Agentenbasierte Simulation als Verfeinerung der Diskreten-Ereignis-Simulation

Wolf-Ulrich Raffel
raffel@inf.fu-berlin.de
Institut für Informatik
Freie Universität Berlin

1. Zusammenfassung

Möchte man das Verhalten eines zu erforschenden Systems nicht direkt an diesem untersuchen, sondern dies mit Rechnerunterstützung tun, so spricht man von Simulation. In der Simulation hat sich seit langer Zeit in der Forschung das Konzept der Diskreten-Ereignis-Simulation etabliert, bei der sich der Zustand beim Auftreten von Ereignissen sprunghaft ändert, während er im Zeitraum zwischen zwei Ereignissen konstant bleibt. Die Modellierung und Simulation komplexer Systeme, die aus mehreren, ggf. hierarchisch strukturierten Bestandteilen bzw. Subsystemen bestehen, stößt in der Diskreten-Ereignis-Simulation schnell an ihre Grenzen, da es schwierig ist, einzelne Systembestandteile unabhängig voneinander zu modellieren.

Daher hat sich in der jüngsten Vergangenheit die Agentenbasierte Simulation entwickelt, in der komplexe Systeme, die aus mehreren miteinander und mit ihrer Umwelt interagierenden Entitäten bestehen, als Multiagentensysteme aufgefasst und modelliert werden. Die am System beteiligten Agenten können dabei Handlungen ausführen, ihre Umgebung wahrnehmen und verändern, auf Veränderungen der Umwelt reagieren und sie verfügen über einen mentalen Zustand, der aus Wissen, Zielen, Erinnerungen und Verpflichtungen bestehen kann.

Allerdings zerfällt die Welt der Simulation nun in zwei Lager: die Anhänger der Diskreten-Ereignis-Simulation, die sich über Jahrzehnte hinweg etabliert hat und die Anhänger der Agentenbasierten Simulation, die strukturierte und realitätsnahe Simulationen erlaubt, aber mit der Diskreten-Ereignis-Simulation nichts gemeinsam hat und somit bei deren Anhängern auf wenig Akzeptanz stößt.

Es wird deshalb hier gezeigt, wie man die Agentenbasierte Simulation als Verfeinerung der Diskreten-Ereignis-Simulation auffassen kann. Zu diesem Zweck wird zunächst das in der Praxis heterogene Konzept der Diskreten-Ereignis-Simulation mathematisch formalisiert. Auf dieser Formalisierung aufbauend wird dann eine Formalisierung für Agentenbasierte Simulation vorgestellt. Man kann agentenbasierte Simulationsmodelle, die dieser Formalisierung gehorchen mit einem hier implementierten Simulator ausführen.

In den vorherrschenden Agentenbasierten Simulatoren muss man als Benutzer das konkrete Simulationssystem in einer allgemeinen Programmiersprache oder einer speziellen Simulationssprache

formulieren. Man kann das System also nicht deklarativ, sondern nur imperativ spezifizieren. Hier wird hingegen eine grafische High-Level-Spezifikationssprache für die Agentenbasierte Simulation vorgestellt, die die Form eines UML-Profils hat, so dass man jedes beliebige UML-Tool zur Spezifikation verwenden kann.

Die Implementierung des in Java geschriebenen Simulators, die den Charakter eines *proof of concept* und nicht eines fertigen Softwareproduktes hat, wurde dahingehend erweitert, dass die mit Hilfe eines UML-Tools erstellte Spezifikation direkt ausführbar ist. Dies wurde erreicht, indem die Ausgabe des UML-Tools in Form der XML-basierten Sprache XMI mittels XSL-Transformationen zunächst in ein XML-basiertes Zwischenformat und dann direkt in vom Simulator verwendbaren Java-Code transformiert wird.

Zunächst wird ein Basismodell für die Diskrete-Ereignis-Simulation entwickelt und mathematisch formalisiert. Dieses Basismodell wird später um einige nützliche, nicht agentenorientierte Erweiterungen angereichert, die dann zur sogenannten Objektbasierten Simulation subsummiert werden. Anschließend wird das Modell der Objektbasierten Simulation agentenorientiert erweitert, so dass an seinem Ende ein formales Modell für die Agentenbasierte Simulation steht. Für Agentenbasierte Simulationssysteme wird eine grafische Spezifikationssprache in Form eines UML-Profils vorgestellt. Ein Simulationssystem, das es ermöglicht, in UML spezifizierte agentenbasierte Simulationsmodelle automatisch ausführen zu lassen, wurde implementiert.

2. Simulation

Bei Simulation geht es darum, ein System, das sich für die Durchführung von realen Tests nicht eignet, durch ein Modell zu beschreiben, das die wesentlichen Eigenschaften des Systems abbildet. Mit Hilfe dieses Modells können dann Experimente durchgeführt werden, deren Ergebnisse Rückschlüsse auf das Verhalten des simulierten Systems zulassen sollen.

