Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 1 - WiSe 2010/11

Christian Thiel

Ein verteiltes Multiagentensystem zur Simulation von Fußgängerströmen

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einleitung

In den vergangenen Jahrzehnten kam es bei Großereignissen wie z. B. der Love Parade in Duisburg im Juli 2010 (21 Tote) oder im Februar 2004 in Mekka (251 Tote) [Pro] immer wieder zu Massenpaniken, denen zahlreiche Menschen zum Opfer gefallen sind.
Auch wenn oft genug eine Verkettung unglücklicher Umstände Ursache der Massenpanik ist, ist eine der Ursachen immer wieder das Fehlverhalten von Sicherheitskräften vor Ort während oder im Vorfeld der Panik. Eine weitere häufige Ursache ist die Fehleinschätzung der geographischen Gegebenheiten im Zusammenhang mit der Menge an erwarteten Personen, was oft schlicht zu einer massiven Überfüllung kritischer Bereiche führt und so Paniken auslöst.

Im Zeitalter der Informationstechnologien erscheint es nun natürlich, solche Massenpaniken zu analysieren und aus ihnen für die Zukunft zu lernen, um weitere Katastrophen in diesem Bereich zu verhindern.
So hat sich das WALK-Projekt der Hochschule für Angewandte Wissenschaften Hamburg zum Ziel gesetzt, ein Toolkit zu entwickeln, welches es Sicherheitskräften und Planern solcher Großereignisse bereits im Vorfeld die Möglichkeit gibt, kritische Bereiche des Geländes ausfindig zu machen und Evakuierungspläne entsprechend anzupassen.

Kernkonzept des WALK-Tools soll es einerseits sein, dass die einzelnen Personen der Simulation nicht, wie in anderen Simulationen, als Partikelsysteme oder Flüssigkeitsmodelle simuliert werden, sondern dass durch ein Multiagentensystem eine maximale Individualität der einzelnen Personen erreicht werden kann und so individuelle, persönliche Eigenschaften für jeden Agenten modelliert und simuliert werden können.
Das zweite Kernkonzept, welches jedoch nicht zentraler Teil dieser Arbeit sein wird, ist die Einbindung eines emotionalen bzw. psychologischen Verhaltensmodelles der Agenten. Auf diese Weise sollen persönliche Charaktereigenschaften wie Stressempfindlichkeit, Angst oder auch Führungskraft die simulation maßgeblich beeinflussen und ein realistisches Bild großer Menschenmassen darstellen.

Diese Arbeit soll nun einen kleinen Einblick in das WALK-Projekt und speziell die geplante Funktionsweise und den Aufbau des Agentensystems liefern, weiterhin auf entstehende Probleme und Risiken eingehen und ebenso aufzeigen, welche Forschungsziele sich hinter dem Projekt verstecken.

2 Die Plattform

Eine Schwierigkeit bei der Realisierung eines solchen Simulationssystems wird sein, dass die Intelligenz der einzelnen Agenten sehr komplex sein wird und letztlich quasi ungeahnte Ausmaße erreichen kann, je mehr Verhaltensregeln, emotionale und psychologische Grundsätze

man implementiert und je realistischer die Umwelt modelliert werden soll. Zur Verdeutlichung der Komplexität ein paar Beispielszenarien, was in so einer Simulation alles berücksichtigt werden kann:

- Ausbruch eines Feuers in einer U-Bahn-Station. Das Feuer breitet sich langsam aus, aber der Rauch behindert mangels ausreichender Belüftungssysteme schnell Atmung und Sicht der Personen auf dem Bahnsteig.
- Ein Vater wird im Fußballstadion direkt nach dem Abpfiff im Gedränge von seinen zwei kleinen Kindern getrennt und gerät in Panik.
- Beim Alstereisvergnügen bricht plötzlich an einer Stelle das Eis und die Menschenmassen verlassen panisch die Eisfläche und verursachen so weitere Eisbrüche.

Dieser Umstand führt dazu, dass größere Simulationen sehr wahrscheinlich nicht auf einem einzigen System ausgeführt werden können, weil die Rechenressourcen dafür nicht ausrechen. Die Simulationsplattform muss also auf mehrere oder besser beliebig viele Systeme verteilt werden können, ohne dass nennenswerte Engpässe und Flaschenhälse entstehen und so auch Simulationen mit mehreren Tausend Agenten möglich sind.

2.1 Gesamtkonzept

Die Agentenplattform an sich wird aus drei Bereichen bestehen.

