Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Wolfram Sokollek

Simulation einfacher Evakuierungsszenarien mit Hilfe intentionaler Agenten

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Wolfram Sokollek

Simulation einfacher Evakuierungsszenarien mit Hilfe intentionaler Agenten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Wolfgang Renz
Zweitgutachter : Prof. Dr. Friedrich Esser

Abgegeben am 6. Juli 2009

Wolfram Sokollek

Thema der Bachelorarbeit

Einfache Evakuierungsszenarien mit Hilfe intentionaler Agenten

Stichworte

Agenten, Agentenorientierte Softwareentwicklung, BDI-Agenten, Emotionstheorien, Evakuierung, Intentionen, OCC-Theorie, Pax, PSI-Theorie, Simulation, Verhaltensregulierung

Kurzzusammenfassung

Diese Arbeit befasst sich mit dem Entwurf und der Entwicklung einer Agentenbasierten Evakuierungssimulation. Der Schwerpunkt liegt auf der Verwendung einer Emotionstheorie, um das Verhalten und die Intentionen der Flüchtenden zu modellieren. Es wurde, mit Hilfe einer Agentenorientierten Entwicklungsmethode, ein Design erstellt, das die Verwendung verschiedener Emotionstheorien ermöglicht. Darauf basierend konnte ein finaler Prototyp realisiert werden, der ein emotionsgesteuertes Fluchtverhalten der Evakuanten generiert. Für die Umsetzung wurde eine Multiagentenplattform verwendet.

Wolfram Sokollek

Title of the paper

Simple Evacuation Scenarios with the aid of Intentional Agents

Keywords

Agents, Agent Oriented Software Engineering, BDI-Agents, Emotional Theories, Evacuation, Intention, OCC-Theorie, Pax, PSI-Theory, Simulation, Behaviour regulation

Abstract

This paper describes the design and development of an agent based evacuation simulation. The main focus is on the usage of an emotional theory to model the behaviour and the intentions of the evacuants. With the aid of an agent oriented engineering methodology, a design it has been developed that allows the usage of different emotional theories. Based on that, a final prototype has been realised that generates an emotional controlled flight behaviour for evacuants. A multi agent platform has been used for the implementation.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einführung

1.1. Problemstellung und Motivation

Die grundlegende Motivation dieser Arbeit ergibt sich aus dem Interesse an folgender Fragestellung: Welche Rolle spielen die Emotionen bei der Verhaltenssteuerung der Menschen und Lebewesen und kann diese Art der Handlungsregulierung ein Vorbild für das Lösen von Problemstellungen der Computerwissenschaften sein?

Anhand einer konkreten Umsetzung einer agentenbasierten Evakuierungssimulation soll diese Fragestellung aufgegriffen werden. Es soll also im Laufe dieser Arbeit ein emotionales Modell, das das Verhalten der simulierten Personen steuert, in Verbindung mit einer agentenbasierten Evakuierungssimulation entwickelt werden. Diese Arbeit stellt eine Machbarkeitsstudie dar. Zu klären ist erstens, ob ein ausgewähltes Multiagentensystem (MAS) und die Agentenorientierte Software Entwicklung (AOSE) gute Werkzeuge für die Erstellung einer Evakuierungssimulationen sind, und zweitens, ob eine psychologische Komponente damit effektiv entwickelt und integriert werden kann. Es wird also mittels einer Agentenorientierten Vorgehensweise ein Design für die Simulation entwickelt und anhand eines Prototypen die Umsetzbarkeit des aufgestellten Designs aufgezeigt. Anhand des folgenden Abschnitts wird die Funktionalität einer Evakuierungssimulation beschrieben.

1.2. Szenario der Evakuierungssimulation

Das System zeigt eine einfache Evakuierungsumgebung, z. B. ein Flugzeug, an. Die simulierten Pax [1] versuchen es durch die Ausgänge zu verlassen. Dabei stoßen sich die Pax gegenseitig an und kollidieren auch mit Hindernissen, wie z. B. Sitzen oder Gegenständen, die im Weg liegen. Vor Gefahren, wie z.B. Feuer, fliehen sie. Die Pax werden von unterscheidbaren Emotionen beeinflusst und handeln emotionsabhängig. Beispielsweise haben sie das Bedürfnis Schmerzen zu vermeiden, können ein panisches Verhalten zeigen oder suchen die Nähe zu affinen Personen. Wenn unerwartete Ereignisse eintreten, werden sie unsicher und zögerlich in ihrem Verhalten. Andererseits werden sie zielstrebig, wenn sie z. B. einen

[1] Pax: (Begriff für Passagier(e) im Fluggewerbe)

Hinweispfeil in Richtung des Ausgangs wahrnehmen. Es kann verschiedene Emotionstypen geben, wie besonders ängstliche oder mutige Pax. Nachdem die Simulation durch den Benutzer gestartet wurde, läuft die Simulation ab. Eine grafische Darstellung des Geschehens informiert den Benutzer über den Simulationsverlauf. Die Simulation endet, wenn alle Pax das Flugzeug verlassen haben.

Die Bedingungen für die Simulation können z. B. durch die Vorgaben der EASA [2] zur Evakuierung von Flugzeugen festgelegt werden (EASA, 2003). Die relevanten Vorgaben sind, dass nur die Hälfte der verfügbaren Notausgänge funktionieren und die Simulation in Dunkelheit durchgeführt wird. Eine gewisse Verteilung der Geschlechter, des Alters und des Gesundheitszustandes der Menschen im Flugzeug soll vorliegen. Gepäckstücke sollen im Flugzeug verteilt sein, die die Wege zu Notausgängen behindern. Alle Personen sind am Anfang auf ihren Sitzen angeschnallt.

Weitere Evakuierungsszenarien sind denkbar, die auch verschiedene Gefahren wie Feuer oder Rauch beinhalten. Diese bei den Evakuierungstests der EASA nicht beachteten Aspekte einer Evakuierungssituation wären mögliche Situationen, die simuliert werden können. Eine Erkenntnis aus Statistiken über reale Evakuierungen (Galea u. a., 2006) ist, dass Pax selten alleine reisen und somit verschiedenen Beziehungen zu anderen Personen wie Familienmitgliedern oder Geschäftspartnern haben. Sie versuchen beieinander zu bleiben und sich gegenseitig zu helfen. Dieses affine Verhalten ist ein weiterer Aspekt, der ins Szenario aufgenommen werden könnte.

1.3. Abgrenzung

Die Entwicklung einer professionellen Evakuierungssimulation dauert mehrere Jahre und benötigt ein ganzes Team von Entwicklern. Es müssen neben der Analyse und Implementierung u. a. eine große Menge an empirischen Daten erhoben und ausgewertet werden, um eine hohe Genauigkeit der Simulationssoftware zu erreichen (Galea u. a., 2006). Aufgrund der zeitlichen Begrenzung dieser Arbeit wird es also nicht möglich sein, eine realistische Evakuierungssimulation zu erstellen. Vielmehr handelt es sich, wie bereits erwähnt, um eine konzeptionelle Studie. Es wird also versucht, die in der Arbeit aufgestellten Konzepte anhand einer (fortgeschrittenen) prototypischen Implementation zu überprüfen.

[2] EASA: (Europäische Agentur für Flugsicherheit)

1.4. Aufbau der Arbeit

Nach der Einführung in fachliche und technische Grundlagen Kapitel 2 (S.13) folgt eine ausführliche Anforderungserhebung Kapitel 3 (S.37) und eine Auswahl der technischen Mittel und fachlichen Ziele für die Umsetzung der Arbeit Kapitel 4. Im Kapitel 5 (S.58) wird das Design des Gesamtsystems entwickelt und ausgewählte Teile werden im Detail betrachtet. Auf dem Design basierend, wird im Kapitel 6 (S.74) zur Realisierung die Umsetzung der Simulation mit verschiedenen Diagrammen erläutert. Im Kapitel 7 (S.82) Test und Bewertung folgt, nach einer Beschreibung des Testvorgehens während der Entwicklung des Protoypen, ein Vergleich zwischen den in der Analyse aufgestellten Anforderungen und deren Umsetzung in Bezug auf das Design und auf die Realisierung durch den finalen Prototypen. Außerdem werden aufgetretene Probleme während des Entwicklungsprozesses analysiert. Die Arbeit endet mit einer Zusammenfassung sowie einem Ausblick auf zukünftige Entwicklungen.

2. Grundlagen

In diesem Kapitel werden die für das Verständnis und die Realisierung dieser Arbeit notwendigen Grundlagen vorgestellt.

Dafür gehen wir als erstes auf die technischen Grundlagen, wie Multiagentensysteme, Agentenorientierte Vorgehensmodelle und Computersimulationen, ein und werfen einen Blick auf den Stand der Technik. Folgend werden die fachlichen Grundlagen dieses Themas wie Evakuierungsszenarien und emotionale Modelle vorgestellt, aus denen im Anschluss auch die abgeleiteten Begrifflichkeiten dieser Arbeit festgelegt werden. Ein Kreuzvergleich zwischen den einzelnen Hauptaspekten unter Berücksichtigung des „State of the Art“ beendet dieses Kapitel.

2.1. Multiagentensysteme

Multiagentensysteme (MAS) sind Systeme, die aus mehreren homogenen oder heterogenen, autonom agierenden und kommunizierenden, kleineren Systemen, bekannt als Agenten, bestehen. Aus Wooldridge (2002). MAS sind Forschungsgegenstand der verteilten KI (Künstlichen Intelligenz). Sie beschäftigt sich damit komplexe Probleme durch Koordination und Kooperation von mehreren Agenten zu lösen. MAS können z.B. dort eingesetzt werden, wo durch einen hohen Grad an Komplexität und Unvorhersehbarkeit der Umwelt kein logisches Schließen mehr möglich ist. Hierbei kann die Lösung des Problems dadurch erreicht werden, dass viele einzelne Agenten mit unterschiedlichen Aufgaben zusammenarbeiten, kommunizieren und sich selbst organisieren. Eine übergeordnete, steuernde Instanz muss dann nicht mehr nötig sein (Ferber, 1999). Eine Herausforderung dabei ist, aus dem mikroskopischen Verhalten der Agenten den gewünschten makroskopischen Effekt des Gesamtsystems zu erzeugen (Klügl, 2006).

2.1.1. Agenten

Es gibt viele verschiedene Arten von Agenten. Wenn hier der Begriff Agent benutzt wird, ist damit der Intelligente Software Agent gemeint.

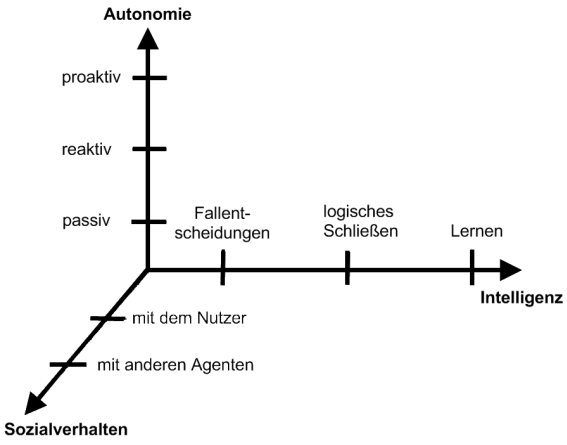

Abbildung 2.1.: Dimensionen der Eigenschaften Intelligenz, Autonomie, Sozialverhalten (Renz und Sudeikat, 2007)

Eine eindeutige Definition für Agenten ist in der Literatur nicht zu finden. Es gibt verschiedene Ansätze. Eine weit verbreiteter und weitgehend akzeptierter, „weicher“ Definitionsversuch ist der von Wooldridge und Jennings (1995). Ein intelligenter Agent ist ein System, das folgende Eigenschaften besitzt:

- *autonomy*: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
- *reactivity*: Intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.
- *proactiveness*: Intelligent agents are able to exhibit goal-directed behaviour by taking the initiative in order to satisfy their design objectives.
- *sozial ability*: Intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

Frei übersetzt bedeutet diese Definition, das intelligente Agenten die Fähigkeit zum selbständigen Handeln (autonomy) besitzen und auf Veränderungen der Umwelt (die Domäne in der sich der Agent befindet) zeitnah reagieren (reactivity). Sie sind in der Lage aus Eigeninitiative ihre Ziele zu verfolgen (proactivness) und interagieren zum Erreichen ihrer Ziele mit anderen Agenten oder auch Menschen (social ability).

Welche Ausprägung die Eigenschaften der Intelligenz, Autonomie und des Sozialverhaltens von Softwareagenten haben können, zeigt die Abbildung 2.1.

Es gibt viele Architekturen die das Agentenparadigma umsetzen. In der technischen Analyse wurde die Architektur auf die Agentenplattform Jadex festgelegt (siehe 4.1). Jadex implementiert die sogenannte BDI-Architektur, die im folgendem Abschnitt erklärt wird.

2.1.2. BDI - Agenten

Eine - für diese Arbeit höchst interessante - Architektur für Agenten ist die Belief Desire Intention (BDI)-Architektur. Sie hat ihre Ursprünge in der Philosophie des Practical Reasoning [1]. Der Ansatz geht davon aus, dass Menschen ihre rationalen Entscheidungen treffen. Diese Entscheidungen basieren auf den Wünschen oder Zielen (Desires), die sie verfolgen, und ihrem momentanen Wissen (Beliefs) über die Umwelt. Das Dokument Bratman u. a. (1988)[2] bietet eine genauere Einführung in diese Philosophie, soll an dieser Stelle aber nicht weiter thematisiert werden. Vielmehr interessiert hier, wie diese Philosophie für Computersysteme umgesetzt wird.

Die Architektur sieht folgende Bestandteile vor:

- Beliefs - Repräsentieren das aktuelle Wissen des Agenten.
- Desires - Die Ziele (Goals), die der Agent verfolgt.
- Intentions - Das momentan aktive Ziel, das die aktuelle Handlungssteurerung übernimmt.

Der Agent bestimmt mit einem Deliberation #2[3]-Prozess, welches Ziel(Desires/Goals) er verfolgt. Hat der Agent die Entscheidung für ein Ziel getroffen, wird es zur aktuellen Intention. Wie die Intention verwirklicht wird, basiert dann wieder auf dem Prozess der Means-End-Analyse [4]. Hier wird je nach Umweltsituation und vorhandenem Wissen ausgewählt, welche Aktion(Plan) als nächstes ausgeführt wird. Die Herausforderung besteht hierbei darin, die richtige Balance zwischen Deliberationzeit und Aktionszeit zu finden. Wird zu oft Deliberation betrieben, kommt der Agent nicht zur Aktion und umgekehrt.

[1] Practical Reasoning: (praktische/rationale Entscheidungsfindung)
[2] Prämiert mit dem 2008 International Foundation of Autonomous Agents and Multi-Agent Systems (IFAAMAS) influential paper award to recognize papers that have had a significant impact on the field of agents and multi-agent systems.
[3] Deliberation: Entscheidungsprozess welches Ziel in Abhängigkeit zur momentanen Situation verfolgt wird
[4] Means-End-Analyse: (Auswahl einer Aktion oder einer Folge von Aktionen die den „Abstand“ zwischen dem angestrebten Ziel und der momentanen Situation verringert)

2.1.3. Goaltypen der BDI-Architektur

Da im späteren Verlauf dieser Arbeit die Agentenplattform Jadex als Entwicklungswerkzeug festgelegt wird (Kapitel 4.1), werden als nächstes die wesentlichen Goaltypen von Jadex beschrieben (vgl. Braubach u. a. (2005, Kapitel 3.2)). Die hier gezeigten Goaltypen finden aber auch in weiteren Umsetzungen der BDI-Architektur Verwendung.

Perform Goal Das Perform Goal bezieht sich direkt auf eine Aktion(Plan), die ausgeführt werden soll;z. B.: Tippe einen Satz. Das Goal ist erfolgreich, wenn die Aktion ausgeführt wurde und schlägt fehl, wenn die Aktion nicht ausgeführt werden kann (keine Tastatur vorhanden) oder keine Aktion zur Ausführung zur Verfügung steht (kein Satz vorhanden).

Achieve Goal Das Achieve Goal bezeichnet einen bestimmten Zustand, der erreicht werden soll;z. B.: Erreiche 50 getippte Seiten (target condition) bis zum Fristende (failure condition). Es bleibt solange aktiv, bis der angestrebte Zustand (target condition) erreicht ist und schlägt fehl, wenn der Zustand als nicht erreichbar eingestuft wird (failure condition, Frist ist abgelaufen).

Query Goal Das Query Goal wird genutzt, um eine Information zu einem bestimmten Thema zu erlangen;z. B.: Wie wird das Query Goal beschrieben? Es ist erfolgreich, wenn die Information zur Verfügung steht, dafür kann eine Aktion ausgeführt werden (betreibe Recherche) oder direkt auf das aktuelle Wissen zurückgegriffen werden.

Maintain Goal Das Maintain Goal bezeichnet einen Zustand, der Erhalten (maintain condition) werden soll;z. B.: Hohe Koffein-Konzentration im Blut. Das Goal wird aktiviert, wenn der Zustand verletzt wird (niedrige Koffein Konzentration im Blut) und Aktionen zum Wiederherstellen des Zustand werden gestartet (Kaffee kochen und trinken). Der Unterschied zu den Achieve und Perform Goals ist, dass das Maintain Goal, auch wenn der Zustand temporär erreicht wurde, aktiv bleibt und kontinuierlich überprüft, ob der Zustand verletzt wurde. Das Maintain Goal kann dennoch als nicht erreichbar eingestuft werden (unmaintainable), wenn in der momentanen Situation keine Aktion zur Verfügung steht (kein Kaffee im Haus), die ausgeführt werden kann.

Ein schönes Beispiel für die Funktionsweise der BDI-Architektur zeigt die Abbildung 2.2

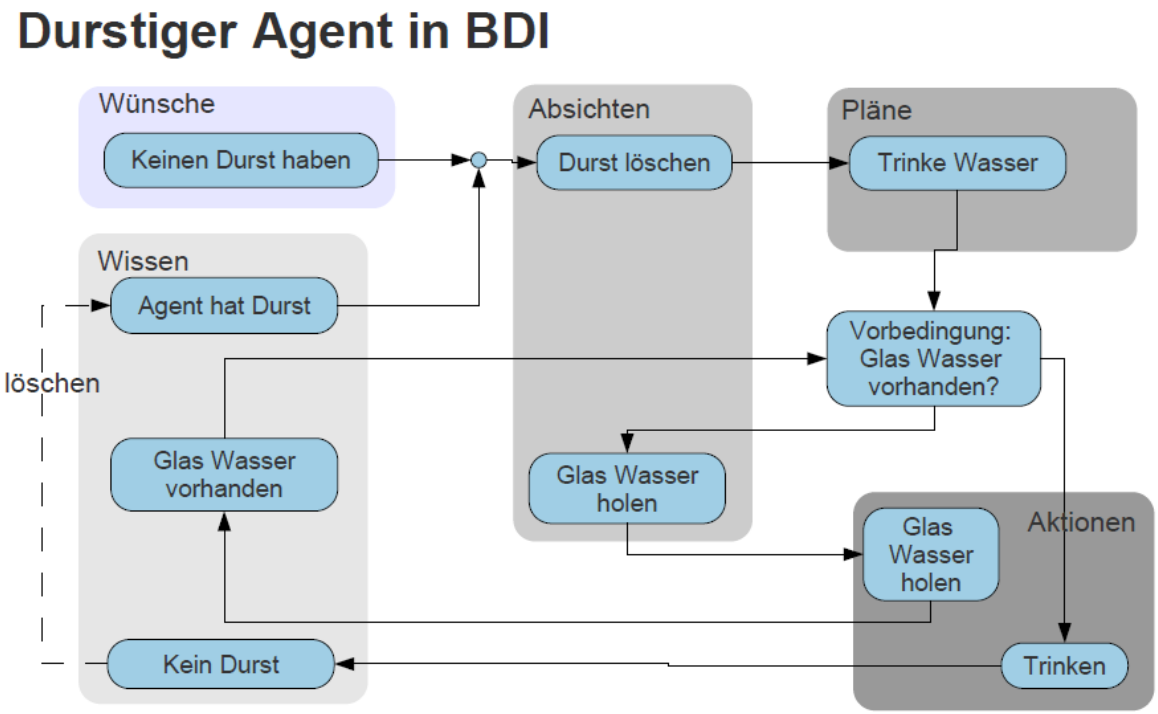

Abbildung 2.2.: Ein Durstiger BDI-Agent (Zapf, 2007)

2.1.4. Agentenorientierte Softwareentwicklung

Für eine neues Programmierparadigma wie die Agentenorientierte Software werden neue Vorgehensweisen benötigt, um erfolgreich komplexe Agentensysteme zu programmieren. Die etablierten Entwicklungsmethoden und Werkzeuge der Objektorientierten Softwareentwicklung reichen an dieser Stelle nicht aus, um alle Aspekte der Agenten zu gestalten. An dieser Stelle wird der Entwicklungsprozess an der Prometheus Agenten Methodologie gezeigt.

2.1.5. Prometheus - Agenten Methodologie

Die Entscheidung für Prometheus ist durch die Festlegung auf die Agentenplattform Jadex im Kapitel Technische Analyse 4.1 begründet. Nach Sudeikat (2004, S.109) ist Prometheus die für Jadex am besten geeignete Methode, da die von Jadex angebotenen Konzepte zur Agentenprogrammierung von der Prometheus-Vorgehensweise am besten unterstützt werden.

