9. Ausblick

Dieses Kapitel reißt verwandte und weiterführende Themen an, die in dieser Arbeit nicht ausführlich behandelt werden konnten. In Abschnitt 9.1 wird eine generische Visualisierung und Sonifizierung für Simulationssysteme behandelt mit der unabhängig von der verwendeten Domäne der Verlauf und das Ergebnis des Simulation visualisiert werden kann. Dazu gehört auch die Verwendung von statistischen Verfahren. Abschnitt 9.2 behandelt die Fragestellung, wie das vorhandene zentralistische Simulationssystem um Aspekte der Verteilten Simulation angereichert werden kann. Abschnitt 9.3 behandelt die Fragestellung, wie die Agentenbasierte Simulation um Echtzeitanforderungen erweitert werden kann. Abschnitt 9.4 hingegen beschäftigt sich mit der Thematik, wie die Formalisierung der Agentenbasierten Simulation zu einer Interaktiven Agentenbasierten Simulation angereichert werden kann.

9.1 Visualisierung

In der in Abschnitt 8.3 vorgestellten Implementierung des Simulators ABSim gibt es nur eine einfache generische Ausgabe des Ablaufs und der Ergebnisse der Simulation. Der Benutzer hingegen hat die Möglichkeit, den Java-Code um eine, an die spezielle Domäne angepasste Visualisierung zu erweitern. Das ist aber keine ideale Lösung, da zum einen das Prinzip, dass der Benutzer nicht selbst programmieren muss aufgegeben wird, zum anderen es sehr aufwendig ist, für jede Simulation eine eigene Visualisierung zu entwickeln.

Deshalb wird vorgeschlagen, eine generische Visualisierung zu verwenden, die es ermöglicht, den Ablauf der Simulation unabhängig von einer konkreten Domäne auf geeignete Art und Weise grafisch darzustellen. Dabei soll es, soweit erforderlich, dem Benutzer möglich sein, Hinweise für die Visualisierung (und ggf. Sonifizierung) ins UML-Modell einzubauen. Eine geeignete Visualisierung ist für die Mensch-Maschine-Interaktion von großer Bedeutung.

In Abschnitt 9.1.1 wird die vorhandene Darstellung der Ergebnisse im Simulator ABSim dargestellt. Abschnitt 9.1.2 stellt Ansätze für eine generische Visualisierung vor, während Abschnitt 9.1.3 Ansätze für einen generischen Statistik-Output vorstellt.

9.1.1 Vorhandene Visualisierung

Im Simulator ABSim besteht die Möglichkeit für den Benutzer, eine auf die jeweilige Domäne zugeschnittene Visualisierung zu implementieren. Dazu muss die abstrakte Klasse *Visualisation* implementiert werden, in der sich abstrakte Methoden befinden, die das Simulationssystem aufruft, um die Visualisierung über Vorkommnisse im Ablauf der Simulation zu unterrichten. In Abbildung 93 sind diese Methoden dargestellt. Dabei wird die Visualisierung mittels *notifyEvent()* bzw. *notifyInternalEvent()* über ein auftretendes Ereignis bzw. internes Ereignis informiert, mittels *notifyERule()* bzw. *notifyARule()* über die Auswahl einer U-Regel bzw. A-Regel sowie mittels *notifyAction()*, *notifyPerception()* bzw. *notifyMessage()* über eine ausgeführte Aktion, eine Wahrnehmung und eine versendete Nachricht.

```
public abstract void notifyEvent(EnvEvent e);

public abstract void notifyInternalEvent(InternalEvent e);

public abstract void notifyERule(EnvRule r);

public abstract void notifyARule(AgentRule r, InternalAgent a);

public abstract void notifyAction(Action a);

public abstract void notifyPerception(Perception p);

public abstract void notifyMessage(Message m);

public abstract void notifyNewState(double clock);

public abstract void notifyEnd();
```