2.1.1 Steuerung

Der SimulationManager ist die steuernde Komponente des gesamten Systems. Alle beteiligten Agenten-Server melden sich bei ihm an und erhalten von ihm Daten, welche Agenten sie für welches Szenario zu starten haben. Der SimulationManager hat so auch immer den Überblick über die Auslastung der einzelnen Knoten und kann ggf. gegensteuern, falls ein Knoten überlastet ist.

Der NameServiceAgent ist für die Namensauflösung verantwortlich. Da die einzelnen Agenten während der Simulation ihre zuständigen Knoten wechseln können, um eine gleichmäßige Auslastung sicherzustellen, muss eine zentrale Komponente für die Namensauflösung zuständig sein, damit für die Kommunikation der Ort des Gesprächspartners ermittelt werden kann.

Der DistributionController ist dafür zuständig, für eine gleichmäßige und effiziente Verteilung der Agenten auf den einzelnen Plattformen zu sorgen, damit weder ein Server überlastet ist noch ein anderer kaum Arbeit hat. Dafür kontrolliert er regelmäßig die Auslastung der Server

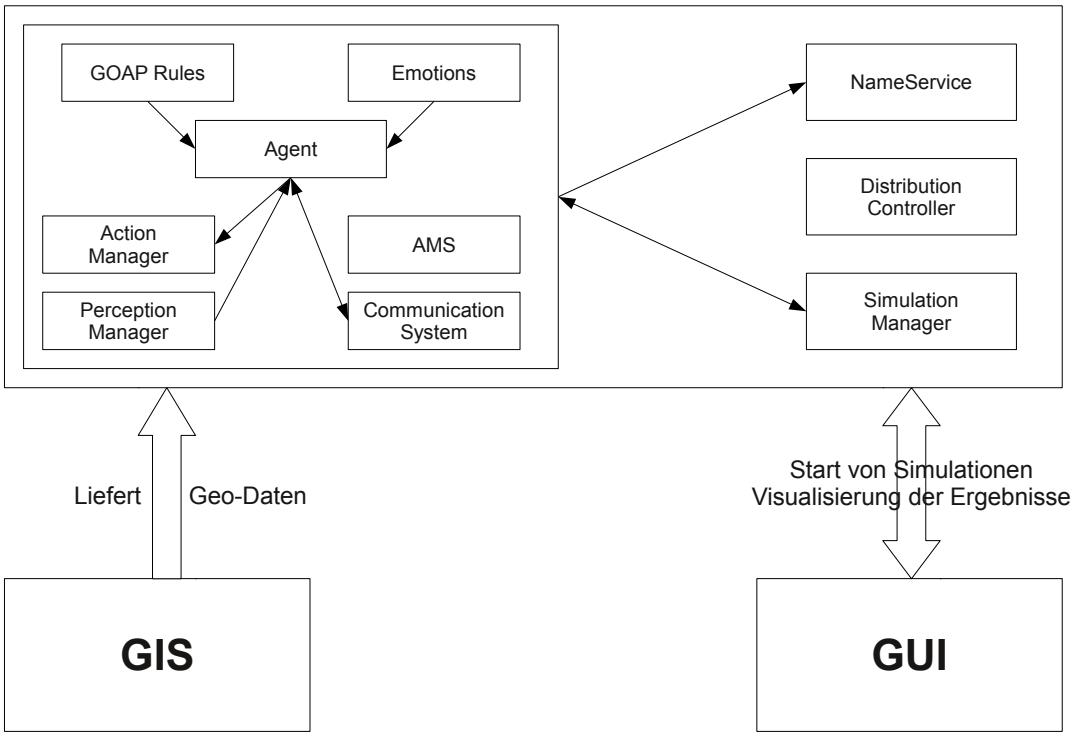

Abbildung 1: Komponenten der Agentenplattform

und analysiert auch die Agenten selbst, um eine effiziente Verteilung zu ermitteln und stößt bei den Plattformen eine Migration der Agenten auf eine benachbarte Plattform an.

2.1.2 Die Agentenplattform

Die Agentenplattform ist für die Laufzeitverwaltung der Agenten verantwortlich. Sie stellt unter anderem sicher, dass neue Agenten gestartet und gesteuert werden können, dass die Agenten untereinander kommunizieren können und versorgt den Eventloop der Agenten mit Ereignissen und organisiert deren Ausführung.

Der PerceptionManager der Agentenplattform ist für die Wahrnehmung der Agenten verantwortlich. Er hat Kontakt mit dem Geoinformationssystem und leite Wahrnehmungsänderungen ind er Umgebung der Agenten als Ereignis an diese weiter.

Der ActionManager ist für die Interaktion der Agenten mit der Umwelt verantwortlich. Jede Handlung, bei der ein Agent seine Umwelt oder seinen nach außen hin sichtbaren Zustand (Aussehen, Position, Geschwindigkeit) verändert, muss vom ActionManager geprüft und bei Erfolg dem Rest der Welt mitgeteilt werden.