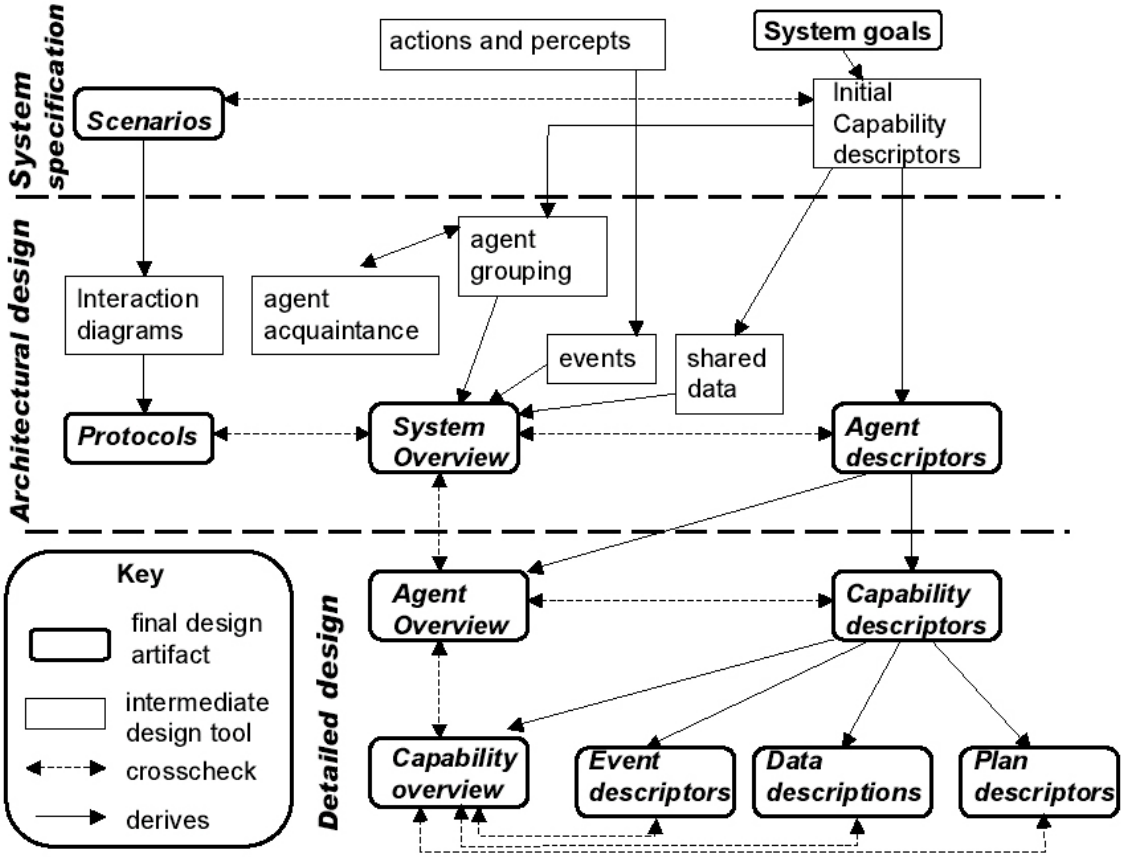

Abbildung 2.3.: Prometheus - drei Phasen (Padgham, 2002)

Die Prometheus Methodologie [5] wurde vom Royal Melbourne Institute of Technology entwickelt. Und ist im Internet zu finden unter http://www.cs.rmit.edu.au/agents/pdt/ Außerdem unterstützt das Prometheus Design Tool (PDT) den Entwicklungsprozess und bietet eine Lösung zum Erstellen von Prometheus Agenten Diagrammen.

Die Vorgehensweise von Prometheus gliedert sich in drei iterative und parallele Phasen.

System Specification Phase Im AnalyseOverview Diagram wird beschrieben, wie das System mit der Umwelt agiert. Hierfür werden die beteiligten Akteure, sowie deren „Actions“ und „Percepts“ identifiziert. Actions sind Aktionen, die die Umwelt beeinflussen, und Percepts sind Wahrnehmungen, die aus der Umwelt kommen. Percepts sind hierbei von Events zu unterscheiden. Events sind interne Nachrichten zwischen einzelnen Agenten und Percepts kommen von Außen in das System.

Parallel dazu werden Szenarios, Goals und Rollen (auch als Functions bezeichnet) analysiert. Goals, die Ziele des Systems, werden zu Rollen gruppiert, um somit eine erste Modularisierung des Systems zu erhalten. Für jedes Szenario kann eine Beschreibung der für den Ablauf des Szenarios nötigen Schritte (Steps) erfolgen. Die Steps bestehen aus Percepts, Actions, Events und Goals oder auch anderen Szenarios, um einen ersten Eindruck der Abfolge von Aktionen zu erhalten. Die Szenarios werden in sogenannte Scopes eingeteilt. Diese sind essential, conditional und optional.

Architectural Design Im Architectural Design werden viele weitere Diagramme erstellt. Die in der ersten Phase identifizierten Rollen werden den identifizierten Agenten zugewiesen (Agent-Role Grouping) und die Zugehörigkeit von Rollen und Daten wird erkannt (Data Coupling). Die Abhängigkeiten der Agenten untereinander wird im „Agent Acquaintance“ Diagramm bestimmt. Außerdem werden Kommunikationsprotokolle im Diagramm „System Overview“ zwischen den beteiligten Agenten angeordnet und mit AUML2 beschrieben.

Detailed Design Im Detailed Design wird jeder einzelne Agent detailliert beschrieben. Hier können Actions, Percepts, Events und Daten einzelnen Plänen zugeordnet werden und zu „Capabilities“ gruppiert werden.

Für ein Verständnis der im weiteren Entwicklungsprozess gezeigten Diagramme wird empfohlen das offizielle Manual zu studieren. Zu finden unter http://www.cs.rmit.edu.au/agents/pdt/docs/PDT-Manual.pdf. Zum tieferen Verständnis von Prometheus bieten sich noch weitere nicht zitierte aber informative Dokumente an, wie Padgham und Winikoff (2004), Henderson-Sellers und Giorgini (2005) und die Dokumentation und

[5] Methodologie: (Methodologie: (Duden2009, 2009) Lehre, Theorie der wissenschaftlichen Methoden)

die Tutorials auf der Homepage des PDT http://www.cs.rmit.edu.au/agents/pdt/.

2.1.6. Multiagentensysteme - State of the Art

> In short, autonomous agents is a field of high vitality, a melting pot of the most advanced ideas from artificial intelligence, economics, game theory, robotics, simulation, linguistics, biology, and many other fields.

Aus Azaiez u. a. (2007, Vorwort von Ladislau Bölöni)

Anwendung von MAS Agentenorientierte Software ist weit verbreitet und etabliert sich mehr und mehr in Wissenschaften und Industrie. Eine der populärsten Anwendungen für Agenten sind z.B. Massenszenen in Filmen, wobei jede einzelne Person als Agent agiert [6]. Auch in der Raumfahrt stellen autonome Agenten eine gute Lösung für z.B. Raumsonden und Roboter dar [7]. MASe sind, wegen ihrer Fähigkeit zur Selbstorganisation, eine gute Wahl, wenn es um Planungstätigkeiten z.B. für die Paketverteilung der Post oder Containermanagment an den Häfen geht. MAS werden auch im Bereich der Simulationen von soziologischen, biologischen und ökologische Systemen verwendet, um aufgestellte Theorien zu überprüfen und neue Erkenntnisse zu gewinnen. Speziell die Bereiche der Evakuierungs-, sowie Crowd-Behavior Simulationen[8] sind ebenfalls Anwendungsgebiet von MAS.

Agentenstandards und Plattformen Die Foundation for Intelligent Physical Agents (FIPA) legt die Standards für MAS fest und sorgt somit für einheitliche Schnittstellen und Funktionalität der Agenten, z.B. für die Agenten Kommunikation (FIPA-Protokolle [9]).

Der FIPA-Standard für Agentenplattformen (Abbildung 2.4) legt die Architektur für MAS fest. Im Wesentlichen legt die Spezifikationen der Plattform fest, dass Erstellung, Lokalisierung, Kommunikation und Zerstörung von Agenten möglich ist. Viele Agenten Plattformen existieren, die die FIPA-Standards umsetzen. Die Plattformen erleichtern die Programmierung von Agenten und bieten eine Laufzeitumgebung für die Agenten an. Auch die Visualisierung der Agenten zur Laufzeit und die damit einhergehende Unterstützung des Debuggings ist oft ein Bestandteil der Plattformen. Es gibt sowohl Open-Source als

[6] umgesetzt zum Beispiel von der Firma ©Massive http://www.massivesoftware.com/film-gallery/

[7] Agenten Technologie wurde z.B. bei der Deep Space 1 Mission verwendet http://ti.arc.nasa.gov/project/remote-agent/

[8] In Crowd-Behavior Simulationen wird z. B. Gruppenverhalten von Menschen erforscht

[9] http://www.fipa.org/repository/aclspecs.html

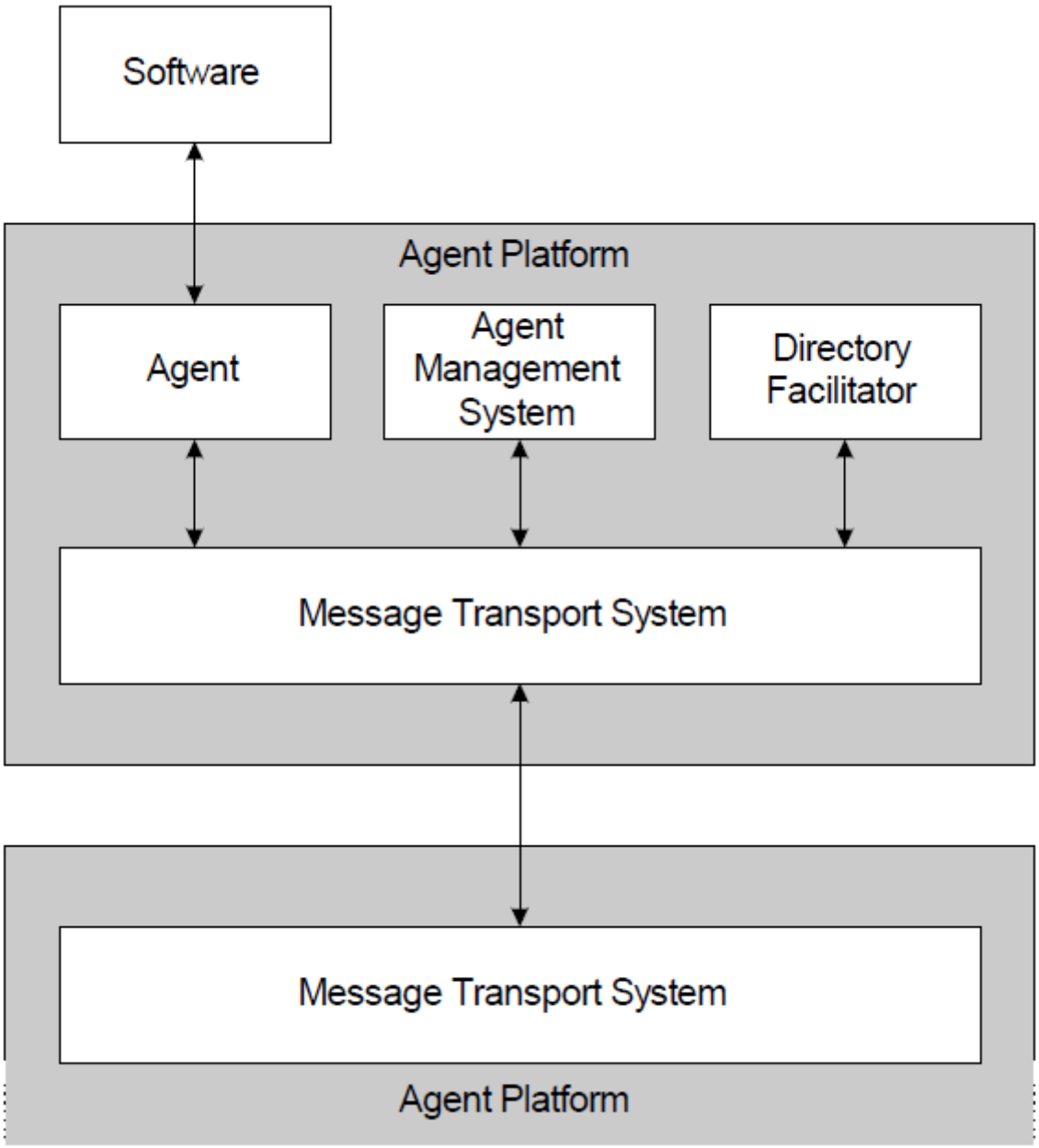

Abbildung 2.4.: Agent Management Reference Model - Agentenplatform als Middleware (Fipa)

auch kommerzielle Lösungen. Jade[10] und darauf aufbauend Jadex [11], das die Jade Plattform um BDI Agenten erweitert, sowie Jason[12] sind Open Source Produkte, um nur einige zu nennen. Ein Übersicht weiterer Plattformen mit kurzer Beschreibung findet sich auf http://www.fipa.org/resources/livesystems.html.

2.2. Evakuierungssimulationen

In diesem Abschnitt wird eine kurze Einführung in (Computer-)Simulationen gegeben, um darauf aufbauend über Evakuierungssimulationen zu informieren.

2.2.1. Simulationen

Stephen Hartmann definiert Simulation folgendermaßen:

> A simulation imitates one process by another process. In this definition, the term „process“ refers solely to some object or system whose state changes in time.

(Hartmann, 1996)

Das Verb „Imitieren“ (imitates) ist bei dieser Definition bewusst gewählt, denn eine Imitation ist, wenn man so will, niemals so gut wie das Original. Daraus folgt, dass eine Simulation eine abstrahierte Abbild des Simulationsgegenstand liefert, also eine Reduktion der realen Welt auf die erkannten simulationsrelevanten Eigenschaften und Vorgänge.

Folgende Abbildung 2.5 verdeutlicht die Beziehung zwischen realer Welt, und einer Simulation. Im Mittelpunkt steht die System-Theorie, die den zu behandelten Aspekt der realen Welt abstrahiert. Die Theorie ist die Basis für die Modellbildung des Simulationsgegenstand. Aus dem erstellten Modell wird die Simulation umgesetzt (bei Computersimulationen implementiert). Zum Schluss werden die gewonnenen Simulationsdaten mit den Daten der Experimente, die in der realen Welt ausgeführt werden, einer Validierung unterworfen und gegebenenfalls neue Hypothesen und Theorien gebildet oder Vorhersagen getroffen (Sargent, 2004).

[10] im WWW unter] http://jade.tilab.com
[11] im WWW unter http://jadex.informatik.uni-hamburg.de/
[12] im Internet http://jason.sourceforge.net/JasonWebSite/Jason%20Home.php

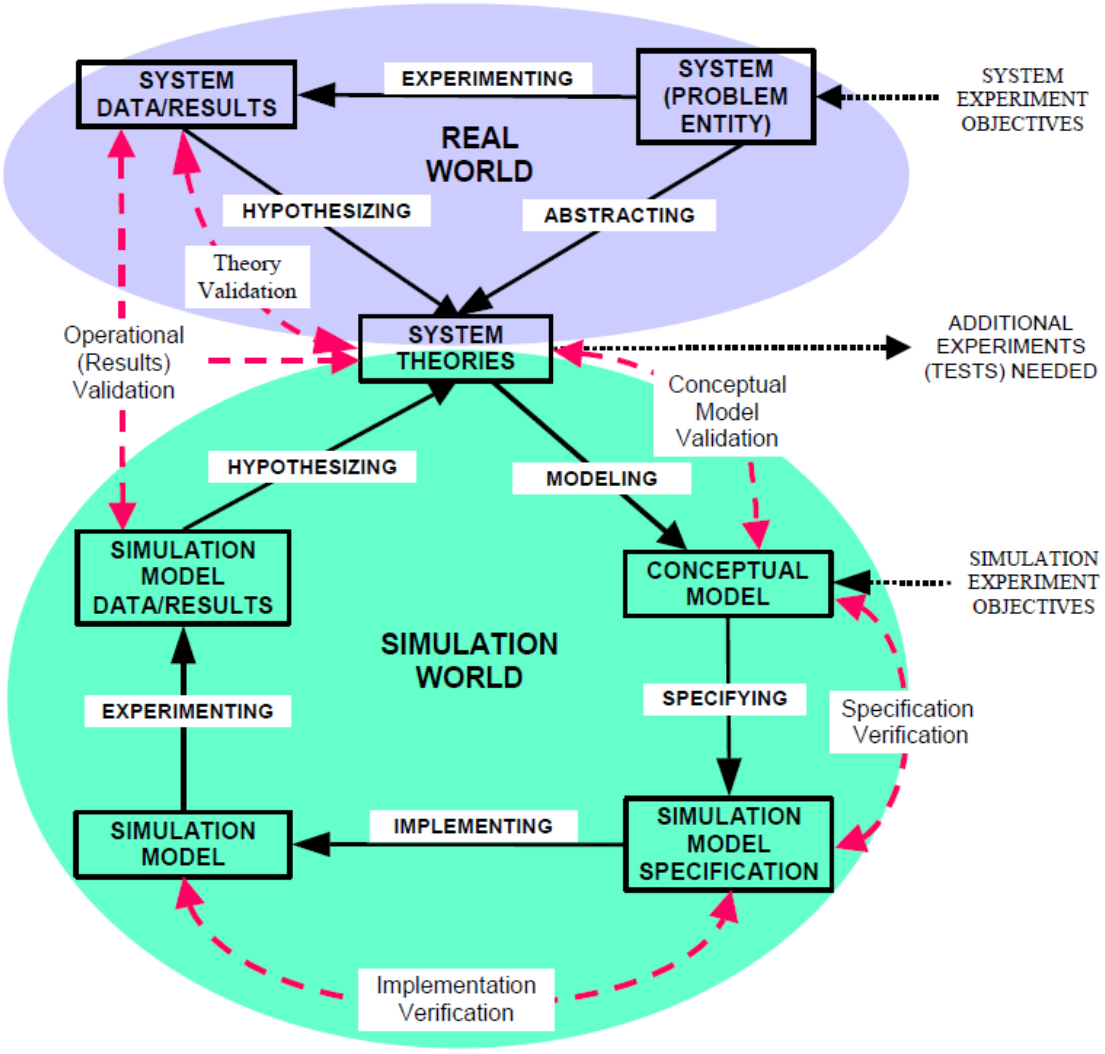

Abbildung 2.5.: Beziehung zwischen realer Welt und Simulation mit Verifikation und Validierung (Sargent, 2004)

Bestandteile einer Simulation Ein Simulationsmodell besteht normalerweise aus simulierten Einheiten und Übergangsregeln. Je nach Art der Simulations können diese Einheiten Variablen, Objekte, Zellen oder Agenten sein. Die Einheiten beschreiben in ihrer Gesamtheit das System. Die Übergangsregeln bestimmen, wie sich die Simulationseinheiten über die Zeit hinweg verändern (Herrler, 2007, S.41)´.

Zeitliche Darstellung der Simulation Ein wichtige Komponente einer Simulation ist der Fortschritt der Zeit und damit die Steuerung der Vorgänge in der Simulation. Es gibt grundsätzlich folgende Optionen:

Kontinuierlicher Zeitfortschritt Bei kontinuierlichen Zeitschritten können Vorgänge eine unendlich kleine Dauer aufweisen. Das heißt, dass sich Zustandsvariablen der Simulation in beliebig kleinen Zeitschritten ändern können. Das mathematische Mittel dieser kontinuierlichen Modelle sind (nichtlineare) Differentialgleichungssysteme, die bei Computersimulationen durch numerische Integration „gelöst“ werden (Herrler, 2007, S.42).

Diskreter Zeitfortschritt Bei diskreten Zeitschritten werden die Variablen der Simulation in diskreten Schritten verändert. Einmal wäre da der konstante Zeitfortschritt bei taktbasierten Simulationen. Dieser steht meistens in einem festen Verhältnis zur Realzeit und bietet somit Vorteile für die direkte Beobachtung der Simulation. Wenn allerdings die Berechnung von bestimmten Vorgängen länger als der Takt dauert, kann unvorhergesehenes Verhalten auftreten oder, im gegenteiligen Fall, Leerlauf entstehen. Anders ist das bei eventbasierter Simulation. Die Steuerung der Simulation geschieht hierbei über Ereignisse(Events). Die auftretenden Ereignisse werden in einer sortierten Liste abgelegt und der Reihe nach abgearbeitet. Je nach Anzahl der Ereignisse dauert die Berechnung der Vorgänge in der Simulation daher länger oder kürzer. Die Zeit streckt oder verkürzt sich für den Beobachter (Herrler, 2007, S.42).

Weitere Eigenschaften von Simulationen Die weiteren allgemeinen Eigenschaften von Simulationen wie mikroskopisch oder makroskopisch, Auflösung und Maßstab werden am Beispiel von Evakuierungssimulationen im folgenden Abschnitt 2.2.2 behandelt.

2.2.2. Evakuierungssimulationen

Evakuierungssimulation/Evacuation Simulations (EvacSims) spielen bei der Planung von Transportmitteln und Gebäuden, aber auch bei der Auswertung von Katastrophen eine große

Abbildung 2.6.: Exodus Evakuierung eines Gebäudes (http://fseg.gre.ac.uk/)

Rolle. Ein großes Personenflugzeug, das für viel Geld entwickelt wird und dann dem Evakuierungstest der EASA[13] nicht gerecht wird, erhält keine Zulassung. Somit besteht ein großes Interesse der Flugzeugindustrie verschiedene Kabinenkonfigurationen am Computer zu testen, bevor die Live-Tests[14] der EASA durchgeführt werden. Ähnliches gilt für die Gebäudekonstruktion.

Auch bei der Aufbereitung von Katastrophen sind EvacSims nützliche Hilfsmittel. Die Evakuierungen der World Trade Center am 11. September 2001 wurden ebenfalls am Computer simuliert. So schlimm diese Katastrophen auch sind, so sind die Erfahrungen der beteiligten Personen eine wichtige Quelle zur Validierung [15] der EvacSim. Eine wissenschaftliche Sammlung und Auswertung dieser Erfahrungen zum 11.September sind in Johnson (2005) und Galea u. a. (2008) zu finden. Die weltweit führende EvacSim ist EXODUS. Entwickelt unter der Leitung von E.R. Galea in der Fire Safety Engineering Group der University of Greenwich[16].

Eigenschaften von Evakuierungssimulationen Das Feld der EvacSim umfasst ein weites Gebiet. Mehrere Eigenschaften der Simulationen lassen sich unterscheiden:

[13] Die wesentlichen Bedingungen sind die Evakuierung des Flugzeuges innerhalb von 90 Sekunden in völliger Dunkelheit mit der Hälfte der verfügbaren Ausgänge (EASA, 2003)

[14] Der Evakuierungstest des A380 ist auf http://www.dumpalink.com/videos/873_people_evacuated_from_a380_in_77_seconds-cbhj.html zu sehen

[15] Validierung: (Das Validieren ist eine Prüftätigkeit, bei der Produkte, Protokolle oder Dokumente in Bezug auf ihre Spezifikationen geprüft werden.)

[16] zu finden unter urlhttp://fseg.gre.ac.uk/exodus/

- **Maßstab (Scale):** Der Maßstab bezeichnet die räumliche und zeitliche Ausdehnung der Simulation. Bei EvacSim geht der Maßstab von Evakuierungen ganzer Landstriche und Städte, die mehrere Tage in Anspruch nehmen kann[17], über die Evakuierung von Großbauten wie Sportstadien und Hochhäuser, was viele Minuten bis Stunden dauert, bis hin zur EvacSim von Transportmitteln wie Schiffen, Flugzeugen oder Bussen, für die eine Dauer von Minuten bis Sekunden angenommen werden kann.
- **Auflösung (Resolution):** Die Auflösung hingegen bezeichnet den Detaillierungsgrad des simulierten Raumes. Hier gibt es Auflösungen in denen der Raum in Felder von der Größe mehrerer Meter aufgeteilt wird (Grid), bis hin zur kontinuierlichen Darstellung[18]
- **Genauigkeit (Fidelity):** Die Genauigkeit der Simulation beruht insbesondere auf der Anzahl der Parameter, die den Simulationsverlauf beeinflussen. Das könnten z. B. Alter, Gewicht, Größe, Mobilität für Personen sein.
- **Mikroskopisch vs. Makroskopisch:** Des Weiteren gibt es mikroskopische oder makroskopische Simulationen. Während die mikroskopische EvacSim z. B. das Verhalten der Menschen aus Sicht jedes Einzelnen mit einer Vielzahl von Parametern steuert, wird bei der makroskopischen Simulation die Bewegung der Menschen mit Hilfe einer funktionalen Analogie berechnet, die das Verhalten der Menschen beispielsweise als Fluss von Partikeln berechnet.

