Agentenbasierte Modellierung und Beschreibungsmittel im Multi-Agenten Simulationskontext

Daniel Glake

Hochschule für Angewandte Wissenschaften Hamburg,
Berliner Tor 7, Hamburg 20099, Deutschland
daniel.glake@haw-hamburg.de

Zusammenfassung. Diese Arbeit befasst sich mit dem Forschungsfeld der Multi-Agenten Programmierung im Kontext von Simulationen durch die Vorstellung einer Reihe von Simulationsbeschreibungssprachen, mit Details besonderer Konstrukte und deren verfolgtem Beschreibungsansatz. Es werden Sprachen aus verstärkt aktuellen Arbeiten betrachtet und ein kleiner Ausschnitt aus älteren Arbeiten. Dazu wird zunächst auf das Problem der bisherigen Simulationsbeschreibung eingegangen sowie Gründe für die eigene Sprachentwicklung genannt, die mit dem Übebrlick eine Ausgangsbasis für das zukünftige Forschungsfeld der Sprachentwicklung, im Rahmen des an der Hochschule für Angewandte Wissenschaften Hamburg entwickelten Multi-Agenten Simulationsframeworks MARS, bildet. Konkrete Forschungsfragen und Arbeiten, mit denen sich befasst werden soll, sind am Schluss genannt.

Schlüsselwörter: Multi-Agent System, Modellierungssprachen, Implementierung, Technologie,

Inhaltsverzeichnis

1 Einleitung

Agenten sind autonome Einheiten die sich in einer Umgebung befinden und darin interagieren. Betrachtet man die Entwicklung von Multi-Agenten-Systemen für Simulationen, haben sie trotz der zunehmenden Beliebtheit und des stärker werdenden Wissenschaftsgebietes der Agenten-basierten Softwareentwicklung mit Simulationsplattformen wie Swarm [MBLA13], ein wiederkehrendes Komplexitätsproblem bei tiefgreifenden Simulationsmodellen [Das14]. Dies wird mit typischen general-purpose Sprachen, in denen diese Plattformen gebaut wurden und mit denen Modelle entwickelt werden, zu einer stärker werdenden technischen Aufgabe, die aufgrund des fehlenden, technischen Hintergrundwissens des Modellierers zumeist erschwert wird, und außerdem keine wissenschaftliche Diskussionsgrundlage bietet, indem man sich über das entwickelte Modell und den Modellcode austauscht [SUZ+14]. Eine hohe Fehlerwahrscheinlichkeit, aufgrund falscher Anwendung etablierter Konzepte aus der Agentenbeschreibung im Modell und damit in den Simulationsergebnissen, führt zu fehlerhaften Schlussfolgerungen [BBD+06], die durch die breite Anwendbarkeit einzelner Ausführungsplattformen für verschiedene Bereiche, Folgen in jeder Teildisziplin (Politik, Chemie, Ökologie etc.) haben kann. Der Aufwand für die Verifikation und Validierung, d.h. die Sicherstellung, dass das Multiagenten-System (MAS) korrekt arbeitet, bleibt jedoch weiterhin notwendig [DF13].

Zusammen mit etablierten Sprachen wie GAML des GAMA Frameworks [GTG+13] und NetLogo als eines der bekanntesten Sprachen und Plattformen [TW04], oder gar abgrenzenden Domänen- spezifischen Sprachen versucht man diese fehlende Entwicklungsunterstützung zu reduzieren und über sprachliche Mittel für Verifikation und Validierung zu ermöglichen [MHS05], sowie Verständ-lichkeit und Benutzbarkeit des zu entwickelten Modells zu erhöhen [Das14]. Dem Modellierer wird damit eine komplexere Simulationsbeschreibung mit unterschiedlichen Betrachtungsebenen, z.B. aus der Multi-Layer Modellierung [GTG+13] ermöglicht, die zudem angepasst an die Beschreibungsart und Notation der Benutzerdomäne ist, und somit eines seiner wichtigsten Werkzeuge darstellt.

Diese Arbeit befasst sich mit der Art und Weise wie agentenbasierte Modellierung im Umfeld von Simulationen aussehen kann, und wie eine eigene sprachliche Umsetzung diese Modellierung unterstützen könnte. Hierzu geht diese Arbeit zunächst auf Aspekte der agentenbasierten Modellierung ein, indem eine historische Hintergrundbetrachtung, mit ersten Notwendigkeiten warum diese gebraucht wurden, vorgenommen wird. Es werden Gründe zur Entwicklungsentscheidung für eigene Sprachen betrachtet sowie die Vergleichskriterien für agentenorientierte Sprachen geschaffen. Die mögliche Sprachunterstützung wird mit Rücksicht auf bereits bestehende Sprachen untersucht, und als Überblick mit einer Menge ausgewählter, bereits existierender Modellierungsbeschreibungssprachen betrachtet, die in ihren Ansätzen stark differenzieren. Den Abschluss der Arbeit bildet ein Ausblick des zukünftigen Forschungsfelds im Rahmen von MARS, wobei unter anderem auf mögliche Forschungsfragen und das weitere Vorgehen eingegangen wird. Eine Erläuterung was das MARS System ist, wird anfänglich nach den vergleichenden Arbeiten und Konferenzen gegeben. Diese Arbeit befasst sich nicht mit den konkreten Ausführungsplattformen, die die angebotenen Konzepte durch Algorithmen und Datenstrukturen umsetzen. Für einen Überblick der verschiedenen Plattformen sei hierfür auf Teile der Arbeit von [BBD+06] und folgerichtig auf Arbeiten zum MARS System verwiesen [HWFTC15] (siehe Abschnitt 3). Ebenso wird nicht auf das ähnliche Gebiet der visuellen Modellierung über eine Graphische Benutzeroberfläche eingegangen, die mit Vorzug ebenfalls eine aktuelle Thematik darstellen, wie mit Aspekten der Arbeit von [BHB+10].

2 Vergleichende Arbeiten und Konferenzen

Wichtige aktuell, vergleichende Arbeiten im Bereich der Sprachen für Multi-Agenten-Systeme sind vor allem Beiträge aus [Das14], die speziell auf das aktuelle Interessengebiet der kognitiven Agenten eingeht, in die auch die vorgestellte Sprache Brahms (siehe Abschnitt 5.2) hineinfällt. Viele ältere Beschreibungsmittel, aber dennoch erwähnenswert, sind die Sprachen die in der Arbeit von [BBD+06] betrachtet werden, und hierbei noch eine Unterscheidung zwischen imperativen und deklarativen Sprachen sowie die Anforderungen und den Bedarf von Sprachmitteln darlegen (siehe Abschnitt 4.2). Die von [EU14] und [GTG+13] halbwegs kürzlich vorgestellten Sprachen gehen auch auf einen anderen wichtigen Aspekt

ein, der Beschreibung des eigentlichen Simulationsexperiments, die mit der Sprache GAML [GTG+13] in dieser Arbeit betrachtet wird (siehe Abschnitt 5.1). Der Beweggrund und die wichtige Rolle der Domänenspezifischen Sprachen zeigen sich auch durch den von Adeline Uhrmacher vorgetragenen Keynote Beitrag mit der Rolle der DSL in der Modellierung und Simulation auf der SIMULTEC 2015, eine der wichtigsten Wissenschaftskonferenzen im Bereich der Simulation und Modellierung. Daneben sind auch die jährliche Winter Simulation Conference (WSC) und die Sommer Simulation Conference (SCS) zu nennen.