Abbildung 93: Methoden der abstrakten Klasse *Visualisation* im Simulator OBSim

Eine konkrete Unterklasse dieser abstrakten Klasse könnte beim Ablauf der Fahrstuhlsimulation z.B. die aktuelle Situation wie in Abbildung 94 zu sehen zeigen. Dabei werden die beiden Fahrstühle jeweils in ihrer aktuellen Position gezeigt. Bei fahrenden Fahrstühlen wird die Fahrtrichtung durch

einen Pfeil gekennzeichnet, Personen im Fahrstuhl und in den Etagen auf einen Fahrstuhl wartend werden als Figuren dargestellt. Eine Anforderung eines Fahrstuhls zur jeweils oberen bzw. unteren seiner beiden angefahrenen Etagen wird durch ein Kästchen oben bzw. unten im Fahrstuhl dargestellt. Das Kästchen ist schwarz bei einer Anforderung von außen und ansonsten weiß. Wartet ein Fahrstuhl auf eine Erlaubnis des anderen Fahrstuhls, so wird dies mit einer Sanduhr gekennzeichnet. Hat sich ein Fahrstuhl vorgemerkt, dass der andere Fahrstuhl auf eine Erlaubnis wartet, so wird dies durch ein Gehirn gekennzeichnet. Der bevorrechtigte Fahrstuhl ist durch eine Sonne gekennzeichnet.

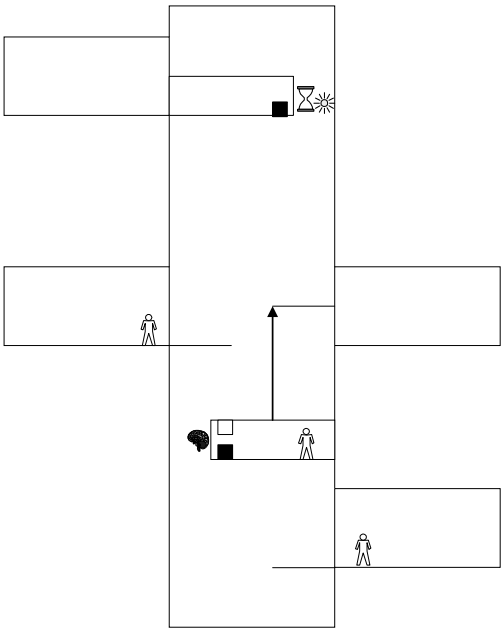

Abbildung 94: Spezialisierte Visualisierung der Fahrstuhlsimulation

Es versteht sich von selbst, dass eine solche Visualisierung nicht automatisch aus den in UML spezifizierten Reaktionsregeln generiert werden kann, da hierzu spezielles Domänenwissen bezüglich der Semantik der einzelnen Zustandsattribute und Ereignisse erforderlich wäre. Der Benutzer müsste stattdessen eine Unterklasse der Klasse Visualisation bilden.

Tut der Benutzer dies nicht, z.B. weil eine domänenspezifische Visualisierung zu aufwendig wäre, wird eine Standardvisualisierung eingesetzt. Diese ist jedoch nicht grafisch, sondern rein textbasiert. Das jeweilige Ereignis bzw. interne Ereignis, die Reaktionsregel bzw. die Aktion, die Wahrnehmung und die Nachricht werden auf der Konsole ausgeben, wobei mit Hilfe von Reflection-Klassen die Attributwerte angezeigt werden. Am Ende jedes Zyklus (bei Aufruf von *notifyNewState()*) wird der komplette Systemzustand zusammen mit der aktuellen Zeit ausgegeben. Im Fahrstuhlbeispiel könnte die Standardvisualisierung z.B. die in Abbildung 95 dargestellte Ausgabe erzeugen. Alternativ dazu könnte eine Standardvisualisierung auch XML-basiert sein, um eine Ex-Post-Statistik zu ermöglichen. Nachteil wäre eine schlechtere Lesbarkeit auf der (textbasierten) Konsole.

```
...
Next internal event from agent Elevator 1:
absimulation.xamples.elevator.PerceptionArrivalAtFloor
  floor: 2

Chosen internal rule of agent Elevator 1:
absimulation.xamples.elevator.PerceptionArrivalAtFloor4

New state at 9785.0:
  absimulation.xamples.elevator.Elevator 1
    position: 2
    standing: true
  absimulation.xamples.elevator.Elevator 2
    position: 3
    standing: true
  absimulation.xamples.elevator.IElevator 1
    preferred: false
    memorized: false
    waiting: false
    position: 2
    standing: true
  absimulation.xamples.elevator.IElevator 2
    preferred: false
    memorized: false
    waiting: false
    position: 3
    standing: true
...
```

Abbildung 95: Generische textbasierte Visualisierung der Fahrstuhlsimulation

Vorteil dieser Visualisierung ist, dass man als Benutzer keinen Mehraufwand für ihre Erstellung hat. Allerdings ist der große Nachteil, dass man keine grafische Veranschaulichung des Simulationsablaufs hat und es damit schwer ist, den Simulationsablauf nachzuvollziehen. Im nachfolgenden Abschnitt werden deshalb Ansätze gezeigt, wie man diese Standardvisualisierung durch grafische Darstellung verbessern kann.

