 = lake
  plankton-bio  ; plankton stock
  plankton-eaten ; plankton consumed by fish agents and school SDs
]
globals [
  cohesion-flag  ; helper variable to tag schools
  school-counter   ; helper variable to set school ID
  counter ; = integer of ticks (ticks have small rounding errors if governed by advance-dt)
;; spatial variables
```

```
  scale-factor   ; conversion factor between metre NetLogo distance units
  attersee-dataset  ; GIS shapefile of lake
  lake-patches   ; all patches inside the lake
  coast-patches  ; all patches adjacent to land
;; SD parameters
  carrying-capacity    ; max plankton per lake patch
  plankton-growth-rate ; growth rate of plankton
  max-fish            ; threshold for max fish in a school
  dt                  ; SD time increment
;; flocking parameters
  vision
  minimum-separation
  max-align-turn
  max-cohere-turn
  max-separate-turn
]
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; SETUP PROCEDURES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;; Setup AB model ;;;;;;;;;;;;;;;;
to setup
  ca
  reset-ticks
  reset-timer
  set counter 0
  ;;;;;; load & display GIS data
  ; loads Attersse.shp into the global variable “Attersee-dataset”
  set attersee-dataset gis:load-dataset "netlogodata/attersee.asc"
  ; defines the extent of the ‘World’ by the bounding box of a dataset
  gis:set-world-envelope (gis:envelope-of attersee-dataset)
  ; calculate scale conversion factor
  let envelope gis:envelope-of attersee-dataset
  let y-extent item 3 envelope - item 2 envelope
  ; scale-factor = 200 (with a world setting of 45 x 90 cells) -> 1 patch is 200m x 200m
  set scale-factor y-extent / (max-pycor + 1)
  ; Copy values from Attersee-dataset into an lake patch variable (lake)
  gis:apply-raster attersee-dataset lake

  ; color lake patches blue and coast patches green
  ask patches [ifelse lake = 1 [set pcolor blue][set pcolor 68]]
  set lake-patches patches with [pcolor = blue]
```

```
  ask lake-patches [if count neighbors with [pcolor = 68] > 0 [set pcolor 97]]
  set coast-patches lake-patches with [pcolor = 97]
  ;;;;;;; setup fish agents
  ;; create and distribute fish
  make-fish initial-number-of-fish
  set cohesion-flag true
  ; define threshold for max fish in a school
  set max-fish 5000
  ; set flocking parameters in metre and degree
  set vision 700
  set minimum-separation 10
  set max-align-turn 2
  set max-cohere-turn 8.5
  set max-separate-turn 3
  ;; start system dynamics setup procedure
  sd-setup
end
;; create and distribute fish (at setup and when switched from SD -> ABM)
to make-fish [number-of-fish]
  create-fish number-of-fish [
    let a one-of lake-patches
    set shape "fish"
    set color orange
    set size 1
    setxy [pxcor] of a [pycor] of a
    set age random-float 6
    set sex random 2  ;female = 0, male = 1
    set color one-of base-colors
    set school-ID 0
    set E-repro 0
  ]
  ask fish [
    find-flockmates
    ]
end
;;;;;;;;;;;;;; Setup SD model ;;;;;;;;;;;;;;;;;
to sd-setup
  ;; the SD increment
  set dt 1
```

```
  ; total plankton biomass in Attersee = app. up to 2300t according to the literature
  set carrying-capacity 2300 / count lake-patches
  ; plankton doubles in app 9 days (growth-rate = 0.01) - estimated value
  set plankton-growth-rate 0.01
  ;; initialize plankton biomass in patches stochastically
  ask lake-patches [ set plankton-bio ((2000 + random 300) / count lake-patches) ]
end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; GO PROCEDURES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;; Control AB model ;;;;;;;;;;;;;;;;;
to go
  ;; performance tracking
  if counter = 0 [reset-timer]
  if counter mod (365 / dt) = 0 [show timer]
  ; execute ABM actions daily
  if counter mod (1 / dt) = 0 [
    ;; execute for fish in ABM mode
    ask fish [
      ; fish turn to flock with others
      flock
      ; move 20m
      move ( 20 )
      ; calculate energy budget
      update-energy
      ;; reproduce, if eligible
      if sex = 0 [reproduce]
      ; grow older
      set age age + 1 / 365  ;one day (=1/365)
      ; die of age or if the fish has no more energy reserves (starvation)
      if age > 6 or E-repro < 0 [ die ]
    ]
    ; agents cluster to schools
    identify-schools
    ; fish join school within sight (vision)
    ask fish [join-school]
    ; school SDs move
    ask schools [
      separate-from-other-schools ; turns to plankton-rich cells and separates from other schools
      move ( 20 )  ; move 20m
    ]
```

```
    ;; too small schools fall apart
    ;if a school is below a certain min-threshold, the school transitions into single agents
    foreach sort schools with [fish-stock < school-threshold] [
      make-fish [ fish-stock ] of ?
      ask ? [die]
    ]
    ;; limit school to a max. size; surplus fish are emitted as agents
    limit-school-size
    ;; diffuse plankton: 10% per month (30 days)
    diffuse plankton-bio (0.1 / 30)
    ; return plankton from non-lake patches back to the lake
    ask patches with [lake != 1 and plankton-bio > 0] [
      ask min-one-of neighbors with [lake = 1] [plankton-bio] [set plankton-bio (plankton-bio + [plankton-bio] of
myself)]
      set plankton-bio 0
      ]
    display-biomass
  ]
  ;; call SD procedures at each dt time increment
  sd-go
  update-plots
  ; stop conditions
  if scenario = "100 years" [
    if ticks >= (365 * 100) [ show timer show ticks stop ]
  ]
  if scenario = "fishing" [
    if (sum [fish-stock] of schools + count fish) > 500 [fishing 25]
  ]
  set counter counter + 1
end
;;;;;;;;;;;;; Control SD model ;;;;;;;;;;;;;;;;
;; Step through the system dynamics model by performing next iteration of Euler's method.
to sd-go
  ;;;;;;; plankton biomass growth in patches
  ask lake-patches [
    ;; compute variable and flow values once per step
    let local-biomass-growth biomass-growth ; logistic growth
    let local-biomass-removed biomass-removed  ; sum of eaten plankton by fish-ABM and school-SD
    ;; update stock values
```