3 MARS

Das in der Arbeit von [HWFTC15] vorgestellte Multi-Agenten Simulations Framework MARS ist ein verteiltes, als Modelling and Simulation as a Service konzipiertes System, zur Durchführung von Simulationen und Ergebnisanalysen. Es verfolgt den Ansatz des Individual Based Modelling [GBD+10] und trennt seine Modellierung in zwei Rollen. Einem eigentlichen Domänen-Modellierer, der das Wissen um die Fachlichkeit bereithält,und einem Modellentwickler, der das erdachte Simulationsmodell in entsprechenden ausführbaren Code, entgegen der MARS Suite und in der zugrundeliegenden general-purpose Sprache C#, verwirklicht. Diese Anforderung an Implementierung, Aspekte der Datenintegration des eigentlichen Modells sowie Ergebnisanalysen und Visualisierungen, werden bisher durch eine Reihe verschiedener Komponenten in einer Pipeline-ähnlichen Struktur, mit zu nennenden Komponenten wie SHUTTLE, GROUND, ROCK sowie des Ausführungskerns LIFE, geregelt. Agenten werden dort in Layern verteilt, dessen Grundgedanke aus der Geoinformatik stammt. Aktuell orientiert sich die MARS-Entwicklung an ein internationales Projekt zur Savannenökologie Südafrikas, bezeichnet als Ars Africae, das übersetzt Adaptive Resilience of Southern African Ecosystems bedeutet [Bus15]. Wesentliches Ziel ist die Erforschung der Savanne und deren Auswirkungen auf klimatische Veränderungen über mehrere Jahre hinweg, was mit der Adaptiven Resilienz insbesondere die Anpassungsfähigkeit auf äußere Einflüsse untersucht, sowohl kurzzeitige- als auch dauerhafte-Anpassungen [LBC+14].

4 Agentenbasierte Modellierung

Die Agentenbasierte Modellierung und Simulation (ABMS) ist wachsender Modellierungsansatz, der seit über 15 Jahren an Bedeutung zunimmt und dieser Trend aufgrund der steigenden Anzahl von Anwendungen, Fachbeiträgen, Konferenzen etc. bisher kein Ende vorhersieht [MN14]. Im Folgenden wird die Entstehungsgeschichte der agentenbasierten Simulation vorgestellt sowie Entwicklungsentscheidungen und Vergleichskriterien in Bezug auf agentenbasierte Simulationssprachen.

4.1 Entstehungsgeschichte

Verschiedene Disziplinen haben zur Entwicklung der agentenbasierten Simulationen beigetragen. Ende der 1940er Jahre haben Arbeiten von Neumann sich bereits mit der Betrachtung selbstreproduzierender Systeme befasst [MBLA13]. Die Idee zahlreiche interagierende Agenten in einem Computermodell zu repräsentieren, resultierte in der Definition von Zellularautomaten, bei dem John Conways Game of Life [MN14] wohl eines der bekanntesten zweidimensionalen Zellularautomaten ist [MN09], die eine Agenteninteraktion repräsentieren und darüber Informationen durch eine gemeinsame Umgebung verfügbar halten. Die Veränder-ung einfacher Regeln auf Zellebene haben zu unvorhergesehenen Strukturen auf der Makroebene geführt. Im Forschungsfeld komplexer Systeme hat die agentenbasierte Simulation starken Einfluss gezeigt und so in diesem Bereich und der anpassbaren Systeme zu einer neuen Betrachtungsform geführt, bei denen dortige Komponenten (Agenten) miteinander agieren können, variierendes Verhalten haben können und die Umgebung verändern können. Aus der Notwendigkeit solche komplexen Systeme besser zu analysieren wurde daraus das bekannte Swarm Framework, vom Santa Fe Institut [MBL+96] entwickelt, das neben dem ursprünglichen Verwendungszweck zudem zur Betrachtung der Emergenz und des Anpassungsvermögens von Syntheseprozessen biologisch-inspirierter Systeme geführt hat [Hol95]. Hieraus wiederum entstanden eine Reihe weiterer Anwendungsgebiete von Modellen für die Börsensimulation [AHL+96] bis zum Verhalten von einzelnen Zellen [EMN+05] [Das14]. Gemein haben all

diese Modelle, dass dort sogenannte nichtlineare Systeme untersucht werden, deren Ausgangssignal nicht immer proportional zum Eingangssignal passen. Agentenbasierte Modelle sind daher eine Möglichkeit zur Betrachtung nichtlinearer Systeme [SUZ+14].

4.2 Sprachentwicklungsentscheidung

Vor zehn Jahren war die Art und Weise, wie Multi-Agenten Systeme implementiert werden sollten, sehr stark auf die manuelle Programmierung des kompletten Agentensystems fokussiert, die mit eigenständigen agenten-orientierten Programmiersprachen oder Sprachkonstrukten vorgenommen oder mit Frameworks angeboten wurden [SUZ+14]. Diese sind nicht explizit für Multi-Agenten-basierte Simulationen gedacht, sondern werden vermehrt als dynamisches System beschrieben [BBD+06]. Das Design und der tatsächliche Code resultieren anschließend in einem höheren Aufwand zur Pflege des entwickelten Simulationsmodells [BHB+10]. Das ist auch dem geschuldet, dass zwar entwickelte Methodiken und Muster für MAS [SS14] in der konzeptionellen Phase berücksichtigt wurden, es jedoch zu Unterschieden in der tatsächlichen Implementierung kommt [BBD+06] [Das14]. Darauf bezogen, versuchen DSLs eine Abstraktion zu bieten und notwendige Engineering-Techniken zu integrieren, um die Domäne korrekt abbilden zu lassen [MHS05] und damit folgernder Erosion entgegenwirkt. Nach [MHS05] liegt die Entscheidung für eine DSL Entwicklung sehr nahe, wenn es eines oder mehre der nachfolgenden Anforderungen zu erfüllen gilt:

- **Notation**: Vereinfachte Notation durch Abstraktion komplexer Syntax und der Ersetzung durch eine naheliegende Sprache, die stärker an den geforderten Kontext angepasst ist, durch entsprechende Bezeichner und domänen-spezifische Operatoren. Insbesondere wenn die vorhandene Sprache sehr hardwarenah ist, liegt ein zusätzlicher Abstraktionsgrad sehr nah
- **Wiederkehrende Aufgaben**: Gerade sehr häufig vorkommende Operationen, die im Kontext der Domäne anfallen, wie im Falle der Agentenbewegung in einem umweltbedingten Modell, mit sich fortbewegenden Agenten, lassen sich diese wiederkehrenden Aufgaben als eigenständigen Teil der Sprache beschreiben, und als einsetzbares Muster hinterlegen. In Zusammenhang von MARS wären dies Bewegungsoperatoren, die beispielsweise im ArsAfricae Modell für einen konkreten Elefanten, eine konkrete Neupositionierung an eine bestimmte Koordinate der Karte vornimmt.
- **Domänen-Spezifische Analysen**: Die Prüfung und Optimierung des geschriebenen Programms werden in der DSL zum Teil vorgenommen, insbesondere wenn Festlegungen, die aus den jeweiligen Anwendungsdomänen, wie der Ökologie hervorgehen und dort bereits anerkannt sind [GBD+10], als Teil der Grammatik aufgenommen werden, und damit diese Fehlerquellen bereits aus der Entwicklung ausgeschlossen werden können.
- **System front-end**: Ergänzend zur Simulationskonfiguration von [EU14] lassen sich Anpassungen an eine Simulation und den Experimenten über eine DSL vereinfachen.
- **Erweiterung**: Neue Erkenntnisse einer Domäne oder Einbindung neuer Techniken im Umgang mit Simulationsmodellen, wie Verhaltensmodellierung durch Ergänzung eines neuen Agententyps nach der BDI-Charakterisierung, werden schneller an die betroffenen Simulationen herangetragen als durch zusätzliche Software.

4.3 Vergleichskriterien

Vergleichskriterien für Sprachen existieren je nach Anwendungsgebiet der Sprache [MHS05]. Im Bereich der Agenten-orientierten Programmierung haben sich, neben Funktionalität und Einfachheit der Sprache, einige weitere Kriterien etabliert, die gezielt auf Aspekte der Ausdrucksstärke und deren Integrierbarkeit abzielt. In Zusammenhang mit MARS sind hierbei vor allem Anforderungen an einen leichten Einstieg in die Sprache unumgänglich, derer , um damit eine breite, internationaler Interessengruppen möglicher Modellierer sowie verschiedene Anwendungsdomänen ansprechen zu können, wie es im Falle des ArsAfricae Projekts bereits geschieht. Angepasst auf diese Einarbeitungsanforderung, lassen Vergleichskriterien für eine zukünftige Modellierungssprachen, mit Kriterien nach [Das14] folgendermaßen unterscheiden:

- **Funktionalität**: Beschreibt die Vielfalt der Verhaltensstrukturen, die eine Sprache anbietet, wie reaktive Agenten, kognitive Agenten, besondere Denkstrukturen (Denkzustände) und soziale Fähigkeiten.

- **Kommunikation**: Beschreibt welche Kommunikationsmechanismen von der Sprache angeboten werden, und wie stark davon abstrahiert worden ist. Dazu zählt Unicast-, Multicast-, Broadcast-Kommunikation etc., die allesamt auch Teil des ACL Standards sind [Das14].
- **Einfachheit**: Gibt an wie gut und schnell man die Sprache benutzen und verstehen kann. Diese Einfachheit zielt in Multi-Agenten Systemen geradewegs auf den eingeschlagenen Modellierungsansatz ab, der von der Sprache vorgegeben wird [SUZ+14].
- **Exaktheit**: Gibt den Grad der beschreibbaren Semantik an sowie die Frage, wie die Sprache formalisiert wurde.
- **Ausdrucksfähigkeit**: Bezeichnet die Vielfalt der beschreibbaren agenten-orientieren Programme, die mit der Sprache entwickelt werden können, was auch die Erweiterungsfähigkeit der Sprache beinhaltet [BBD+06]
- **Umfang**: Beschreibt ob mit der Sprache neue Sprachkonstrukte gebaut werden können, die aus den Basiskomponenten entstanden sind.
- **Verifizierbarkeit**: Drückt aus ob der eingeschlagene Modellierungsansatz formal geprüft werden kann, oder von vornherein in der Sprachgrammatik fixiert ist, was zumal bei DSL's eine Einsatzmöglichkeit darstellt [DF13] [MHS05].
- **Software Engineering Prinzipien**: Software Engineering Konzepte, wie Vererbung, Kapselung, Modularität etc. können im Design der Sprache eingepflegt sein oder auch nicht.
- **Sprachintegration**: Gibt an ob und wie die Sprache in bereits etablierte Programmiersprachen wie Java integriert wird, oder ob sie auf anderem Weg die Möglichkeit bietet Funktionalitäten davon nutzen zu können.

Welche der Kriterien einen höheren Stellenwert erhält bleibt in der Verantwortung der Modellierungsdomäne. In Bezug auf das MARS System jedoch sind vor allem Kriterien, wie Software Engineering Prinzipien sowie Ausdrucksfähigkeit und Einfachheit ein wichtiges Merkmal, für eine mögliche MARS-DSL, da diese einerseits einen leichten Einstieg für internationale und stark interdisziplinäre Teams ermöglicht und darüberhinaus durch die hohe Integrationsanforderung verschiedener Anwendungsdomänen (Biologie, Ökologie, Katastrophenschutz etc.) [HWFTC15] eine größere Vielfalt bieten muss. Zugunsten dieser Anforderung ist die Betrachtung der Sprachgranularität daher eine wichtige Frage. Die verschiedenen Formen zur Integration einer DSL sind nach [MHS05] die Erweiterung, Spezialisierung oder partielle Verwendung einer existierenden Sprache. Wie weit eine neue DSL sich in eine existierende Sprache integrieren lässt oder von einer völlig neuen Sprache davon abgrenzt, liegt in der Anforderung wie kennzeichnend die Sprache für die Domäne ist, und bis wohin sich die Domäne erstreckt und was modelliert werden soll [MHS05]. So existiert in der Multi-Agenten Simulationswelt wie auch in der Welt der Programmiersprachen ein Unterschied aus:

- Domänen-spezifischen-Simulationssprachen, die an konkreten Simulationsdomänen, wie der Ökologie oder Wirtschaft etc. angepasst sind und darin existierenden Objekte in der Sprache repräsentieren [SCH09].
- Domänen-unabhängige-Simulationssprachen, die im allgemein zumeist direkte Erweiterungen zu general-purpose Sprachen sind und dort das agenten-basierte-Paradigma ergänzen, wie SARL aus der Java Welt [RGG14] oder die Swarm Abstraktion MAML [GKF99] für Objective-C [MBLA13].
- Unterstützende Sprachen zu Simulationsexperimenten, als eine explizit auf die Beschreibung des Simulationsexperiments ausgelegte Sprache, die keine Modellbeschreibung vornimmt sondern stattdessen die Ausführung und Ergebnisverarbeitung der Simulation betrachtet, wie es zum Beispiel mit der Sprache SESSL der Fall ist [EU14].