9.1.2 Generische Visualisierung

Das Problem einer geeigneten generischen Visualisierung ist schwierig, da aus dem UML-Modell die Semantik einzelner Ereignisse nicht erschließbar ist.

Als Ansatz für die Visualisierung sind geeignete Darstellungsparameter mit entsprechenden ontologischen Kategorien zu verknüpfen, z.B. haben *Substantials* räumliche Ausdehnung und Koordinaten, während *Occurrences* raumzeitliche Ausdehnung und Koordinaten haben (siehe [DHH+01, GGM+02]). Es müsste daher eine Systematik der Darstellungsoptionen entwickelt werden, die die wichtigsten Simulationsabstraktionen (z.B. keine Ausdehnung, keine Form, keine Farbe, etc.) beinhaltet.

Die in Abschnitt 7.1 vorgestellte Simulationsspezifikationssprache müsste dann um Visualisierungs- und Sonifizierungskonstrukte erweitert werden. So könnte beispielsweise mit Stereotypes zwischen Entitäten mit und ohne physikalische Koordinaten unterschieden werden. Bei den Attributen würde mit Hilfe von Stereotypes ausgedrückt werden können, dass es sich um eine physikalische Koordinate handelt. Die entsprechenden Entitäten könnten dann am Bildschirm entsprechend ihrer aktuellen Koordinate dargestellt werden. Die Frage der Bildschirmanordnung der Entitäten ohne physikalische Koordinaten wäre aber weiterhin offen.

Ein spezielles Problem ist die Darstellung mentaler und sozialer Entitäten (wie z.B. Überzeugungen, Gefühlszustände, Verpflichtungen, Kommunikationshandlungen, etc.). Hier kämen auch Möglichkeiten der Sonifizierung in Frage. Nachrichten könnten beispielsweise aber auch durch einen Pfeil oder Blitz vom Sender zum Empfänger dargestellt werden.

Aktionen und damit verbundene Wahrnehmungen könnten durch gleiche Farben oder durch Verbindungslinien gekennzeichnet sein.

9.1.3 Statistik

Zur Darstellung der Ergebnisse des Ablaufs der Simulation gehört auch eine generische statistische Auswertung. Im Gegensatz zur grafischen Visualisierung stellt diese aber kein schwieriges Problem

dar. Im Folgenden wird dargelegt, welche Daten als Ausgabe der Statistik bei der Durchführung einer Simulation unter anderem interessant sein könnten.

Für jeden Ereignistyp:

- Anzahl des Auftretens
- Zeitpunkt des ersten/letzten Auftretens
- kürzester/längster Zeitabstand zwischen zwei Auftreten

Für jeden Parameter eines Ereignisses:

- Verteilung der Werte
- maximaler/minimaler/Durchschnittswert (bei numerischen Parametern)

Die gleichen Daten sind auch für interne Ereignisse, Aktionen, Wahrnehmungen und Nachrichten interessant, wobei hier noch die Verteilung auf die einzelnen Agenten interessant ist, z.B.

Für jeden Aktionstyp

- minimale/maximale/durchschnittliche Anzahl der Durchführungen durch Agenten

Für jedes Attribut des Zustands eines Objekts und des externen und des internen Zustands eines Agenten:

- Verteilung der Werte (gemessen an Simulationszeit)
- maximaler/minimaler/Durchschnittswert (bei numerischen Parametern)

Auch die ausgewählten Regeln durch den Umgebungssimulator und die Agentensimulatoren bieten zahlreiche Möglichkeiten für die statistische Untersuchung.

Ein weiterer interessanter Aspekt der Statistik wäre die Definition von Kosten wie z.B. Wartezeiten als Zeit zwischen zwei Ereignissen. Dabei ist zwischen Fällen zu unterscheiden, in denen man die Zeit zwischen einem Ereignis und einem dem Ereignis zuordenbaren späteren Ereignis eines anderen Typs misst (z.B. im Supermarkt die Zeit zwischen *Kunde_kommt_an* und dem entsprechenden *Bedienung_beendet*) und Fällen, in denen man die Zeit zwischen einem Ereignis und dem nächsten Ereignis eines anderen Typs misst (z.B. im Fahrstuhl-Beispiel die Zeit zwischen *Wird_angefordert* und *Fährt_an*).

9.2 Verteilte Simulation

Dieser Abschnitt behandelt die Fragestellung, wie das vorhandene zentralistische Simulationssystem um Aspekte der Verteilten Simulation angereichert werden kann.