Vergleiche Klüpfel (2003, S.7,S.10,S.11)

2.3. Emotionale Systeme

Das rationale Denken und die Emotionen der Menschen sind gerade in der Geschichte der Wissenschaften meist als Widerspruch betrachtet worden (Schneider, 2005, S.9). Mittlerweile ändert sich dieses Bild der Verknüpfung von Ratio und Emotion aber und Erkenntnisse verbreiten sich, dass Emotionen eine Voraussetzung der Ratio sind. Menschen, bei denen ein für sekundäre[19] Emotionen zuständiger Teil des Gehirns verletzt wurde, können durch logisches Folgern keine Entscheidungen mehr treffen. Daraus entstand die Schlussfolgerung, dass Emotionen einen entscheidenden Beitrag für unser Verhalten liefern. Man könnte somit

[17] Gemeint ist hier nicht die Berechnungsdauer der Simulation, sondern wie lange solch ein Vorgang in der realen Welt dauert.

[18] Hier gibt die endliche Fließkommaarithmetik des verwendeten Computersystems natürlich auch eine Grenze vor.

[19] Nach Damasio (2001) unterscheidet man zwischen primären und sekundären Emotionen, wobei die primären Emotionen die unwillkürlichen, angeborenen, körperlichen Reaktionen sind und die sekundären Emotionen aus Denkprozessen entstehen und direkt mit erworbenen Erfahrungen verknüpft sind.

Emotionen als eine durch die Evolution geprägte Heuristik für unsere Handlungsregulation betrachten.

Die Erforschung der zugrunde liegenden Prozesse beschäftigt die Neurowissenschaften. Die Prozesse werden auch mit Hilfe der Computerwissenschaften erforscht und die Erkenntnisse in der künstlichen Intelligenz genutzt.

Es gibt viele Theorien die zum Ziel haben die Systematik von Emotionen zu klären. Unterschiedliche und gegensätzliche Vertreter dieser Theorien sind die PSI Theorie von Dörner sowie die Emotionstheorie von Ortony, Clore und Collins. Auf diese Theorien wird im folgenden Abschnitt genauer eingegangen.

2.3.1. Emotionstheorie von Ortony, Clore und Collins (1988)

Ortony, Clore und Collins (OCC) haben bei der Entwicklung ihrer Emotionstheorie explizit angestrebt eine für den Computer implementierbare Theorie der Emotionen zu erschaffen. Nicht um eine emotionale Maschine zu schaffen, sondern um die Aspekte der emotionsbasierenden Entscheidungsfindung der künstlichen Intelligenz näher zu bringen (Rübenstrunk, 1998, Kapitel 4.1) nach Ortony u. a. (1988).

Appraisal-Theorien

Die OCC Emotionstheorie basiert auf und erweitert die Bewertungstheorien (appraisal theories). Diese Theorien gehen davon aus, dass der Mensch eine Situation kognitiv einschätzt und auf dieser Einschätzung basierend eine Emotion ausgelöst wird. Intensität und Qualität der Emotion ist direkt abhängig von der kognitiven Bewertung der Situation oder eines Objekts. Die Bewertungstheorien werden in Schneider (2005, S.19-S.22) beschrieben und bieten eine gelungene Zusammenfassung für tiefere Einblicke.

Emotionsgruppen

Im OCC Emotionsmodell werden nach Schneider (2005, S.24) für die Bewertung der emotionsauslösenden Situation drei grundsätzliche Emotionsgruppen erkannt. Die ereignisfundierten Emotionen, die handlungsfundierten Emotionen und die objektfundierten Emotionen. Ob eine Emotion eine positive oder negative Ausprägung erfährt, hängt des weiteren von der Erwünschtheit, der Lobwürdigkeit und der Attraktivität der Ursache ab.

Die Abbildung 2.7 (S.28) zeigt die Struktur von Emotionstypen bei der Emotionstheorie von Ortony, Clore und Collins

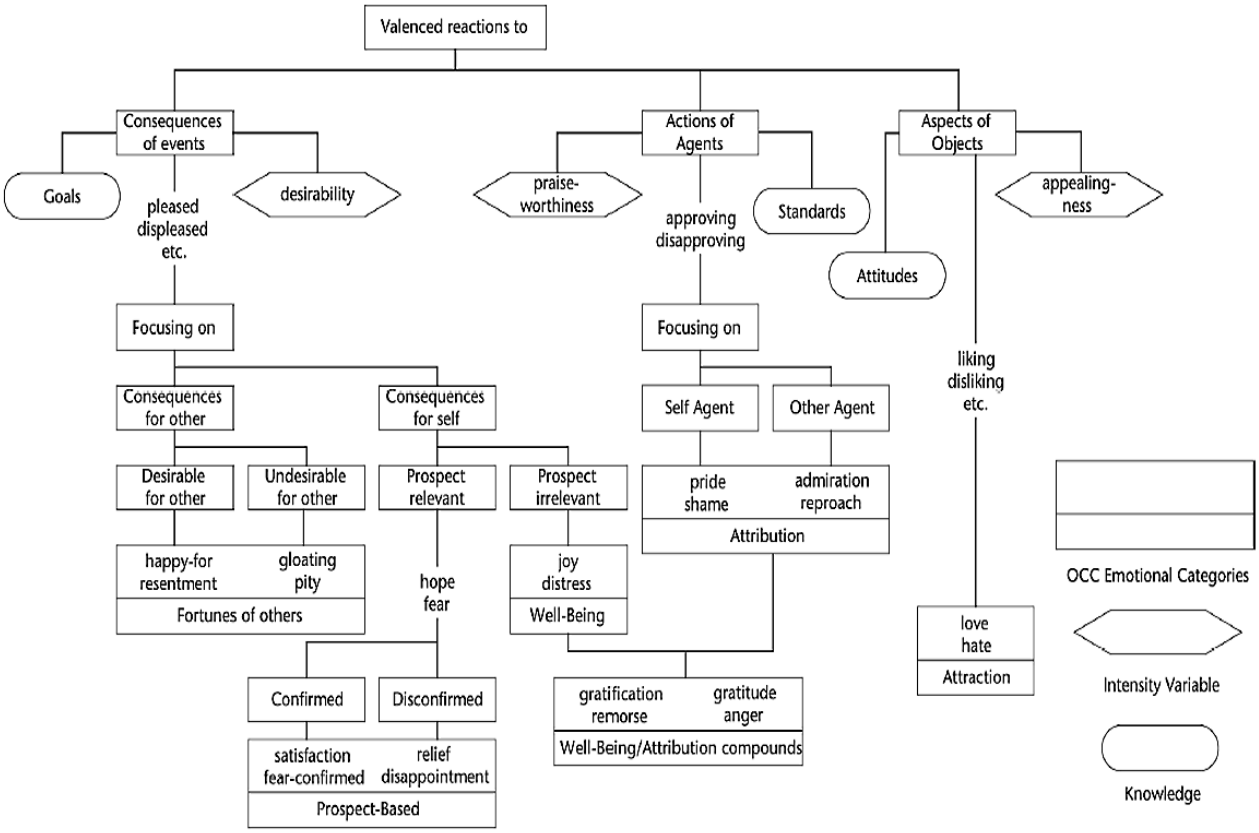

Abbildung 2.7.: Das OCC-Modell (Bartneck, 2002)

Der linke Zweig der Abbildung 2.7 zeigt die *Ereignisfundierten Emotionen.* Sie werden nach ihrer Erwünschtheit bzw. nach ihrer Unerwüschtheit bewertet. Das bedeutet, dass bei der Wahrnehmung eines Ereignisses [20] es danach beurteilt wird, ob es die eigenen Wünsche und Ziele unterstützt oder behindert.

Der Mittlere Zweig der Abbildung 2.7 zeigt die *Handlungsfundierten Emotionen.* Sie referenzieren die Aktionen (oder das Unterlassen) von Akteuren und bewerten diese nach der Lobwürdigkeit. Der Grad der Lobwürdigkeit hängt davon ab, in wie weit die Handlung einer (gesellschaftlichen) Norm entspricht.

Im rechten Zweig der Abbildung 2.7 geht es um *Objektfundierte Emotionen.* Sie beziehen sich auf Personen, Tiere oder Gegenstände. Die Bewertung erfolgt danach, ob die Person dem Objekt positive oder negative Eigenschaften zuspricht (attitude). Das Objekt wirkt anziehend, wenn es positive Eigenschaften hat, und abstoßend, wenn es negative Eigenschaften hat.

Aus der Abbildung 2.7 (S.28) sind außerdem sechs Emotionskategorien zu entnehmen, die beim Kognitionsprozess jeweils einem Ereignis, einer Aktion oder einem Objekt zugeordnet werden und eine positive oder negative Ausprägung als konkrete Emotion haben. Für Ereignisorientierte Emotionen sind nach Schneider (2005, S.26) Wohlergehensemotionen (Well-Being), die als konkrete Ausprägung Freude und Leid haben, Empathieemotionen (fortunes for other) mit den Ausprägungen Mitfreude und Mitleid, sowie erwartungsfundierte Emotionen (Prospect-Based) mit den konkreten Emotionen Hoffnung und Furcht.

2.3.2. PSI-Theorie von Dörner

> „Bei der PSI-Theorie handelt es sich um ein Modell der Integration von Perzeption, Emotion, Kognition, Motivation und Aktion für die menschliche Handlungsregulierung. Ganz im Sinne der Konzeption zu einer «Theoretischen Psychologie» werden hier keine psychologischen «Module» schwerpunktmäßig erforscht oder ausgelassen. Das Hauptaugenmerk liegt immer in dem Zusammenspiel der verschiedenen Komponenten und ihrer wechselseitigen Beeinflussung. [...] Die PSI-Theorie und ihre Implementation beschreibt und erklärt die menschliche Handlungsregulation durch Formalisierung der psychischen Prozesse in ihrer Gesamtheit“

Zitiert aus Detje (1999).

Im Gegensatz zur Emotionstheorie von Ortony, Clore und Collins modelliert Dörner in seiner PSI-Theorie Emotionen nicht als konkrete Artefakte eines Kognitionsprozesses, sondern

[20] Ereignisse sind in diesem Kontext Sachverhalte, für die kein Urheber existiert.

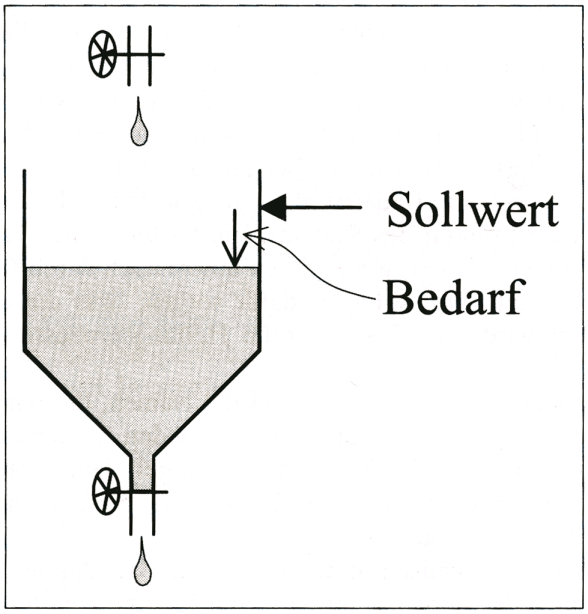

Abbildung 2.8.: Bedarf (Dörner, 2003, S.114)

wählt einen systemischen Ansatz. Autonomes und emotionales Verhalten resultiert hierbei aus dem komplexen Zusammenspiel von verschiedenen Prozessen, die vielfältig ineinander greifen und rückkoppeln.

Bedürfnisse in der PSI-Theorie

Dörner identifiziert als Kern seiner Handlungsregulation bestimmte Bedürfnisse. Diese differenzieren sich nach existentiellen und informellen Bedürfnissen. Die existentiellen sind Hunger, Durst und Schmerzvermeidung. Die informellen Bedürfnisse sind Affilitation (Maß der Gruppenzugehörigkeit), Kompetenz (Maß der Erfüllung der eigenen Bedürfnisse) und Bestimmtheit (Maß zur Vorhersagbarkeit der Umwelt und des eigenen Handelns). In bestimmten Unmweltsituationen kann es nun zu einer Reduzierung oder Zunahme der verschiedenen Bedürfnisse kommen. Beispielsweise kommt es zu einer Reduktion des Bedürfnisses Schmerzvermeidung, wenn eine Verletzung stattfindet. Eine spezielle Rolle haben hierbei die Kompetenz und die Bestimmtheit. Sie sind so etwas wie der allgemeine „Lagebericht“ über das Verhältnis des Systems zu seiner Umwelt. Diese „Berichte“ führen zu bestimmten Modulationen des Verhaltens und der inneren Steuerungsprozesse (Dörner, 2003). Das Verhalten wird über die Parameter Aktiviertheit, Selektionsschwelle und Auflösungsgrad gesteuert.

Ein Bedürfnis wird als eine Art Kessel abgebildet. Jeder Kessel ist bildhaft gesprochen mit

einer dem Bedürfnis entsprechenden Flüssigkeit gefüllt. Jeder Kessel hat außerdem einen Sollwert(siehe Abbildung 2.8 (S.30)). Unterschreitet der Pegel der Flüssigkeit den Sollwert, wird aus dem Bedürfnis ein Bedarf. Aus diesem Bedarf entsteht ein Motiv, das schlussendlich eine konsumatorische Endhandlung bewirkt und den Pegelstand des Bedürfnisses wieder anhebt. Vergleiche hierzu Schneider (2005, S.33)

Dringlichkeit Die Dringlichkeit des Bedürfnisses B bestimmt sich logarithmisch aus der Abweichung A zum Sollwert [21].

$$B = \log A$$

Handlungsregulation

Bei verschiedenen Bedürfnissen können mehrere Motive miteinander im Konflikt stehen. Um nun zu entscheiden welches Motiv die aktuelle Handlungssteuerung übernimmt, wird die Dringlichkeit für jeden Motivator berechnet. Das läuft folgendermaßen ab: Motive sind mit verschiedenen Zielen und/oder Plänen assoziiert, die im Endeffekt das zugrunde liegende Bedürfnis füllen. Die Motivstärke M ergibt sich aus der Bedürfnisstärke B (s. o.) multipliziert mit der Erfolgserwartung E, in der aktuellen Situation, einen der assoziierten Pläne oder Ziele umsetzen zu können.

$$M = B * E$$

In bestimmten Umweltsituationen könnte es zu einem ständigen Wechsel der Motive kommen. Um das zu verhindern wird eine Selektionsschwelle eingeführt. Ein aktives Motiv mindert die Stärke anderer Motive um einen bestimmten Betrag. Dieser Betrag ist umso geringer, je niedriger die Kompetenz ist. Bei niedriger Kompetenz wird eine häufigere Neuorientierung und ein häufigerer Wechsel der Motive die Folge sein, als bei hoher Kompetenz. Das daraus resultierende unterschiedliche Verhalten kann, bei niedriger Kompetenz, beispielsweise als Unsicherheit in einer unvorhersehbaren Umwelt interpretiert werden. Damit geht eine ständige Überwachung der Umgebung einher. Bei hoher Kompetenz ergibt sich dementsprechend das stetige Verfolgen der Ziele bei bekannter Umwelt.

[21] Berechnet wird die Stärke des Bedürfnisses, durch den Logarithmus der Summe der Bedarfssignale, die innerhalb eines bestimmten Zeitintervall eintreffen. Dies entspricht dem Weber-Fechner Gesetz http://de.wikipedia.org/wiki/Weber-Fechner-Gesetz aus der Psychophysik (Schneider, 2005, S.33).

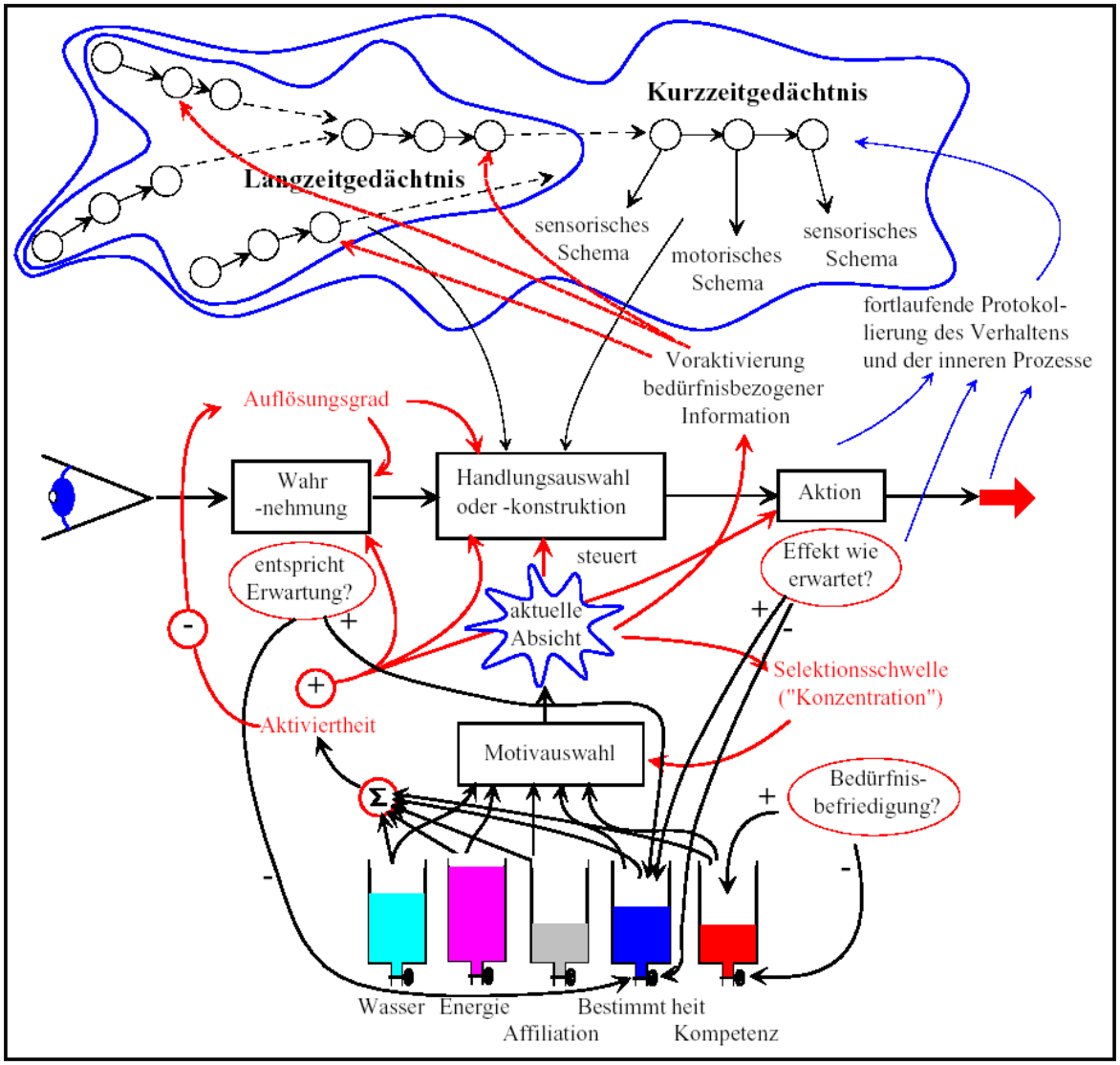

Abbildung 2.9.: Bedürfnisse (Dörner, 2003)

Das Kompetenzbedürfnis - Unsicher oder Selbstbewusst

Die Kompetenz wird nach einen ganz bestimmten Prinzip gefüllt und reduziert. Und zwar beeinflusst der Pegelstand des Kompetenzbedürfnisses die Zu- und Abnahme des Pegels. Ein hoher Grad an Kompetenz wird nur wenig von reduzierenden Ereignissen beeinflusst, aber er verstärkt die Zunahme der Kompetenz bei steigernden Ereignissen. Das ist vergleichbar mit dem Verhalten eines selbstbewussten Menschen, den so schnell nichts aus der Ruhe bringt. Bei niedriger Kompetenz verhält es sich genau anders herum. Ein Ereignis, dass für eine Kompetenzsteigerung sorgt, wird wenig beachtet, aber ein verringerndes Signal fällt umso stärker ins Gewicht. Hier kann der Vergleich zu einem unsicherem Menschen herangezogen werden. Dieser lässt sich auch durch Erfolge nur langsam wieder aufbauen. Betrachtet man die Abbildung 2.9, wird deutlich das eine Kompetenzsteigerung auch eine Bedürfnisbefriedigung ist und sich somit selbst verstärkt.

Emotionen in der PSI-Theorie

Emotionen werden bei Dörner nicht als eigenständige Objekte realisiert, sondern Emotionales Verhalten entsteht aus dem komplexen Zusammenspiel des Gesamtsystems. Eine wesentliche Rolle spielen hier die Bedürfnisse nach Bestimmtheit und nach Kompetenz. Sie stellen das Verhältnis des Systems zu seiner Umwelt dar und beeinflussen sein Verhalten. Der Pegel des Bestimmtheitskessels liefert Informationen über die Voraussagbarkeit der Umwelt und der Pegelstand des Kompetenzkessels informiert über die Bewältigbarkeit von Problemen (Dörner, 2003). Verschiedene Verhaltensparameter werden u. a. durch diese informellen Bedürfnisse beeinflusst. Diese Parameter sind die oben erwähnte Selektionsschwelle, die Aktiviertheit (Arousal) und der Auflösungsgrad.

Verhaltensweisen in der PSI-Theorie

Verschiedene Verhaltensweisen werden in der PSI-Theorie beschrieben, um die informellen Bedürfnisse Kompetenz und Bestimmtheit zu befriedigen. Die Abbildung 2.10 zeigt wie der Kompetenzpegel (Competence level) und Bestimmtheitspegel (Certainty level) Einfluss auf die Verhaltenstendenz haben.

Sicherungsverhalten (Safeguard activities) Bei niedriger Kompetenz und Bestimmtheit steigt die Tendenz zum Sicherungsverhalten. Sicherungsverhalten bedeutet, dass die Umgebung genau analysiert wird und das eigene Handeln bewertet wird. Dieses Verhalten könnte als Emotion Angst oder Unsicherheit gedeutet werden.