5 Modellierungsbeschreibungssprachen

Als Basis für die Eigenentwicklung einer Simulationssprache ist es sinnvoll, sowohl die wesentliche Ausrichtung der Sprache als auch interessante Sprachkonzepte zu untersuchen. Im Folgenden wird ergänzend zu der Arbeit von [BBD+06] ein Überblick einiger, verschiedener, bereits existierenden Multi-Agenten Sprachen gegeben, welche über die Jahre entwickelt wurden und dabei Problemstellungen mit unterschiedlichen Ansätzen lösen. Die Auswahl wurde mit Rücksicht auf das zukünftige Sprachdesign für die

MARS-DSL getroffen und ist mithin interessant, da sie entweder direkt auf diese Arten von Simulationen abzielen, die bereits mit MARS entwickelt werden wie dem Savannenökomodell [HWFTC15], wie es am ehesten mit der Sprache GAML der Fall ist, die dem GIS-Layer Konzept folgt und sowohl Modellierungsaspekte als auch die anschließende Datenanalyse in die Sprache mit einbezieht, ermöglicht wird dies auch mit dem MARS Framework [HWFTC15]. Überdies sind Sprachen ausgewählt, deren Konzepte mit MARS bereits umgesetzt sind oder für für zukünftige Weiterentwicklung interessant sind.

5.1 GAma Modelling Language

GAML ist eine agenten-orientierte Sprache des Multi-Agenten Frameworks GAMA, dass vor allem zur Modellierung expliziter, spatialer agenten-basierter Simulationsmodelle gedacht ist [GTG+13]. Die Sprache ist Teil des GAMA Framework und basiert auf einem XML Formalismus zur Definition reaktiver Agenten und seiner Umwelt, in der Form eines Entscheidungssystems. GAML ist eine agenten-orientierte Sprache und erweitert das Paradigma um eine Integration von Beschreibungsmitteln des umgebenden Experiments. Die Trennung zwischen Modell und Experiment wird damit vermindert und gliedert Aspekte, wie Visualisierungen und statistische Auswertungen, in die Sprache und in die Verantwortung des Modellierers mit ein [GTG+13] [GBD+10].

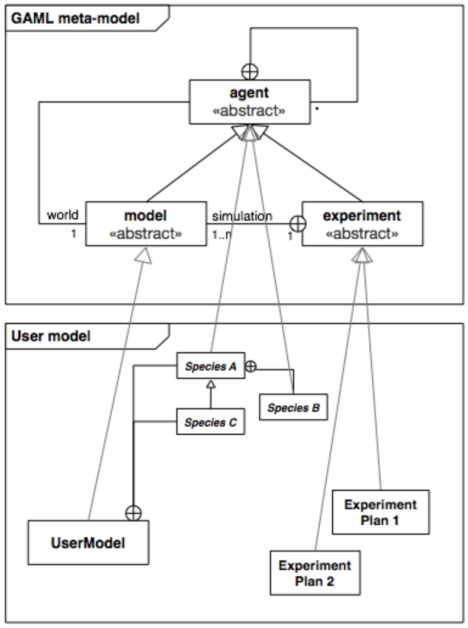

Abb. 1. Modell der GAML Sprache, mit einer eingeschränkten Sicht auf das GAML-Metamodell und der zugehörigen Elemente eines Simulationsmodells. Quelle: [GTG+13]

Während in der Objektorientierung Klassen zur Spezifikation von Objekten genutzt werden, werden in GAML dagegen jegliche Agenten als Spezies definiert. Um zu sehen was sie wissen werden Spezies durch *attributes* beschrieben, um zu sehen was sie können werden sie durch *actions* beschrieben, und um zu sehen was sie zum aktuellen Zeitpunkt tun mit *behaviors* beschrieben. Darüberhinaus erlaubt die Sprache eine Reihe zusätzlicher Eigenschaften, wie Ausbreitungsweise von Agenten oder das Scheduling, zur Manipulation der Aktivierungsreihenfolge. Ein Besonderheit bei der Spezies ist zugleich die Festlegung eines bestimmten Typs, der beispielsweise die Agententopologie beschreibt, was spatiale Eigenschaften (Wahrnehmung und Bewegung) der Spezies festlegt. Die Sprache ist dahingehend offen, dass es mithilfe der Spezies unterschiedliche Modellierungsansätze unterstützt. Neben der eigentliche Multi-Agenten

Simulation erlaubt es die Erweiterung des Ansätze auf ein sogenanntes Multi-Layer Modelling, was der gleichen Modellierung wie bei Multi-Agenten Systemen entspricht, nun jedoch unter Berücksichtigung hierarchischer Strukturen, mit Zeit, Raum und unterschiedlichem Verhalten. Betrachtet man die Biologie, kann eine Simulation beispielsweise vom einfachen Gewebe, hin zu einer Zelle, bis hin zu einem Molekül reichen und muss jeweils mit unterschiedlichen Zeiten, Räumen und Verhalten berücksichtigt werden [BLP04] [GTG+13].

Das Modell ist in GAML in drei Teile gegliedert. Der sogenannte Modell Header enthält Meta-Daten des kompletten Modells, wie den Modellnamen und Imports von anderen existierenden Modellen sowie die Definition eines globalen Agenten, der als Repräsentant des Modells verwendet wird. Ein weiterer Teil befasst sich mit der Deklaration verschiedener Spezies, die die angesiedelten Agenten des Modells darstellen. Und ein dritter Teil enthält die Deklaration jeglicher Experimente, die für das Modell ausgeführt werden sollen. Diese Trennung ist zugleich auch ein Problem, da durch das Aufweichen der Trennung zwischen Experiment und Modell, es nicht mehrere Modelle für das gleiche Experiment geben kann. Es sind lediglich mehrere Experimente für ein Modell möglich.

Die Agenten werden pro Zeitschritt ausgeführt, und basierend auf deren internen Zustand wird ein entsprechender ausführbarer Task ausgewählt, der zum aktuellen Zeitpunkt möglich wäre. Ein besonderes Feature der Sprache, was neben GAML unter anderem auch in ELMS wiederzufinden ist, ist die Möglichkeit zur impliziten oder passiven Agenteninteraktion durch das Stigmergie Kommunikationskonzept. Die Agenten können, allein durch Wahrnehmung und Manipulation ihrer Umwelt miteinander agieren und erlauben somit besonders die Modellierung von Schwarmverhalten, durch die Freigabe von Signalen in die Umwelt um mehrere Agenten gleichzeitig ansprechen zu können. Bisherige Kommunikation findet dagegen überwiegend explizit statt, d.h. eine direkte Agent-zu-Agent Kommunikation, und nicht indirekt über die Umwelt [Das14] [Sch13].