Bei Verteilung kann man zwischen Verteilung des Modells und Verteilung der Simulationsausführung unterscheiden. Unter Verteilung des Modells versteht man die Aufteilung des Modells in verschiedene Entitäten, was der Grundidee der Objektbasierten und der Agentenbasierten Simulation entspricht. Unter Verteilung der Simulationsausführung versteht man hingegen die gleichzeitige Ausführung der Simulation auf mehreren Rechnern.

Bei Ghosh [Gho02] wird eine verteilte Simulation eines großen verteilten, aus vielen Subsystemen bestehenden Systems als realistischer angesehen. Diese Ansicht wird hier nicht geteilt, da die beiden Aspekte der Verteilung weitestgehend orthogonal zueinander sind.

Selbst für z.B. die Simulation unzuverlässiger Verteilter Systeme (Verlust von Nachrichten, Ausfall von Rechnern) ist es zweckmäßiger, eine zentralistische Simulation (oder eine Simulation auf einem zuverlässigen Verteilten System) durchzuführen und dabei den Verlust von Nachrichten und den Ausfall von Rechnern mit einer Wahrscheinlichkeitsverteilung zu modellieren als die Simulation auf einem verteilten Rechnersystem ablaufen zu lassen und dabei z.B. die Nachrichten, die auf dem realen verteilten System verloren gehen, als in der Simulation verloren gegangen anzusehen. Eine solche Vorgehensweise wäre nur dann realistisch, wenn man sie direkt auf dem Zielsystem, also auf dem Verteilten System, das untersucht werden soll, durchführt. Dann kann man aber nicht von einer Simulation, sondern von einem Probelauf sprechen.

Mögliche Gründe für Verteilung der Simulationsausführung sind zum einen die Beschleunigung der Simulation und zum anderen Fehlertoleranz. Will man den Ablauf der Simulation beschleunigen, entweder um Echtzeitanforderungen zu erfüllen oder um schneller zu einem Ergebnis zu kommen, so bietet sich Verteilung an, ebenso, wenn man Fehlertoleranz erreichen will, also die Möglichkeit, mit dem Ausfall eines Rechners oder der Verbindung zu einem Rechner umzugehen.

Eine Verteilte Simulation ist ein spezieller Verteilter Algorithmus, so dass sich die selben Verteilungsmöglichkeiten ergeben wie allgemein für Verteilte Systeme (siehe z.B. [TS03, CDK01]). Mit Hilfe von Systemen wie z.B. Pangaea [Spie02] gibt es auch die Möglichkeit, ein zentralistisches Java-Programm automatisch zu verteilen.

Aufgrund der Aufteilung der Agentenbasierten Simulation in den Umgebungssimulator und die Agentensimulatoren ergibt sich in natürlicher Art und Weise die Möglichkeit der Verteilung. Damit aber die Verteilung optimal ausgenutzt werden kann und Umgebungs- und Agentensimulatoren gleichzeitig rechnen, ist es vorteilhaft, dass der Umgebungssimulator den Agentensimulatoren in Schritt 3 nicht erst die Menge von Wahrnehmungen des Agenten in diesem Zyklus (die er im vergangenen Zyklus ermittelt hat), sondern bereits die Menge der Wahrnehmungen im nächsten Zyklus (die er gerade in Schritt 2 ermittelt hat) sendet. Da sich an der Ausführungssemantik nichts ändern soll, darf der Agent diese Wahrnehmungen erst bei der Berechnung im darauf folgenden Zyklus verwenden. In Abbildung 96 und Abbildung 97 ist der Ablauf der Simulation mit den nötigen Synchronisationen im zentralen und verteilten Fall als UML-Aktivitätsdiagramm dargestellt.

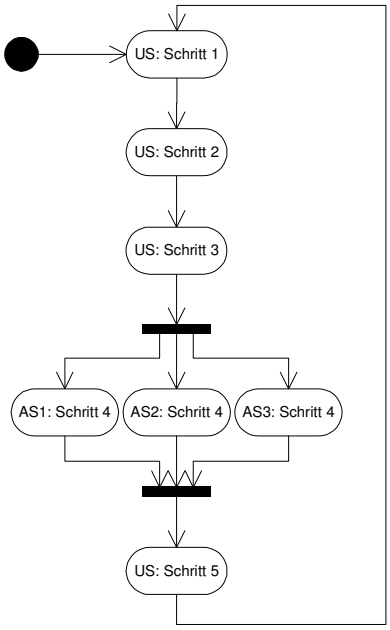

Abbildung 96: Ablauf der Simulation im zentralen Fall als UML-Aktivitätsdiagramm

Im zentralen Fall ist der Umgebungssimulator mit den Schritten 1 bis 3 an der Reihe, danach die Agentensimulatoren mit ihrem Schritt 4 (nacheinander oder parallel in mehreren Threads) und dann wieder der Umgebungssimulator mit Schritt 5. Eine Aufteilung auf mehrere Threads würde auf einem Einprozessorrechner aufgrund zusätzlichem Aufwand in der Threadverwaltung die Simulation leicht verlangsamen, allerdings bei einem Mehrprozessorrechner die Simulation erheblich beschleunigen.