2.1.3 Der Agent

Ein Hauptanliegen des Designs ist es, ein Agentensystem zu schaffen, welches sich von der konkreten Architektur des Systems der Entscheidungsfindung unabhängig macht. Der Unterbau stellt eine ereignisorientierte Ausführungs-Architektur zur Verfügung, auf welcher jedes andere beliebige System der künstlichen Intelligenz aufgesetzt werden kann, sodass ebenso evaluiert werden kann, welches Konzept für konkrete Fälle besser geeignet ist.

Nichtsdestotrotz ist einer der Hauptforschungspunkte des WALK-Projektes die Einbindung von emotionalen und psychologischen Verhaltensmodellen in die Entscheidungsfindung eines Agenten. In diesem Bereich hat Arne Klingenberg in seiner Bachelorarbeit [Kli09] bereits einige Vorarbeit geleistet, welche während des Projektes genutzt werden soll, um, basierend auf dem GOAP-Konzept [Jef], eine deklarative Agentensprache zu entwickeln. Diese Agentensprache soll den Endbenutzern des Simulations-Werkzeuges die Möglichkeit geben, schnell und ohne großes Informatiker-Fachwissen, die Handlungen und Aktionen des Agenten und die Einflüsse psychologischer Modelle zu zu entwickeln. Die Entwicklung des eigentlichen Agenteninnenlebens ist jedoch nicht Teil dieser Arbeit und befindet sich zudem noch in der frühen Entwicklungsphase und wird deshalb hier nicht näher erläutert.

Zusätzlich zu den bereits genannten internen Komponenten der Agentenplattform gibt es noch zwei weitere externe Komponenten, mit denen das System zusammenarbeiten muss, welche jedoch nicht Teil dieser Arbeit sind:

2.1.4 Geoinformationssystem

Das Geoinformationssystem liefert sämtliche Wahrnehmungsdaten auf Basis von geographischen Daten wie Gebäude- oder Geländeplänen und beinhaltet einen Simulator für z. B. auch Rauchentwicklung oder Feuerausbreitung. Aktueller Stand dieses Projektes ist es, dass für die Modellierung der Daten und Gebäudepäne das Dateiformat IFC [bui] und für die bereitstellung der Daten der openBIM-Server [Ope] verwendet werden soll. Der Zugriff auf diese Daten soll mit einer SQL-ähnlichen Sprache erfolgen (für weitere Informationen siehe auch [Bal11]).

2.1.5 GUI

Die GUI ist die Darstellungskomponente, welche sowohl live als auch bereits vergangene Simulationen visualisieren kann. Über die GUI werden neue Simulationen gestartet und gesteuert. Die Kommunikation zwischen GUI und dem Agentensystem wird über eine Webservice-Schnittstelle realisiert, bei der die GUI mit jedem Agenten als auch mit den einzelnen Plattformen Kontakt aufnehmen, Daten abrufen und in begrenztem Maße Daten und Aufgaben beeinflussen kann.

Eine möglichst lose Kopplung der GUI an die anderen beiden Komponenten und die universelle Webservice-Kommunikation erlaubt es, mehrere Arten von Benutzerinterfaces zu entwickeln. So soll neben dem üblichen PC-tauglichen Interface auch eine Microsoft-Surface Bedienung entstehen und es soll ebenso untersucht werden, ob sich ein mobiler Zugang via Smartphone als machbar erweist. Vor allem letzteres stellt eine hohe Herausforderung an das Design der Webservice-Schnittstelle, was Datenvolumen und Latenzzeiten angeht.

2.2 Verteiltes Wissen & Wahrnehmung

Eine der zentralen Herausforderungen in der Entwicklung des vorgestellten Multiagentensystems ist, dass das System keine zentrale, kontrollierende Einheit haben wird, welche den gesamten Zustand des Systems kennt und so einen eindeitigen *Ground Truth* darstellt. Das hat einerseits den Vorteil, dass kein Flaschenhals entsteht, wenn die Anzahl der Agenten in größeren Simulationen die Kapazität eines einzigen Systems übersteigt. Auf der anderen Seite eröffnet eine Verteilung des Weltzustandes allerdings einige Probleme, die noch gelöst werden müssen:

- Eine Verteilung und Replikation von Daten über das Netzwerk birgt immer die Gefahr von Inkonsistenzen innerhalb des Systems, da bei der Verteilung der Daten immer Latenzen entstehen
- Funktionen, die auf konsistente Zustände angewiesen sind (z. B. Kollisionserkennungen) werden komplizierter
- Das Sperren von kritischen Objekten wird aufwändiger

Dieser Abschnitt stellt vor, wie die Wahrnehmung und Aktionsausführung des Agenten realisiert werden soll und welche entsprechenden Schnittstellen dem eigentlichen Agenten dafür zur Verfügung gestellt werden sollen.