5.2 Brahms

Brahms ist eine Multi-Agenten Sprache, die anfangs als Sprache zur Simulation der menschlichen Arbeitsweisen und sozialen Zusammenhänge von Gruppen entwickelt wurde. Sie ist als BDI-Sprache entworfen und ursprünglich für verhaltens-basierte Modelle zur Simulation von Geschäftsprozessen entwickelt worden [Das14]. In der Sprache als sogenannte work practice bezeichnet, werden kontextabhängige Aktivitäten von Gruppen und von Personen beschrieben, die sich untereinander koordinieren um die Aktivitäten abarbeiten zu können. Kern und einer der wesentlich verlockenden Konstrukte der Sprache sind die BDI-Agenten, die über definierte Aktivitäten und dahinter liegende Ziele, als Basis für die Sprache, fungieren, unc damit kognitive Fähigkeiten in die Agenten integrieren. Eine Aktivität dient in der Sprache als Abstraktion zu einer Aktion aus dem realen Leben. Die zu den BDI-Agenten gehörenden Beliefs und zusätzlich konkrete Fakten der Welt lassen sich initial beschreiben, und sich während der Simulationsausführung ändern. In Brahms werden Beliefs als individuelle Interpretation der tatsächlichen Fakten gesehen und sollen dadurch Eigenheiten von Menschen repräsentieren. Neue Beliefs werden als Konsequenz aus ausgeführten Aktivitäten oder Gedankengängen (THOUGTFRAMES) erzeugt, die wiederum ausgeführt werden, wenn definierte Vorbedingungen passen. Auf der anderen Seite werden tatsächlich ausführbare Aktivitäten in Form sogenannter WORKFRAMES beschrieben. Dieser "WORKFRAME"wird ebenfalls durch erfüllte Vorbedingungen möglich und daraus folgert eine konkrete primitive oder komponierte Aktivität. Diese verbrauchen Zeit und Ressourcen und können im Falle der komponierten Aktivitäten auch benutzerdefiniert sein. Interessant ist zudem ein weiteres Sprachkonstrukt (DETECTABLES), zur Erkennung neuer Fakten und zur Unterbrechung von gerade ausgeführten Aktionen, womit es ermöglicht wird ein reaktives Verhalten zu beschreiben.

Angelehnt an Jason [BBD+06] ist Brahms ebenfalls eine Abstraktion und baut auf der Sprache Java auf. Es werden partielle Anteile implementiert, da es möglich ist Java Funktionen innerhalb der komponierten Aktivitäten nutzen zu können [MHS05], um beispielsweise eine einfache Konsolenausgabe auszugeben. Das Java Erbe hat außerdem den Einsatz von Konzepten, wie Vererbung, Abstraktion, Modularität, Overloading, Kapselung, Generics und Fehlerbehandlung in der Sprache verankert, was wiederum Wissen von den jeweiligen Benutzer über die Softwareentwicklung voraussetzt [Das14].

GROUPS are composed of
AGENTS having
BELIEFS and doing
ACTIVITIES executed by
WORKFRAMES defined by
PRECONDITIONS, matching agents beliefs
PRIMITIVE ACTIVITIES
COMPOSITE ACTIVITIES, decomposing the activity
DETECTABLES, including INTERUPTS, IMPASSES
CONSEQUENCES, creating new beliefs and/or facts
DELIBERATION implemented with
THOUGHTFRAMES defined by
PRECONDITIONS, matching agents beliefs
CONSEQUENCES, creating new beliefs

Abb. 2. Taxonomie der Sprache Brahms. Quelle: [Das14]

5.3 SARL

Die Sprache SARL ist, neben Brahms und GAML, eines der aktuellen Arbeiten zum Themengebiet der Agentensprachen, die als general purpose Sprache für diverse Anwendungsfelder geeignet ist und sich nicht nur auf die Simulation beschränkt [RGG14]. SARL baut auf der Janus-Ausführungsplattform für Multi-Agenten-Systeme auf und verfolgt die Modellierungssicht der typischen Agenten-Umgebungs-Interaktion, die auch in [MN14] erläutert wird. Agenten selbst werden durch Fähigkeiten beschrieben, die wiederum durch definierte, ausführbare Aktionen implementiert werden. Die Umgebung wird von Agenten als abstraktes Ganzes, im Unterschied zum bisherigen Layer-Design von MARS [HWFTC15] betrachtet. Die Implementierung von SARL erfolgte auf Basis des Xtext Werkzeugs, wurde nach einer LL* Grammatik erstellt, übersetzt seinen SARL Code in einen Java-ausführbaren Code und erlaubt die Entwicklung von eigenen Domänen-spezifischen Sprachen auf Basis der SARL Sprache, dessen Anforderung sich gleichermaßen in der zukünftigen MARS-DSL wiederfinden wird (siehe Abschnitt 6).

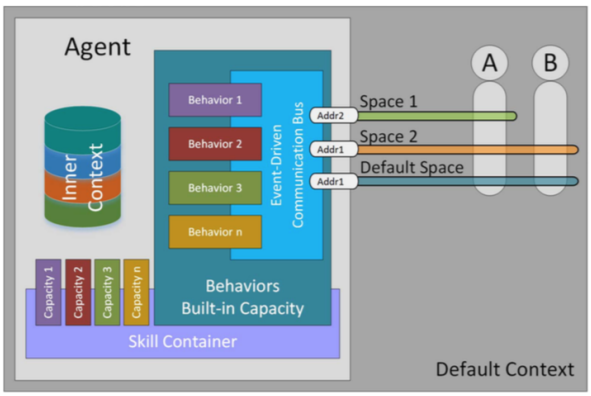

Abb. 3. Modell eines Agenten der Sprache SARL. Quelle: [RGG14]

Abbildung 3 ist die Vorstellung des Agenten in der Sprache SARL. Dieser wird als eine Menge von Capacities (Skill Container) beschrieben, deren vereinzelte Spezifikation die Menge der konkreten ausführbaren Aktionen ist, die eine Veränderung im Agentensystem oder der Umwelt hervorrufen. SARL legt zudem eine Menge vordefinierter Fähigkeiten fest die allen gemein sind und nutzt eine Besonderheit in der Interaktion zwischen Agenten. Während andere general-purpose Sprachen auf die Verwendung der

standardisierten Protokolle wie FIPA und ACL abzielen [SCH09], verfolgt SARL die Idee einer Definition von vereinzelten Umgebungen, bei dem der Scope über Prädikate beschrieben wird und es mögliche Event-Empfänger einschränkt. Somit lassen sich sowohl Gruppen ähnlich dem Multicast bilden und gezielt die Nachrichten verteilen. Gemeinsam sind sie zudem über einen allgemeinen Umgebungskontext als gemeinsamen Raum verbunden. Diese Freiheit erlaubt SARL damit auch gewisse Erweiterungen um andere Interaktionskonzepte je nach Bedarf einzupflegen, weswegen diese Spracherweiterung hier vor allem einen Grund für MARS liefert, diesen Lösungsansatz und deren Anforderungen, angesichts der Interdisziplinarität des Einsatzgebiets vorbehaltlos mit aufzunehmen. Ein ähnliches gruppen-orientiertes Konzept findet sich bereits in MARS, durch die Ähnlichkeit der GIS-Layer Aufteliung und der darauf folgernden Agentenverteilung, wieder [HWFTC15].