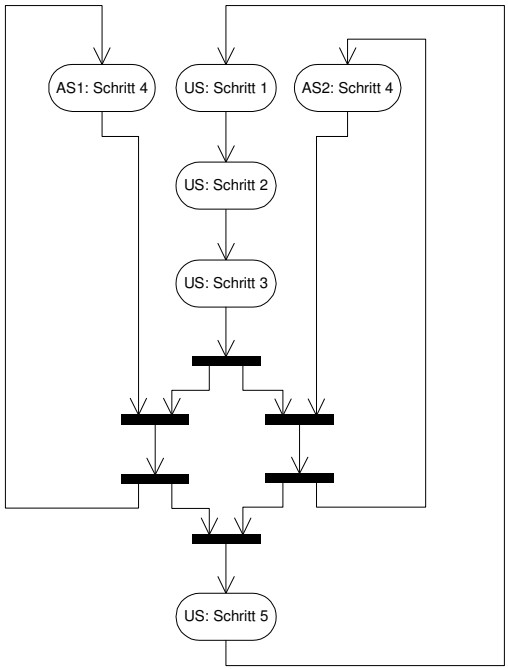

Abbildung 97: Ablauf der Simulation im verteilten Fall als UML-Aktivitätsdiagramm

Im verteilten Fall kann jeder Agentensimulator parallel zu den Schritten 1 bis 3 des Umgebungssimulators seinen Schritt 4 ausführen, da er ja bereits im vorherigen Zyklus die Wahrnehmungen für diesen Zyklus empfangen haben. Wenn der Umgebungssimulator seinen Schritt 3 ausgeführt hat, kann er bereits mit dem Schritt 4 des darauf folgenden Zyklus beginnen. Der Umgebungssimulator hingegen muss warten, bis alle Agentensimulatoren ihren Schritt 4 ausgeführt haben, bevor er mit Schritt 5 fortfahren kann. Da die Agentensimulatoren aber ihren Schritt 4 parallel zu den Schritten 1 bis 3 ausführen können, ist hier mit keiner Wartezeit oder zumindest mit einer geringeren als im zentralen Fall zu rechnen.

Aus Gründen der Übersichtlichkeit ist in Abbildung 97 der Startzustand nicht dargestellt. Die Simulation beginnt beim Umgebungssimulator mit Schritt 1 und bei den Agentensimulatoren mit Schritt 4.

9.3 Echtzeitsimulation

Dieser Abschnitt behandelt die Fragestellung, wie die Agentenbasierte Simulation um Echtzeitanforderungen erweitert werden kann. Wie bereits in Abschnitt 4.3.1.3 dargestellt, kann der Fortschritt der Zeit in einer Simulation schneller oder langsamer als der Fortschritt der realen Zeit sein. Es gibt aber Fälle, in denen man die Simulationszeit genauso schnell voranschreiten lassen möchte wie die reale Zeit. Man spricht dann von Echtzeitanforderungen.

Läuft die Simulation von sich aus schneller ab als in Realzeit, so sind Echtzeitanforderungen leicht zu erfüllen, indem man die Simulation einfach am Ende jedes Zyklus so lange ausbremst, bis die Zyklendauer auch in realer Zeit abgelaufen ist. Läuft sie hingegen langsamer ab, so muss man versuchen, die Simulation zu beschleunigen, etwa durch Verteilung, wie in Abschnitt 9.2 beschrieben.

Echtzeitanforderungen können nur dann entstehen, wenn ein Benutzer oder andere Systeme am System aktiv teilnehmen und somit von außen Ereignisse erzeugen, auf die das Simulationssystem reagieren muss. Man spricht dann von Interaktiver Simulation, die in Abschnitt 9.4 ausführlich behandelt wird.

Echtzeitanforderungen ohne Interaktion hat man normalerweise nur, wenn die Visualisierung in Echtzeit ablaufen soll. Ist das nicht möglich, da die Simulation zu langsam abläuft, kann man nach dem Ablauf der Simulation die Visualisierung in Echtzeit ablaufen lassen.