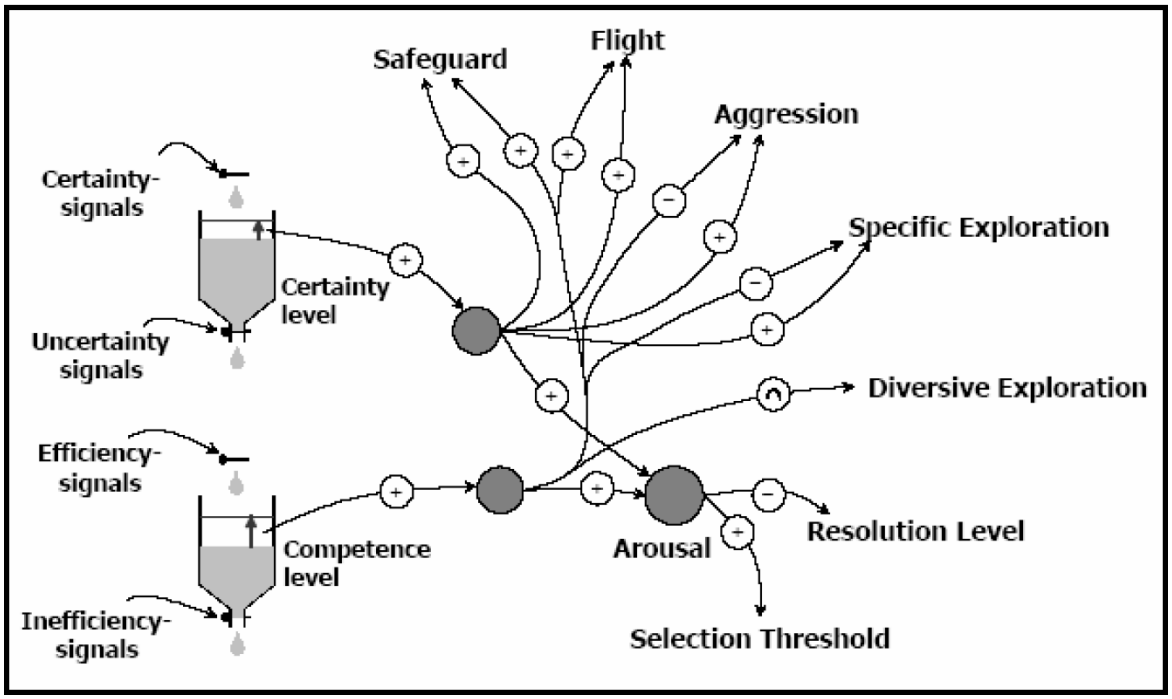

Abbildung 2.10.: Verhaltenstendenzen (Dörner u. a., 2003, S.76)

Tendenz zur Flucht (Flight) Bei niedriger Kompetenz und Bestimmtheit steigt die Tendenz zur Flucht.

Tendenz zur Aggression (Aggression) Bei niedriger Bestimmtheit, aber hoher Kompetenz (große Bewältigbarkeit der Probleme), steigt die Aggressionstendenz. Aggressives Verhalten, z. B. das Zerstören eines Gegenstandes, ist ein großes Bestimmtheitssignal. Denn die Bestimmtheit steigt, wenn Aktionen, die man ausführt, zum gewollten Ergebnis führen.

Tendenz zur spezifische Exploration (Specific Exploration) Spezifische Exploration bezeichnet ein Verhalten, das man mit Suche nach Selbstbestätigung vergleichen könnte. Es wird ausgelöst, wenn das Bedürfnis nach Bestimmtheitssignalen steigt. Vandalismus wäre eine Form dieser Selbstbestätigung oder auch das Sich-Zurückziehen in eine sichere Umgebung.

Tendenz zur diversiven Exploration (Diversive Exploration) Diversive Exploration ist ein Mittel, um seine Kompetenz zu steigern. Die Tendenz zu diesem Verhalten steigt, wenn die Kompetenz niedrig, aber nicht zu niedrig ist. Die diversive Exploration bezeichnet das

Suchen/Erlangen von neuen Erfahrungen und Erkenntnissen. Hier setzt sich der Ausführende einer unbekannten und potenziell gefährlichen Situation aus, aber steigert so durch neue Erfahrungen seine Kompetenz.

Aktiviertheit (Arousal) Aktiviertheit ist ein Maß für die Dringlichkeit einer Bedürfnisbefriedigung. Steigt die Aktiviertheit, steigt auch die Selektionsschwelle (Threshold) und sorgt somit für eine Fokussierung auf ein Ziel. Außerdem wird die Auflösung (Resolution) reduziert. Damit geht ein ungenaueres kognitives Planen der Aktionen einher. Dies führt zu einer schnellen Reaktion, aber dafür ungenauen Verhaltensweise.

2.3.3. Zusammenfassung

Die Theorie von Orthony et al. kategorisiert Emotionen nach Art und Ausprägung. Des weiteren gehen Orthony et al. davon aus, dass Ziele, Normen und Eigenschaften von Personen und Objekten vorhanden sind, anhand derer die Ereignisse, Handlungen oder Objekte emotional bewertet werden können. Die Theorie bedient sich einer reduktionistischen Herangehensweise.

Die PSI-Theorie hingegen verwendet einen holistischen Ansatz und entwirft ein System, das vielfältig ineinander rückgekoppelte und voneinander abhängige Teile besitzt. Der Schwerpunkt liegt hierbei auf den Beziehungen der Teilsysteme untereinander. Die PSI-Theorie umfasst das ganze System. Die Datenrepräsentation für Erinnerung und Pläne findet mit Hilfe neuronaler Netze statt und ermöglicht (im Zusammenspiel mit den anderen Systemteilen) ein Lernverhalten. Die Erklärung dieser Teile ist für diese Arbeit nicht vorgesehen, aber findet sich bei Interesse in den zitierten Werken.

2.4. State of the Art

An dieser Stelle folgt eine Beschreibung des State of the Art (Stand der Technik/Forschung) in den Bereichen Intelligente Agenten in Evakuierungssimulationen, Agenten in Verbindung mit Emotionstheorien und Emotionale Agenten in Evakuierungssimulationen.

2.4.1. Agenten und Emotionale Systeme

Die BDI-Agenten Architektur bietet durch das von der Menschlichen Handlungsregulation inspirierte Konzept bereits eine gute Grundlage, um Emotionale Systeme zu realisieren. Man-

che Ansätze für Emotionale Agenten gehen darauf zurück die BDI-Architektur um emotionale Komponenten zu erweitern (Emotional Belief Desire Intentional (EBDI)-Architektur (Jiang, 2007) und PECS Architektur (Schmidt, 2008).

Ein großer Forschungsbereich für Emotionale Agenten ist die Mensch-Maschinen Kommunikation. Für die Benutzerschnittstellen werden Emotionale Agenten in Betracht gezogen. Vorstellbar sind z. B. virtuelle Klassenkameraden für E-Learning Systeme. Die Architektur für solche Systeme wird in Ghasem-Aghaee u. a. (2008) vorgestellt. Emotionale Agenten werden auch bei Computerspielen verwendet, um die virtuellen Charaktere lebendiger erscheinen zu lassen [22] oder unterschiedliche Charaktertypen zu realisieren. Die interessante Fragestellung, ob Emotionale Agenten auch für Handlungsregulationen, die nichts mit menschlichem Verhalten zu tun haben, eingesetzt werden können, wird u. a. in Hannon (2003) am Beispiel eines intelligenten Hauses betrachtet. Des Weiteren wird die Steuerung von autonomen Robotern, die in der komplexen unvorhersehbaren Realwelt zurecht kommen sollen, mittels emotionaler Konzepte erforscht((Hirth u. a., 2008)).

2.4.2. Agenten und Evakuierungssimulationen

Die weit verbreitete Evakuierungssimulation Exodus sieht in ihrer Architektur ein Behavior-Subsystem vor[23]. Inwieweit bei diesem Subsystem aber die Definition intelligenter Agenten von Wooldrigde und Jennings (siehe 2.1) angesetzt werden kann, bleibt unklar. Nicht zuletzt auch, weil es sich um ein Lizenz Produkt handelt, dass seine internen Strukturen nicht vollständig Preis gibt. Dennoch werden intelligente Softwareagenten in vielen Evakuierungssimulationen genutzt, um individuelles Verhalten der Menschen zu simulieren. Die Agententechnologie macht es möglich mikroskopische Computersimulationen auch für Großraumevakuierungen, wie beim Auftreten eines Tsunamies oder einer Flutkatastrophe, zu modellieren. Mikroskopisch bedeutet, dass jeder einzelne Agent seinen individuellen Fluchtweg plant und unterschiedliches Verhalten zeigt (siehe hierzu Lämmel und Nagel (2008)).

2.4.3. Emotionale Systeme, Agenten und Evakuierungssimulationen

In vielen Evakuierungssimulationen wird das individuelle Verhalten der Flüchtenden durch (viele) Verhaltensparameter bestimmt. Ein emotionales Konzept, wie das von Orthony, Clore und Collins oder die PSI-Theorie von Dörner in Verbindung mit einer Evakuierungssimulation, wurde bei den Recherchen nicht gefunden.

[22] http://www.cs.northwestern.edu/~forbus/c95-gd/lectures/The_Sims_Under_the_Hood_files/v3_document.htm
[23] http://fseg.gre.ac.uk/exodus/work.html

3. Anforderungserhebung

In diesem Kapitel werden nun die Anforderungen erhoben, dazu werden als erstes fachliche und technische Anforderungen identifiziert. Danach findet eine genaue Analyse der wichtigsten Anforderungen statt.

3.1. Fachliche Anforderungen

Die fachlichen Anforderungen ergeben sich aus dem Thema dieser Arbeit. Eine Beschreibung der angestrebten Simulation findet man in der Einführung 1.2 (S.10). Wäre dies eine Objektorientierte (OO) Arbeit, würden als erstes Use Cases aufgestellt werden. Da es hier aber um ein Agentensystem geht, und die Agenten autonom und unabhängig vom Benutzer agieren, werden keine Use Cases erhoben. Auch die Anforderungen unterscheiden sich von den Anforderungen in OO Systemen. Als Anforderungen gelten hier die Ziele, die ein Agent verfolgt. Die Klassifizierung der Anforderungen orientiert sich an der Vorgehensweise zur Entwicklung von Agentensystemen - Prometheus 2.1.4.

3.1.1. Bewegung im Evakuierungsgegenstand

Die Pax [1] müssen sich im Evakuierungsgegenstand gezielt bewegen können und Hindernisse wie Stühle oder auch andere Pax umgehen können, um den Ausgang zu erreichen.

Klassifikation: essential

3.1.2. Sinnvolle Reaktion auf Kollisionen

Die Pax sollen mit Gegenständen und anderen Pax kollidieren können. Bei Kollisionen zweier Pax sollte eine kommunikative Konfliktlösung initiiert werden.

Klassifizierung: essential

[1] Pax: (Begriff für Passagier(e) im Fluggewerbe)

3.1.3. Emotional gesteuertes Fluchtverhalten

Das ist die zentrale Anforderung. Die Pax sollen den Evakuierungsgegenstand emotional gesteuert verlassen können. Emotional gesteuert bedeutet hier, dass die Pax menschlich nachvollziehbar auf Veränderungen, andere Pax oder Ereignisse in der Umgebung reagieren und zielstrebig das Flugzeug verlassen. Daraus folgt, dass sie ihren Wirkungsbereich einschätzen und emotional bewerten können müssen. Um dann ein, der Bewertung entsprechendes, Verhalten zu zeigen.

Klassifizierung: essential

3.1.4. Simulation von Gefahren

Es können Gefahren simuliert werden, die den Verlauf der Simulation beeinflussen; z. B. Feuer oder Rauch. Auf diese Gefahren reagieren die Pax mit Angst oder Panik.

Klassifizierung: conditional.

3.1.5. Verschiedene Emotionstypen

Verschiedene Emotionstypen sind denkbar, wie besonders mutige oder besonders ängstliche Pax.

Klassifizierung: conditional

3.1.6. Flugzeugcrew organisiert Evakuierung

Die Crew organisiert die Evakuierung und hilft anderen Pax das Flugzeug zu verlassen. Die Crew Mitglieder öffnen die Notausgänge und weisen den Pax in ihrer Umgebung auf den Notausgang hin. Bevor sie selber das Flugzeug verlassen, überprüfen sie, ob alle Pax das Flugzeug verlassen haben. Die Crew soll auch emotional gesteuert werden.

Klassifizierung: optional.

3.1.7. Verschiedene Anfangskonfigurationen

Der Benutzer wählt über eine GUI eine Anfangskonfiguration aus und legt damit die Simulations-Eigenschaften (properties) fest. Eine Vielzahl verschiedener Einstellungsmöglichkeiten sind hier denkbar:

- Auswahl vorgegebener Evakuierungsszenarien z.B. nach EASA- Standard EASA (2003).
- Festlegung der Anzahl der Pax und Verteilung nach Geschlecht, Alter, Beziehungen zu anderen Pax und Emotionstyp
- Anzahl der Crew Mitglieder
- Festlegung der funktionierenden Notausgänge
- Anzahl und Position der Hindernisse
- Anzahl und Position von Feuer oder anderen Gefahren

Klassifizierung: optional

3.2. Technische Anforderungen

Die Technischen Anforderungen bestimmen die technischen Voraussetzungen, die für das Erfüllen der fachlichen Anforderungen nötig sind.

3.2.1. Multiagentenplattform bzw.-sprache

Eine Multiagentenplattform bzw.-sprache soll genutzt werden, die die Umsetzung der fachlichen Anforderungen ermöglicht. Die hier gestellten fachlichen Anforderungen sind generell mit allen gängigen Agentenplattformen realisierbar. Deshalb wird sich die Auswahl auf die technischen Anforderungen und auf Aspekte der Einarbeitungszeit und Erfahrung richten. Die technischen Anforderungen, die die Agentenplattform erfüllen sollte sind folgende:

Kommunikation zwischen den Agenten Da die Pax zur Konfliktlösung bei Kollisionen oder zum Informationsaustausch kommunizieren, sollte die Kommunikation der Agenten untereinander von der Agentenplattform unterstützt werden. Hier sind Standards erwünscht, wie die Kommunikationsprotokolle der FIPA (siehe Kapitel 2.1.6 (S.20))

Klassifizierung: essential

Graphische Aufbereitung des Simulationsgeschehens Die GUI sollte das Gesamtsystem zu jeder Runde der Simulation darstellen. Das ist unabdingbar für das Debugging und Testen in Agentensystemen.

Zwei Varianten der Darstellung sind denkbar. Einmal könnte die Darstellung der Simulation in „Echtzeit“, also zur Simulationszeit, ablaufen oder es könnte erst simuliert und danach dargestellt werden. Das bedeutet, dass die Bewegungen und Zustände der Pax gespeichert werden müssen.

Klassifizierung: essential (eine Art der Darstellung)

Physik Berechnungen Das Berechnen von Kollision und realistischen Bewegungen der Passagiere und von Feuer und Rauch sollte möglich sein. Das Einbeziehen einer Physic Engine wäre hier denkbar. Die Agentenplattform sollte das Einbeziehen von externen Systemen ermöglichen.

Klassifizierung: essential

Simulationsumgebung Eine Simulationsumgebung kann in der ausgewählten Agentenplattform enthalten sein. Die Simulationsumgebung sollte zum einen die zeitliche Steuerung der Simulation ermöglichen und zum anderen die grafische Darstellung des Simulationsgegenstandes erleichtern.

Klassifizierung: conditional

3.2.2. Anforderungen Übersicht

Die fachlichen und technischen Anforderungen sind nun erhoben und werden hier in einer Tabelle 3.1 übersichtlich aufgelistet.

Auf diesen Anforderungen aufbauend, kann jetzt die technische Analyse (Kapitel 4) durchgeführt werden.

Fachliche Anforderung	**Scope**
Navigation in der Simulationsumgebung	essential
Sinnvolle Reaktion auf Kollisionen	essential
Emotional gesteuertes Fluchtverhalten	essential
Simulation von Gefahren	conditional
Verschiedene Emotionstypen	conditional
Flugzeugcrew organisiert Evakuierung	optional
Verschiedene Anfangskonfiguration	optional
Technische Anforderung	
Multiagentenplattform	essential
Kommunikation zwischen den Agenten	essential
Grafische Aufbereitung des Simulationsgeschehens	essential
Physik Berechnungen	essential
Simulationsumgebung	conditional

Tabelle 3.1.: Anforderungen Übersicht

4. Technische Analyse

In diesem Kapitel werden die wichtigsten der identifizierten fachlichen und technischen Anforderungen analysiert. Es werden verschiedene Lösungsvorschläge aufgestellt und anhand der aufgestellten Anforderungen bewertet und ausgewählt.

Als erstes wird die Multiagentenplattform ausgewählt, da viele weitere Anforderungen und deren Umsetzung davon abhängen, welche Strukturen diese vorgibt.

Danach werden die technischen Anforderungen der Simulationsumgebung, der Physic Engine, sowie die grafische Aufbereitung des Simulationsgeschehens diskutiert.

Zum Schluss werden die wichtigsten fachlichen Anforderungen aufgegriffen und Herangehensweisen für das System Design entwickelt.

4.1. Auswahl der Agentenplattform

Es gibt eine Vielzahl von Agentenplattformen, die für die Umsetzung der aufgestellten Anforderungen in Frage kommen. Alle vorhandenen Lösungen gegeneinander abzuwägen wäre ein eigenständige Arbeit und ist hier nicht vorgesehen.

Einige Kandidaten wären Jadex[1], Jason[2], und JADE[3], sowie die Agentbasierte Simulationsumgebung Swarm[4].

Da die zeitliche Begrenzung dieser Arbeit ein tieferes Einarbeiten in unbekannte Agentenlösungen verhindert, fällt die Entscheidung auf die Agentenplattform Jadex, da Vorkenntnisse vorhanden sind und somit die Einarbeitungszeit kurz ist. Jadex wurde an der Universität Hamburg entwickelt und ist eine zu Jade kompatible Agentenplattform, die zusätzlich den BDI - Standard implementiert. Jadex basiert auf Java, welches durch z.B. AWT, Swing o.ä. Packages geeignet ist, um eine GUI zu implementieren. Jadex implementiert die

[1] zu finden unter http://jadex.informatik.uni-hamburg.de/bin/view/About/Overview
[2] zu finden unter http://jason.sourceforge.net/
[3] zu finden unter http://sharon.cselt.it/projects/jade/
[4] zu finden unter http://sharon.cselt.it/projects/jade/

BDI-Agenten und erleichtert das Starten und Beenden sowie Debuggen der BDI-Agenten durch vielfältige grafische Werkzeuge. Zum Implementieren eines Jadex Agenten werden zwei Dinge benötigt. Erstens die ADF, in dieser Datei wird die Agentenstruktur mittels XML beschrieben. Die ADF dient Hauptsächlich zur Beschreibung von Beliefs, Goals, Plans und Events. Zweitens die Implementierung der konkreten Aktionen (Pläne), die der Agent ausführen kann. Die Aktionen werden als Java Klasse beschrieben und von den in der ADF beschriebenen Plänen referenziert. Eine ausführliche Dokumentation der Jadex Plattform findet man auf der Webseite unter http://jadex.informatik.uni-hamburg.de/bin/view/Resources/Online+Documentation.

Des weiteren bietet Jadex ein Modulierungskonzept an. Es können Goals und Pläne in eine sogenannte Capability definiert werden. Diese Capability wird in einer ADF definiert und kann von einer anderen ADF eingebunden werden. So stehen die Elemente der Capability dem Agenten zur Verfügung. Vergleichbar ist dieses Konzept mit der Vererbung bei der Objektorientierung.

4.1.1. Machbarkeitsstudie Jadex

Um die Machbarkeit der Simulation mit Jadex besser einschätzen zu können, wurde ein technischer Prototyp entwickelt, das HunterPrey Beispiel[5] von Jadex zur Grundlage hat. Dieser technische Prototyp basierend auf der Jadex Version 0.96 und wurde eingesetzt, um folgende Dinge zu implementieren:

- schematische Darstellung eines Flugzeug ähnlichen Raumes
- Bewegung eines Pax in der Umgebung
- Eine einfache Kollisionserkennung
- Eine einfache Emotion
- Ein Feuer

Die Abbildung 4.1 zeigt die erfolgreiche Umsetzung des Prototypen.

Zu sehen sind die Sitzreihen und der Mittelgang des Flugzeuges. Die Pax können sich im Flur nach links und rechts bewegen, in den Sitzreihen nur nach oben und unten. Zwei Pax können nicht gleichzeitig ein Feld belegen. Damit wurde eine Kollisionserkennung umgesetzt. Ein statisches Feuer wird dargestellt. Wenn der Pax in die Nähe eines Feuers kommt, wird seine interne Variable „afraidness“ auf true gesetzt. Dadurch wird das Maintain-Goal (Abschnitt

[5] Das HunterPrey Beispiel implementiert das Jäger Beute Szenario in einer gridbasierten einfachen Umgebung online ausführbar unter http://jadex.informatik.uni-hamburg.de/bin/view/Usages/Examples

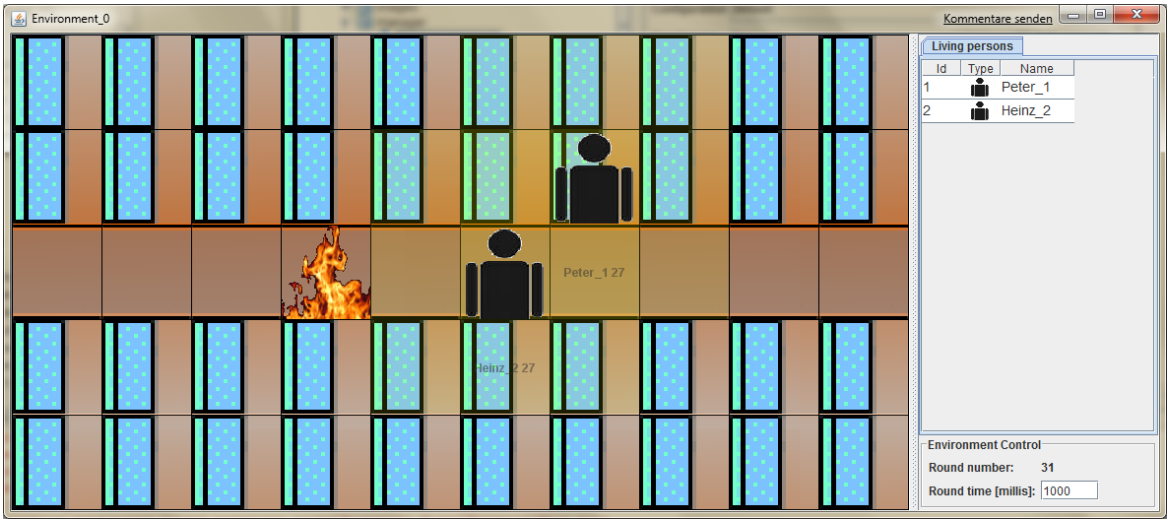

Abbildung 4.1.: Prototyp Visualisierung

2.1.3 (S.16)) „stay_save“ aktiviert und der Plan „flee“ ausgelöst. „Flee“ lässt den Pax vom Feuer weg laufen.