```
    ;; use temporary variables so order of computation doesn't affect result.
    let new-plankton-bio ( plankton-bio + local-biomass-growth - local-biomass-removed )
    if new-plankton-bio > 0
      [set plankton-bio new-plankton-bio]
  ]
  ;;;;;; fish population growth in schools
  ask schools [
    ;; death because of age: fish die at the age of 6 years (2190 days)
    ;set fish-growth-lst reset-lst fish-growth-lst 2190
    ;; compute variable and flow values once per step
    fish-growth
    fish-die-of-age
    fish-starve

    ;; update juvenile fish stock
    let new-juvenile-fish-stock ( sum sublist fish-growth-lst ((2190 - 1460) * (1 / dt)) (2190 * (1 / dt)) )
    set juvenile-fish-stock new-juvenile-fish-stock
    if juvenile-fish-stock < 0 [show "juvenile stock is negative!"]
    ;;update adult fish stock
    let new-adult-fish-stock ( sum sublist fish-growth-lst ((0) * (1 / dt)) ((2190 - 1461) * (1 / dt)) )
    set adult-fish-stock new-adult-fish-stock
    if adult-fish-stock < 0 [show "adult stock is negative!"]
    ;;update fish stock
    let new-fish-stock ( juvenile-fish-stock + adult-fish-stock )
    set fish-stock new-fish-stock
    ; min size of 0.3 to be visible
    set size 0.3 + (fish-stock / (school-threshold * 200))
  ]
  tick-advance dt
end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; HELPER PROCEDURES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; report 180° turn true, if fish approaches the shore
to-report turn-180
  ifelse [lake] of patch-ahead (20 / scale-factor) != 1 [report true] [report false]
end
;; update energy budget of fish agents
; plankton controls the energy budget of fish in the following order: maintenance, growth and reproduction (Sibly et al., 2013)
to update-energy
```

```
  set E-ingested 2.7 * (plankton-bio / carrying-capacity) / (plankton-bio / carrying-capacity + 2)  ;; after Sibly 2013
  ; the normalisation constant is set to 0.005; body mass is estimated to be 100g * age
  set E-survive 0.005 * (age * 100)^ 0.75                    ;; after Sibly 2013
  ; under opimal conditions, after survival costs: 50% of the available energy goes into growth and 50% into reproduction
  set E-growth 0.5 * (0.9 - E-survive)                       ;; maximum E-ingest = 0.9
  ; The reproduction energy includes maturation and reproduction; it is cumulative over the lifespan of a fish.
  if age >= 4 [set E-repro E-repro + E-ingested - E-survive - E-growth ]
end
;; fish reproduction
;probability of reproduction when annual offspring is 5 (=5/365)
;at least 2 flockmates, age >= 4 and female gender are prerequisite for reproduction
to reproduce
  if (random-float 1 < (5 / 365)) and (count flockmates > 5) and (age >= 4) and (sex = 0) and (E-repro > 15) [
    set E-repro E-repro - 15
    hatch-fish 1 [
      let a one-of coast-patches
      set sex random 2
      set age 0
      setxy [pxcor] of a [pycor] of a
      set color one-of base-colors
      set school-ID 0
      set E-repro 0
    ]
  ]
end
;; identify agent fish schools and replace by school agent
to identify-schools
  ;; tag 'new' fish
  if cohesion-flag [
    ask turtles [ set color one-of base-colors ]
    set cohesion-flag false
  ]
  ;; set my color to the color of my nearest flockmate
  ask fish [ if any? flockmates [ set color [color] of min-one-of flockmates [distance myself] ] ]
  ;; link with flockmates in a radius of 300m
  ask fish [ create-links-with flockmates in-radius (300 / scale-factor) ]
  ;; every 5th tick, the fish in each school get a unique school-ID (otherwise, we end up with fish schools of the exact threshold size)
```

```
if counter mod 5 = 0 [
  ask fish [ set-school-ID school-counter ]
]
;; replace identified fish schools with a single school agent
foreach sort fish with [school-ID > 0] [
 if ? != nobody [
   ask ? [
     let x count fish with [school-ID = [school-ID] of ?]
     if x >= school-threshold [
       hatch-schools 1 [
         let school-fish fish with [school-ID = [school-ID] of myself]
         ;create fish growth list of zeros with length max possible age of fish
         set fish-growth-lst[]
         repeat 2190 * (1 / dt)[
           set fish-growth-lst lput 0 fish-growth-lst
         ]
         ;parameterise fish growth list based on age of school fish
         foreach sort school-fish [
           ;get position in fish growth list to add new fish
           let pos (length fish-growth-lst - (int ([age] of ? * 365 * (1 / dt))))
           if pos = 0[ set pos 1 ]
           ;create new fish growth list by adding fish to school (+1)
           set fish-growth-lst replace-item (pos - 1) fish-growth-lst ((item (pos - 1) fish-growth-lst) + 1)
         ]
         ; update school stocks
         set juvenile-fish-stock sum sublist fish-growth-lst ((2190 - 1460) * (1 / dt)) (2190 * (1 / dt))
         set adult-fish-stock sum sublist fish-growth-lst ((0) * (1 / dt)) ((2190 - 1461) * (1 / dt))
         set fish-stock juvenile-fish-stock + adult-fish-stock
         ; define display settings of school
         set shape "circle"
         set size fish-stock
         ; kill fish agents
         ask school-fish [die]
       ]
     ]
   ]
 ]
]
;; reset school-ID for the next tick
```

```
  ask fish [set school-ID 0]
  ;; reset links
  ask links [die]
en
;; assign a unique school ID
to set-school-ID [tag]
  if school-ID = 0 [
    set school-ID tag
    ask link-neighbors [
      set school-ID tag
      ask link-neighbors [set school-ID tag]
    ]
    set school-counter school-counter + 1
  ]
end
;; schools swim towards plankton, but separate from other schools
to separate-from-other-schools
  ; turn towards the most plankton-rich cell
   turn-towards (towards max-one-of neighbors [plankton-bio]) 15
  ; separate from nearby schools
  if any? other schools in-radius (200 / scale-factor)  [
    turn-away ([heading] of min-one-of other schools in-radius (200 / scale-factor) [distance myself]) 25
  ]
end
;; agents move
; the speed parameter may differ between single fish and school agents
to move [dist]
  ; move [dist] metre per day or turn around at shore
  ifelse [lake] of patch-ahead ((2 * dist) / scale-factor) != 1
    [set heading heading + 180 ]
    [fd dist / scale-factor]
end
;; 'vacuum cleaner' function: if a fish agent is close to a school, it joins the school SD and the agent is deleted
to join-school
  ; look for nearby school
  if any? schools with [fish-stock < max-fish] in-radius (vision / scale-factor)[
    ;select a school in vision radius for join
    let target-school one-of schools with [fish-stock < max-fish] in-radius (vision / scale-factor
    ;get fish growth list of selected school
```