Andere Fälle von Echtzeitanforderungen sind nicht plausibel. Findet keine Interaktion statt und gibt es keine in Echtzeit ablaufende Visualisierung, so ist das entscheidende Kriterium eine Verkürzung der Ablaufzeit der Simulation. Insofern könnte es auch hier z.B. wünschenswert sein, die Dauer des

Ablaufs einer Simulation mit einer Stunde Simulationszeit von 65 Minuten auf 60 Minuten zu verkürzen, aber eine weitere Verkürzung auf 55 Minuten wäre noch wünschenswerter.

9.4 Interaktive Simulation

Dieser Abschnitt beschäftigt sich mit der Thematik, wie die Formalisierung der Agentenbasierten Simulation zu einer Interaktiven Agentenbasierten Simulation angereichert werden kann. Dazu wird zunächst in Abschnitt 9.4.1 eine kurze Einleitung in Interaktive Simulation gegeben. Abschnitt 9.4.2 stellt mögliche Gründe für Interaktive Simulation dar. In Abschnitt 9.4.3 wird dann zwischen drei Formen Interaktiver Simulation unterschieden. Abschnitt 9.4.4 untersucht vorhandene Interaktive Simulationssysteme. In Abschnitt 9.4.5 wird dann die Möglichkeit der Erweiterung der Agentenbasierten Simulation auf Interaktive Agentenbasierte Simulation untersucht.

9.4.1 Einleitung

Das Paradigma der Agentenorientierung eröffnet neue Möglichkeiten zur Erforschung der Interaktiven Simulation, in der Akteure und technische Systeme mit einem Simulationssystem interagieren. Interaktive Simulation ist nicht nur die Basis für viele Computerspiele, sondern auch für neuere Forschungen in der Simulation sozialer und ökonomischer Systeme. In der Agentenbasierten Simulation gibt es eine natürliche Modularisierung: die Gesamtsimulation besteht aus dem Umgebungssimulator und den Agentensimulatoren. Man kann sich vorstellen, dass in einem Simulationslauf die Rolle eines Agentensimulators auch durch sein reales Pendant (Mensch, Tier, Softwaresystem oder Maschine) gespielt werden kann. Der reale Akteur würde dann als Subsystem an der Simulation des Gesamtsystems teilnehmen.

Wie bereits in Abschnitt 1.5.2 dargestellt werden in [MD+02] Anforderungen an Plattformen für Agentenbasierte Simulation zusammengestellt. Insbesondere sollen sie

- kontrollierbare Simulationen mit wiederholbaren Ergebnissen ermöglichen;
- asynchronen Nachrichtenaustausch zwischen Agenten zulassen;
- außer externen Ereignissen (wie z.B. Nachrichtenaustausch, Handlungen von Agenten) auch interne/kognitive Ereignisse beobachtbar machen;
- verschiedene (vom Simulator kontrollierte und nicht vom Simulator kontrollierte) ‚Umgebungen' in die Simulation einbinden können;
- Mechanismen bereitstellen, die eine Intervention des Experimentators in das Simulationsgeschehen ermöglichen.

Wie die Autoren bemerken, werden speziell die letzten zwei dieser Anforderungen von den heute verfügbaren Plattformen nicht erfüllt. Sie verlangen eine gewisse Unterstützung der Interaktiven Simulation, wobei Interaktivität bei ihnen nicht als generisches Konzept betrachtet wird.

9.4.2 Gründe Interaktiver Simulation

In der Literatur über Interaktive Simulation werden Systeme beschrieben, die außer dem Zweck der Erforschung eines Realsystems verschiedenen anderen Zwecken dienen, z.B.

- zum Lernen-Was (Wissensaneignung)
- zum Lernen-Wie (Steigerung der Handlungskompetenz)
- zur Durchführung von Wettbewerben
- zum Spielen
- zur Produktion von interaktiven Kunstwerken.

Diese Zwecke können sich überlappen und sind mit unterschiedlichen Anforderungen an die Interaktivitätsmöglichkeiten und mit unterschiedlichen Rollen verknüpft. Bei einer Simulation zur Erforschung eines Realsystems gibt es die Rollen des Experimentators (derjenige, der die Simulation durchführt) und der Simulationsteilnehmer, bei einer Simulation zum Lernen gibt es die Rollen des Tutors und des/der Lernenden, bei einer Simulation zur Durchführung von Wettbewerben gibt es die Rollen des Wettbewerbsleiters und der Teilnehmer, bei einer Simulation zur Durchführung von

Spielen gibt es die Rollen des Spielleiters und des Spielers (bzw. der Spieler), und bei einer Simulation zur Produktion von interaktiven Kunstwerken gibt es die Rollen des Regisseurs und der Mitspieler.