Erkenntnisse aus dem Prototypen

Es folgt eine Beschreibung der Erkenntnisse aus der Implementierung des Prototypen.

Darstellung und Kollisionserkennung Die schematische, gridbasierte Darstellung des Flugzeuges im technischen Prototypen ist zwar einfach, bedeutet aber auch, dass die Simulation sehr statisch wirkt. Würde ein kontinuierlicher Raum als Darstellung für den finalen Prototyp gewählt werden, könnte eine Physic Engine die Kollisionsberechnung übernehmen. Für den finalen Simulations Prototypen ist daher eine genauere kontinuierliche Darstellung angestrebt, sowie die Benutzung einer Open Source Physic Engine.

Daraus ergibt sich auch die Anforderung, ein relativ genaues Modell des Evakuierungsgegenstandes (Flugzeug) als Polygonmodell umzusetzen, um ihn kontinuierlich abzubilden (Kapitel 6.2 (S.78)). Außerdem sollte eine Schnittstelle für die Physic Engine in das Design der Simulation integriert werden und eine passende Physic Engine ausgewählt werden 4.3. Der große Vorteil der Physic Engine besteht in der realistischeren Darstellung von Bewegungen und Kollisionen. Auch Dinge wie Explosionen und deren Kraftauswirkung auf Passagiere ließen sich mit relativ kleinem Aufwand integrieren.

Emotionen Die im Prototyp implementierte Emotionssteuerung hat funktioniert, macht aber keine Aussage darüber, ob emotionale Steuerung in einer komplexeren Umgebung möglich ist, da nur eine Boolesche Variable (afraidness) benutzt wurde. Eine Erkenntnis aus dem Prototyping für die Emotionsimplementierung ist, dass das Maintain-Goal (Abschnitt 2.1.3 (S.16)) ein geeigneter Kandidat für die Implementierung von emotionalen Bedürfnissen zu sein scheint.

Kommunikation Im Prototypen wurde auch Kommunikation zwischen den Agenten umgesetzt. Hierfür wurde die Protocols-Capability[6] genutzt, die die Funktionen der FIPA-Kommunikationsprotokolle kapselt. Im Prototypen hat der Pax Agent vom Environment Agenten seine aktuelle Sicht auf die Umwelt über das Inform-Protokoll erhalten.

4.1.2. Fazit

Die Agentenplattform Jadex wurde ausgewählt und anhand des Prototypen gezeigt, dass Jadex geeignet ist, diese Simulation zu verwirklichen. Die technischen Anforderungen der Kommunikation zwischen Agenten und die grafische Aufbereitung werden problemlos unterstützt.

4.2. Analyse der Simulationsumgebung

Nachdem der technische Prototyp mit der Jadex Version 0.96 fertig entwickelt war, wurde die Jadex Version2 Beta2 vom 15. Januar 2009 zum Download bereit gestellt. Jadex Version 2 Beta2 implementiert ein Simulationspaket. Das Paket bietet ein Gerüst für die Umsetzung einer Simulation an. Die Abbildung 4.2 bietet einen Überblick über die grobe Architektur der Simulationsumgebung.

Darstellung Die Darstellung der Simulation wird durch offene Grafikstandards wie OpenGl[7] und der Java Schnittstelle dafür, Jogl[8], unterstützt. Die Simulationsumgebung läuft, wenn keine Grafikbeschleunigung zur Verfügung steht, aber auch ohne Grafikbeschleunigung. Eine klar definierte Schnittstelle ermöglicht die grafische Anzeige der verschiedenen

[6] Eine genaue Beschreibung der Protocols Capability findet man im Kapitel 16.3 unter http://jadex.informatik.uni-hamburg.de/bin/view/User+Guide/16+Predefined+Capabilities+(old)

[7] zu finden unter http://www.opengl.org/

[8] zu finden unter https://jogl.dev.java.net

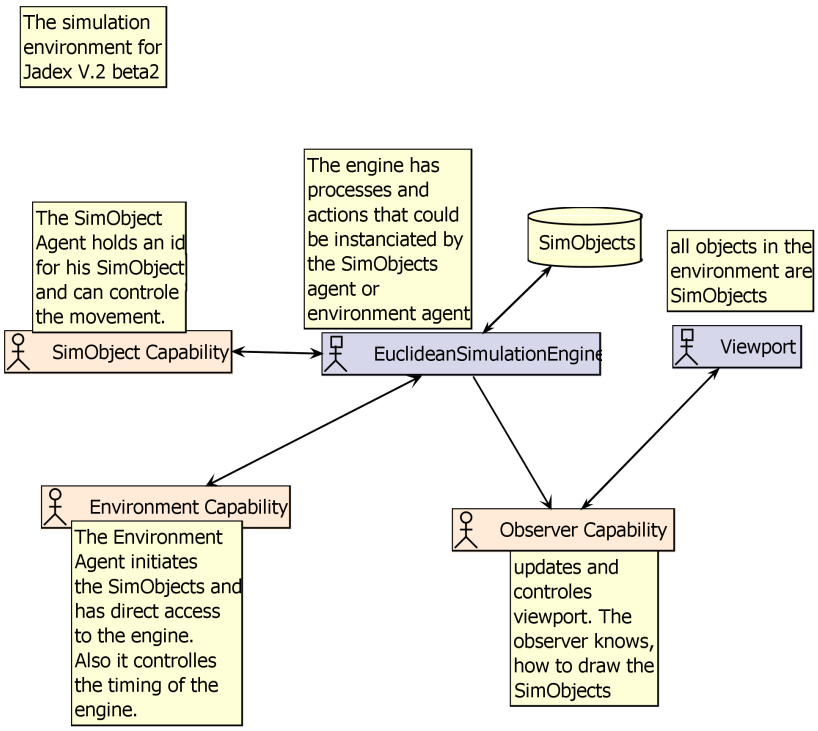

Abbildung 4.2.: System Übersicht - Simulationsumgebung Jadex V.2 Beta2

Simulationsobjekte. Die SimObserver Capability bietet diverse Goals zur Initialisierung der grafischen Anzeige und regelt die Aktualisierung der GUI (Viewport).

Zeitliche Steuerung der Simulation Die zeitliche Steuerung der Simulation wird durch die SimEnvironment-Capability übernommen und kann im Jadex Werkzeug für Simulationen gesteuert werden. Hier kann zwischen kontinuierlichem, taktgesteuertem und eventbasiertem Simulationsablauf (Siehe 2.2.2) umgeschaltet werden.

Processes und Actions Die Environment Capability bietet Goals zum Hinzufügen von Actions und Processes zur Environment Engine. Actions implementieren das Interface ISimAction und werden zur Ausführung bestimmter (einmaliger) Aktionen genutzt. Beispielsweise zum Verlassen des Flugzeuges durch die Tür. Die Processes implementieren das Interface IEnvironmentProcess und werden zu jeder Runde der Simulation einmal ausgeführt.

Simulations Objekte Eine SimObject-Capability bietet u. a. Goals zur Bewegung der Simulations-Objekte sowie Goals zur An- und Abmeldung bei der Simulationsengine. Außerdem beinhaltet die SimObject Capability ein Goal, um eine Action des Environments aus-

zuführen („environment_perform_action“). Dem Simulationsobjekt können zusätzlich noch „Task“ (ISimObjectTask) zugewiesen werden, die in jeder Runde ausgeführt werden, bis sie erfüllt sind.

Risiko - Analyse Eine Beta Version zu benutzen ist ein Risiko, da die Version fehlerhaft sein kann und/oder wenig Dokumentation zur Verfügung steht.

Jadex Version 2 Beta2 vs. Jadex Version 0.96 Themen wie der zeitliche Ablauf der Simulation und die grafische Darstellung sind sehr umfangreich und auch nicht Kerngebiet dieser Arbeit. Somit ist die Arbeitsersparnis, die mit der Verwendung der Simulationsumgebung einhergeht, den Aufwand zur Einarbeitung und die Risiken wert. Die Verwendung der Simulationsumgebung von Jadex Version2 Beta2 ermöglicht, eine Konzentration auf die wesentlichen Punkte dieser Arbeit - die emotionale Komponente. Die Entscheidung fällt also auf die Jadex Version 2 Beta2.

Risikominimierung Um das Risiko zu minimieren, wurde Rücksprache mit den Entwicklern von Jadex[9] gehalten und die implementierten Beispiele genau studiert, um die Funktionalität der Simulationsumgebung zu verstehen. Dabei ergab sich, dass die meisten Konzepte aus der Version 0.96 erhalten worden sind. Zusätzlich bietet die Simulationsumgebung ein leistungsfähiges Konzept für die Implementation der Evakuierungssimulation.

4.3. Physic Engine

Um die Kollisionen und Bewegungen der Pax möglichst realistisch zu gestalten, bietet sich eine Physic Engine an. Um zu bestimmen, welche Physic Engine hier zum Einsatz kommt, werden, nach einer kurzen Beschreibung von Physic Engines, die Anforderungen an die Physic Engine erhoben und eine passende Physic Engine ausgewählt. Danach wird noch auf Problematiken bei der Nutzung der ausgewählten Physic Engine eingegangen.

Eine Physic Engine (bedeutet soviel wie: Physikmodul) ist meist Teil eines Computerprogrammes, das die Simulation physikalischer Prozesse[10] zum Ziel hat. Sie berechnet des weiteren verschiedene Eigenschaften der Objekte, wie z. B. den Impuls. Ziel ist es, die Programmierung zu vereinfachen und eine realistische Umgebung zu vermitteln.

[9] Lars Braubach und Alexander Pokahr von der Universität Hamburg
[10] z. B. basierend auf der Newtonschen Mechanik

Anforderung	JBox2D	Phys2D
Statische und dynamische Körper	ja	ja
Komplexe Körper	ja	ja
Pax über Kollision informieren	nicht offensichtlich	ja, über Event
Java	ja	ja
Klare Schnittstelle	nein	ja
Unterstützung von Feuer und Rauch	nein	nein

Tabelle 4.1.: Anforderungen Physic Engine

Anforderungen an die Physic Engine Folgende Anforderungen werden an die Physic Engine gestellt:

- Unterstützung von statischen(unbewegten) und dynamischen(bewegten) Körpern.
- Berechnung von Kollisionen komplexer [11] Körper. Klassifikation: must have
- Bei aufgetretener Kollision informieren des Pax. Klassifikation: must have
- In Java geschrieben, da Jadex auch auf Java basiert und hier auf eine Middleware verzichtet werden soll (KIS-Prinzip). Klassifikation: must have
- Klar definierte Schnittstelle zum Anbinden an die Simulationsumgebung. Klassifikation: must have
- Unterstützung von Feuer und Rauch. Klassifikation: nice to have

Auswahl der Physic Engine Es wurden zwei Physic Engines untersucht und auf die Anforderungen überprüft. Die erste ist JBox2D [12]. Sie ist eine Java-Portierung der performanten auf C++ basierenden Box2D Physic Engine. Trotz der Java Portierung ist der C++ Stil beibehalten worden und daher für einen Java Programmierer mit Einarbeitungszeit verbunden. Die Nutzung von Box2D erschien im Test kompliziert und war wenig intuitiv. Die zweite untersuchte Physic Engine ist Phys2D [13] ist nicht ganz so performant wie JBox2D, basiert aber auf den gleichen Prinzipien. Phys2D bietet eine klare Schnittstelle und ist im gewohnten Java Stil implementiert. Die Funktionsweise der Engine ist nach wenigen Minuten Einarbeitungszeit verständlich.

[11] mit Komplex sind hier Polygone gemeint
[12] Zu finden unter http://www.jbox2d.org/
[13] http://www.cokeandcode.com/phys2d/

Wahl der Physic Engine Die Wahl fällt auf Phys2D, da die einfachere Schnittstelle die Einarbeitungszeit gering hält und die Benutzung vereinfacht.

Konsequenzen aus der Nutzung einer Physic Engine Die Jadex Simulationsumgebung bietet in der SimObject-Capability bereits Goals zur Bewegung der Simulationsobjekte (im folgenden SimObjecte) an. Da diese Goals die Bewegung der SimObjecte unabhängig von der Physic Engine realisieren, bedarf es entweder einer Synchronisation der Bewegung der Physikobjekte und der SimObjecte, oder die Bewegungs-Goals der Capability werden neu definiert mit dem Ziel, die Steuerung der Bewegung nur noch über die Physic Engine zu realisieren.

Synchronisation der Bewegung zwischen Physikobjekten und Simulationsobjekten von Jadex Die Synchronisation der Objekte ist solange simpel und fehlerfrei, bis eine Kollision auftritt. Die SimObjecte werden über die Parameter Velocity und Position gesteuert. Diese Parameter finden sich auch bei den Physikobjekten. Solange eine geradlinige Bewegung vorliegt, könnte die Velocity und die Position der SimObjecte die Vorgabe für die Physikobjekte sein und diese bei Abweichungen angleichen. Sollte nun eine Kollision registriert werden, könnten für eine kurze Zeit die Parameter der Physikobjekte die Steuerung übernehmen. Ein Problem hierbei ist, dass während die Physic Engine die Kontrolle über die Steuerung der Objekte übernimmt, die SimObjecte keinen Einfluss mehr auf ihre Bewegungen haben. Bei ständigen Kollisionen, z. B. in einer dichten Paxmenge, würde die Kontrolle nicht mehr an die SimObjecte zurück gegeben werden. Damit wären die Pax nicht mehr zu steuern. Es bedarf also einer zusätzlichen Kontrolle, die feststellt, ob gehäufte Kollisionen auftreten. Diese Kontrolle gibt dann gegebenenfalls die Steuerung der Bewegung an die Pax zurück.

Bewegung nur über Physic Engine Die Bewegung der Pax nur über die Physic Engine zu realisieren, würde bedeuten, dass die SimObject Capability Goals zur Bewegung nicht genutzt werden könnten oder zumindest umgeschrieben werden müssten. Eine Möglichkeit wäre, die Steuerung der Bewegung nur über Kräfte, die auf das Physikobjekt wirken, zu realisieren (Phys2D bietet hierfür die Methode body.setForce(Vector2) an). Diese Realisierung käme dem Beschleunigen und Bremsen des menschlichen Ganges sehr nahe. Die Physic Engine hat dann die Möglichkeit, die Auswirkung dieser Kräfte zu bestimmen und in eine Bewegung umzusetzen. Dadurch würden sehr weiche Bewegungen entstehen. Beim Auftreten vieler Kollisionen (dichte Paxmenge) könnten diese Kräfte trotzdem die Bewegung der Pax in eine Richtung lenken. Die Synchronisation der Position wäre nur von den Physikobjekten zu den SimObjects nötig.

Auswahl eines Synchronisationsverfahrens Die Synchronisation von Bewegungen zwischen verschiedenen Welten (Simulationsengine und Physic Engine) ist kompliziert und führt zu vielen Schwierigkeiten. Daher fällt die Wahl darauf, die Steuerung der Bewegung an die Physic Engine zu delegieren und somit eine realistische Darstellung von Bewegungen zu ermöglichen.

Fazit Es wurde die Phys2D Physic Engine ausgewählt, sowie Möglichkeiten zur Synchronisation der Bewegung von SimObjects mit den Physikobjekten vorgestellt. Die Verantwortung für Bewegungen wurde abschließend an die Physic Engine delegiert.

4.3.1. Analyse der grafischen Aufbereitung des Simulationsgeschehens

Als erstes wird an dieser Stelle geklärt, wie realistisches Flugzeugmodell in das System integriert werden kann. Als zweites wird über die Darstellung des Ablaufes der Simulations diskutiert. Das Thema der grafischen Aufbereitung sollte möglichst einfach gehandhabt werden, da es nicht die zentrale Anforderung der Arbeit ist.

Flugzeugmodell Zur Darstellung eines Evakuierungsgegenstandes, hier ein Flugzeug bzw. der Innenraum des Flugzeuges, werden Daten benötigt, mit denen sich das Flugzeug darstellen lässt. Der Einfachheit halber wird hier auf eine dreidimensionale Darstellung verzichtet und nur zweidimensional modelliert. Es besteht die Möglichkeit ein Bild des Innenraums als Wavefront Standard 3D Object Format (OBJ)-Format[14] abzuspeichern und mit Hilfe eines Java OBJ-Parsers einzulesen und daraus geometrische Objekte zu generieren. Diese können dann von der Simulationsengine angezeigt werden.

Darstellung des Ablaufs des Simulationsgeschehens Die beweglichen Objekte der Simulation können ihre Positionen in Abhängigkeit zum zeitlichen Verlauf der Simulation ändern. Zwei Möglichkeiten bestehen, den Ablauf darzustellen.

„Echzeit"-Darstellung Die Darstellung könnte synchron zum Simulationsablauf (zur Laufzeit) angezeigt werden. Das bedeutet, dass die Positionen der beweglichen Objekte zu jeder Simulationsrunde direkt an die Darstellung gereicht werden. Wenn die Performanz des Systems allerdings gering ist, führt das zu einer sehr langsamen Darstellung.

[14] OBJ ist ein unkomprimiertes Datenformat für grafische Modelle. Spezifikation unter http://www.martinreddy.net/gfx/3d/OBJ.spec

Darstellung im Nachhinein Die Darstellung im Nachhinein bedeutet, dass erst simuliert wird. Die Positionen und weitere darstellungsrelevante Eigenschaften der beweglichen Objekte müssten während jeder Simulationsrunde in einer beliebigen Datenstruktur abgelegt werden. Unabhängig von der Simulation kann nun im Nachhinein der Ablauf beliebig oft angesehen werden.

Fazit Die Architektur des Systems sollte beide Darstellungsvarianten ermöglichen. Die Simulationsumgebung von Jadex beinhaltet nur eine Darstellung zur Laufzeit. Jadex ermöglicht einem aber durchaus die Daten der Simulation zu speichern und im Nachhinein anzuzeigen. Das wäre aber mit einigem Entwicklungsaufwand verbunden. Um möglichst wenig Aufwand in die Darstellung zu stecken, wird hier zunächst nur die Darstellungsfunktion der Simulationsumgebung von Jadex zur Laufzeit genutzt.

4.4. Emotional gesteuertes Fluchtverhalten

Die Pax sollen ihre momentanes Verhalten an ihrem emotionalen Status orientieren.

4.4.1. Auswahl des emotionalen Modells

Folgende Anforderungen stellen sich an das emotionale Modell.

- Es soll emotionales Verhalten hervorbringen können. Denn es ist offensichtlich, dass eine Emotion, ohne ein daraus resultierendes Verhalten gerade in einer Evakuierungssituation, keinen Sinn macht.
- Es muss bei konkurrierenden Verhaltenszielen auswählen können. Eine Situation kann mehrere Emotionen auslösen; Ein denkbarer Konflikt wäre in einer Evakuierungssituation einerseits das Fluchtbedürfnis (Angst) und andererseits das Bedürfnis einer anderen Person zu helfen (Mitleid).
- Verschiedene Emotionstypen sollten grundsätzlich realisiert werden können.

Emotionstheorie von Ortony, Clore und Collins versus PSI-Theorie von Dörner Da die Emotionstheorie von OCC (siehe 2.3.1 (S.27)) keine Aussage darüber macht, wie aus den analysierten Emotionen ein Verhalten resultiert, ist die erste Anforderung bereits fast ein Ausschluss-Kriterium für das OCC Modell. „Fast“, da es natürlich möglich wäre, für jede der vielen unterschiedlichen Emotionen bei OCC eine andere Verhaltensweise zu implementieren. Das zweite Kriterium der konkurrierenden Verhaltensziele wird ebenfalls nicht direkt von

der Emotionstheorie von OCC unterstützt. OCC sprechen zwar von einer Hierarchie der Ziele und Wünsche der Person, beschreiben aber nur sehr oberflächlich, wie diese umgesetzt wird Schneider (2005, Kapitel 10).

Die PSI-Theorie von Dörner 2.3.2 (S.29) unterstützt Verhaltenssteuerung und bietet auch ein Konzept zur Lösung des Problems der konkurrierenden Ziele. Die verschiedenen Emotionstypen können durch unterschiedliche Einstellung der Reduzierung bzw. Zunahme von Bedürfnissen entstehen. Somit ist die PSI-Theorie das Mittel der Wahl.

Funktionsweise der PSI-Theorie Die Funktionsweise der PSI-Theorie wurde in den Grundlagen 2.3.2 (S.29) ausführlich erläutert. Zusammenfassend werden hier nochmal die relevanten Eigenschaften für die Pax in der Evakuierungssimulation genannt:

- Verhalten des Pax wird durch Bedürfnisse gesteuert.
- Relevante existentielle Bedürfnisse: Schmerzvermeidung (engl.: Physical Integrity)
- Relevante informelle Bedürfnisse: Kompetenz, Bestimmtheit, Affilitation
- Die informellen Bedürfnisse steuern folgende relevanten Verhaltenstendenzen: Sicherungsverhalten, Flucht, Aggression, spezifische Exploration, diversive Exploration
- Die Bedürfnisse steuern die Aktiviertheit. Die Aktiviertheit bestimmt die Selektionsschwelle und die Auflösung.

Risiko Analyse für die PSI-Theorie Die PSI-Theorie ist ein ganzheitlicher Ansatz für emotional gesteuertes Verhalten. Daher ist sie eine Theorie „aus einem Guss„ und widerspricht dem Prinzip der Modularisierung (Schneider, 2005, Kapitel 10). Es wird im Umfang dieser Arbeit nicht möglich sein, die in der PSI-Theorie genutzten neuronalen Netze für die Erinnerung, Wahrnehmungen, Lernverhalten sowie das dynamische Planen zu implementieren. Somit ist es fraglich, ob es möglich ist, nur die Teilbereiche der holistischen Theorie umzusetzen, die für die Evakuierungssimulation nötig sind und trotzdem ein emotional gesteuertes Verhalten zu generieren.

4.5. Analyse der Navigation im Simulationsgegenstand

Die Navigation oder zielgerichtete Bewegung der Pax im Simulationsgegenstand ist eine Voraussetzung für das erfolgreiche Verlassen des Flugzeuges. Jeder Pax hat einen Sichtbereich und kann darauf (und auf seiner Erinnerung) basierend die nächsten Schritte planen. Des weiteren werden im Evakuierungsgegenstand in regelmäßigen Abständen Leuchtpfeile

aufgestellt sein, die eine Orientierung des Pax ermöglichen. Trotzdem ist ein Algorithmus nötig, der das Umgehen von Hindernissen ermöglicht. Da in dieser Simulation auf eine kontinuierliche Umgebung gesetzt wird, ist diese Anforderung nicht trivial. Durch die vielen Pax auf engem Raum ändern sich außerdem die Bedingungen ständig.