Ein interessanter Aspekt aus SARL ist der ungebundene Agententyp. Agenten lassen sich sowohl in reaktiver als auch in proaktiver Form beschreiben, womit die Sprache versucht im Allgemeinen anerkannte Eigenschaften von Agenten als betrachtete Entität [MN14] zu bewahren und so neben der Verhaltenssteuerung auch Autonomie und Soziale Agenten-Interaktionsfähigkeiten beibehält. Das bezieht sich auch auf die vorgegebene Art wie Agenten miteinander in der Umgebung kommunizieren. Sie verständigen sich über die definierte Events, die in die Umgebung getragen werden, ähnlich dem Stigmergiekonzpet aus GAML (siehe Abschnitt 5.1 und [Sch13]), und von potenziellen Zuhörern empfangen und verarbeitet werden. Die Verarbeitung der Events wird dabei durch keine festgelegte Kontrollschleife geregelt (Agent Lifecycle), sondern lediglich durch zwei ausgewählte Ereignisse sowohl gestartet und parametrisiert (Initialize-Event) als auch der darunterliegenden Plattform mitgeteilt, dass der Agent beendet werden soll (killme-Event).

5.4 Environment Description Language for Multi-Agent Simulation

Im Gegensatz zu Brahms und Jason ist ELMS eine auf die Beschreibung der Umgebung ausgelegte Sprache und betrachtet die Simulation primär aus Sicht der Agentenumgebung. Sie ist Teil des MASSOC Projekts [OBdRC04] und wird dort als Ergänzung zur AgentSpeak(L) Sprache verwendet. ELMS sieht eine strikte Trennung von Umgebung und Agenten vor und zergliedert die Umgebung nicht in eigenständige Agenten, sondern als mehrere, einzelne, reaktive Objekte, der verschiedenen Eigenschaften zugeordnet werden können, und mit denen die Agenten über Ein- und Ausgabekanäle interagieren können.

Eine, mit ELMS definierte Umgebung wird nach den von [RNI95] definierten Eigenschaften klassifiziert. Zwei interessante Merkmale, die sich in der Sprache wiederfinden, sind die Festlegung von Eigenschaften, welche von Agenten wahrgenommen werden können und welche nicht, sodass der Zugriff eines Agenten zur Umgebung vom Modellierer abhängt. Zudem neigt die ELMS Umgebung dazu, als diskrete Event Simulation aufgefasst zu werden, wenn diese insbesondere einen physischen Raum repräsentiert, was durch ein definierbares zwei- bzw. dreidimensionales Grid geschieht.

ELMS baut seine Sprache aus einer XML-Syntax zusammen und bietet Konstrukte zur eingeschränkten Agenten-Beschreibung (ergänzend zu AgentSpeak(L)) an, die jegliche Anteile enthält, die mit der Interaktion der Umgebung zu tun hat. Darunter Agentenbewegungen und das Sehen und Erkennen von Objekten aus der Umgebung, sowie Aktionen die Veränderungen der Umgebung verursachen. Es lassen sich, wie erwähnt, die wahrnehmbaren Eigenschaften beschreiben, und außerdem eine Vielzahl an optionalen Parametern wie für das Umgebungs-Grid, das insbesondere für geographische Simulationen gedacht ist, einstellen. Über das Grid lassen sich spatiale Aspekte modellieren, darunter Positionierungen von Agenten und Objekten mit Markierungen von einzelnen Grid-Zellen.

5.5 Multi-Agent Modelling Language

MAML, für Muli-Agent Modelling Language, ist eine domänen-spezifäische Modellierungssprache zum Modellierunsframework Swarm, die als Sprachabstraktion für die in Swarm verwendete Sprache Objective-C entwickelt wurde. In der aktuellen Version wird die Objective-C Sprache um verschiedene Beschreibungsformen erweitert [GKF99]. Es verfolgt den Ansatz der diskreten Event Simulation wie es bereits von Swarm vorgegeben wird, und lässt sich über einen eigenen Compiler xmc in den entsprechenden Swarm Objective-C Code übersetzen. Der zusätzliche Einsatz eines C-Compilers ist weiterhin notwendig,

um damit, neben der Vereinfachung, auch alle Features der darunterliegenden Sprache für den MAML-Modellierer zur Verfügung zu stellen. Unter anderem bietet Swarm angepasste Zufallszahlengeneratoren und vorgefertigte Statistikwerkzeuge an, die in der Sprache in anderer Form verwendet werden können.

```
@agent Student : Person {
  @var protected: List marks; // list of marks received so far
  @sub: (void) receiveMark: (int) mark { [marks add: mark]; }
  @sub static: (void) goToSchool { ... }
  /* etc. */
  @schedule cyclic(1440) {
     480: @to self goToSchool;
     840: @to self goHome;
     850: @to mother giveLunch;
     960: @to self doHomework; }
}
```

Abb. 4. Beispiel eines beschreibbaren MAML-Agenten, mit ausführbaren Plänen und einem Scheduling, dass eintreffende Ereignisse zu einem Zeitstempel zuordnet. Quelle: [GKF99]

MAML ist eine agentenorientierte Modellierungssprache und lässt seine Agenten, aufgrund der vielen Ähnlichkeiten zu Objekten, als eigene Objekte repräsentieren. Über Nachrichtenhändler werden eintreffende Ereignisse, dessen Zeitstempel zum aktuellen übereinstimmt abgefangen und verarbeitet, und erlaubt damit die Festlegung des Verhaltens über Regeln und Zustandsänderung des Agenten. Agentenfeatures, die bereits aus Swarm bekannt sind, wie die sogenannten action-groups, zur Beschreibung von Plänen, werden von MAML geerbt [MBLA13] und sollen einen leichteren Umstieg ermöglichen. Der Unterschied von MAML zu Swarm, ist der zusätzliche Abstraktionsgrad als domänen-spezifische Sprache, die Swarm weiter spezialisiert und das Paradigma der agenten-orientierten Modellierung innerhalb der diskreten Event Simulation einfügt. Bereits implementierte Swarm-Sprachfeatures, wie Pläne und das Scheduling von Ereignissen, werden in MAML unter anderer Notation angeboten und sollen damit eine Vereinfachung darstellen. Der Modellierer kann sich hierdurch in seiner Domänen bewegen ohne durch den objektorientierten Overhead beeinflusst zu werden. Hintergrund der Sprache ist der Gedanke, dass man häufig Änderungen am Simulationsmodell vornimmt und das Experimente von Änderungen ausgehen, was von [Das14] bestätigt wird. Daher ist eine einfache Pflege und Verständlichkeit des Modells notwendig, die durch Wegfall diverser Swarm-Implementationsdetails und dem Fokus auf ein gemeinsames, agentenbasiertes Verständnis unter mehreren Personen, erfüllt werden sollte.

```
@agent Student : Person {
  @var protected: List marks; // list of marks received so far
  @sub: (void) receiveMark: (int) mark { [marks add: mark]; }
  @sub static: (void) goToSchool { ... }
  /* etc. */
  @schedule cyclic(1440) {
     480: @to self goToSchool;
     840: @to self goHome;
     850: @to mother giveLunch;
     960: @to self doHomework; }
}
```

Abb. 5. Beispiel einer MAML-Plan- und Eventscheduling Definition, mit zeitgesteuerten Events und Events die vom MAML Plan ausgehen: [GKF99]

Die Idee einer zusätzlichen Sprachabstraktion macht MAML durchaus für eigene Entwicklungen interessant, um durch die Sprachmittel eigene Konzepte einzubauen und vorhandene Konzepte in anderer Notationsform anbieten zu können. Es ist jedoch nicht sichergestellt, dass mit MAML automatisch eine Sprachvereinfachung entsteht, auch wenn über das agentenbasierte Paradigma weitläufig, eine gemeinsame Kommunikationsbasis, ähnlich der Mathematik entsteht [GBD+10]. Es muss sich weiterhin mit der Sprache und den Konstrukten auseinander gesetzt werden.