2.2.1 Das Wahrnehmungsmodell

Das Wahrnehmungsmodell versucht sich weitesgehend am C4 Modell [BID+01] und seiner Sensordefinition zu orientieren. Beim Start eines Agenten meldet sich dieser beim Perception-Manager an und bekommt eine Liste verfügbarer Sensoren zurück geliefert. Diese Sensoren stellen die Wahrnehmungsschnittstellen des Agenten dar. Er kann sich bei diesen Sensoren anmelden und abonniert damit quasi den Sensor und bekommt bei jeder Änderung des Sensors eine Ereignismeldung, welche er intern bearbeiten kann. Alternativ kann der Agent auch manuell on demand den Sensor nach seinem aktuellen Wert fragen und bekommt dann alle Daten, die der Sensor gerade beinhaltet.

Die Art der Wahrnehmungen, die der Sensor liefert, unterscheiden sich von Sensor zu Sensor:

- Objekte: Einige Sensoren (z.B. für physikalische Hindernisse) stellen Wahnehmungen als Objekte dar. Der Sensor liefert in einem Ereignis oder bei Direktabfrage dementsprechend eine Liste von wahrgenommenen Objekten zurück, welche widerrum sensorabhängig Daten enthalten
- Einzelwerte: Sensoren wie Temperatursensoren basieren biologisch gesehen weder auf Augen noch Ohren und liefern daher weder Positions noch Richtungsangaben sondenr nur einen stationären Zustandswert (meist wohl eine Zahl)
- Flächenwerte: Sensoren, welche biologisch gesehen visuell arbeiten, aber ihre Daten nur schlecht als Objekte darstellen können (z.B. Hochwasser, Untergrundbeschaffenheit), liefern bei Abfrage ein mehrdimensionales Array zurück, welches Flächen- bzw. Raumdaten enthält.

Mit diesen drei Grundtypen können nun so gut wie alle denkbaren Arten von Sensoren gebaut werden, welche dann von den Agenten benutzt werden können:

- **AgentSensor:** Liefert den externen Zustand aller Agenten, welche sich im Sichtbereich des Agenten befinden
- **ObstacleSensor:** Liefert Position, Form und Art aller für den Agenten sichtbaren Hindernisse
- **ObjectSensor:** Liefert wie der ObstacleSensor die Eigenschaften sichtbarer Objekte und zusätzlich noch mögliche Aktionen, die mit diesem Objekt ausgeführt werden können
- **TemperaturSensor:** Liefert die aktuell vom Agenten empfundene Temperatur als Zahlwert
- **GroundHeightSensor:** Liefert eine Heightmap des Sichtbereiches zurück

- **ForceSensor:** Liefert aktuell auf den Agenten von außen einwirkende Kräfte (mit Wirkungsvektor und Stärke)

Die Wissensquellen des PerceptionManagers können abhängig vom Sensor ebenfalls unterschiedlich sein. Sensoren wie der ObstacleSensor, welcher hauptsächlich Wände und Möbelstücke darstellen soll, basiert hauptsächlich auf den Daten, die das Geoinformationssystem liefert. Andere Sensoren, wie z. B. der AgentSensor basiert auf Daten, die er von den verschiedenen ActionManagern gesendet bekommt.

Das bereits in Abschnitt 2.1.4 beschriebene Geoinformationssystem (GIS), welches für einige Sensoren Daten liefert, teilt seine angebotenen Informationen in sogenannte Information Layer auf. Jeder dieser Layer enthält inhaltlich getrennte Daten, so kann ein Layer z. B. den Blueprint eines Gebäudes darstellen, ein anderer Layer bietet detailiertere Informationen über Bodenbeschaffenheit oder über Objekte, mit denen in gewisser Weise interagiert werden kann (z. B. Feuermelder, Fahrstühle oder auch Wegweiser). Andere Layer sind eher dynamischer Natur und können sich stetig ändern. So unterliegen die Layer über Rauch in der Luft oder Temperatur nach dem Ausbruch eines Feuers kontinuierlicher Veränderung, welche durch einen Simulator innerhalb des GIS berechnet wird.