In der Simulationstheorie unterscheidet man zwischen diskreten und kontinuierlichen Simulationssystemen, wobei sich die Begriffe auf die Art der Änderung des Systemzustands beziehen. Bei diskreten Systemen wird zwischen ereignisorientierten, prozessorientierten und transaktionsorientierten Ansätzen zur Simulation unterschieden.

Am weitesten verbreitet ist diskrete Simulation unter Verwendung des ereignisorientierten Modells, die diskrete ereignisorientierte Simulation, auch kurz *Diskrete-Ereignis-Simulation* genannt. In ihr werden Zustandsänderungen durch das Auftreten von Ereignissen, die zu bestimmten Zeitpunkten stattfinden vorgenommen, während zwischen zwei Ereignissen der Zustand unverändert bleibt. Ändert man in der Simulation den Zustand gemäß des Auftretens eines Ereignisses, so spricht man auch von der *Ausführung* des Ereignisses. Man betrachtet beim ereignisorientierten Modell eine Ereignismenge, in der sich alle noch nicht ausgeführten Ereignisse befinden. In einem Ausführungsschritt wird das Ereignis aus der Ereignismenge mit dem frühesten Zeitpunkt ausgewählt und ausgeführt, wodurch sich nicht nur der Zustand ändern kann sondern außerdem neue Ereignisse in die Ereignismenge eingefügt werden. Bei solchen Ereignissen spricht man von *Folgeereignissen* des ausgeführten Ereignisses. Das ereignisorientierte Modell zeichnet sich durch seine klare verständliche Struktur aus, die es auch ermöglicht, eine einfache mathematische Formalisierung zu finden.

3. Agentenbasierte Simulation

Da die Modellierung und Simulation komplexer Systeme, die aus mehreren, ggf. hierarchisch strukturierten Bestandteilen bzw. Subsystemen bestehen in der Diskreten-Ereignis-Simulation schnell an ihre Grenzen stößt, weil es schwierig ist, einzelne Systembestandteile unabhängig voneinander zu modellieren, hat sich in der Praxis ein neues Paradigma, die Agentenbasierte Simulation entwickelt.

In der Agentenbasierten Simulation geht es darum, komplexe Realsysteme, die aus miteinander und mit ihrer Umwelt interagierenden Entitäten bestehen, als *Multiagentensysteme* zu interpretieren und gemäß dieser Interpretation zu simulieren. Im Vergleich zu vielen traditionellen Methoden der Simulation – wie kontinuierliche Simulation (Differentialgleichungen), Diskrete-Ereignis-Simulation, zelluläre Automaten und Spieltheorie – ist die Agentenbasierte Simulation weniger abstrakt und realitätsnäher.

Agentenbasierte Simulation wird heute in vielen Bereichen der Forschung angewendet, insbesondere in der Biologie, der Bioinformatik, den Wirtschaftswissenschaften, den Sozialwissenschaften und den Ingenieurwissenschaften.

Für die Agentenbasierte Simulation sind insbesondere die folgenden Anforderungen von Belang:

- der Simulationsspezifikationssprache und der Simulator basieren auf einem agentenorientierten Metamodell
- die Simulationsspezifikationssprache ist deklarativ
- es gibt eine visuelle, UML-basierte Simulationsspezifikationssprache

Die Forderung nach einem Metamodell entspringt aus dem Wunsch, einem Simulator eine theoretische Grundlage zu geben, was in der Praxis oft nicht der Fall ist. Durch die Deklarativität der Simulationsspezifikationssprache sollen Spezifikation und Ausführung voneinander entkoppelt werden, so dass es auch möglich wäre, ein einmal spezifiziertes System in verschiedenen Simulationsplattformen auszuführen. Außerdem ist bei einer deklarativen Spezifikation natürlicher und gegenüber einer imperativen Spezifikation ist die Lesbarkeit erhöht. Der Vorteil bei einer visuellen Spezifikationssprache ist, dass sie leichter erfassbar ist und damit die Simulationsmodelle leichter kommunizierbar sind. UML-Basierung ist sinnvoll, da sich UML als Standard in Forschung und Industrie etabliert und die Konzepte der Softwaremodellierung der letzten Jahrzehnte in sich aufgenommen hat.

Die genannten Anforderungen werden von den verfügbaren Systemen gar nicht oder nur teilweise erfüllt. Das hier entwickelte Simulationssystem erfüllt alle genannten Anforderungen. Es basiert auf einem Agentenbasierten Metamodell. Die entwickelte grafische High-Level-Spezifikationssprache für die Agentenbasierte Simulation ist deklarativ, da sie auf Reaktionsregeln basiert und hat die Form eines UML-Profils, so dass man jedes beliebige UML-Tool zur Spezifikation verwenden kann.