```
    let target-fish-lst [fish-growth-lst] of target-school
    ;get position in fish growth list to add new fish
    let pos (length target-fish-lst - int (age * 365 * (1 / dt)))
    if pos = 0 [ set pos 1 ]
    ;add fish to school by updating fish growth list (+1)
    set target-fish-lst replace-item (pos - 1) target-fish-lst ((item (pos - 1) target-fish-lst) + 1)
    ask target-school [set fish-growth-lst target-fish-lst]
    ;add newly joined fish to school stocks
    let new-fish self
    ask target-school [
      set juvenile-fish-stock sum sublist fish-growth-lst ((2190 - 1460) * (1 / dt)) (2190 * (1 / dt))
      set adult-fish-stock sum sublist fish-growth-lst ((0) * (1 / dt)) ((2190 - 1461) * (1 / dt))
      set fish-stock juvenile-fish-stock + adult-fish-stock
    ]
    ;delete fish agent
    die
  ]
end
;; if a school grows beyond a certain max-threshold, fish agents leave the school until the school is below the
max-threshold again.
to limit-school-size
  ; count total number of fish in lake that are beyond the max-threshold
  let x int sum [ fish-stock ] of schools with [fish-stock > max-fish] - count schools with [fish-stock > max-fish] *
max-fish
  ; create that many fish agents
  repeat x [make-fish 1]

  ; delete that many fish from too large school SDs
  foreach sort schools with [fish-stock > max-fish] [
    ask ? [
      ; remove random fish from the fish-growth-lst and create agent fish instead
      while [fish-stock - max-fish > 0] [
        ; draw a random position in the growth list
        let ran random (365 * 6 * (1 / dt))
        ; remove selected fish-stock from list (-> replace random position with 0)
        set fish-growth-lst replace-item ran fish-growth-lst 0
        set fish-stock sum fish-growth-lst
      ]
      set juvenile-fish-stock ( sum sublist fish-growth-lst ((2190 - 1460) * (1 / dt)) (2190 * (1 / dt)) )
```

```
      set adult-fish-stock ( sum sublist fish-growth-lst ((0) * (1 / dt)) ((2190 - 1461) * (1 / dt)) )
    ]
  ]
end
;; fishing removes all fish except for a few 'remaining' fish
to fishing [remaining-fish]
  ; exit this function if there are already less fish than the ones that should remain
  if count fish + sum [fish-stock] of schools < remaining-fish [stop]
  ; if there are enough fish agents: kill all but the number that should remain
  if count fish > remaining-fish [
    while [count fish > remaining-fish][
      ask one-of fish [die]
    ]
  ]
  ; if there are not enough single fish agents: create some
  if count fish < remaining-fish [
    make-fish (remaining-fish - count fish)
  ]
  ; kill all schools
  ask schools [die]
end
;;;;;;;;;;;;;;;;;;;;;; FLOCKING Behaviour (from the NetLogo Library) ;;;;;;;;;;;;;;;;;;;;;;
to flock  ;; fish procedure
  find-flockmates
  if any? flockmates
    [ find-nearest-neighbor
      ifelse distance nearest-neighbor < (minimum-separation / scale-factor)
        [ separate ]
        [ align
          cohere ] ]
end
to find-flockmates  ;; fish procedure
  set flockmates other fish in-radius (vision / scale-factor)
end
to find-nearest-neighbor ;; fish procedure
  set nearest-neighbor min-one-of flockmates [distance myself]
end
;;; SEPARATE
to separate  ;; fish procedure
```

```
  turn-away ([heading] of nearest-neighbor) max-separate-turn
end
;;; ALIG
to align  ;; fish procedure
  turn-towards average-flockmate-heading max-align-turn
end
to-report average-flockmate-heading  ;; fish procedure
  ;; We can't just average the heading variables here.
  ;; For example, the average of 1 and 359 should be 0,
  ;; not 180.  So we have to use trigonometry.
  let x-component sum [dx] of flockmates
  let y-component sum [dy] of flockmates
  ifelse x-component = 0 and y-component = 0
    [ report heading ]
    [ report atan x-component y-component ]
end
;;; COHERE
to cohere  ;; fish procedure
  turn-towards average-heading-towards-flockmates max-cohere-turn
end
to-report average-heading-towards-flockmates  ;; a-fish procedure
  ;; "towards myself" gives us the heading from the other a-fish
  ;; to me, but we want the heading from me to the other a-fish,
  ;; so we add 180
  let x-component mean [sin (towards myself + 180)] of flockmates
  let y-component mean [cos (towards myself + 180)] of flockmates
  ifelse x-component = 0 and y-component = 0
    [ report heading ]
    [ report atan x-component y-component ]
end
;;; HELPER PROCEDURES
to turn-towards [new-heading max-turn]  ;; fish procedure
  turn-at-most (subtract-headings new-heading heading) max-turn
end
to turn-away [new-heading max-turn]  ;; fish procedure
  turn-at-most (subtract-headings heading new-heading) max-turn
end
;; turn right by "turn" degrees (or left if "turn" is negative),
;; but never turn more than "max-turn" degrees
```