5.6 Functional Agent-Based Language for Simulations

Aufbauend auf der Multi-Agent Simulation Suite (MASS) wurde die Sprache FABLES (Functional Agent-Based Language for Simulations) entwickelt. Die FABLES Sprache besteht aus drei wesentlichen Teilen und verwendet ein hybrides Paradigma aus funktionaler, objektorientierter und zum Teil imperativer Programmierung [GBK+05]. Die Objektorientierung wird verwendet zur Spezifikation von Agenten und dessen Umgebung, der funktionale Anteil beschreibt die funktionalen Beziehungen der beschriebenen Agenten und der Umgebung durch mathematische Konzepte, und der imperative Teil beschreibt die Dynamik der Simulation über die Zeit. Ein verfügbares Eclipse Plugin erlaubt die Spezifikation von FABLES Modellen und die direkte Ausführung des Modells, durch die Übersetzung in Java Code über einen Codegenerator. Interessant hierbei ist der starke Fokus und die Verwendung auf vorteilhafter Aspekte der genannten Programmierparadigmen. Während die OO weiterhin als Repräsentationsform für Agenten und Co. dient, sind anstelle von OO-Konzept zur Beschreibung von Beziehung mathematische Konstrukte wie Sequenzen, Mengen und Relationen eingesetzt. Hierdurch lassen sich wesentlich präzisere Beschreibungen vornehmen, die zudem von einer größeren Benutzermenge verstanden werden, mit der Mathematik als gemeinsame Sprache. Die zusätzliche Trennung des Simulationsverhaltens über die Zeit durch imperative Aspekte von der Struktur bietet darüberhinaus den Vorteil die Komplexität weiter zu reduzieren ohne das Modell hinfällig werden zu lassen.

6 Ausblick und künftige Forschung

Die an der Hochschule für Angewandte Wissenschaften Hamburg existierende MARS Forschungsgruppe [HWFTC15] hat das Ziel eine allgemein verwendbare Multi-Agenten Umgebung zu schaffen, die eine möglichst rasche Modellentwicklung ermöglichen soll, indem bereits vorkommende Modelle, als wiederverwendbare Komponenten für die eigene Modellentwicklungen herangezogen werden können. Eine beispielsweise auf die Savannenökologie zugeschnittene Sprache erlaubt damit die Nutzung vorgefertigter Agenten im Kontext des ArsAfricae Modells (vordefinierte Elefantenagenten oder Baumagenten), sodass sich eine intentionale Programmierung ergibt [Das14] und über eine Model-to-Model Transformation diese Sprache zunächst in eine allgemeine MARS-DSL übersetzt und anschließend in ausführbaren C# Code gebracht wird.

Mit dem Fokus auf die Wiederverwendbarkeit existierender Modelle und einer dennoch weitreichenden Abdeckung von möglichen Modellierungsdomänen bedarf es einer general purpose Multi-Agenten Plattform. Welche Modellieurngskonzepte jedoch mit dem MARS System erreicht werden können und wie eine dafür ausgelegte Modellierungs-Beschreibungssprache aussehen kann, sowie dahinterliegende Komponentenabhängigkeiten hinsichtlich einer genügsamen Simulationsperformance ausgenutzt werden kann, sind zentrale Herausforderungen und Fragen, mit denen ich mich künftig beschäftigen werde. Insbesondere sind folgende Forschungsfragen interessant:

- Welche Beschreibungskonzepte sind für eine abstrakte Metasprache interessant, die als general-purpose Sprache für jegliche Domänen einsetzbar ist?
- Wie sieht das Metamodell des MARS Systems aus?
- Wie kann eine neuartige agentenbasierte Modellierungssbeschreibungssprache für MARS aussehen und welches Modellierungsprotokoll kann und soll verfolgt werden [GBD+10]?
- Welche Modellierungssichten existieren für das MARS System, die vor allem auf die Art und Weise abzielen, wie Agenten und ihre Umgebung betrachtet werden können? Diese Frage ist insbesondere dann interessant, wenn Simulationen mit Raumbezug entwickelt werden sollen, wie im Falle ökologischer Simulationen wo Bewegung im Raum ein zentrales Modellelement ist [GBD+10].

– Welche Beschreibungsmittel stehen im MARS System bereits zur Verfügung, und welche neuen Beschreibungsformen lassen sich daraus entwickeln, die ggf. in der Sprache als Konstrukt zur Verfügung stehen?
– Welche zukünftigen Arbeiten können in Verbindung mit Qualitätssicherung zur Korrektheit des Modells durchgeführt werden?
– Wie kann eine Beschreibungssprache als Werkzeug in das bestehende MARS System integriert, sowohl technisch als auch im MARS-Workflow [HWFTC15] werden?

Besondere Anhaltspunkte, die einen Start für eine domänen-spezifische Sprache darstellen, sind die in dieser Arbeit vorgestellte SARL Sprache für Janus. Der dortige Abstraktionsschritt und das Ziel dahinter, den Kern des Einsatzes einer Simulation nicht zu vernachlässigen, wird eine wichtige Anforderungen für die Sprachentwicklung darstellen, dass vor allem im Austausch mit Interessenten an dem MARS-System verbessert werden kann. Darüber hinaus ist die Sprache GAML von Bedeutung, da sie viele Aspekte enthält, mit der sich das MARS-System auszeichnet. Der Schwerpunkt auf ökologische Simulationen über GIS-Layer und die Möglichkeit zur Multi-Layer Modellierung sind sowohl wegen der existierenden epidemischen Modelle als auch wegen des ArsAfricae Modells wichtig [Bus15], weil damit auf Details im Modell eingegangen werden kann, und somit wertvolle Ergebnisse auf Makroebene untersucht werden können [GTG+13].