Die Diskrete-Ereignis-Simulation wird erst in ihrer einfachsten Form formalisiert, woraus ein Basismodell der Diskreten-Ereignis-Simulation entsteht. Es werden dann Erweiterungen des Basismodells der Diskreten-Ereignis-Simulation vorgenommen. Dabei sind drei mögliche Erweiterungen der objektorientierte Systemzustand, die Unterscheidung zwischen exogenen und Folgeereignissen sowie die Ersetzung der Angabe von altem und neuem Zustand durch Angabe von Zustandsbedingung und Zustandseffekt. Das Prinzip der auf der Diskreten-Ereignis-Simulation aufbauenden Objektbasierten Simulation vereinigt diese Erweiterungen in sich. In der auf der Objektbasierten Simulation aufbauenden Agentenbasierten Simulation wird das Modell der Objektbasierten Simulation um die Modellierung von Nachrichten und um die Trennung zwischen externem und internem Agentenzustand, die die Einführung eines Umgebungssimulators nach sich zieht, erweitert.

Die hier formalisierte Agentenbasierte Simulation beinhaltet im wesentlichen die Aufteilung des zu simulierenden Systems in aktive Entitäten (Agenten) und passive Entitäten (Objekte). In der Simulation gibt es einen Umgebungssimulator, der die Umgebung sowie die (passiven) Objekte verwaltet sowie für jeden Agenten einen Agentensimulator. Die Simulation läuft in Zyklen ab, wobei der Ablauf eines Zyklus in der Simulation wie in Abbildung 1 dargestellt aussieht.

Schritt 1	Der Umgebungssimulator ermittelt alle in diesem Zyklus stattfindenden Ereignisse (die *aktuellen Ereignisse*). Dabei handelt es sich um: a) die Aktionen, die die Agenten am Ende des vergangenen Zyklus ausgeführt haben b) Ereignisse aus der Ereignismenge des Umgebungssimulators, die in diesen Zyklus fallen c) exogene Ereignisse, die in diesen Zyklus fallen.
Schritt 2	Der Umgebungssimulator berechnet aus dem aktuellen Umgebungszustand und den aktuellen Ereignissen einen neuen Umgebungszustand, eine Menge von Folgeereignissen sowie für jeden Agenten seine Wahrnehmungen. Zu den Wahrnehmungen gehört auch der Empfang von Nachrichten, die andere Agenten versendet haben.
Schritt 3	Der Umgebungssimulator sendet jedem Agentensimulator die (ggf. leere) Menge von Wahrnehmungen des durch ihn simulierten Agenten.
Schritt 4	Jeder Agentensimulator (auch die, die eine leere Menge von Wahrnehmungen empfangen haben) bestimmt zunächst die Menge der in diesem Zyklus stattfindenden internen Ereignisse (die *aktuellen internen Ereignisse*). Dabei handelt es sich um: a) die empfangenen Wahrnehmungen b) interne Zeitereignisse aus der Zeitereignismenge des Agentensimulators, die in diesen Zyklus fallen c) periodische interne Zeitereignisse, die in diesen Zyklus fallen. Anschließend berechnet er aus dem aktuellem internen Agentenzustand und aus den aktuellen internen Ereignissen einen neuen internen Agentenzustand, eine Menge von internen Folge-Zeitereignissen sowie eine Menge von Aktionen, die der durch ihn simulierte Agent durchführt. Zu den Aktionen gehören auch an andere Agenten versendete Nachrichten. Die (ggf. leere) Menge von Aktionen sendet er an den Umgebungssimulator.
Schritt 5	Der Umgebungssimulator wartet ab, bis er von allen Agenten jeweils die Menge ausgeführter Aktionen empfangen hat und setzt die Zeit weiter. Die Menge der ausgeführten Aktionen wird im darauffolgenden Zyklus berücksichtigt.

Abbildung 1: Ablauf eines Zyklus in der Agentenbasierten Simulation

Konkrete Agentenbasierte Simulationssysteme können durch Angabe der verwendeten Agenten- und Objekttypen, der Ereignis-, Wahrnehmungs- und Aktionstypen, der konkreten Agenten und Objekte, der auftretenden exogenen Ereignisse sowie der Reaktionsregeln für den Umgebungssimulator und für die Agentensimulatoren vollständig spezifiziert werden. Diese Spezifikation lässt sich in natürlicher Art und Weise in der Sprache UML durchführen. Dabei werden die Agenten-, Objekt- Ereignis-, Wahrnehmungs- und Aktionstypen mit Hilfe von Stereotypes unterschieden. Reaktionsregeln werden als n-äre Assoziationen zwischen den beteiligten Typen ausgedrückt, wobei Bedingungen, unter denen die Regeln gelten, die Form von den Assoziationsenden zugeordneten OCL-Ausdrücken haben.