4.6. Wegfindungs Algorithmen

Der A* Algorithmus basiert auf einem Knotennetz (Graph) und findet in diesem einen optimalen Weg (vgl. hierzu (Russel und NORVIG, 2003)). Um also den A* Algorithmus benutzen zu können, müsste ein Knotennetz über den kontinuierlichen Simulationsraum gelegt werden. Einen informierten Wegfindungsalgorithmus wie den A* Algorithmus zu verwenden, ist also für eine kontinuierliche Umgebung nicht ohne größeren Aufwand zu bewerkstelligen. Da die Pax zusätzlich ihr nächstes Bewegungsziel emotional gesteuert finden sollen und nicht auf eine optimale Wegfindung zurückgreifen sollen, wird von der Verwendung des A* Algorithmus an dieser Stelle Abstand genommen.

Stattdessen wird die Wegfindung der Pax von den aktuellen Verhaltenstendenzen und dem Wissen über ihre Umwelt gesteuert. Sollte ein Pax z.B. das Verhalten der diversiven Exploration ausüben, wird er versuchen neue Erkenntnisse zu erlangen. Dies könnte in der Suche nach Richtungssignalen, die zum nächsten Notausgang weisen, verwirklicht werden. Damit der Pax nun aber nicht ständig gegen Gegenstände rennt, ist ein Algorithmus nötig, der das verhindert.

Steering Behaviors (Steuerungs- oder Lenkverhalten) Es gibt sogenannte Steering Behavior Algorithmen, die durch Beschleunigungsveränderungen und Steuerungskräfte die aktuelle Bewegung eines Objektes[15] beeinflussen. Die für das Verhalten der Pax relevanten Steering Behaviors sind Seek/Flee/Arrival, Obstacle Avoidance und Separation.

Nach Reynolds (1999) ist die Funktionsweise der Algorithmen folgende:

Arrival/Seek Behavior Das Ziel dieser Algorithmen ist das Ansteuern eines Ziels. Hierfür wird der Beschleunigungsparameter so manipuliert, dass der Pax in die Richtung des Ziels abgelenkt wird siehe Abbildung 4.3.

Bei Seek wird das Ziel mit voller Geschwindigkeit angesteuert. Das führt dazu, dass der Pax über das Ziel hinausschießt. Diese Verhaltensweise ist erwünscht, wenn ein Pfad aus

[15] Dieses Objekt wird als Vehicle bezeichnet und besitzt Parameter wie Beschleunigung, Masse, Position, Max-Speed, Max-Beschleunigung und Orientierung

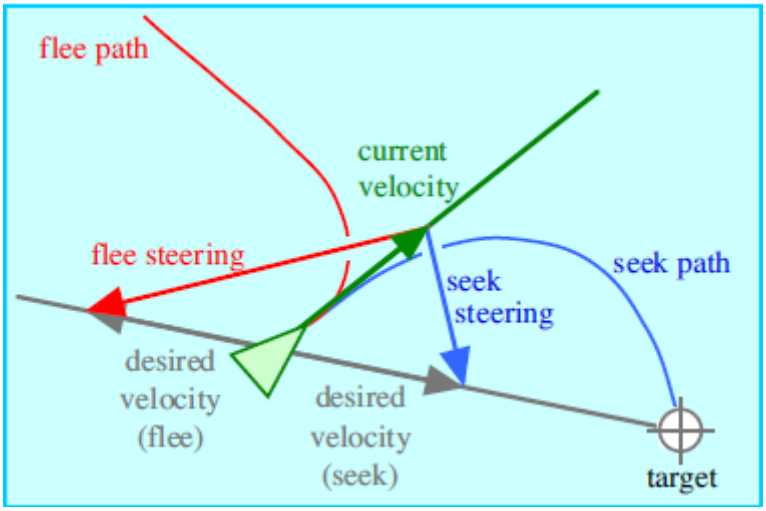

Abbildung 4.3.: Seek Flee (Reynolds, 1999)

mehreren Zielpunkten verfolgt werden soll, ohne zu bremsen. Arrival hingegen bremst, in Abhängigkeit zur Entfernung zwischen Pax und Ziel ab, um asymptotisch das Ziel zu erreichen[16]

Flee wird, wie auf Abbildung 4.3 dargestellt, durch eine Umkehrung des Seek Verhaltens erreicht.

Obstacle Avoidance Obstacle Avoidance (Hindernis Umgehung) ermöglicht dem Pax Hindernisse zu umgehen. Das Ziel dieses Verhaltens ist, einen imaginären zylinderförmigen Raum vor dem Pax frei von Hindernissen zu halten (siehe Abbildung 4.4). „Vor“ dem Pax bedeutet, in Bewegungsrichtung des Pax. Die Länge des Zylinders hängt dabei davon ab, wie schnell und beweglich der Pax ist. Die Breite des Zylinders entspricht der Breite des Pax[17]. Findet nun eine Überschneidung eines Hindernisses mit dem Zylinder statt, wird eine Kurskorrektur veranlasst. Die Korrektur wirkt dabei vom Hindernis weg. Somit wird eine frühzeitige Kollisionsvermeidung realisiert.

Separation Separation ermöglicht einem Pax in einem dichten Gedränge möglichst viel Abstand zu den anderen Pax zu halten. Das ist nötig, um ständige Kollisionen mit den anderen Pax zu vermeiden. Für jeden Pax in der näheren Nachbarschaft wird eine abstoßende

[16] Wegen der asymptotische Annäherung an das Ziel muss, um festzustellen, wann das Ziel erreicht ist, eine Toleranzgrenze eingeführt werden.

[17] bzw. der Breite der Bounding Box des Pax. Eine Bounding Box ist z. B. ein Kreis, der um ein komplexes Polygonmodell gelegt wird. Die Bounding Box wird dazu verwendet Kollisionsberechnungen zu vereinfachen.

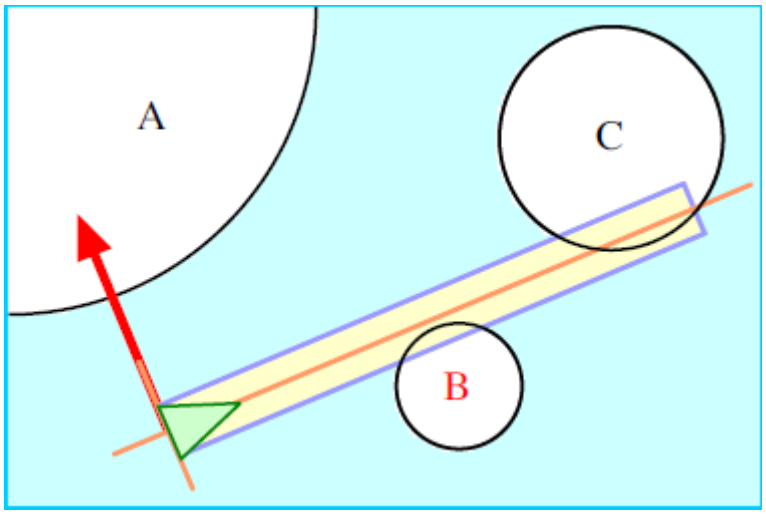

Abbildung 4.4.: Obstacle Avoidance (Reynolds, 1999)

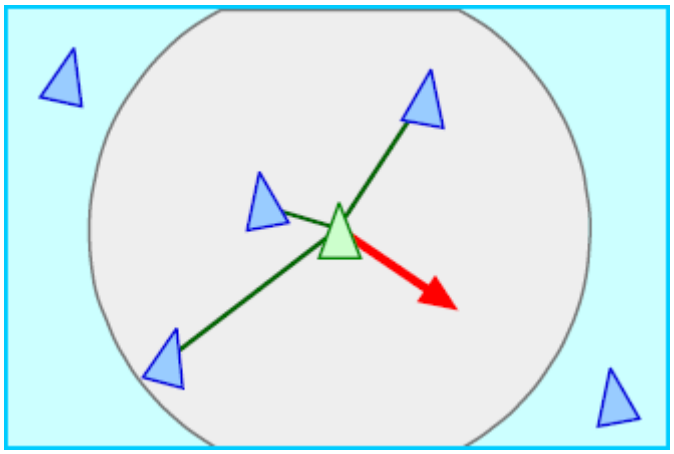

Abbildung 4.5.: Separation (Reynolds, 1999)

Kraft in Abhängigkeit zum Abstand berechnet und die Summe der Kräfte als Bewegungskorrektur genutzt (siehe Abbildung 4.5).

Zusammenführung der Steering Behaviors Die einzelnen Algorithmen sollten, um unterschiedliches Verhalten des Pax zu unterstützen, je nach Situation an- oder ausgeschaltet oder in Kombination genutzt werden können; z. B. ist das „Seek“ Verhalten nur sinnvoll, wenn gleichzeitig das „Obstacle Avoidance“ Verhalten aktiviert ist. Befindet sich der Pax in einem dichten Gedränge, wird zudem noch das „Separation“ Verhalten aktiviert, um sich möglichst viel Platz zu verschaffen. Die resultierende Bewegungskorrektur aus mehreren parallelen Steering Behaviors kann z. B. mittels einer (gewichteten) Summe aus allen Bewegungskorrekturen der einzelnen Verhaltensweisen berechnet werden. Reynolds (1999, Seite 17) Eine interessante Demonstration der drei hier genannten Verhaltensweisen in Kombination ist als Java-Applet auf `http://www.red3d.com/cwr/steer/Doorway.html` zu finden.

Fazit Die Navigation des Pax im Flugzeug erfolgt durch zwei Mechanismen. Zum einem wird das nächste Bewegungsziel durch die Emotionale Verhaltenssteuerung bestimmt. Zum Anderen wird die Ausführung der Bewegung durch die Kombination verschiedener Steering Behavior ermöglicht.

4.7. Analyse der Reaktion auf Kollisionen

Verschiedene Arten von Kollisionen sind zu unterscheiden.

- Vereinzelte Kollisionen: Wenn der Pax mit einem Gegenstand kollidiert, muss die Bewegungsrichtung angepasst werden. Bei Kollisionen mit einem anderem Pax sollte eine Kommunikation zur Konfliktlösung stattfinden, wobei in Abhängigkeit von den Verhaltensparametern einer der Pax dem anderen Pax den Vortritt lässt.
- Gehäufte Kollisionen: Bei häufigen Kollisionen, wie in einer dichten Warteschlange, sollten die Kollisionen weitestgehend ignoriert werden und die Bewegungsrichtung beibehalten oder neu auf das Ziel ausgerichtet werden.

4.8. Zusammenfassung

In diese Kapitel wurden die wichtigsten technischen und fachlichen Anforderungen detailliert analysiert. Darauf basierend wurde eine Auswahl von Arbeitsmitteln getroffen und es

wurden grundlegende Eigenschaften für das Design der Evakuierungssimulation und die Realisierung des finalen Prototypen festgelegt:

- Auswahl von Jadex als Agentenplattform
- Benutzung der Jadex Version2 Beta2 aufgrund der integrierten Simulationsumgebung
- Benutzung der Physic Engine Phys2D
- Entscheidung für einen kontinuierlicher Evakuierungsgegenstand in Form eines Flugzeuginnenraums
- Verwendung der PSI-Theorie als Basis für die Verhaltenssteuerung
- Nutzung von Steering Behavior zur Bewegungssteuerung
- Kommunikative Lösung für Kollisionskonflikte zwischen den Pax

Nun wird im folgenden Kapitel 5 das Design des Systems erläutert.

5. Design

In diesem Kapitel wird das Design des Gesamtsystems mit verschiedenen beschrieben und durch Diagramme veranschaulicht. Da die Wahl der zugrunde liegenden Agentenplattform auf Jadex gefallen ist, wird als Vorgehensmodell für Multiagentensysteme Prometheus, in Verbindung mit dem PDT, genutzt (siehe Grundlagen 2.1.4 (S.17)). Leider können hier nicht alle Schritte, die das Prometheus Vorgehensmodell vorsieht, dargestellt werden. Weil es den Rahmen dieser Arbeit sprengen würde. Das Design basiert auf den fachlichen und technischen Anforderungen und beinhaltet auch Aspekte, die im Prototyp aus zeitlichen Gründen nicht implementiert werden konnten. Als erstes wird hier eine Architektur-Übersicht über das System gegeben. Dabei werden die Agenten und ihre Zusammenspiel erläutert. Danach geht es an das detaillierte Design der Agenten und eines zentralen Kommunikationsprotokolls. Zu erwähnen bleibt noch, dass standardmäßig auf Englisch programmiert wurde und somit viele Bezeichnungen in diesem Kapitel in Englisch gehalten sind.

5.1. Architektur des Multi Agenten Systems

Die gewählte Architektur des MAS orientiert sich zwar an der Simulationsumgebung von Jadex 2 Beta. Es wurde hier jedoch, um eine Abhängigkeit von der Simulationsumgebung zu verhindern, ein Design gewählt, das auch ohne die Simulationsumgebung umsetzbar ist.

Die identifizierten Akteure und Beziehungen zwischen den Akteuren werden auf der Abbildung 5.1 dargestellt.

- Pax Agent : Alle Pax in dem Flugzeug basieren auf der gleichen Agenten Implementation. Daher ist nur ein Pax Agent im Diagramm dargestellt. Das angestrebte unterschiedliche Verhalten der Pax wird durch Modulation verschiedener Parameter erreicht. Zum einen äußere Parameter wie Gewicht, Alter, maximale Geschwindigkeit und Größe. Zum anderen innere Parameter, wie eine starke Tendenz zum Fluchtverhalten (ängstlicher Typ) oder diversiven Exploration (mutiger Typ).
- Emotional System Agent: Für die inneren Parameter ist der Emotional System Agent zuständig. Jeder Pax Agent „besitzt“ einen Emotional System Agent, der die Funktionalitäten des PSI System kapselt. Der Emotional System Agent ist aus Gründen

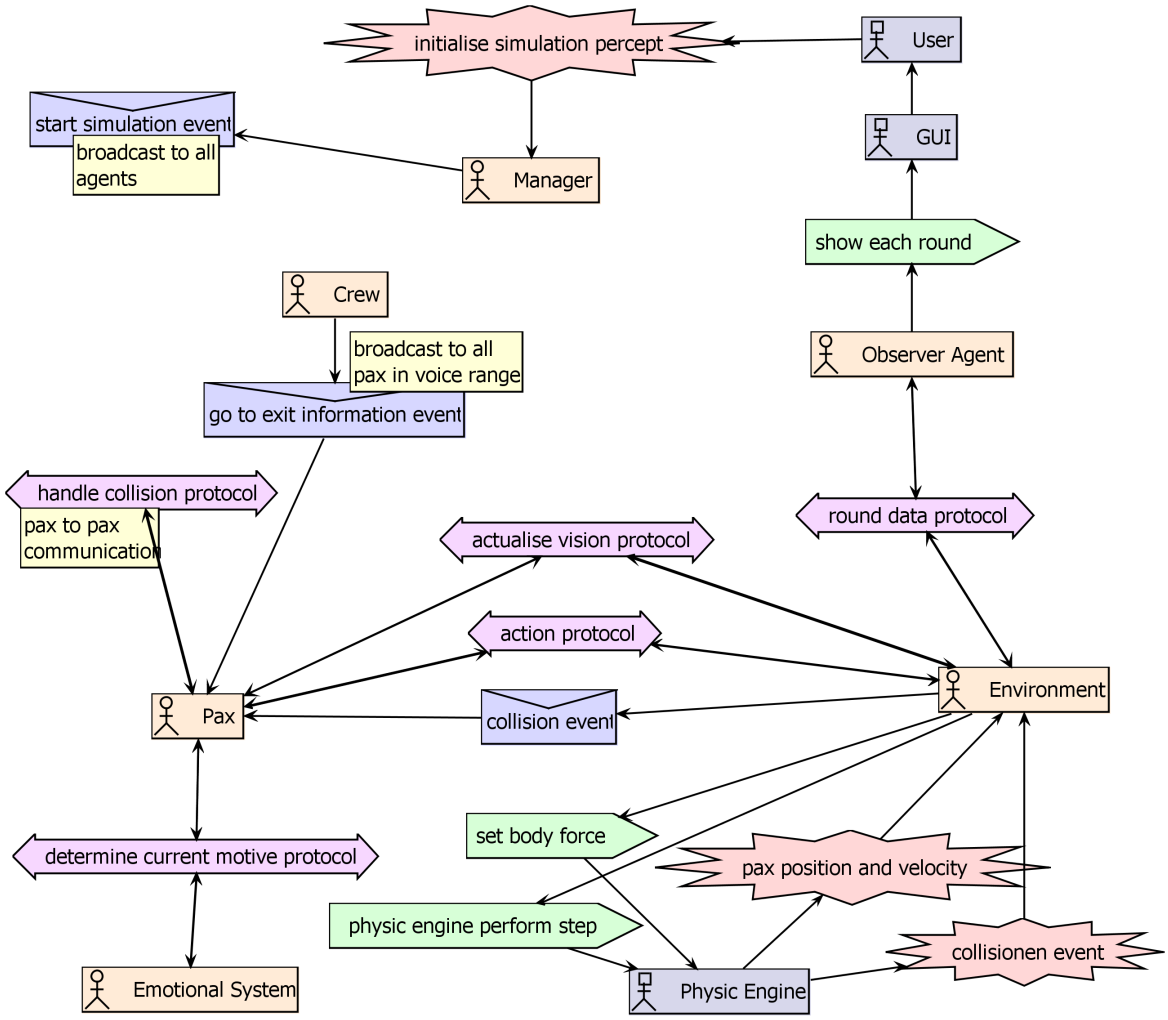

Abbildung 5.1.: System - Übersicht

der losen Kopplung als eigenständiger Agent geplant. Das garantiert eine leichte Austauschbarkeit verschiedener Emotionaler Systeme oder Typen.

- Crew Agent: Der Crew Agent erweitert das Verhalten des Pax um die Funktionen der Crew-Mitglieder und steht ebenfalls stellvertretend für alle anderen Crew-Mitglieder.
- Environment Agent: Der Environment Agent hat Zugriff auf die Simulationsobjekte des Simulationsgegenstand. Er kontrolliert den Simulationsablauf und die Physic Engine.
- Physic Engine: Die Physic Engine berechnet die Bewegungen der Pax und informiert über Kollisionen.
- GUI Agent: Der GUI Agent verwaltet, wie zu erwarten, die Darstellung auf der GUI.
- Manager Agent: Der Manager Agent initialisiert alle anderen Agenten.

Des weiteren werden das Zusammenspiel und die Abhängigkeiten zwischen den einzelnen Akteuren kurz erläutert. Eine genaue Beschreibung des wichtigsten hier erwähnten Kommunikationsprotokolls („determine current motive protocol“) folgt im Abschnitt 5.2.6 (S.70).

User und Manager Agent Basierend auf der Anforderung „Verschiedene Anfangskonfigurationen“ wählt der User eine Simulationskonfiguration aus und startet den Manager Agent. Daraufhin empfängt der Manager Agent das „initialise simulation percept“ vom User und startet die Simulation, indem er an alle Pax und das Environment das „start simulation event“ sendet.

Pax Agent und Environment Agent Der Pax Agent kommuniziert über zwei Protokolle mit dem Environment. Zum einen dem „action protocol“, um Bewegungen oder andere Aktionen im Environment zu realisieren, und zum anderen dem „actualise vision protocol“, um die aktuelle Sicht („currentView“) auf die Umwelt vom Environment zu erhalten. Das „collision event“ informiert den Pax über Kollisionen.

Pax Agent und Pax Agent Gemäß der Anforderung der Reaktion auf Kollisionen kommuniziert der Pax mit anderen Pax über das „handle collision protocol“, um Kollisionskonflikte mit anderen Pax zu lösen.

Pax Agent und Emotional System Agent Eine Trennung zwischen Pax Agent und Emotional System Agent ist bewusst gewählt, um eine lose Kopplung der Systemteile zu gewähren. Der Pax kommuniziert mit seinem „Emotional System“ über das „determine current motiv protocol“, um das aktuelle Handlungsmotiv für die Situation emotional zu bestimmen.

Pax Agent und Crew Agent Der Crew Agent informiert alle in seiner „Stimmreichweite“ liegenden Pax über einen funktionstüchtigen Ausgang mittels dem „go to exit information event“

Environment Agent und Physic Engine Der Environment Agent teilt der Physic Engine mit, wenn ein Pax beschleunigen oder bremsen möchte („set body force“ action). Im Gegenzug berechnet die Physic Engine Kollisionen und teilt diese dem Environment mit („collision event“ percept). Außerdem wird die Position der Pax und anderer beweglicher Gegenstände dem Environment mitgeteilt.

Environment Agent und GUI Agent Der GUI Agent aktualisiert regelmäßig (z. B. 30 mal pro Sekunde) seine Daten über die Simulations Objekte über das Protokoll „round data protocol“.

GUI Agent und GUI Der GUI Agent aktualisiert während des Simulationsablauf die GUI über die „show each round“ action.

GUI und User Der User sieht den Simulationsablauf auf der GUI.

5.2. Design der Agenten

Um das Design der Agenten darzustellen, wird erstens ihre Goal Hierarchie dargestellt. Hierbei ist interessant, dass hierarchisch höher gestellte Goals das „Was soll erreicht werden?“ beschreiben und die darunter gestellten Goals das „Wie soll es erreicht werden?“ beschreiben. Als zweitens wird das detaillierte Design des Agenten beschrieben. Hierbei wird ein Blick auf die innere Struktur des Agenten geworfen. Es wird festgelegt, welche Events, Goals und Daten zu welchen Plänen oder Capabilities gehören.

5.2.1. Detailliertes Design des Pax Agent

Der Pax Agent betreibt intensive Kommunikation mit seinem inneren Emotional System Agent, um die aktuelle Handlung zu bestimmen. Außerdem tauscht er sich mit dem Environment Agenten aus, um Aktionen zu realisieren oder Informationen über seine Umgebung zu beziehen.