7 Zusammenfassung

Diese Arbeit hat aktuelle und verschiedene Beschreibungssprachen für Multi-Agenten Simulationen diskutiert und gezeigt, dass sich jede Sprache in Bezug auf deren Abstraktion, dem verfolgten Modellierungsprinzip sowie den verfügbaren Sprachkonstrukten unterscheidet. Der aktuelle Fokus liegt vermehrt darauf, für bereits existierende Frameworks eine eigene Beschreibung anbieten zu können und Sprachmittel für das umgebende Simulationsmodell bereitzustellen. Eine derzeitige Herausforderung zeigt sich durch die fehlende Unterstützung von Qualitätssicherung, durch Debugging- und Test-Werkzeugen, die auch mit der Verteilungseigenschaft der Multi-Agenten Systeme umgehen können. Des Weiteren fehlt das Zutun in der Beschreibung von Multi-Agenten Organisationen sowie der Multi-Agenten-Umgebung. Zwar gab es Versuche wie mit ELMS (siehe Abschnitt 5.4) und GAML (siehe Abschnitt 5.1), jedoch bleibt das Ziel der Integration jeweiliger Beschreibungsmittel von Multi-Agenten Organisationen und Multi-Agenten-Umgebungen, um damit das gesamte Beschreibungsmittel zu einer Reife für wissenschaftliche Entwicklungen voranzubringen. Mit dem hier vorgestellten MARS Framework versucht man diese Unterstützung in zukünftigen Arbeiten zu erreichen, und sowohl ein übergreifendes Beschreibungsmittel als auch ein domänenspezifisches Mittel zu entwickeln.

Literatur

AHL+96. W Brian Arthur, John H Holland, Blake LeBaron, Richard G Palmer, and Paul Tayler. Asset pricing under endogenous expectations in an artificial stock market. *Available at SSRN 2252*, 1996.

BBD+06. Rafael H Bordini, Lars Braubach, Mehdi Dastani, A El F Seghrouchni, Jorge J Gomez-Sanz, Joao Leite, Gregory O'Hare, Alexander Pokahr, and Alessandro Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30(1), 2006.

BHB+10. Tristan Behrens, Koen V Hindriks, Rafael H Bordini, Lars Braubach, Mehdi Dastani, Jürgen Dix, Jomi F Hübner, and Alexander Pokahr. An interface for agent-environment interaction. In *Programming multi-agent systems*, pages 139–158. Springer, 2010.

BLP04. François Bousquet and Christophe Le Page. Multi-agent simulations and ecosystem management: a review. *Ecological modelling*, 176(3):313–332, 2004.

Bus15. Jan Busch. Ein spatio-temporales data warehouse fur multiskalige, heterogene okologische daten. 2015.

Das14. Mehdi Dastani. *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

DF13. Jürgen Dix and Michael Fisher. Specification and verification of multi-agent systems. *Multiagent Systems. MIT Press, Cambridge*, 2013.

EMN+05. Thierry Emonet, Charles M Macal, Michael J North, Charles E Wickersham, and Philippe Cluzel. Agentcell: a digital single-cell assay for bacterial chemotaxis. *Bioinformatics*, 21(11):2714–2721, 2005.

EU14. Roland Ewald and Adelinde M. Uhrmacher. Sessl: A domain-specific language for simulation experiments. *ACM Trans. Model. Comput. Simul.*, 24(2):11:1–11:25, February 2014.

GBD+10. Volker Grimm, Uta Berger, Donald L DeAngelis, J Gary Polhill, Jarl Giske, and Steven F Railsback. The odd protocol: a review and first update. *Ecological modelling*, 221(23):2760–2768, 2010.

GBK+05. László Gulyás, Sándor Bartha, Tamás Kozsik, Róbert Szalai, Attila Korompai, and Gábor Tatai. The multi-agent simulation suite (mass) and the functional agent-based language of simulation (fables). In *Ninth Annual Swarm Users/Researchers Conference (SwarmFest 2005), Turin, Italy*. Citeseer, 2005.

GKF99. László Gulyás, Tamás Kozsik, and Sándor Fazekas. The multi-agent modeling language. In *Proceedings of Proceedings of the 4th International Conference on Applied Informatics (ICAI)*, pages 43–50, 1999.

GTG+13. Arnaud Grignard, Patrick Taillandier, Benoit Gaudou, Duc An Vo, Nghi Quang Huynh, and Alexis Drogoul. Gama 1.6: Advancing the art of complex agent-based modeling and simulation. In *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, pages 117–131. Springer, 2013.

Hol95. John H Holland. *Hidden order: How adaptation builds complexity*. Basic Books, 1995.

HWFTC15. Christian Hüning, Jason Wilmans, Nils Feyerabend, and Thomas Thiel-Clemen. Mars–a next-gen multi-agent simulation framework, 2015.

LBC+14. Igor Linkov, Todd Bridges, Felix Creutzig, Jennifer Decker, Cate Fox-Lent, Wolfgang Kroger, James H. Lambert, Anders Levermann, Benoit Montreuil, Jatin Nathwani, Raymond Nyer, Ortwin Renn, Benjamin Scharte, Alexander Scheffler, Miranda Schreurs, and Thomas Thiel-Clemen. Changing the resilience paradigm. *Nature Clim. Change*, 4(6):407–409, 06 2014.

MBL+96. Nelson Minar, Roger Burkhart, Chris Langton, Manor Askenazi, et al. The swarm simulation system: A toolkit for building multi-agent simulations. Santa Fe Institute Santa Fe, 1996.

MBLA13. N Minar, R Burkhart, C Langton, and M Askenazi. The swarm simulation system: A toolkit for building multi-agent simulations, 1996. *URL http://citeseerx. ist. psu. edu/viewdoc/summary*, 2013.

MHS05. Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, December 2005.

MN09. Charles M. Macal and Michael J. North. Agent-based modeling and simulation. In *Winter Simulation Conference*, WSC '09, pages 86–98. Winter Simulation Conference, 2009.

MN14. Charles Macal and Michael North. Introductory tutorial: Agent-based modeling and simulation. In *Proceedings of the 2014 Winter Simulation Conference*, pages 6–20. IEEE Press, 2014.

OBdRC04. Fabio Y Okuyama, Rafael H Bordini, and Antônio Carlos da Rocha Costa. Elms: an environment description language for multi-agent simulation. In *Environments for Multi-Agent Systems*, pages 91–108. Springer, 2004.

RGG14. Saul Rodriguez, Nicolas Gaud, and Stephane Galland. Sarl: a general-purpose agent-oriented programming language. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 3, pages 103–110. IEEE, 2014.

RNI95. Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27, 1995.

SCH09. Maarten Sierhuis, William J. Clancey, and Ron J.J. Hoof. *Multi-Agent Programming:: Languages, Tools and Applications*, chapter Brahms An Agent-Oriented Language for Work Practice Simulation and Multi-Agent Systems Development, pages 73–117. Springer US, Boston, MA, 2009.

Sch13. Christian Schlette. *Anthropomorphe Multi-Agentensysteme: Simulation, Analyse und Steuerung*. Springer-Verlag, 2013.

SS14. Onn Shehory and Arnon Sturm. Agent-oriented software engineering. *Reflections on architectures, methodologies, languages, and frameworks*, 2014.

SUZ+14. Alexander Steiniger, Adelinde M. Uhrmacher, Sabine Zinn, Jutta Gampe, and Frans Willekens. The role of languages for modeling and simulating continuous-time multi-level models in demography. In *Proceedings of the 2014 Winter Simulation Conference*, WSC '14, pages 2978–2989, Piscataway, NJ, USA, 2014. IEEE Press.

TW04. Seth Tisue and Uri Wilensky. Netlogo: Design and implementation of a multi-agent modeling environment. In *Proceedings of agent*, volume 2004, pages 7–9, 2004.