```
to turn-at-most [turn max-turn]  ;; fish procedure
  ifelse abs turn > max-turn
    [ ifelse turn > 0
        [ rt max-turn ]
        [ lt max-turn ] ]
    [ rt turn ]
End
;;;;;;;;;;;;;;;;;;;;;;;; System Dynamics - report flows ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; fish inflow: exponential growth (growth depends on fish stock)
to fish-growth
  ;; compute growth increment: annual offspring of female fish is 5 (= 5 fish per 365 days)
  let fish-growth-increment  ((adult-fish-stock / 2) * (5 / 365) * dt * [plankton-bio] of patch-here / carrying-capacity)
  ; append growth increment to list
  set fish-growth-lst lput fish-growth-increment fish-growth-lst
end
;; fish outflow I (die because of age): linear decline - die after 6 years (2190 days)
; implemented as system delay
to fish-die-of-age
   ; remove death increment from list
   set fish-growth-lst remove-item 0 fish-growth-lst
end
;; fish outflow II (starvation): decline is coupled to plankton stock
to fish-starve
  ;; starvation: each element in the fish-growth-lst is multiplied by the local starvation-rate
  ;let starvation-rate  ( 1 - [plankton-bio] of patch-here / carrying-capacity) * 0.00825
  let starvation-rate  ( 1 - ((plankton-bio / carrying-capacity) ^ 0.033 )) * 0.24 ; 0.15 -> 640,000; 0.2 -> 576,000; 0.23 -> 522,000; 0.24 -> ; 0.25 -> 463,000
  ; update fish growth list
  set fish-growth-lst map [? * (1 - starvation-rate * dt)] fish-growth-lst
end
;; Report value of biomass inflow: resource limited, logistic growth
to-report biomass-growth
  report ( plankton-bio * plankton-growth-rate * ( 1 - plankton-bio / carrying-capacity ) ) * dt
end
;; Report value of biomass outflow: decline coupled to fish stock
; plankton eaten (= sum of fish in ABM and school-SDs  x  daily food rate)
to-report biomass-removed
  report ((count fish-here + sum [fish-stock] of schools-here) * 0.0000027 * dt)
```

```
end
;;;;;;;;;;;;;;;;;;;;;;;;;;;; DISPLAY ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; display plankton biomass stocks (spatial SD)
to display-biomass
  let min-biomass 1500 / count lake-patches
  let max-biomass 2300 / count lake-patches
  ask lake-patches [ set pcolor scale-color blue plankton-bio min-biomass max-biomass ]
end
; Written by: Gudrun Wallentin and Christian Neuwirth
; Refer to the following publication to cite the model:
;    Wallentin, G., and Neuwirth, C. (2016) "Dynamic hybrid modelling: switching between AB and SD designs of a predator-prey model". Ecological Modelling (under review)
; This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0
```

8.2 Code repliziertes Modell (GAMA)

```
/* Master-Thesis: Replication of the NetLogo File 004_agent-spatstock - superagent-spatstock */
/* Source: Wallentin, Gudrun & Neuwirth, Christian. (2016). Dynamic hybrid modelling: Switching between AB and SD designs of a predator-prey model. Ecological Modelling. 345. 10.1016/j.ecolmodel.2016.11.007. */
/* Author: Katharina Rybnicek */

model replication_004
global {
	// load & display GIS data
	// 200x200 per cell
	file attersee <- grid_file("../includes/attersee.txt");
	geometry shape <- envelope(attersee);

	// global lake variables
	list<CA> lake_cells;
	list<CA> coast_cells;
	geometry lake;
	geometry coast;
	int number_lake_cells;

	// global plankton variables
	float t; // current time of the equation system integration
	float SD_timestep <- 0.5;
	float plankton_growth_rate;
	float diff_rate <- 0.1 / 30; // plankton diffusion rate 10 % per month (0.1 / 30)
```

```
		float carrying_capacity_cell;


		// global fish variables
		int init_nb_fish <- 25;
		float max_speed <- 20.0; // default value of step = 1.0; https://gama-platform.org/wiki/ManipulateDates
		float starvatedFishes <- 0.0;
		float reproducedFishes <- 0.0;

		// flocking parameters
		// default value for distance is meter; https://gama-platform.org/wiki/UnitsAndConstants
		float vision <- 700.0;
		float vision_school <- 300.0;
		float min_separation <- 10.0;
		float max_align_turn <- 2.0;
		float max_cohere_turn <- 8.5;
		float max_separate_turn <- 3.0;
		float max_separate_turn_school <- 25.0;
		float min_separation_school <- 200.0;

		// global school variables
		int school_min <- 50; // treshold for min. number of fisch in school
		int school_max <- 5000; // threshold for max. number of fish in school

		// global time variables
		string scenario <- "100 years"; // simulation period

		// performance parameters
		float start_time;
		float performance;
		float current_time;
		float previous_time;
		float elapsed_minutes;

		// output data variables
		int total_fishes_in_school <- 0;
		int total_fishes_outside_school <- init_nb_fish;
		int total_fishes <- total_fishes_outside_school;
		int total_school_species <- 0;
```

```
int all_fishes <- 0;

// setup model
init {
	start_time <- machine_time; // performance tracking
	do init_lake;
	do init_fish;
}

// initialise lake
action init_lake {
	ask CA {
		is_lake <- false;
		is_coast <- false;
		if (grid_value = 1) {
			is_lake <- true;
			if sum(neighbors collect (each.grid_value)) < 8 {
				is_coast <- true;
			}
		}

		// color lake patches blue and coast patches white
		if (is_lake = true) {
			color <- #blue;
		}
		if (is_coast = true) {
			color <- #white;
		}
		if (not (is_lake or is_coast)) {
			color <- #gray;
		}
	}
	lake_cells <- CA where (each.is_lake);
	coast_cells <- CA where (each.is_coast);
	number_lake_cells <- length(lake_cells);
	lake <- geometry(lake_cells);
	coast <- geometry(coast_cells);
	plankton_growth_rate <- 0.01;
```

```
		// originial attersee.asc file could not be processed by GAMA, because it was too large
		// solution: resampled the original atteresse.asc with ArcGIS (200x200)
		// number of lake cells differs therefore from NetLogo about 1.2 % due to geometry scaling
		carrying_capacity_cell <- 2300.0 / number_lake_cells;
		ask lake_cells {
			plankton <- ((2000 + rnd(300)) / number_lake_cells);
		}
	}

	// diffuse plankton: 10% per month (30 days)
	reflex new_value {
		ask (lake_cells) {
			diff_p <- plankton * diff_rate;
		}
	}

	reflex diff_plankton {
		diffuse var: diff_p on: lake_cells avoid_mask: true;
	}

	// create and distribute fish
	action init_fish {
		do createFishes(init_nb_fish);
	}

	// fish join school within sight (vision)
	reflex capture_fish_to_school {
		ask school_species {