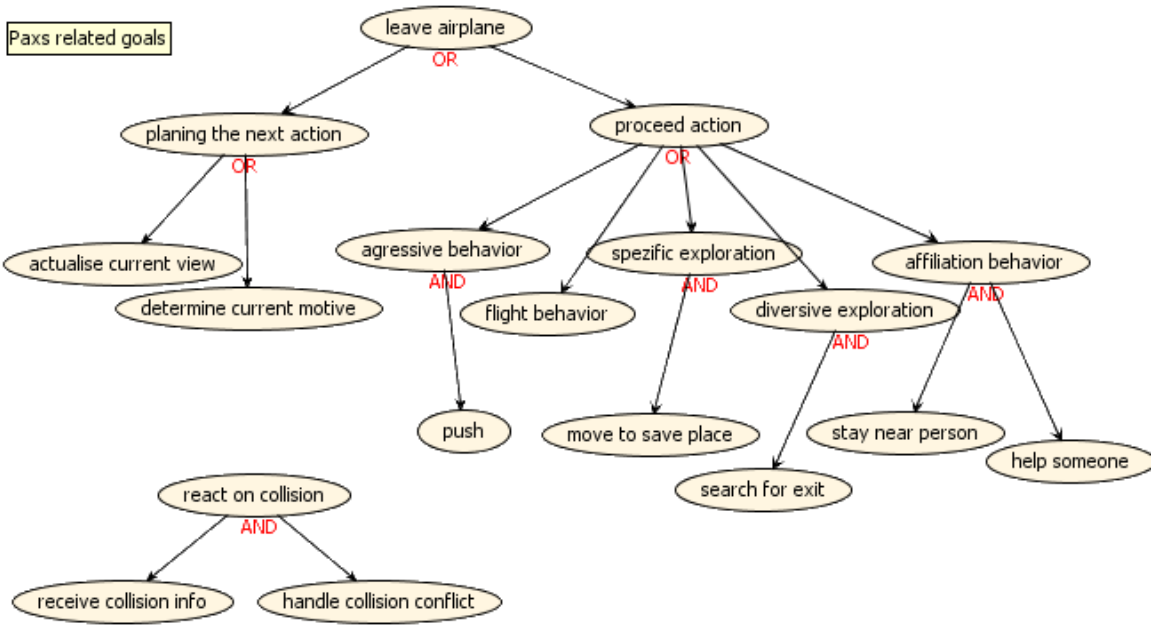

Abbildung 5.2.: Pax Agent Goal Hierarchie

Goal Hierarchie Die Abbildung 5.2 zeigt die Hierarchie der Goals des Pax Agenten. An oberster Stelle steht das Ziel „leave Airplane“. Zur Erfüllung dieses Ziels stehen zwei Verhaltensweisen zur Verfügung. Wenn keine aktuelle Aktion zur Verfügung steht, muss der nächste Schritt erst einmal geplant werden (linker Zweig der Hierarchie). Dazu wird das „planing next action“ Goal genutzt. Dieses Goal sorgt dafür, dass die Sicht auf die Umwelt aktualisiert wird („actualise current view“) und darauf basierend das aktuelle Motiv ausgewählt wird („determine current motive“). In Abhängigkeit des aktuellen Motivs und der Verhaltensparameter kann die nächste Aktion bestimmt und ausgeführt werden (rechten Seite der Hierarchie). Hier stehen die verschiedenen Verhaltensweisen zur Auswahl. Sie entsprechen den in der PSI-Theorie definierten Verhaltenstendenzen, siehe Abschnitt 2.3.2 (S.29). Jedes Verhalten führt zu anderen Subzielen, bis hin zu einer konkreten Handlung, wie Bewegungen oder Öffnen eines Notausgangs. Die „react on collision“ Goalstruktur ist einfacher aufgebaut. Sie besteht aus dem Empfangen von Kollisionsinformation („receive collision info“) und aus dem Auflösen möglicher Kollisionskonflikte („handle collision“). Dieser Fall könnte eintreten, wenn zwei Pax von verschiedenen Seiten gleichzeitig die selbe Position betreten wollen und sich gegenseitig blockieren. Einer muss dem Anderen den Vortritt lassen (hier könnte die Aggressionstendenz eine Rolle spielen), um den Konflikt aufzulösen.

Innere Struktur Das detaillierte Design 5.3 zeigt, wie die Kommunikation gehandhabt wird und welche Daten dabei benötigt werden. Um die Abbildung zu verstehen, sollte sie im Zu-

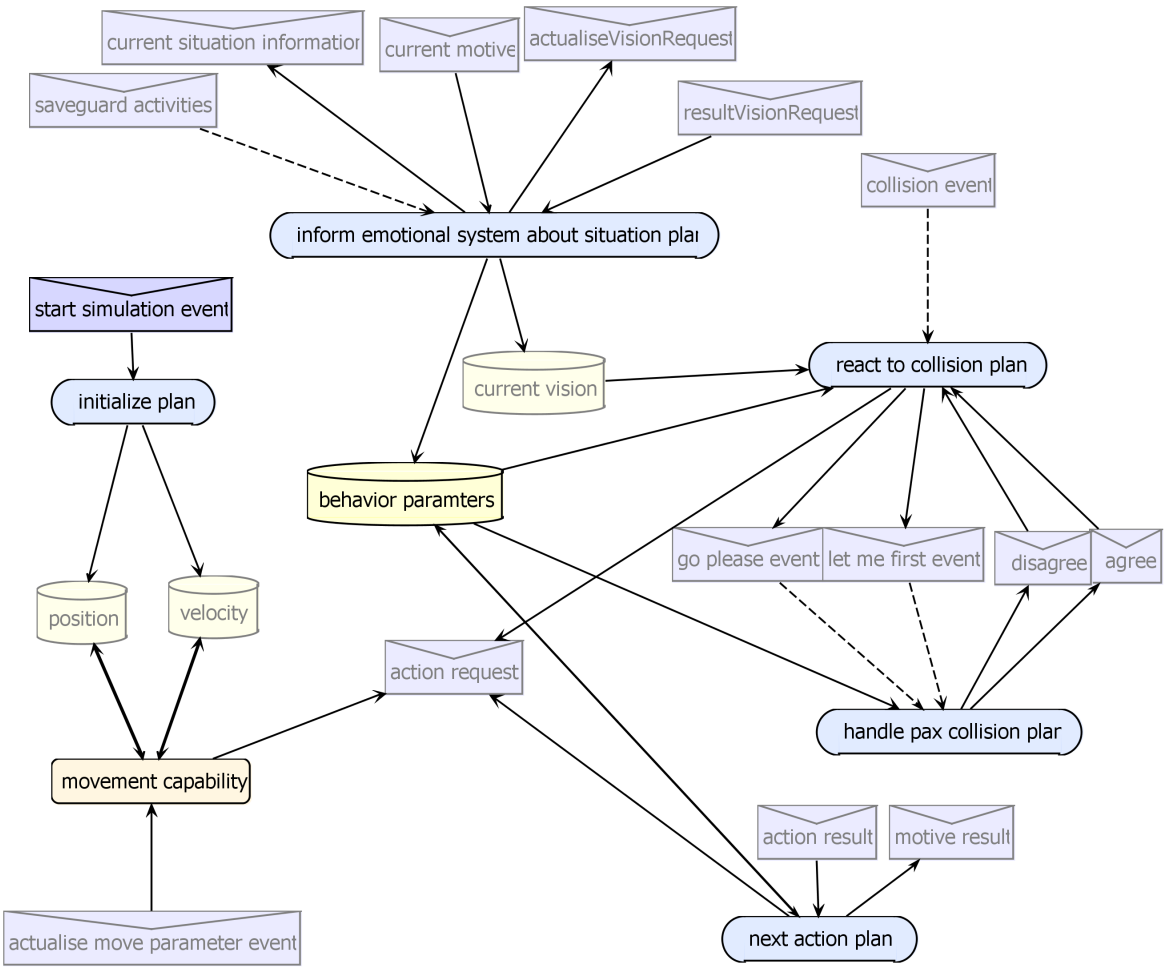

Abbildung 5.3.: Detailliertes Design des Pax Agenten

sammenhang mit den Kommunikationsprotokoll „determine current motive protocol“, das im Abschnitt 5.2.6 erläutert wird, betrachtet werden.

Der „inform emotional system about situation plan“ ist dafür zuständig, die aktuelle Situation an das Emotionale System zu senden. Dafür wird die aktuelle Sicht aktualisiert („actualise vision request“), analysiert und das Ergebnis versendet.

Der „next action plan“ erhält das aktuelle Motive („current motive“) mitsamt den „behavior paramters“ vom Emotional System Agent. Daraufhin wird ein dazu passender Plan ausgewählt und ausgeführt. Dafür wird eine (oder mehrere) „action request“ ans Environment geschickt. Wurde die Aktion beendet, wird das Ergebnis der Aktion ans Emotionale System gesendet. Ein Sonderfall liegt vor, wenn das empfangene Motiv bereits ausgeführt wird. Das bedeutet, dass sich der Emotionale Zustand nicht geändert hat, daher wird keine weitere Deliberation betrieben und das bisherige Ziel weiter verfolgt.

Der „react to collision plan“ wird durch den Empfang eines „collision events“ ausgelöst. Das Event informiert den Pax über eine Kollision die aufgetreten ist. Fand der Zusammenstoß mit einem Gegenstand statt, wird eine Aktion ausgewählt, z. B. eine Bewegungskorrektur, und über das „action request“ versendet. Ist die Kollision aber mit einer Person aufgetreten und besteht ein Bewegungskonflikt mit der Person oder besteht die Gefahr einer Kollision, wird in Abhängigkeit der „behavior parameter“ ein „go ahead“ oder ein „let me first event“ versendet. Bei der empfangenden Person löst das den „handle collision plan“ aus, der je nach „behavior parameter“ auf Zustimmung („agree“) oder Ablehnung(„disagree“) stoßen kann.

Die „movement capability“ beinhaltet die Pläne und Ziele der in der „technischen Analyse“ vorgestellten Steering Behaviors, siehe Abschnitt 4.6 (S.53). Bewegungen des Pax werden über diese Capability gesteuert.

5.2.2. Detailliertes Design des Emotional System Agenten

Der Emotional System Agent ist für jeden Pax das „Unterbewusstsein“. Er steuert die Requirements und die Verhaltensparamter des Pax.

Goal Hierarchie Die Abbildung 5.4 zeigt die Hierarchie der Goals des Emotional System Agenten. Das übergeordnete Ziel „determine current motivator and behavior parameters“ soll die Verhaltensweise der Pax bestimmen. Einen großen Einfluss darauf haben die untergeordneten „maintain“ Ziele. Sie bestimmen die Aktivität und Stärke der Motivatoren und sorgen somit dafür, dass die Bedürfnisse des Pax erfüllt werden.

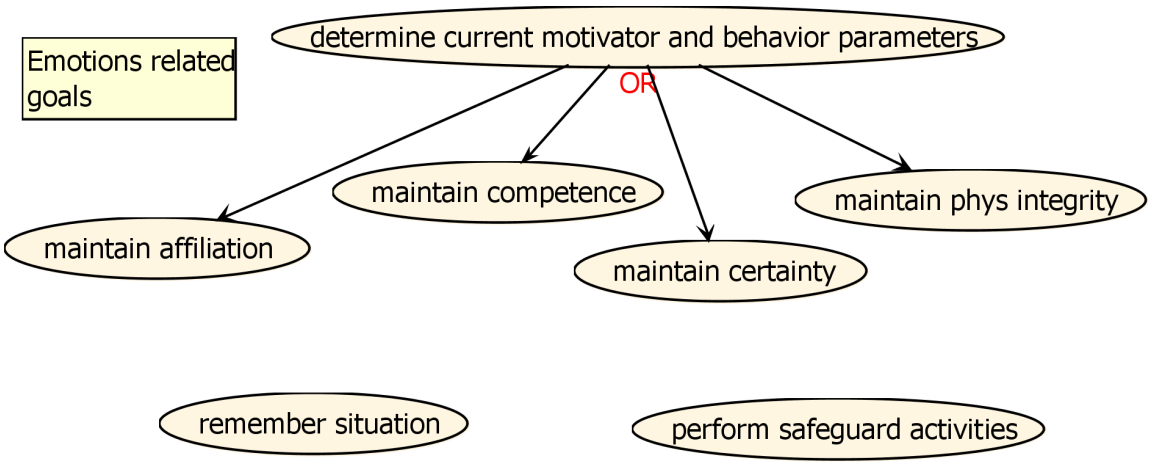

Abbildung 5.4.: Goal Hierarchie Emotional System Agent

„perform saveguard activities“ sorgt dafür, dass regelmäßig die erforderlichen Aktivitäten zur Absicherung vorgenommen werden. Das bedeutet, dass die Pax die eigene Situation einschätzen und gegebenenfalls das aktuelle Verhalten anpassen müssen.

Das Ziel „remember situation„ sorgt dafür, dass Situationen gemäß ihrer emotionalen Wirkung gespeichert werden. Dieses „Erinnern“ ist nötig, um z. B. nicht immer wieder in dasselbe Feuer zu laufen.

Innere Struktur Die Abbildung 5.5 zeigt die innere Struktur des Emotional System Agenten. Ein wichtiger Bestandteil ist der „saveguard activities plan“. Er steuert die Häufigkeit der Neubewertung der aktuellen Situation des Pax. Die Frequenz der „safeguard activities“ hängt, wie in den Grundlagen der PSI-Theorie beschrieben (Abschnitt 2.3.2), von den Steuerungsparametern ab.

Der „analyse current situation plan“ empfängt Informationen über die aktuelle Situation und beeinflusst daraufhin die Requirements. Die Informationen könnten z. B. eine Verletzungsignal beinhalten, weil der Pax in den Wirkungsbereich eines Feuers gelaufen ist. Daraufhin würde das Requirement Physical Integrity reduziert werden.

Der „current motive plan“ bestimmt, basierend auf den Requirements und Steuerungsparametern, die Motivstärken und wählt das aktuelle Motiv aus (siehe Pseudocode Algorithm 1). Außerdem werden die „behavior parameters“ berechnet. Zuletzt wird das Motiv über die „current motive“ Nachricht an den Pax verschickt.

Der „remember motive success or failure plan“ empfängt das „motive result“ vom Pax und

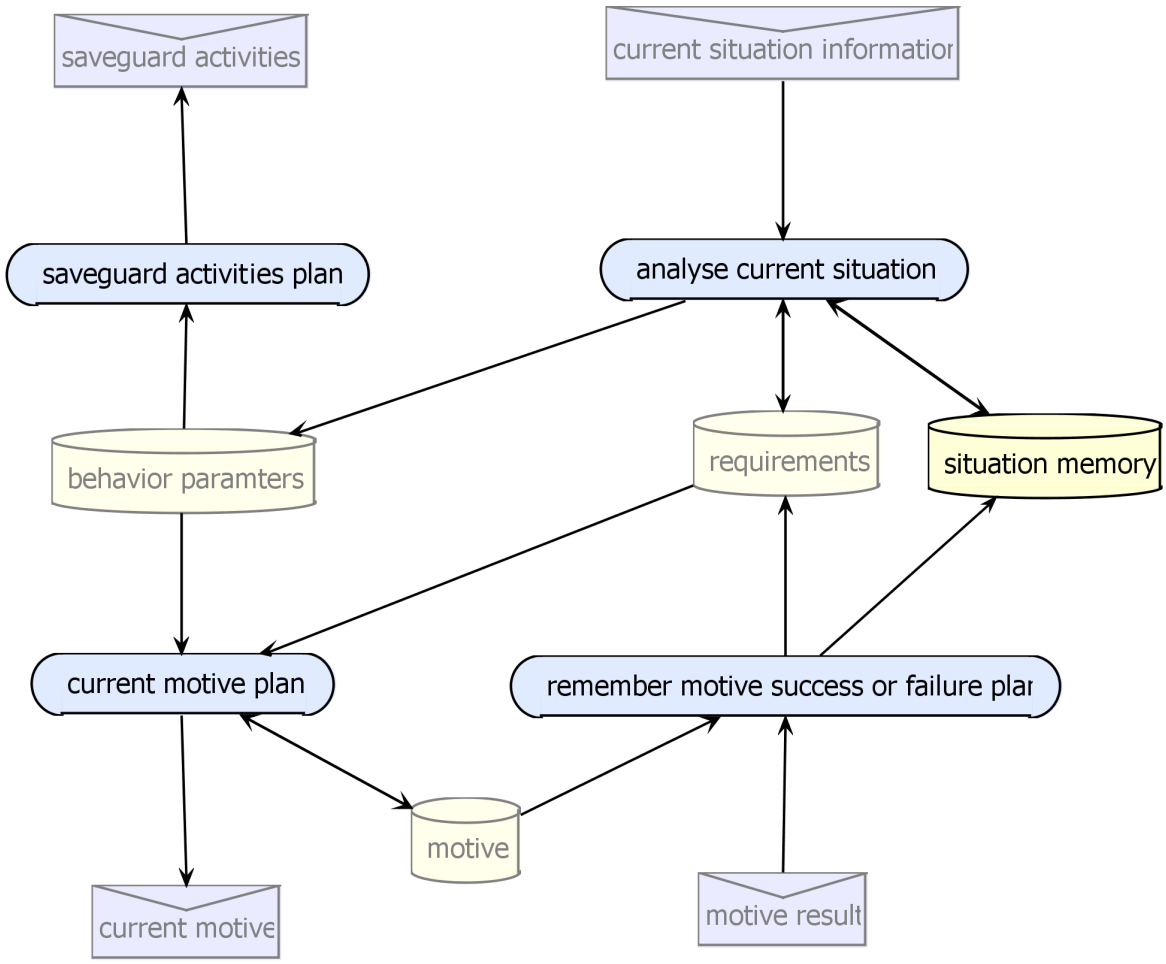

Abbildung 5.5.: Detailliertes Design des Emotional System Agent

bewertet daraufhin das Motiv in Verbindung mit der aktuellen Situation positiv oder negativ. Diese Bewertung wird benötigt, um die Erfolgswahrscheinlichkeit der Motive zu berechnen.

Algorithm 1 Determine Current Motive Plan

```
1: {Berechnen der Steuerungsparameter}
2: Erstelle eine Liste aller aktiven Motivatoren;
3: arousal = Summe der Stärke der Motivatoren;
4: selection threshold = log(arousal/Anzahl aktiver Motivatoren);
5: resolution = max(1, 1/log(arousal));
6: {Berechnen der Erfolgswahrscheinlichkeit der Motivatoren}
7: for jeden Motivator do
8:     Ähnliche Situation = z. B. (Aktuelle Anzahl in der Umgebung Pax - Erinnerung Anzahl
       Pax) * resolution <= 1;
9:     if Ähnliche Situation gefunden then
10:        Dringlichkeit = log(Stärke des Motivators)*(Erfolgswahrscheinlichkeit der erinnerten
           Situation);
11:    else
12:        Dringlichkeit = log(Stärke des Motivators) * Zufallszahl zwischen 0 und 1;
13:    end if
14: end for
15: erhöhe Dringlichkeit des aktive Motivators um selection threshold;
16: neuer Motivator = Motivator mit höchster Dringlichkeit;
```

Pseudocode zur Auswahl des aktiven Motivators Der Pseudocode Algorithm 1 zeigt, wie mit Hilfe der Steuerungsparameter arousal, selection threshold und resolution der Motivator bestimmt wird. Die Schwierigkeit hierbei besteht darin, den Wertebereich der Steuerungsparameter so zu bestimmen, dass sie den richtigen Einfluss besitzen. Der Pseudocode macht einen Vorschlag, wie diese Gewichtungen ausfallen könnten. Es kann aber sein, dass sich in der Praxis zeigt, dass andere Wertebereich und Gewichtungen für die Steuerungsparameter gewählt werden.

5.2.3. Detailliertes Design des Environment Agenten

Der Environment Agent ist dafür zuständig, die Simulation ablaufen zu lassen. Er steuert die Physic Engine und verwaltet die Simulationsobjekte.

Goal Übersicht Die Abbildung 5.6 zeigt die Goal Hierarchie des Environment Agenten. Das oberste Ziel des Environment Agenten ist es, die Simulation ablaufen zu lassen („run

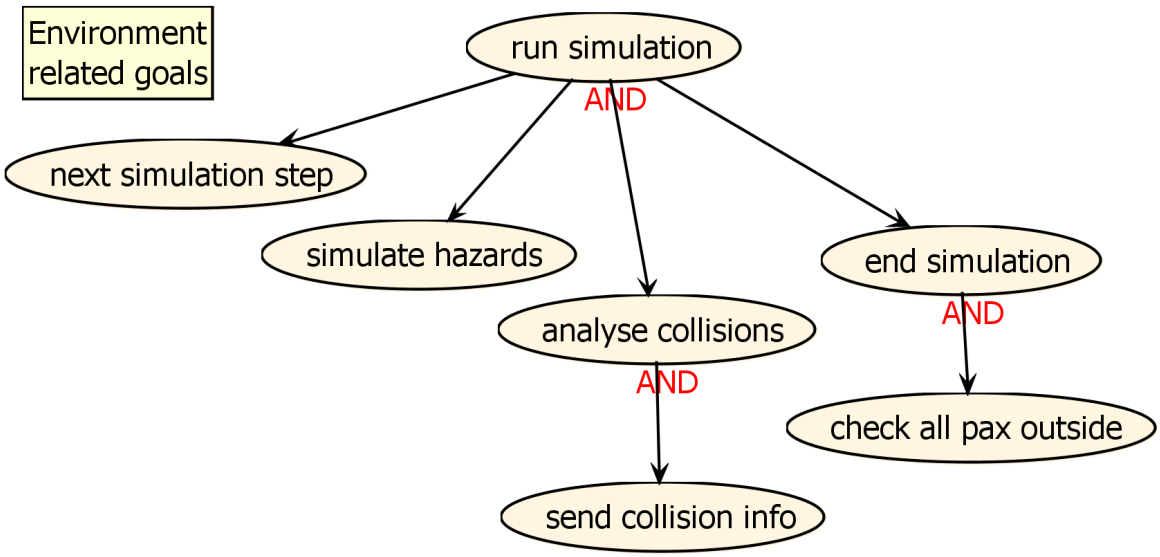

Abbildung 5.6.: Goal Hierarchie des Environment Agenten

simulation“). Dieses Goal wird realisiert, indem für jeden Simulationsschritt („next simulation step“), die Gefahren(„simulate hazards“) simuliert werden, die Kollisionen überprüft werden („analyse collisions“) und, wenn die Endbedingung erfüllt ist, die Símulation beendet wird. Diese Endbedingung ist, dass alle Pax (die noch bewegungsfähig sind) das Flugzeug verlassen haben.

Innere Struktur Die Innere Struktur des Environment Agenten wird auf der Abbildung 5.7 dargestellt. Um die Simulation ablaufen zu lassen, wird durch einen Zeitgeber das „next round event“ produziert. Daraufhin wird der „run simulation plan“ aktiviert. Dieser Plan sorgt für die Ausführung der oben beschriebenen Goals; z. B. sorgt die Aktion „physic engine perform step“ dafür, dass die Physic Engine einen weiteren Schritt berechnet. Des weiteren werden die Pax Aktionen im „handle passenger action plan“ ausgeführt. Diese Aktionen können eine Bewegungskorrektur sein, die an die Physic Engine weitergeleitet wird („set body force“), oder das Öffnen und Benutzen eines Notausgangs. Sollte die Physic Engine Kollisionen berechnet haben („collision event“) , wird diese Information im „inform pax about collision plan“ aufbereitet und an die beteiligten Pax weitergeleitet. Außerdem werden in jeder Runde die Bewegungsparameter für jeden Pax aktualisiert „send pax position plan“. Die Bewegungsparamter stammen aus der Physic Engine und werden über die Perzeption „pax position and velocity“ an den Plan übergeben. Der „send pax current vision plan“ wird vom „actualise vision request“ aktiviert. Daraufhin generiert der Environment Agent die aktuelle Sicht für den Pax und schickt sie als Antwort zurück an den Pax(„result vision request event“).