Simulationszweck	*Rollen*
Erforschung eines Realsystems	Experimentator, Simulationsteilnehmer
Lernen	Tutor, Lernende
Durchführung von Wettbewerben	Wettbewerbsleiter, Teilnehmer
Durchführung von Spielen	Spielleiter, Spieler
Produktion von interaktiven Kunstwerken	Regisseur, Mitspieler

Abbildung 98: Simulationszweck und Rollen in der Interaktiven Simulation

9.4.3 Formen Interaktiver Simulation

Der Begriff *Interaktivität* wird mit verschiedenen Bedeutungen verwendet. Bei einer Simulation zum *Lernen-Was* (Wissensvermittlung) kann die Interaktivität darauf beschränkt sein, Lernenden die Möglichkeit zu geben, durch die direkte Manipulation von Simulationsobjekten in das Simulationsgeschehen zu intervenieren. Diese Art der Interaktivität, *Benutzer-Intervention*, wird mit den Funktionen einer (grafischen) Benutzerschnittstelle realisiert. Bei einer Simulation zum *Lernen-Wie* (Steigerung der Handlungskompetenz) geht es um eine andere Form von Interaktivität: hier werden Lernende als *intentionale Systeme*[99], die mit der simulierten Umgebung und ggf. anderen an der Simulation teilnehmenden intentionalen Systemen interagieren, in die Simulation eingebunden. Ein Beispiel für diese Art der Interaktivität, *Benutzer-Partizipation*, sind Flugsimulatoren. Bei einer Simulation zum *Spielen* geht es in der Regel ebenfalls um Benutzer-Partizipation, die mit geeigneten Benutzerschnittstellen realisiert wird. Demgegenüber geht es bei einer Simulation zur *Durchführung von Wettbewerben* allgemeiner um *System-Partizipation*, da neben Menschen auch künstliche intentionale Systeme, wie Softwaresysteme oder Maschinen, teilnehmen können. Simulationssysteme mit System-Partizipation benötigen eine anspruchsvollere Architektur: sie werden als verteilte Systeme mit offenen Schnittstellen und standardisierten Interaktionsprotokollen realisiert.

9.4.4 Interaktive Simulationssysteme

Beispiele für Interaktive Simulationssysteme, die nur die Benutzer-Intervention unterstützen, sind *InterSim* [KO+98], ein System zur Erstellung von Simulationen zum Lernen, *AgentSheets* [Rep00], ein Tool zur Erstellung Agentenbasierter Simulationen in der Form von Java-Applets, und *Jane* [PF99], ein Programmier-Framework für die Visualisierung und Benutzer-Intervention bei parallelen und verteilten Simulationen.

Ein Beispiel für ein Interaktives Simulationssystem, das sowohl Benutzer-Intervention als auch Benutzer-Partizipation unterstützt, ist *CourseWare* [Cour04, Boo99]. Bei einer CourseWare-Simulation kann der Benutzer z.B. die Rolle eines Bürgermeisters spielen, der durch umweltpolitische Maßnahmen das Aussterben bedrohter Arten verhindern soll (Benutzer-Partizipation). Bei anderen CourseWare-Beispielen kann man in die Simulation (z.B. mit Hilfe der Maus) eingreifen, ohne selbst eine Rolle zu spielen (Benutzer-Intervention). So kann man eine Halbkugel, auf der Gänseblümchen wachsen, durch Drehungen unterschiedlichem Lichteinfall aussetzen und die entsprechenden Auswirkungen beobachten oder man kann durch Einstellung gewisser Parameter die Entwicklung einer Bakterienkultur beeinflussen.