Um zu zeigen, dass es tatsächlich möglich ist, mit Hilfe eines UML-Modells ein ausführbares agentenbasiertes Simulationssystem zu spezifizieren, wird als *proof of concept* ein Simulationssystem entwickelt.

Der Kern des Simulators ist eine Java-Programmbibliothek, die eine Klasse enthält, die die Ausführung eines agentenbasierten Simulationsmodells organisiert. Ein konkretes Simulationssystem wird ausgeführt, indem die verwendeten Typen (Agenten- und Objekttypen, Ereignis-, Wahrnehmungs- und Aktionstypen) als Unterklassen der generischen Typklassen realisiert werden. Um zu erreichen, dass man als Modellierer vollständig in UML spezifizieren kann und dieses Modell dann direkt ausführen kann, wird als das Format XMI verwendet, eine standardisierte XML-basierte Sprache, die eine textuelle Repräsentation eines UML-Modells darstellt. Viele der heutigen UML-Tools verwenden XMI direkt für die Speicherung der UML-Modelle oder bieten zumindest die Möglichkeit, ein UML-Modell in das Format XMI zu exportieren. Mit Hilfe von XSL-Transformationen - eine XML-Anwendung, die es ermöglicht Dokumente aus einem XML-basierten Format in ein anderes Format gemäß gewisser Transformationsregeln zu überführen - wird dann das im XMI-Format vorliegende UML-Modell zunächst in ein XML-basiertes Zwischenformat und dann direkt in vom Simulator verwendbaren Java-Code transformiert.

4. Literatur

[Cormas00] C. Le Page, F. Bousquet, I. Bakam, A. Bah, C. Baron: CORMAS: A multiagent simulation toolkit to model natural and social dynamics at multiple scales. Präsentiert beim Workshop "The ecology of scales", Wageningen (Niederlande), 2000.

http://cormas.cirad.fr/pdf/cormasw.pdf

[Geh98] H. Gehring: Operations Research – Simulation, Fernuniversität Hagen, 1998.

[GK94] M.R. Genesereth, S.P. Ketchpel: Software agents. Communication of the ACM, 37, 7, S. 48-53, 1994.

[Hay95] B. Hayes-Roth: An architecture for adaptive intelligent systems. Artificial Intelligence, 72, S. 329-365, 1995.

[Kil00] R.A. Kilgore: Silk, Java and object-oriented simulation, ThreadTec Inc., Proceedings of the 2000 Winter Simulation Conference,2000.

http://www.informs-cs.org/wsc00papers/037.PDF

[Klü01] F. Klügl: Multiagentensimulation, Addison-Wesley Verlag, 2001.

[Lie95] F. Liebl: Simulation - Problemorientierte Einführung, Oldenbourg Verlag, 1995.

[MadKit00] J. Ferber, O. Gutknecht, F. Michel: MadKit Development Guide, 2002.

http://www.madkit.org/madkit/doc/devguide/devguide.html

[MD+02] M. Marietto, N. David, J. Sichman, H. Coelho: Requirements Analysis of Multi-Agent-Based Simulation Platforms: State of the Art and New Prospects, Universität Sao Paulo, 2002.

http://www.pcs.usp.br/~jaime/papers/marietto_mabs02_e.ps

[Ngu04] D.M.Nguyen: Objektorientierte Diskrete Ereignissimulation basierend auf UML-Klassendiagrammen und Reaktionsregeln, Diplomarbeit, Freie Universität Berlin, 2004.

[OMG00] OMG: OMG Unified Modeling Language Specification, Version 1.4, 2000.

http://www.omg.org

[Pag91] B. Page: Diskrete Simulation, Springer-Verlag, 1991.

[Sau99] T. Sauerbier: Theorie und Praxis von Simulationssystemen, Vieweg-Verlag, 1999.

[SeSAm02] F. Klügl: Introductory slides to SeSAm, 2002.

http://www.simsesam.de

http://ki.informatik.uni-wuerzburg.de/~sesam/tutorials/shortIntroduction/

[Sho93] Y. Shoham: Agent-oriented programming. Artificial Intelligence, 60, S. 51-92, 1993.

[Swarm96] N. Minar, R. Burkhart, C. Langton, M. Askenazi: The Swarm Simulation System: A Toolkit For Building Multi-Agent Simulations, 1996.

www.swarm.org/archive/overview.ps

[Wag97a] G. Wagner: Vivid Agents: How They Deliberate, How They React, How They Are Verified, Universität Leipzig, 1997.

[WJ95] M. Wooldridge, N. Jennings: Intelligent Agents – Theories, Architectures and Languages. Springer Lecture Notes in Artificial Intelligence (LNAI) 890, 1995.