		// select all fishes in radius for join
			list<fish_species> fish_species_in_range <- fish_species at_distance vision_school;
			loop fish over: fish_species_in_range {
				if (fish_stock < school_max) {
					do capture_fish(fish);
				}
			}
		}
	}
```

```
	// too small schools fall apart
	// if a school is below a certain min-threshold, the school transitions into single agents
	reflex dissolve_schools {
		int fishesToCreate <- 0;
		ask school_species {
			if (fish_stock < school_min) {
				fishesToCreate <- fishesToCreate + int(fish_stock);
				do die;
			}
		}
		do createFishes(fishesToCreate);
	}

	// create and distribute fish
	action createFishes (int number) {
		create fish_species number: number {
			location <- any_location_in(one_of(lake));
			speed <- max_speed;
			age <- rnd(6.0);
			if flip(0.5) {
				sex <- "f";
			} else {
				sex <- "m";
			}
			E_repro <- 0.0;
		}
	}

	// create a school if fish has enough flockmates
	reflex check_if_create_school {
		ask fish_species {
			list<fish_species> fish_species_list_to_be_captured <- flockmates where
(!dead(each));
			if (length(fish_species_list_to_be_captured) > school_min - 1) { // fish belongs to
school as well
				create school_species {
					// set location of school to the fish location
					location <- myself.location;
					speed <- max_speed; // the speed parameter may differ between
single fish and school agents
```

```
                                        add myself to: fish_species_list_to_be_captured;
                                        loop fish over: fish_species_list_to_be_captured {
                                                do capture_fish(fish);
                                        }
                                }
                        }
                }
        }

        // stop condition
        reflex check_100_years when: (scenario contains "100 years" and cycle > 36500) {
                do pause;
        }

        // performance tracking
        reflex compute_data {
                float timesnap <- machine_time;
                elapsed_minutes <- (machine_time - start_time) / (1 #minute * 1000);
                total_fishes_in_school <- int(sum_of(school_species, each.fish_stock));
                total_fishes_outside_school <- length(fish_species);
                total_fishes <- total_fishes_in_school + total_fishes_outside_school;
                total_school_species <- length(school_species);
        }

        reflex output_data {
save [cycle, int(self), starvatedFishes, reproducedFishes, length(fish_species), total_fishes_in_school,  sum(CA
collect each.plankton), length(school_species), fish_species count (each.age >= 4 and each.age < 6)] to:
"result/fish.csv" type: "csv" rewrite: false;
        }
}
species fish_species skills: [moving] {
        float size <- 150.0;
        rgb color <- #orange;
        float age <- #years;
        string sex; // "f" or "m"

        // energy variables
        float E_ingested;
        float E_survive;
```

```
	float E_growth;
	float E_repro;

	// flocking variables
	list<fish_species> flockmates <- [] update: fish_species at_distance vision;
	fish_species nearest_neighbour;
	int avg_head;
	int avg_twds_mates;
	// move 20m / day
	reflex swim {
		do flock;
		point next_location <- {self.location.x + 2 * speed * step * cos(heading), self.location.y + 2 * speed * step * sin(heading)};
		if lake_cells overlapping next_location != [] {
			do move;
		} else {
			heading <- heading + 150 + rnd(60);
		}
	}

	// calculate energy budget
	// plankton controls the energy budget of fish in the following order: maintenance, growth and reproduction (Sibly et al., 2013)
	reflex update_energy {
		// Identify current cell of fish
		CA cell <- CA(location);
		if (cell = nil) { //cell was sometimes nil...why?
			return;
		}
		E_ingested <- 2.7 * (cell.plankton / carrying_capacity_cell) / (cell.plankton / carrying_capacity_cell + 2); // after Sibly 2013

		// the normalisation constant is set to 0.005; body mass is estimated to be 100g * age (in years)
		E_survive <- 0.005 * (age * 100) ^ 0.75; // after Sibly 2013

		// under opimal conditions, after survival costs: 50% of the available energy goes into growth and 50% into reproduction
		E_growth <- 0.5 * ((0.9) - E_survive); // maximum E-ingest = 0.9
```

```
		// the reproduction energy includes maturation and reproduction; it is cumulative over the lifespan of a fish
		if age >= 4 {
			E_repro <- E_repro + E_ingested - E_survive - E_growth;
		}
	}
	// reproduce, if eligible
	// probability of reproduction when annual offspring is 5 (=5/365)
	// at least 2 flockmates, age >= 4 and female gender are prerequisite for reproduction
	reflex reproduce {
		if ((rnd(1.0) < (5 / 365)) and (length(flockmates) > 5) and (age >= 4) and (sex = "f") and (E_repro > 15)) {
			E_repro <- E_repro - 15;
			create fish_species number: 1 {
			// locate in one of coast patches
				location <- any_location_in(one_of(coast));
				if flip(0.5) = true {
					sex <- "f";
				} else {
					sex <- "m";
				}
				age <- 0.0;
				E_repro <- 0.0;
				speed <- max_speed;
			}
		}
	}
	// grow older
	// die of age or if the fish has no more energy reserves (starvation)
	reflex age_and_die {
		age <- age + 1 / 365;
		if (age > 6 or E_repro < 0) {
			do die;
		}
	}
	// ********** FLOCKING **********
	action flock {
		if (not empty(flockmates)) {
			do find_nearest_neighbour;
```

```
                if (distance_to(self, nearest_neighbour) < min_separation) {
                        do separate;
                } else {
                        do align;
                        do cohere;
                }
                do move speed: speed;
        }
}
action find_nearest_neighbour {
        nearest_neighbour <- flockmates closest_to self;
}

// reflex used when the separation is applied to change the velocity of the boid
action separate {
        do turn_away(nearest_neighbour.heading, max_separate_turn);
}

// reflex to align the boid with the other boids in the range
action align {
        avg_head <- int(avg_mate_heading());
        do turn_towards(float(avg_head), max_align_turn);
}

// reflex to apply the cohesion of the boids group in the range of the agent
action cohere {
        avg_twds_mates <- int(avg_heading_towards_mates());
        do turn_towards(float(avg_twds_mates), max_cohere_turn);
}

action turn_away (float new_heading, float max_turn) {
        float subtract_headings <- heading - new_heading;
        if (subtract_headings < -180) {
                subtract_headings <- subtract_headings + 360;
        }
        if (subtract_headings > 180) {
                subtract_headings <- subtract_headings - 360;
        }
```