Die *High Level Architecture (HLA)* [HLA04] ist eine unter der Führung des U.S. Defense Modelling and Simulation Office entwickelte und als IEEE-Standard anerkannte Architektur zur Realisierung von verteilten Simulationssystemen auf der Basis objekt-orientierter Softwaretechnik mit dem ausdrücklichen Ziel, die Wiederverwendbarkeit und die Interoperabilität von Simulationsmodellen zu ermöglichen. Sie unterstützt Interaktive Simulationen mit System-Partizipation, sofern die einzubindenden Fremdsysteme sich an die HLA-Norm halten. Alle Entitäten eines Realsystems wer-

[99] Systeme, die am besten dadurch erklärt werden können, dass man ihnen einen mentalen Zustand unterstellt und annimmt, dass ihre Handlungen auf Wahrnehmungen, Überzeugungen und Absichten beruhen [Den71].

den in einem HLA-Simulationsmodell als Instanzen entsprechender Objektklassen repräsentiert. Eine HLA-Simulation (‚federation') besteht aus mehreren Modellen bzw. Programmen (‚federates'), die verteilt ausgeführt werden und mit Hilfe eines speziellen Nachrichtenaustauschmechanismus (‚Runtime Infrastructure, RTI') interagieren. Die im RTI enthaltenen Funktionen zum Zeitmanagement erlauben lokal unterschiedliche logische Zeiten (asynchrone verteilte Simulation) und verschiedene Zeitprogressionsverfahren (stetig und ereignisbasiert) bei Wahrung einer globalen logischen Zeit.

9.4.5 Interaktive Agentenbasierte Simulation

Zu den bereits in Abschnitt 1.6 dargestellten Anforderungen für Agentenbasierte Simulation kommen bei der angestrebten Interaktiven Agentenbasierten Simulation die Unterstützung aller drei Formen der Interaktivität hinzu, so dass die folgenden Anforderungen von Belang sind:

- die Simulationsspezifikationssprache und der Simulator basieren auf einem agentenorientierten Metamodell
- die Simulationsspezifikationssprache ist deklarativ
- es gibt eine visuelle, UML-basierte Simulationsspezifikationssprache
- alle drei Formen der Interaktivität (Benutzer-Intervention, Benutzer-Partizipation, System-Partizipation) werden unterstützt

Die Übersicht aus Abbildung 3, in der bereits die nicht die Interaktivität betreffenden Anforderungen für agentenbasierte Simulationssysteme untersucht wurden, wird in Abbildung 99 um die Anforderungen der Unterstützung der Formen der Interaktivität erweitert. Dabei wird deutlich, dass auch diese Anforderungen gar nicht oder nur teilweise erfüllt werden.

Anforderung	*Swarm*	*RePast*	*Spades*	*SeSAm*	*MadKit*	*CORMAS*	*RoboCup*	*TAC*
Metamodell	-	√	√	-	√	√	-	-
Deklarativ	-	-	-	√	-	-	-	-
Visuell	√	√	-	√	-	√	-	-
UML-basiert	-	-	-	-	-	-	-	-
Benutzer-Intervention	√	√	-	√	-	-	-	-
Benutzer-Partizipation	-	-	-	-	-	-	√	-
System-Partizipation	-	-	√	-	√	-	√	√

Abbildung 99: Vergleich einiger agentenbasierter Simulationssysteme

Zum Zwecke der Erweiterung der in dieser Arbeit vorgestellten Agentenbasierten Simulation zur *Interaktiven Agentenbasierten Simulation (IABS)* müssten Möglichkeiten einer systematischen Behandlung der Interaktivität in der Simulation untersucht werden. Das Hauptziel wäre die Entwicklung eines generischen Metamodells für die IABS auf der Basis der vorgestellten agentenorientierten Modellierungssprache, das ein abstraktes Ausführungsmodell für den Simulator beinhaltet. Der wesentliche Unterschied zum ABS-Metamodell wird sein, dass die eingebundenen Systeme (Mensch, Tier, Softwaresystem oder Maschine) nicht der Kontrolle des Simulators unterliegen. Anstelle eines Agentensimulators tritt hier das reale System. Zu untersuchen ist dabei vor allem:

- Ist es zweckmäßig, das eingebundene System im Umgebungssimulator und in den Agentensimulatoren zu repräsentieren? Wenn ja, wie kann das eingebundene System aus der Sicht des Umgebungssimulators und der Agentensimulatoren repräsentiert werden?
- Wie müssen die Schnittstellen zwischen dem Simulationssystem und dem eingebundenen System im Vergleich zu den Schnittstellen zwischen Umgebungssimulator und Agenten-simulatoren gestaltet werden?

Im Fall der Benutzerpartizipation geht es vor allem um die aus dem Gebiet der Human Computer Interaction (HCI) bekannte Frage der Benutzermodellierung. Im Fall der Systempartizipation geht es

sowohl um die mentalistische Repräsentation der beteiligten Systeme als auch um eine geeignete Agentenkommunikationssprache zum Nachrichtenaustausch sowie um darauf beruhende Interaktionsprotokolle. Die wichtigsten Simulationskontexte müssten im Metamodell in geeigneter Weise berücksichtigt werden.