2.2.2 Das Aktionsmodell

Die Handlungen von Agenten bestehen im Wesentlichen aus Aktionen, die sie ausführen. Dabei ist die Definition einer Aktion eine Handlung des Agenten, welche nach außen hin sichtbar ist und so die Wahrnehmung anderer Agenten direkt oder indirekt beeinflussen kann (z. B. Bewegung des Agenten oder Veränderung externer Objekte wie Türen oder ähnlichem). Interne Handlungen wie Schlussfolgerungen oder Zielentscheidungen verändern nicht die Wahrnehmung anderer Agenten, da diese auch gar nicht wissen dürfen, was andere Agenten denken.

Für die Ausführung solcher Aktionen ist der ActionManager zuständig. Dieser besitzt eine Schnittstelle zur Ausführung von externen Aktionen der Agenten und eine Menge vordefinierter Aktionen, die Agenten ausführen können. Die Menge dieser Aktionen ist durch die feste Compilierung vorgegeben, jedoch können einige Aktionen zur Laufzeit parametrisiert werden (z.B. eine Richtungsänderungsaktion bekommt einen Richtungsvektor übergeben).

Erstellt ein Agent eine Aktion und übergibt sie dem ActionManager, versucht dieser die Aktion auszuführen. Im Falle eines Erfolges wird dem Agenten sofort eine positive Rückmeldung gegeben, im Falle eines Fehlschlages entsprechend eine Fehlermeldung, welche auch den Grund des Fehlschlages angibt. Aktionen wie das Benutzen von externen Gegenständen erfordern eine Kommunikation mit dem GIS, auf welchem die Objekte verwaltet werden. Da der

Agent während der Wartezeit jedoch nicht komplett blockieren darf, gibt der ActionManager eine Art *TRYING*-Code zurück und liefert das endgültige Ergebnis der Aktion nachträglich durch eine Ereignismeldung an den Agenten aus.

Das Ergebnis der Aktion bzw. die Veränderung der Umwelt wird durch den ActionManager und den PerceptionManager ebenfalls allen anderen Agenten mitgeteilt, für die diese Veränderung sichtbar ist.

Für das Problem der Synchronisation der einzelnen Knoten existiert zur Zeit noch kein konkretes Konzept bzw. es stehen noch einige Konzepte und Ideen im Raum, welche diskutiert und evaluiert werden müssen. Zum einen währe hier das Konzept der vom Department of Defense entwickelte *High Level Architecture* [DFW97] [DFW98], welches eine große Simulation in viele kleine isolierte Simulationen zerlegt, welche untereinander Informationen und Objekte austauschen. Eine weitere Idee ist eine hochtransaktionale Datenbank, welche zugleich eine alternative Schnittstelle für die GUI wäre, sollte sich eine HTTP-Schnittstelle als unzureichend erweisen. Sollte sich in Tests ein stetiges, ereignisgesteuertes System als zu unpraktisch erweisen, bleibt als Alternative noch ein Schritt-synchronisiertes System wie in [Nod01] als Simulation für Fußballspiele benutzt.

Alle Aktionen, die in der Simulation stattfinden, werden vom ActionManager mit exaktem Zeitpunkt protokolliert, sodass eine komplette Historie der Simulation entsteht und nachträglich analysiert und visualisiert werden kann.

2.3 Verteilung der Agenten

Ein wichtiger Punkt in einem solchen verteilten Multiagentensystem ist die Verteilung der Agenten auf den einzelnen Plattform-Knoten, da diese ausschlaggebend dafür ist, wie effizient die Agenten miteinander kommunizieren können und wie groß der Aufwand für PerceptionManager und ActionManager ist, Wahrnehmungen und Aktionen zu verwalten.
Es ist schnell offensichtlich, dass Agenten, die viel miteinander zu tun haben, sei es, weil sie direkt miteinander kommunizieren oder weil ihre ausgeführten Aktionen untereinander sichtbar sind und publiziert werden müssen, auf dem gleichen Knoten liegen sollten. So ist der Kommunikationsaufwand lediglich auf die lokale Plattform beschränkt ist und belastet nicht die Netzwerkstruktur, was zudem zusätzliche Latenzen zur Folge hätte, die die Simulation beeinflussen können. Auch ist es für den PerceptionManager von Vorteil, wenn er nur einen kleinen Teil des GIS überwachen muss (nämlich die Gebiete, die seine Agenten auch sehen) und nicht das komplette GIS.