```
		do turn_at_most(int(subtract_headings), max_turn);
	}

	float avg_mate_heading {
		if (flockmates = nil) {
			write ("Flockmates of fish " + name + " is NIL!");
		}
		list<fish_species> flockmates_insideShape <- flockmates where (each.destination != nil);
		if (flockmates_insideShape = nil) {
			flockmates_insideShape <- [];
		}
		float x_component <- sum(flockmates_insideShape collect (each.destination.x -
each.location.x));
		float y_component <- sum(flockmates_insideShape collect (each.destination.y -
each.location.y));
		if (x_component = 0 and y_component = 0) {
			return heading;
		} else {
			return atan2(y_component, x_component);
		}
	}

	action turn_towards (float new_heading, float max_turn) {
		float subtract_headings <- new_heading - heading;
		if (subtract_headings < -180) {
			subtract_headings <- subtract_headings + 360;
		}
		if (subtract_headings > 180) {
			subtract_headings <- subtract_headings - 360;
		}
		do turn_at_most((int(subtract_headings)), max_turn);
	}

	float avg_heading_towards_mates {
		// note: 0-heading direction in GAMA is east instead of north!
		float x_comp <- mean(flockmates collect (cos(towards(self.location, each.location))));
		float y_comp <- mean(flockmates collect (sin(towards(self.location, each.location))));
		if (x_comp = 0 and y_comp = 0) {
			return heading;
```

```
                } else {
                        return atan2(y_comp, x_comp);
                }
        }

        action turn_at_most (int turn, float max_turn) {
                if abs(turn) > max_turn {
                        if turn > 0 {
                        //right turn
                                heading <- heading + max_turn;
                        } else {

                        //left turn
                                heading <- heading - max_turn;
                        }
                } else {
                        heading <- heading + turn;
                }
        }

        // fish visualisation settings
        aspect default {
                draw line([location, destination]) end_arrow: 2 color: color;
        }
}
species school_species skills: [moving] {
        CA richest_cell; // richest plankton cell
        school_species nearest_school;

        float fish_stock; // total count of fishes in school
        float juvenilefish_stock; // count of juvenile fishes in school
        float adultfish_stock; // count of adult fishes in school
        list<float> fish_growth_list <- list_with(2190, 0.0); // used a list (same as in netlogo) to simulate a
school.
          // An agent based version would be too slow.
        // capture a fish into school and remote the original agent from the model (die)
        action capture_fish (fish_species fish) {
                ask fish {
                        // get the position in fish growth list to add new fish
```

```
                        // the position is a value for the age of the fish
                        int index <- int(age * 365);
                        //add fish to school by updating fish growth list (+1)
                        put myself.fish_growth_list[index] + 1 at: index in: myself.fish_growth_list;
                        // delete fish agent
                        do die;
                }
                // fish_growth_list contains juveline fishes at the beginning (contrary to netlogo)
                juvenilefish_stock <- sum_of(fish_growth_list[0::1460], each);
                adultfish_stock <- sum_of(fish_growth_list[1461::2190], each);
                fish_stock <- adultfish_stock + juvenilefish_stock;
        }
        reflex swim {
                do turn_towards_richest_cell;
                point next_location <- {self.location.x + 2 * speed * step * cos(heading), self.location.y + 2 *
speed * step * sin(heading)};
                if lake_cells overlapping next_location != [] {
                        do move;
                } else {
                        heading <- heading + 150 + rnd(60);
                }
        }
        // fish inflow: exponential growth (growth depends on fish stock)
        reflex reproduce {
                float r_fish <- 5 / 365;
                float fish <- adultfish_stock / 2;
                float fishGrowthIncrement <- ((adultfish_stock / 2) * (5 / 365) * CA(location).plankton /
carrying_capacity_cell);

                // reproduced fishes will be added at the beginning of the list
                // the list is shiftet to the right
                add fishGrowthIncrement to: fish_growth_list at: 0;

                // the oledst fishes (at the right) will be removed
                remove last(fish_growth_list) from: fish_growth_list;

                // compute stocks
                do computeStocks;
                reproducedFishes <- reproducedFishes + fishGrowthIncrement;
```

```
                write ("reproduced Fishes: " + reproducedFishes );
        }
        action computeStocks {
                // compute stocks
                juvenilefish_stock <- sum_of(fish_growth_list[0::1460], each);
                adultfish_stock <- sum_of(fish_growth_list[1461::2190], each);
                fish_stock <- adultfish_stock + juvenilefish_stock;
        }
        // fish outflow II (starvation): decline is coupled to plankton stock
        reflex starvate_fishes {
                float starvation_rate <- (1 - ((CA(location).plankton / carrying_capacity_cell) ^ 0.033)) *
0.24;
                list<float> newfish_growth_list <- [];

                // update fish_growth_list by reducing the value according to starvation rate
                loop s over: fish_growth_list {
                        newfish_growth_list << s * (1 - starvation_rate);
                }
                // [ 0.2, 0.5, 0, 0, 0, 1.0 ]
                // [ 0.18, 0.45, 0, 0, 0, 0.9 ] starvation rate 10 % -> 0.1

                // update list
                fish_growth_list <- newfish_growth_list;

                // compute stocks
                do computeStocks;
                starvatedFishes <- starvatedFishes + starvation_rate;
                write ("starvated Fishes: " + starvatedFishes );
        }
        // swim towards plankton
        // a fish school can correct its direction up to 15° to turn towards the most plankton rich cell
        action turn_towards_richest_cell {
                richest_cell <- CA(location).neighbors with_max_of (each.plankton);
                if (richest_cell = nil) {
                        return;
                }
                point cell <- richest_cell.location;
                float deltaX <- cell.x - location.x;
                float deltaY <- cell.y - location.y;
```

```
		float deg <- atan2(deltaY, deltaX) - heading; // in radians
		if (deg < -180) {
			deg <- deg + 360;
		}
		if (deg > 180) {
			deg <- deg - 360;
		}
		do school_turn_at_most(deg, 15.0);
	}
	action school_turn_at_most (float turn, float max_turn) {
		if abs(turn)  > max_turn {
			if turn > 0 {

