10. Zusammenfassung

In dieser Arbeit wurde eine Form der Agentenbasierten Simulation vorgeschlagen, die eine Verfeinerung der Diskreten-Ereignis-Simulation darstellt, da sie die Grundbegriffe der Diskreten-Ereignis-Simulation differenziert und verfeinert. Sie besitzt eine formale Ausführungssemantik, die auf einer Ausführungssemantik für Objektbasierte Simulation basiert, die ihrerseits auf der Diskreten-Ereignis-Simulation basiert.

Während sich die Diskrete-Ereignis-Simulation, in der sich der Systemzustand beim Auftreten von Ereignissen sprunghaft ändert, seit langer Zeit etabliert hat, hat sich in der jüngsten Vergangenheit die Agentenbasierte Simulation entwickelt, in der komplexe Systeme, die aus autonomen Entitäten bestehen, als Multiagentensysteme modelliert werden.

Bisher wurden die Simulationsparadigmen der Diskreten-Ereignis-Simulation und der Agentenbasierten Simulation als getrennte und voneinander unabhängige Ansätze betrachtet. Bisherigen Agentenbasierten Simulationssystemen (Swarm, RePast, Spades, SeSAm, MadKit, CORMAS) liegt kein Metamodell und damit auch keine formale Ausführungssemantik zugrunde oder man kann nicht deklarativ, visuell und UML-basiert spezifizieren.

Die in dieser Arbeit vorgestellte Agentenbasierte Simulation beinhaltet die Aufteilung des zu simulierenden Systems in aktive Entitäten (Agenten) und passive Entitäten (Objekte). Dabei besitzen die Agenten einen externen (physischen) und einen internen (mentalen) Zustand. In der Simulation gibt es einen Umgebungssimulator, der die Umgebung – die Objekte und die externen Agentenzustände – verwaltet. Ferner gibt es für jeden Agenten einen Agentensimulator, der den internen Agentenzustand verwaltet. Die Simulation läuft in Zyklen ab. Dabei ermittelt der Umgebungssimulator in jedem Zyklus die aktuellen Ereignisse, führt Veränderungen in der Umgebung durch und teilt den Agentensimulatoren ihre Wahrnehmungen ihrer Umgebung (darunter auch empfangene Nachrichten) mit. Die Agentensimulatoren ermitteln in jedem Zyklus aktuelle interne Ereignisse, führen Veränderungen des internen Zustands durch und teilen dem Umgebungssimulator ihre durchgeführten Aktionen (darunter auch gesendete Nachrichten) mit.

Die Agentenbasierte Simulation basiert auf der Objektbasierten Simulation, die wiederum auf der Diskreten-Ereignis-Simulation basiert. Für die Diskrete-Ereignis-Simulation wurde ein mathematisches Basismodell erstellt. Dieses wurde um drei Erweiterungen angereichert: objektorientierter Systemzustand, Unterscheidung zwischen exogenen Ereignissen und Folgeereignissen und Angabe von Zustandsbedingung und -effekt. Diese Erweiterungen wurden in der Objektbasierten Simulation zusammengeführt, die wiederum um zwei Erweiterungen angereichert wurde. Diese sind zum einen die Modellierung von Nachrichten und zum anderen die Trennung von externem und internem Agentenzustand bzw. Umgebungs- und Agentensimulatoren. Es wurde gezeigt, wie man ein Agentenbasiertes Simulationsmodell in ein äquivalentes Objektbasiertes Simulationsmodell und ein Objektbasiertes Simulationsmodell in ein äquivalentes Simulationsmodell der Diskreten-Ereignis-Simulation transformieren kann. Somit stellt die Agentenbasierte Simulation eine Verfeinerung der Diskreten-Ereignis-Simulation dar.

Zur (vollständigen) visuellen Spezifikation eines konkreten Agentenbasierten Simulationssystems wurde eine UML-basierte Spezifikationssprache entwickelt, die Reaktionsregeln zur Verhaltensspezifikation der Simulatoren verwendet. Sie werden in UML als n-äre Assoziationen zwischen den beteiligten Agenten-, Objekt-, Ereignis-, Wahrnehmungs-, Aktions- und Nachrichtentypen ausgedrückt. Bedingungen und Effekte der Regel werden als den Assoziationsenden zugeordneten OCL-Ausdrücke spezifiziert. Ferner wurde eine AORML-basierte Spezifikationssprache entwickelt, die zwar als agentenorientierte Sprache eine natürlichere Modellierung erlaubt, aber aufgrund mangelnder Toolunterstützung sich momentan noch nicht für eine automatisierte Weiterverarbeitung eignet.

In dieser Arbeit wurden als ausführlich behandeltes Beispiel Fahrerlose Transportsysteme gewählt, die zudem mittels UML, AUML und AORML objekt- und agentenorientiert modelliert wurden. Aufgrund der Möglichkeiten der dezentralen Steuerung und ihrer natürlichen Aufteilung in Fahrzeuge, Maschinen, Fahrkurs und Steuerungssystem sind Fahrerlose Transportsysteme besonders gut als Demonstrationsbeispiel für Agentenbasierte Simulation geeignet. Ferner wurden Fahrstuhlsysteme und Warteschlangen im Supermarkt als Beispiele verwendet.

Als *proof of concept* wurde ein Simulationssystem entwickelt, dessen Kern eine Java-Programmbibliothek ist, deren Klassen mittels Vererbung verfeinert werden, um die konkreten Agenten-, Objekt-, Ereignis-, Wahrnehmungs-, Aktions-, Nachrichtentypen sowie die konkreten Regeln darzustellen. Mit Hilfe dieses Simulationssystems wurde ein komplexes dezentral gesteuertes Fahrerloses Transportsystem simuliert und auf seine Effizienz untersucht. Aus einem in der UML-basierten Spezifikationssprache modellierten visuellen Simulationsmodell kann mit Hilfe der gängigen UML-Tools ein textuelles Simulationsmodell in der XML-basierten Sprache XMI generiert werden. Mittels XSL-Transformationen wird dieses Modell zunächst in ein XML-basiertes Zwischenformat und dann direkt in vom Simulator verwendeten Java-Code transformiert. Somit ist es möglich, ein visuell spezifiziertes Simulationsmodell automatisch in ein Simulationsprogramm zu transformieren und dieses vom Simulator ausführen zu lassen.