Das Hauptentscheidungsmerkmal einer Partitionierung der Agentenmenge ist die direkte und indirekte Kommunikation der Agenten untereinander. Da diese Kommunikation jedoch direkt abhängig von den Sichtbereichen der einzelnen Agenten ist, kann das Partitionierungsmerkmal

schlicht auf die Position der Agenten im Geländeplan reduziert werden. Hierbei spielt jedoch nicht nur die reine Position als Ortsvektor eine Rolle sondern auch eine eventuelle Aufteilung des Gesamtgebietes in einzelne Häuser, Stockwerke und Räume, da Wände und Decken den Sichtbereich zusätzlich einschränken.
Eine weitere Schwierigkeit bei der Partitionierung ist, dass Agenten ihre Position und entsprechende Sektoren bzw. Räume höchstwahrscheinlich mehrfach während einer Simulation wechseln und damit gerechnet werden muss, dass die Agenten sich im Laufe der Simulation stark vermischen. Aus diesem Grund ist es erforderlich, dass zum einen Agenten die Fähigkeit besitzen, während der Simulation von einer Plattform zu einer anderen migrieren zu können, ohne dass die Handlungen des Agenten oder seine Kommunikationen und Wahrnehmungen beeinträchtigt werden. Zum anderen muss während der Simulation immer wieder neu entschieden werden, welcher Agent auf welchem Knoten am Besten ist.

Zusammenfassend ergibt sich folgende Anforderungsliste an einen Verteilungsalgorithmus:

- Die Anzahl der entstehenden Partitionen muss vorgebbar sein, da die Anzahl der teilnehmenden Plattformen ebenfalls statisch ist
- Die Größe der Partitionen muss an die Rechenleistung der einzelnen Knoten angepasst sein, sodass die Lastverteilung möglichst homogen ist
- Das Partitionierungsverfahren muss insoweit “träge” sein, dass Agenten an den Partitionsgrenzen nicht zu oft den Knoten wechseln müssen.
- Das Verfahren muss möglichst schnell sein und kaum Rechenleistung erfordern, da die periodische Berechnung sonst zu viel leistung in Anspurch nehmen würde
- Das Verfahren muss neben der geographischen Position des Agenten auch die Unterteilung des Gebietes in einzelne Sektoren berücksichtigen, da diese einen maßgeblichen Einfluss auf die Sichtbereiche haben.
- Der Algorithmus muss NICHT zwingend deterministisch sein, da die Partitionierung nur zur Leistungssteigerung des gesamtsystems dient und es keinen direkten Einfluss auf die Simulation hat, auf welcher konkreten Plattform ein Agent rechnet

Mit Hilfe dieser Liste können die meisten üblichen Partitionierungsverfahren wie K-Means [Clu], DBSCAN [Wys07] oder Hierarchische Cluster [WZ01] ausgeschlossen werden, da diese Verfahren bereits an den ersten beiden Anforderunge scheitern. Verfahren, die rein auf dem direkten Kommunikationsverhalten der Agenten aufbauen und mathematisch analysieren, ob sich eine Migration des Agenten auf eine andere Plattform rechnen würde (vgl. [SBK06a]), fallen ebenfalls aus, weil die Analyse zuvor eine aufwändige Datensammelarbeit erfordern würde.

Eher anbieten würde sich ein auf den einzelnen Sektoren basierender, hierarchischer Ansatz, welcher zuerst die Agenten eines Sektors bzw. Raumes oder einer Etage zusammenfasst und

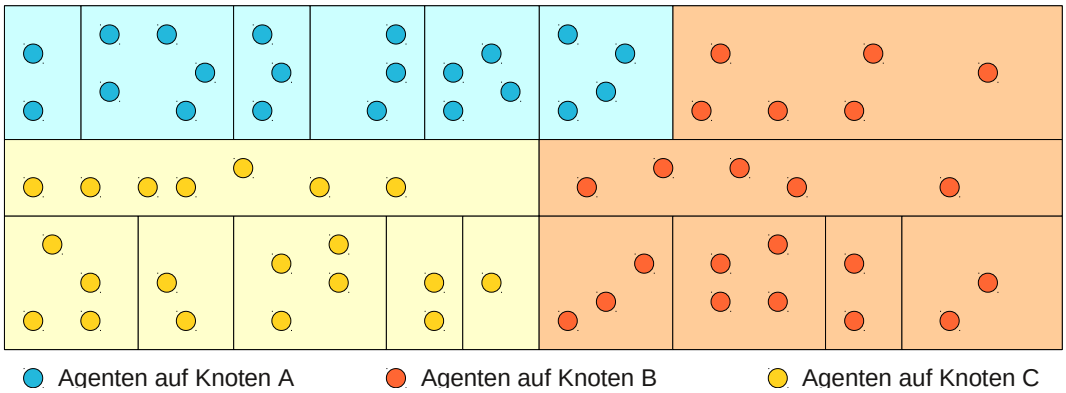

Abbildung 2: Beispielerteilung der Agenten auf 3 Knoten

dann mit Nachbarsektoren verbindet (siehe dazu auch Abb. 2). Wie genau ein solches Verfahren aussehen könnte, wird sich im Laufe des Projektes noch herausstellen.