				// right turn
				heading <- heading + max_turn;
			} else {

				// left turn
				heading <- heading - max_turn;
			}
		} else {
			heading <- heading + turn;
		}
	}
	// avoid collision with other schools
	// a fish school can correct its direction up to another 25° to turn avoid collision with other schools
	reflex collision_detect when: (length(school_species) > 1) {
		do find_nearest_school;
		float distance <- distance_to(self, nearest_school);
		if (distance < min_separation_school) {
			do separate_school;
		}
	}
	// schools separate from other schools
	action separate_school {
		do turn_away_school(nearest_school.heading, max_separate_turn_school);
	}
	action turn_away_school (float new_heading, float max_turn) {
		float subtract_headings <- heading - new_heading;
```

```
                if (subtract_headings < -180) {
                        subtract_headings <- subtract_headings + 360;
                }
                if (subtract_headings > 180) {
                        subtract_headings <- subtract_headings - 360;
                }
                do school_turn_at_most(subtract_headings, max_turn);
        }
        // limit school to a max. size; exceeding fishes are emitted as agents
        // if a school grows beyond a certain max-threshold, fish agents leave the school until the school is below the max-threshold again
        reflex emit_fishes when: (fish_stock > school_max) {
                int fishestoCreate <- int(fish_stock) - school_max;
                // taken from NetLogo. Note: not efficient
                loop while: (fish_stock > school_max) {
                        int index <- rnd(0, 2189);
                        fish_growth_list[index] <- 0;
                        fish_stock <- sum_of(fish_growth_list, each);
                }
                create fish_species number: fishestoCreate {
                        location <- any_location_in(one_of(lake));
                        speed <- max_speed;
                        age <- rnd(6.0);
                        if flip(0.5) = true {
                                sex <- "f";
                        } else {
                                sex <- "m";
                        }
                        E_repro <- 0.0;
                }
                // compute stocks
                do computeStocks;
        }
        // look for nearby school
        action find_nearest_school {
                nearest_school <- list(school_species) closest_to self;
        }
        aspect school_default {
                draw circle(50) color: #yellow;
```

```
	}
}
grid CA file: attersee neighbors: 8 {
	bool is_lake;
	bool is_coast;
	float diff_p <- 0.0;
	float plankton <- 0.0;

	// plankton biomass growth in patches
	// logistic growth
	equation logistic_growth {
		// evolution of plankton during an integration time step
		diff(plankton, t) = plankton * plankton_growth_rate * (1 - plankton / carrying_capacity_cell);
	}
	reflex colour_CA {
		if (is_lake = true) {
			color <-
			rgb((255 * (plankton - 0.95 * carrying_capacity_cell) / (0.05 *
carrying_capacity_cell)), 255, (255 * (plankton - 0.95 * carrying_capacity_cell) / (0.05 *
carrying_capacity_cell)));
		} else {
			color <- #gray;
		}
	}

	// compute plankton eaten by all fishes belonging to this cell
	reflex plankton_feed {
		if (is_lake = true) {
			plankton <- plankton - ((length(fish_species overlapping self) +
sum_of(school_species overlapping self, each.fish_stock)) * 0.0000027);
		}
	}
	reflex solving when: true {
		if (is_lake = true) {
			solve logistic_growth method: "rk4" step_size: SD_timestep;
		}
		if (plankton < 0.0) {
			plankton <- 0.0;
		}
	}
```

```
}
experiment Main_Experiment type: gui {
	parameter "Number of fish: " var: init_nb_fish;
	reflex compacting when: every(50 #cycle) {
		do compact_memory();
	}
	output {
		monitor "number of lake patches" value: length(lake_cells);
		monitor "number of coast patches" value: length(coast_cells);
		monitor "years" value: int(floor(cycle / 365)) refresh: every(365 #cycle);
		monitor "number of fish" value: total_fishes;
		monitor "number of Schools" value: total_school_species;
		monitor "number of fish in schools " value: total_fishes_in_school;
		monitor "number of fish outside schools " value: total_fishes_outside_school;
		monitor "number of adult fishes" value: fish_species count (each.age >= 4 and each.age <
6);
		monitor "plankton biomass" value: sum(CA collect each.plankton);
		// map
		display map {
			grid CA;
			species fish_species aspect: default;
			species school_species aspect: school_default;
		}
		// charts
		display chart_fish_population refresh: every(10 #cycle) {
			chart "Fish-Plankton-Model" type: series x_label: "cycle" y_label: "size" size: {0.5,
0.5} position: {0, 0} {
				data "fish population" value: total_fishes color: #red;
			}
		}
		display chart_school refresh: every(50 #cycle) {
			chart "Fish-Plankton-Model" type: series x_label: "cycle" y_label: "size" size: {0.5,
0.5} position: {0, 0} {
				data "school" value: total_school_species color: #blue;
			}
		}
		display chart_fishes_outside_school refresh: every(10 #cycle) {
			chart "Fish-Plankton-Model" type: series x_label: "cycle" y_label: "size" size: {0.5,
0.5} position: {0, 0} {
```

```
                                        data "fishes outside school" value: total_fishes_outside_school color:
#brown;
                                }
                        }
                        display chart_fishes_inside_school refresh: every(10 #cycle) {
                                chart "Fish-Plankton-Model" type: series x_label: "cycle" y_label: "size" size: {0.5,
0.5} position: {0, 0} {
                                        data "fisches in school" value: total_fishes_in_school color: #black;
                                }
                        }
                        display chart_plankton refresh: every(10 #cycle) {
                                chart "Fish-Plankton-Model" type: series x_label: "cycle" y_label: "size" size: {0.5,
0.5} position: {0, 0} {
                                        data "plankton" value: sum(CA collect each.plankton) color: #green;
                                }
                        }


                        display chart_adult_fishes refresh: every(50 #cycle) {
                                chart "Fish-Plankton-Model" type: series x_label: "cycle" y_label: "size" size: {0.5,
0.5} position: {0, 0} {
                                        data "adult fishes not in school" value: fish_species count (each.age >= 4
and each.age < 6) color: #blue;
                                }
                        }
                        display chart_performance refresh: every(50 #cycle) {
                                chart "Fish-Plankton-Model" type: series x_label: "cycle" y_label: "size" size: {0.5,
0.5} position: {0, 0} {
                                        data "performance" value: elapsed_minutes color: #black;
                                }
                        }
                }
}
experiment my_batch_experiment type: batch repeat: 5 keep_seed: false until: (cycle = 36500) {
//parameter 'school treshold' var: school_min among: [25,75,100];
}
```

8.3 Flocking-Modell