3 Forschungsaspekte und Risiken

3.1 Forschungsziele

Einer der wichtigsten Forschungspunkte im Teilprojekt *Agentenplattform* des *WALK*-Projektes ist die Synchronisation der einzelnen Knoten miteinander und wie es erreicht werden kann, dass zwischen den Zeitpunkt einer Aktionsausführung und dessen Wahrnehmung durch andere Agenten möglichst wenig Zeit vergeht und so möglichst wenige Konflikte entstehen und wie entstandenen Konflikten umgegangen werden soll.

Ebenfalls ein Aspekt, den es zu untersuchen gilt, ist der in Abschnitt 2.1.5 erwähnte Zugriff auf die Simulation durch mobile Endgeräte und die daraus folgende Limitierung in Netzwerk- und Rechenleistung sowie Bedienkomfort. Dieser Punkt spielt vor allem eine Rolle für die Idee, das entstehende Tool in ferner Zukunft für eine Vor-Ort-Echtzeit-Analyse größerer Menschenmassen zu nutzen, damit die örtlichen Sicherheitskräfte computergestützt Pläne und Strategien dynamisch an die Gegebenheiten anpassen können.

Den ersten praktischen Punkt, den es im Laufe des Projektes zu untersuchen gilt, ist die Verifizierung der *Richtlinie für Mikroskopische Entfluchtungsanalysen* [Ini]. Diese Richtlinie spezifiziert diverse Begriffe, Simulationen und Tests, mit dessen Hilfe Evakuierungssimulationen auf ihre Leistung hin untersucht werden können.

Nicht zu vergessen sind natürlich die Forschungsaspekte außerhalb der Informatik, für welche das Projekt hauptsächlich gedacht ist. Ist das Projekt an sich ein Erfolg, können mit dem entstandenen Tool aufwändige Verhaltenssimulationen durchgeführt werden. Es kann nicht nur simuliert und analysiert werden, wie sich Menschen unterschiedlichen Charakters in großen Menschenmassen verhalten und wie Paniken entstehen, sondern auch, wie diese mit gezielten Veränderungen in der Umgebung bereits im Vorfeld verhindert werden. Auch, ob durch das Einschleusen ausgebildeter Sicherheitskräfte eine Massenpanik noch in der Entstehung abgemildert werden kann, könnte mit dem Simulationstool erforscht werden.

3.2 Risiken

Eines der größten Risiken des Projektes ist sicherlich die Frage nach der ausreichenden Performance des Gesamtsystems und ob die bisher offenen Probleme beim verteilten Gesamtzustand und bei der Verteilung der Agenten hinreichend gelöst werden können, um einen reibungslosen Simulationsablauf zu gewährleisten.

Ein weiterer Risikofaktor ist sicherlich auch, dass das Projekt nur ein Teilprojekt ist und sich mit einigen anderen Projekten in das Großprojekt WALK eingliedert und so abhängig ist von der Entwicklung und dem Erfolg dieser Projekte (z. B. den GIS-Projekt für die Beschaffung der Geländedaten).

4 Ausblick

Wie in den vorangegangenen Kapiteln deutlich geworden ist, haben wir innerhalb des WALK-Projektes und in den einzelnen Teilprojekten eine ganze Menge Arbeit vor uns. Da das WALk-Projekt zudem ganz am Anfang steht, ist noch nicht abschätzbar, welche Aufwände und Ideen im Laufe der Projektentwicklung noch hinzu kommen und welche weiteren Teilprojekte sich diesem angliedern.
Vor allem die Ausweitung des Anwendungsbereiches der Simulation und und die Anzahl der simulierten äußeren Einflüsse auf das Verhalten von Menschenmassen und die Genauigkeit dieser Einflüsse können es erfordern, dass das System durch weitere interne und externe Komponenten immer komplexer und aufwändiger wird, allerdings auch immer bessere Ergebnisse liefert.

Literatur

[Bal11] Mariusz Baldowski. Entwicklung eines 3D-Geoinformationssystem für Gefahrensituationen im In- und Outdoorbereich im Rahmen von WALK, 2011.

[BID+01] Robert Burke, Damian Isla, Marc Downie, Yuri Ivanov, and Bruce Blumberg. Creature smarts: The art and architecture of a virtual brain. In *IN PROCEEDINGS OF THE COMPUTER GAME DEVELOPERS CONFERENCE*, pages 147–166, 2001.

[bui] buildingSMART International. Industry Foundation Classes. CTAN http://www.iai-tech.org/products/ifc_specification.

[Clu] Team Clusteranalyse. K-Means. CTAN http://www-m9.ma.tum.de/material/felix-klein/clustering/.

[DFW97] Judith S. Dahmann, Richard M. Fujimoto, and Richard M. Weatherly. The department of defense high level architecture. In *Proceedings of the 29th conference on Winter simulation*, WSC '97, pages 142–149, Washington, DC, USA, 1997. IEEE Computer Society.

[DFW98] Judith Dahmann, Richard M. Fujimoto, and Richard M. Weatherly. The dod high level architecture: an update. In *Proceedings of the 30th conference on Winter simulation*, WSC '98, pages 797–804, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[Ini] Initiatoren des RiMEA-Projektes. Richtlinie für Mikroskopische Entfluchtungsanalysen v2.2.1. CTAN http://www.rimea.de/.

[Jef] Jeff Orkin. Goal Oriented Action Planning. CTAN http://web.media.mit.edu/~jorkin/goap.html.

[Kli09] Arne Klingenberg. Prototypische Entwicklung eines emotionalen Agenten auf der Basis des Goal Oriented Action Plannings, 2009.

[MN06] Charles M. Macal and Michael J. North. Tutorial on agent-based modeling and simulation part 2: how to model with agents. In *Proceedings of the 38th conference on Winter simulation*, WSC '06, pages 73–83. Winter Simulation Conference, 2006.

[MN07] Charles M. Macal and Michael J. North. Agent-based modeling and simulation: desktop abms. In *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*, WSC '07, pages 95–106, Piscataway, NJ, USA, 2007. IEEE Press.

[Nod01] Itsuki Noda. Framework of distributed simulation system for multi-agent environment. In *RoboCup 2000: Robot Soccer World Cup IV*, pages 229–238, London, UK, 2001. Springer-Verlag.

[OOvSB03] Elth Ogston, Benno Overeinder, Maarten van Steen, and Frances Brazier. A method for decentralized clustering in large multi-agent systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS '03, pages 789–796, New York, NY, USA, 2003. ACM.

[Ope] Open Source BIM foundation. Open Source Building Information Modelserver. CTAN http://www.bimserver.org.

[PHM07] Patrick Peschlow, Tobias Honecker, and Peter Martini. A flexible dynamic partitioning algorithm for optimistic distributed simulation. In *Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation*, PADS '07, pages 219–228, Washington, DC, USA, 2007. IEEE Computer Society.

[Pro] Prof. G. Keith Still PhD BSc FIMA. International Crowd Disasters. CTAN http://www.gkstill.com/CrowdDisasters.html.

[SAH03] Anthony Steed and Roula Abou-Haidar. Partitioning crowded virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '03, pages 7–14, New York, NY, USA, 2003. ACM.

[SBK06a] Tino Schlegel, Peter Braun, and Ryszard Kowalczyk. Towards autonomous mobile agents with emergent migration behaviour. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, page 585–592, New York, NY, USA, 2006. ACM.

[SBK06b] Tino Schlegel, Peter Braun, and Ryszard Kowalczyk. Towards autonomous mobile agents with emergent migration behaviour. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, pages 585–592, New York, NY, USA, 2006. ACM.

[SL02] Susan M. Sanchez and Thomas W. Lucas. Exploring the world of agent-based simulations: simple models, complex analyses: exploring the world of agent-based simulations: simple models, complex analyses. In *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*, WSC '02, pages 116–126. Winter Simulation Conference, 2002.

[ST07] Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. *Graph. Models*, 69:246–274, September 2007.

[TCP06] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 1160–1168, New York, NY, USA, 2006. ACM.

[TGB09] Flora Ponjou Tasse, Kevin Glass, and Shaun Bangay. Simulating crowd phenomena in african markets. In *Proceedings of the 6th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, AFRIGRAPH '09, pages 47–52, New York, NY, USA, 2009. ACM.

[TM06] Toshihiro Takahashi and Hideyuki Mizuta. Efficient agent-based simulation framework for multi-node supercomputers. In *Proceedings of the 38th conference on Winter simulation*, WSC '06, pages 919–925. Winter Simulation Conference, 2006.

[Tro09] Klaus G. Troitzsch. Perspectives and challenges of agent-based simulation as a tool for economics and other social sciences. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 35–42,

Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[Wys07] Jan Wyszynski. Dichte-basierte Clusterverfahren (G)DBSCAN, 2007.

[WZ01] Michael Wiedenbeck and Cornelia Züll. Klassifikation mit Clusteranalyse: Grundlegende Techniken hierarchischer und K-means-Verfahren. *ZUMA How-to-Reihe, Nr 10*, 2001.