```
/**
* Name: flocking
* Author: Gudrun WALLENTIN, Dept. of Geoinformatics - Z_GIS, University of Salzburg
* Description: Reynold's boids model tranfserred to GAMA according to the algorithm used in NetLogo
* Tags: boids model, flocking model
```

```
*/

model flocking

global torus: false {
		//  parameters
		int number_of_fish <- 25 min: 3 max: 60;
		float min_separation <- 3.0  min: 0.1  max: 10.0 ;
		int max_separate_turn <- 5 min: 0 max: 20;
		int max_cohere_turn <- 5 min: 0 max: 20;
		int max_align_turn <- 8 min: 0 max: 20;
		float vision <- 30.0  min: 0.0  max: 70.0 ;

		// initialise model
		init {
		// create and distribute fish
		create fish number:number_of_fish;
		}
}

// declare agents, cells and their behaviour
		//  fish agents
		species fish skills: [ moving ] {
		// fish attributes
		float size <- 2.0;
		rgb colour <- #black;

		 // flocking variables
		 list<fish> flockmates ;
		 fish nearest_neighbour;
		 int avg_head;
		 int avg_twds_mates ;

		// flocking movement
		reflex flock {
		// in case all flocking parameters are zero wander randomly
			if (max_separate_turn = 0 and max_cohere_turn = 0 and max_align_turn = 0 ) {
				do wander amplitude: 120;
				}
				// otherwise compute the heading for the next timestep in accordance to
my
					flockmates
				else {
				// search for flockmates
				do find_flockmates ;
				// turn my heading to flock, if there are other agents in vision
				if (not empty (flockmates)) {
					do find_nearest_neighbour;
					if (distance_to (self, nearest_neighbour) < min_separation) {
						do separate;
					}
					else {
						do align;
						do cohere;
					}
					// move forward in the new direction
						do move;
					}
```

```
                                        // wander randomly, if there are no other agents in vision
                                        else {
                                                do wander amplitude: 120;
                                        }
                                }
        }

                //flockmates are defined spatially, within a buffer of vision
                action find_flockmates {
                    flockmates <- ((fish overlapping (circle(vision))) - self);
                }

                //find nearest neighbour
                action find_nearest_neighbour {
                    nearest_neighbour <- flockmates with_min_of(distance_to (self.location,
each.location));
                }

        // separate from the nearest neighbour of flockmates
        action separate  {
                do turn_away (nearest_neighbour towards self, max_separate_turn);
        }

        //Reflex to align the boid with the other boids in the range
        action align  {
                avg_head <- avg_mate_heading () ;
                 do turn_towards (avg_head, max_align_turn);
        }

        //Reflex to apply the cohesion of the boids group in the range of the agent
        action cohere  {
                avg_twds_mates <- avg_heading_towards_mates ();
                do turn_towards (avg_twds_mates, max_cohere_turn);
        }

        //compute the mean vector of headings of my flockmates
        int avg_mate_heading {
                list<fish> flockmates_insideShape <- flockmates where (each.destination != nil);
                float x_component <- sum (flockmates_insideShape collect (each.destination.x -
                   each.location.x));
                float y_component <- sum (flockmates_insideShape collect (each.destination.y –
                   each.location.y));
                //if the flockmates vector is null, return my own, current heading
                if (x_component = 0 and y_component = 0) {
                        return heading;
                }
                //else compute average heading of vector
                else {
                        // note: 0-heading direction in GAMA is east instead of north! -> thus +90
                        return int(-1 * atan2 (x_component, y_component) + 90);
                }
        }

        //compute the mean direction from me towards flockmates
        int avg_heading_towards_mates {
                float x_component <- mean (flockmates collect (cos (towards(self.location,
each.location))));
```

```
                float y_component <- mean (flockmates collect (sin (towards(self.location,
each.location))));
                //if the flockmates vector is null, return my own, current heading
                if (x_component = 0 and y_component = 0) {
                        return heading;
                }
                //else compute average direction towards flockmates
                else {
                // note: 0-heading direction in GAMA is east instead of north! -> thus +90
                return int(-1 * atan2 (x_component, y_component) + 90);
                }
        }

        // cohere
        action turn_towards (int new_heading, int max_turn) {
                int subtract_headings <- new_heading - heading;
                if (subtract_headings < -180) {subtract_headings <- subtract_headings + 360;}
                if (subtract_headings > 180) {subtract_headings <- subtract_headings - 360;}
                 do turn_at_most ((subtract_headings), max_turn);
        }

        // separate
        action turn_away (int new_heading, int max_turn) {
                int subtract_headings <- heading - new_heading;
                if (subtract_headings < -180) {subtract_headings <- subtract_headings 360;}
                if (subtract_headings > 180) {subtract_headings <- subtract_headings -360;}
                do turn_at_most ((-1 * subtract_headings), max_turn);
        }

        // align
        action turn_at_most (int turn, int max_turn) {
                if abs (turn) > max_turn {
                if turn > 0 {
                        //right turn
                        heading <- heading + max_turn;
                        }
                else {
                        //left turn
                        heading <- heading - max_turn;
                        }
                }
                else {
                        heading <- heading + turn;
                }
}
        // fish visualisation settings
        // default arrow
           aspect arrow {
                draw line([location, {location.x - size * cos(heading), location.y - size * sin(heading)}])
                   begin_arrow: 1 color: colour;
                }
                // alternative arrow
                aspect arrow2 {
                if (destination != nil) {
                        draw line([location, destination]) end_arrow: 2 color: colour;
                        }
                }
                // additional vision buffer
```

```
                        aspect buffer {
                        draw location + circle (vision) color: colour ;
                        }
        }
// simulation settings
experiment simulation type:gui {

        //user defined parameters
        parameter 'iniatial number of animals' var: number_of_fish;
        parameter 'Max cohesion turn' var: max_cohere_turn ;
        parameter 'Max alignment turn' var:  max_align_turn;
        parameter 'Max separation turn' var: max_separate_turn;
        parameter 'Minimal Distance'  var: min_separation;
        parameter 'Vision' var: vision;
        output {
                // map
                display map  {
                        species fish aspect: arrow;
                        //species fish aspect: buffer transparency:0.8;
                }
        }
}
```