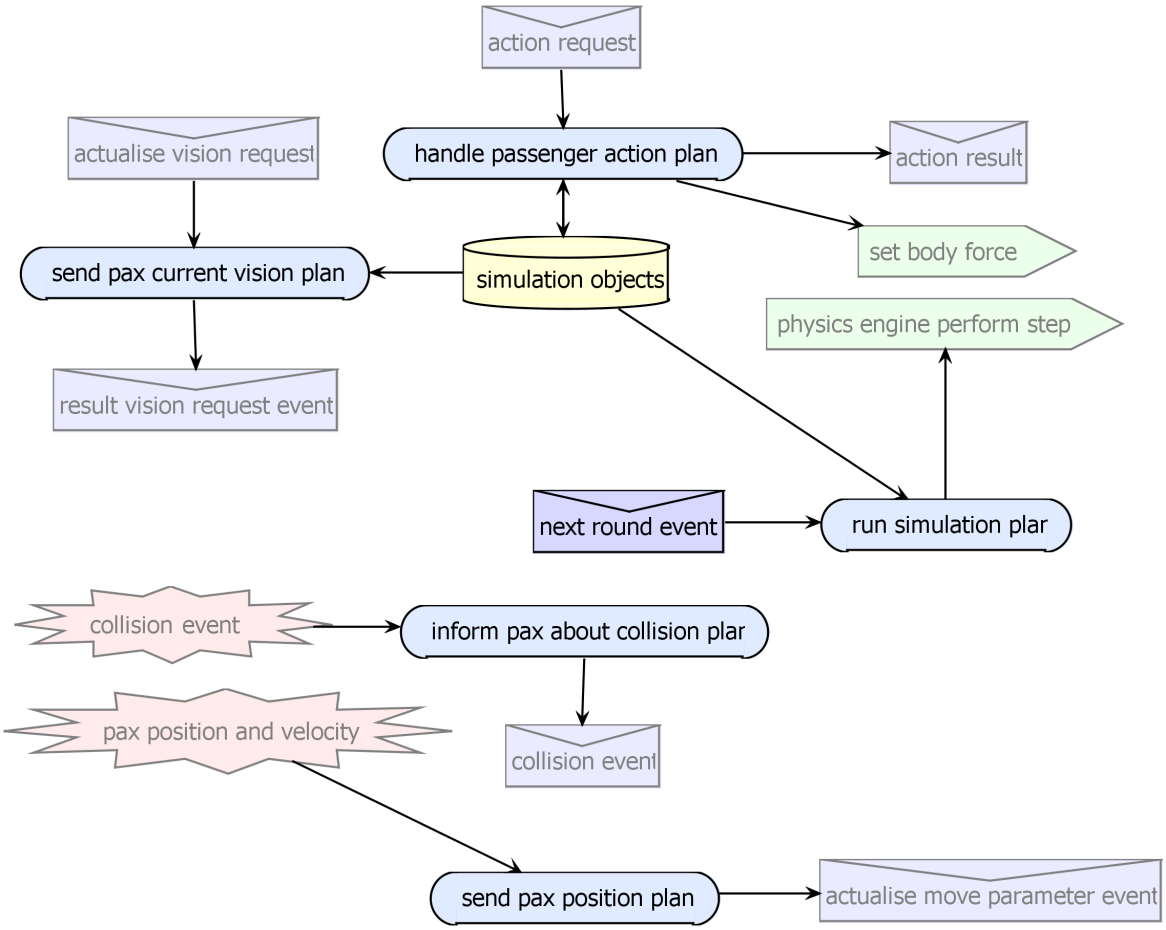

Abbildung 5.7.: Detailed Design Environment Agent

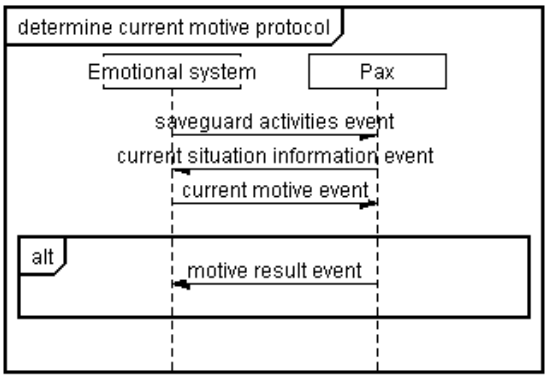

Abbildung 5.8.: Bestimmung des aktuellen Handlungsmotives

5.2.4. Design des Manager Agenten

Der Manager Agent ist sehr einfach aufgebaut und wird daher nur kurz beschrieben. Er besitzt ein Goal: „initialise simulation“. In dem zugehörigen Plan startet er alle Pax Agenten, den Environment Agenten und den Observer Agenten. Wenn alle Agenten initialisiert sind, sendet er das „start simulation event“ an alle Agenten und beendet sich danach selbst.

5.2.5. Design des Observer Agenten

Der Observer Agent wird an dieser Stelle ebenfalls nur kurz beschrieben. Der Observer Agent initialisiert die GUI und aktualisiert mit einer bestimmten Häufigkeit (z. B. 30 mal pro Sekunde) die Positionen und weitere Parameter der dynamischen Objekte in der Simulation und vermittelt sie an die GUI.

5.2.6. Kommunikation zwischen Pax Agent und Emotional System Agent

An dieser Stelle werden das zentrale Kommunikationsprotokoll „determine current motive protocol“ genau beschrieben. Die Nachrichten basieren auf den FIPA-Kommunikationsstandards[1] und die dargestellten AUML Diagramme sind mit dem PDT erstellt worden.

Die Abbildung 5.8 zeigt die sequenzielle Abfolge von Kommunikationsakten zwischen dem Pax und seinem Emotion System (vgl. Abbildung 5.1 (S.59)). Diese Abfolge ist nötig, um das

[1] http://www.fipa.org/repository/aclspecs.html

aktuelle Handlungsmotiv des Pax zu bestimmen. Das Emotionale System beginnt den Kommunikationsakt mit der Request Nachricht „saveguard activites event“. Diese Nachricht soll den Pax veranlassen, die aktuelle Handlung zu unterbrechen und seine Umgebung wahrzunehmen. Das Ergebnis dieser Wahrnehmung wird dann mit der „current situation information“ Nachricht zurück übermittelt. Nun analysiert das Emotionale System diese Informationen und reagiert mit entsprechenden Veränderungen der Requirements auf die aktuelle Situation. Daraus resultiert das aktuelle Motiv und wird mit der Perform-Nachricht “current motive“ zurück an den Pax gesendet. Daraufhin führt der Pax eine dem Motiv entsprechende Handlung aus. Wenn er die Handlung beendet hat, sendet der Pax ein Resultat („motive result“) zurück an das Emotional System. Findet ein Motivwechsel statt bevor die Handlung beendet werden konnte, wird das Resultat negativ sein.

Weitere Protokolle Alle hier nicht genannten Kommunikationsprotokolle bestehen aus einer Anfrage und einer zugehörigen Antwort (FIPA - Request). Sie können im Abschnitt 5.1 (S.58) nachvollzogen werden.

5.2.7. Ablauf der Simulation

Um eine Vorstellung von dem Ablauf der Simulation zu erhalten, wird als nächstes ein UML-Sequenz-Diagramm gezeigt. Es bildet einen typischen Ablauf einer Simulationsrunde und deren Kommunikation in zeitlicher Abfolge ab, siehe Abbildung 5.9. Als Akteure dienen hier das Emotionale System eines Pax, ein zweiter Pax, das Environment und das Physic Engine Modul.

Der Ablauf, der hier vorgestellt wird, betrifft eine häufige Situation in der Simulation. Pax1 versucht sich zu einer Position zu bewegen. Bevor er die Position erreichen kann, kollidiert er mit einem anderen Pax.

Das Sequenz Diagramm beginnt damit, dass das Emotionale System des Pax1 eine Aufforderung zum Sicherungsverhalten (safeguard activity) versendet. Der Pax aktualisiert die Sicht auf seine Umwelt (current view) und schickt Informationen über diese zurück an das Emotionale System. Das entspricht dem Verhalten des „determine current motive protocol“. Daraufhin bestimmt das Emotionale System den aktuellen Motivator und die Verhaltensparameter. In diesem Fall besteht das Bedürfnis nach Vorhersagbarkeit (certainty) der Umwelt in Verbindung mit einem hohen Aggressionspotential. Sind diese Informationen beim Pax angekommen, bestimmt er seine nächste Handlung und setzt eine Bewegungsaufforderung beim Environment ab. Ein Timer informiert das Environment darüber, dass die nächste Runde berechnet werden kann. Daraufhin wird die Berechnung des nächsten Schrittes bei der PhysicEngine aktiviert. Allerdings wird eine Kollision berechnet und der Pax1 darüber informiert. Der informiert sein Emotionales System darüber, dass er das Motiv nicht zum Abschluss

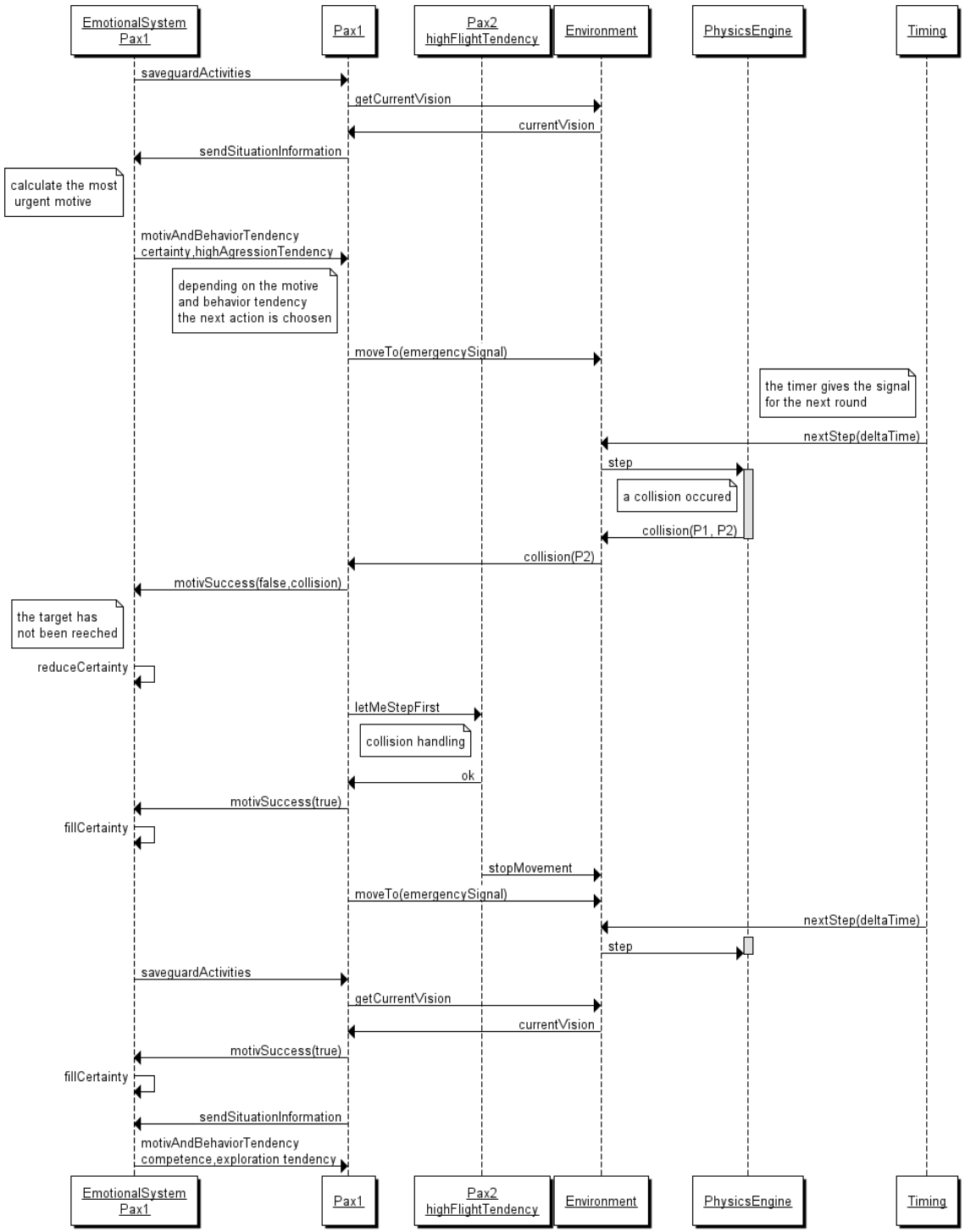

Abbildung 5.9.: Sequenzdiagramm einer Kollisionssitation

bringen konnte. Das Emotionale System reduziert das Certainty Requirement[2]. Da der Pax in einen aggressiven Zustand ist, lässt er dem anderen Pax2 keinen Vortritt, sondern drängelt sich selbst vor. Der andere Pax ist in einem ängstlichen Zustand und lässt den Pax1 ziehen. Daraufhin kann Pax1 einen Erfolg an sein Emotionales System melden und sein vorheriges Ziel wieder ansteuern. Hat er sein Ziel nach dem nächsten Simulationsschritt erreicht, wird ein weiterer Erfolg an das Emotionale System vermeldet. Außerdem wird wieder das regelmäßig auftretende Sicherungsverhalten ausgeführt und das Emotionale System über den Umweltzustand informiert. Daraufhin findet ein Motivwechsel zur Kompetenz statt.

[2] Damit einher geht auch immer eine Reduzierung der Kompetenz, was hier aus Gründen der Übersichtlichkeit nicht berücksichtigt wurde.

6. Ausgewählte Aspekte der Realisierung

Das Design wird in diesem Kapitel durch eine prototypische Implementierung überprüft. Es werden verschiedene Aspekte der Realisierung des finalen Prototypen vorgestellt. Das komplette System kann wegen seines Umfangs an dieser Stelle nicht in jedem Detail erklärt werden. Daher werden die wesentlichen nicht trivialen Teile der Realisierung dargestellt. Dazu gehören der Emotionale System Agent und die Umsetzung der PSI-Requirements und Motivauswahl, sowie die Bewegungssteuerung der Pax mit Hilfe der Physic Engine, und die Umsetzung der Darstellung eines Flugzeuges.

6.1. Implementation der PSI-Theorie

Die Implementierung der PSI-Theorie besteht aus Bedürfnissen(Requirements) und Motivatoren. Außerdem enthält die Umsetzung einen Auswahlmechanismus für den dringendsten Motivator und die Verhaltensparameter.

6.1.1. Umsetzung PSI-Theorie als Handlungssteuerung

Die Requirements der PSI-Theorie wurden als Java Objekte realisiert. Das UML-Klassen-Diagramm für das Emotionale System wird in der Abbildung 6.1 dargestellt.

Zu erklären bleibt bei der Abbildung wohl nur, dass jedem Requirement ein Motivator zugeordnet wird. Der konkrete Motivator „LogarithmMotivator“ implementiert das logarithmische Verhalten der Dringlichkeit des Motivators, siehe Abschnitt 2.3.2 (S.29).

Auswahl des aktiven Motivs Der Emotional Agent verfolgt die Maintain-Goals (Abschnitt 2.1.3), die verschiedenen Requirements gefüllt zu halten (siehe Design 5.2.2). Wenn nun

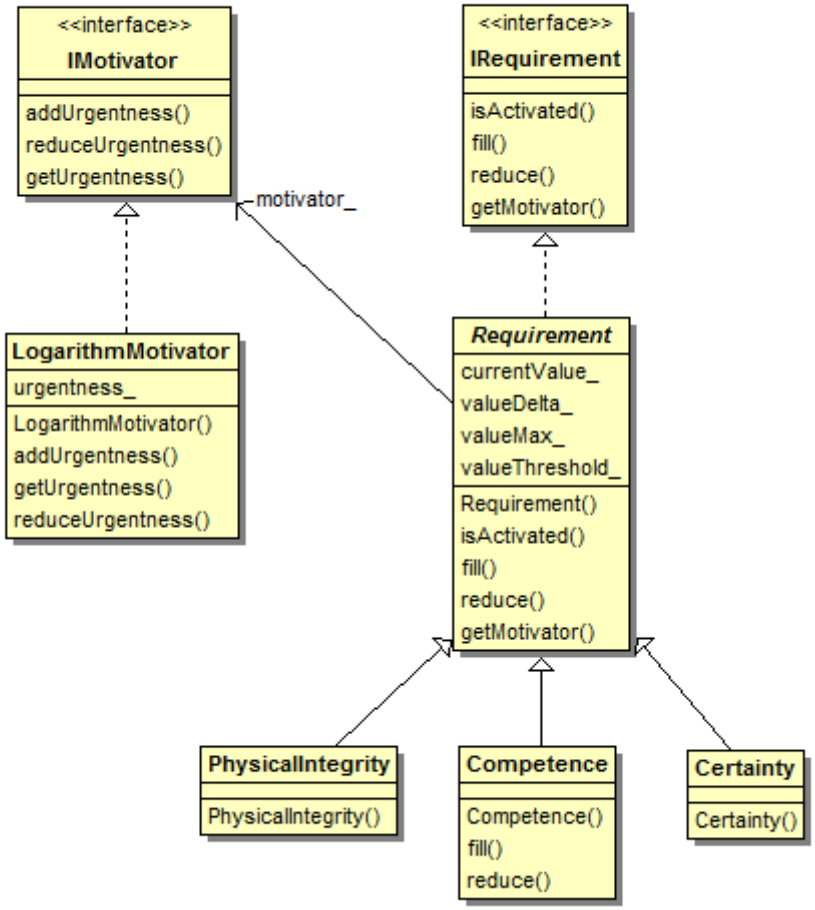

Abbildung 6.1.: Bedürfnis (Requirement) und Motivator des Emotionalen Systems

ein Bedürfnis oder mehrere Bedürfnisse ihren Schwellenwert unterschreiten, wird ein Meta-Goal[1] ausgelöst. Der Plan des Meta-Goals entscheidet, welches Motiv das Dringendste ist. Außerdem werden die aktuellen Verhaltenstendenzen berechnet. Das dringendste Motiv bestimmt, welcher Plan ausgeführt wird. Dieser Plan informiert den Pax Agent per Nachricht über das ausgewählte Motiv und die Verhaltenstendenzen. Der Pax legt nun seinen Fokus darauf, passend zum Motiv und den Verhaltensparametern, eine konsumatorische Endhandlung auszuführen, siehe Abschnitt 6.1.2.

Reduktion und Füllen der Bedürfnisse Die Reduktion und das Füllen von Requirements wurde ebenfalls mit Hilfe von Goals und Plänen realisiert. Hierbei ist es wichtig zu beachten, wie die verschiedenen Bedürfnisse sich gegenseitig beeinflussen; z. B. bedeutet jede Zunahme oder Reduzierung eines Requirements auch eine Zunahme oder eine Reduzierung der Kompetenz. Die Goals „reduceRequirement“ und „fillRequirement“ werden von einem Pax Agenten direkt über eine FIPA Request Nachricht angefragt am Emotionalen System Agenten. Somit kann der Pax die Requirements von außen beeinflussen. Anders als im Design beschrieben, muss deshalb im Emotional System Agenten kein Goal zur emotionalen Bewertung der aktuellen Situation eingefügt werden. Diese Abweichung vom Design ist bewusst gewählt, um den Implementierungsaufwand für den finalen Prototypen zu reduzieren. Die Abweichung geht allerdings mit einer engeren Kopplung einher, da der Pax Agent jetzt die Struktur der Requirements kennen muss.

Das konkrete Füllen und Reduzieren löst der Pax im finalen Prototypen bei bestimmten Situationen aus; z. B. gibt es eine Reduzierung der Bestimmtheit(Certainty), wenn eine Kollision aufgetreten ist. Auch wenn eine angestrebte Handlung unterbrochen oder fehlgeschlagen ist, wird die Bestimmtheit reduziert. Im Gegenzug wird bei jeder Handlung, die erfolgreich abgeschlossen wurde, die Bestimmtheit erhöht, z. B. das erfolgreiche Erreichen eines angestrebten Ortes. Bei Kontakt mit Feuer wird die Erfüllung des Bedürfnisses nach Schmerzvermeidung (Physical Integrity) massiv reduziert. Zu erwähnen ist an dieser Stelle noch, dass Schmerzvermeidung automatisch gefüllt wird, wenn keine Schmerzen vorhanden sind. Dafür ist der „reduceConstantlyPainPlan“ im Emotional System Agent zuständig.

6.1.2. Vom Motivator zur Handlung

Der Pax empfängt die Nachricht vom Emotional System Agent, welcher Motivator die Handlungssteuerung im Moment übernimmt. Anhand der übermittelten Verhaltenstendenzen und

[1] Meta-Goals werden genutzt, um z. B. zu entscheiden, welcher von mehreren verfügbaren Plänen zu einem Goal ausgeführt werden soll. Eine genaue Beschreibung findet man unter http://vsis-www.informatik.uni-hamburg.de/projects/jadex/jadex-0.94x/userguide/goals.html

dem Motiv kann die nächste Handlung ausgeführt werden. Hier ist die Abbildung 2.10 der PSI-Verhaltensregulierung für die Umsetzung heranzuziehen. Der folgende Pseudocode Algorithm 2, macht deutlich, wie aus dem Motiv und den Verhaltenstendenzen eine Handlung entsteht.

Algorithm 2 Plan zur Bestimmung der nächsten Handlung

Require: Eine Nachricht ist eingetroffen, die das Motiv und die Handlungstendenzen enthält. Die Handlungstendenzen sind Aggressivität, Flucht, Erkunden und Aktiviertheit

```
1: if Motiv == Physical Integrity then
2:     if Flucht > Aggressivität then
3:         Das nächste Ziel ist die Flucht vor der Schmerzquelle
4:     else
5:         Das nächste Ziel ist der Kampf gegen die Schmerzquelle; z. B. andere Pax
6:     end if
7: end if
8: if Motiv == Certainty then
9:     if Erkunden stärker als Flucht und Aggression then
10:        Das nächste Ziel ist die Erkundung der Umgebung
11:    else
12:        if Aggression > Flucht then
13:            Zeige Aggressives Verhalten
14:        else
15:            Rückzug zu einem sicheren Ort
16:        end if
17:    end if
18: end if
19: if Motiv == Competence then
20:    if Aktiviert sehr hoch then
21:        Das nächste Ziel ist die Erkundung der Umgebung
22:    else
23:        Zeige ängstliches Verhalten
24:    end if
25: end if
```

Der Pseudocode zeigt, wie mit Hilfe der Verhaltensparameter und Motive unterschiedliches Verhalten umgesetzt werden kann. Man sieht, dass bereits viele unterschiedliche Verhaltensweisen entstehen. Die konkrete Ausführung der einzelnen Verhaltensweisen wird dann in Bezug auf die aktuelle Situation geschehen. Im Rahmen dieser Arbeit ist es nicht möglich, alle Verhaltensweisen zu programmieren. Daher wurden im finalen Prototypen nur die Verhaltensweisen Flucht und Erkunden implementiert.

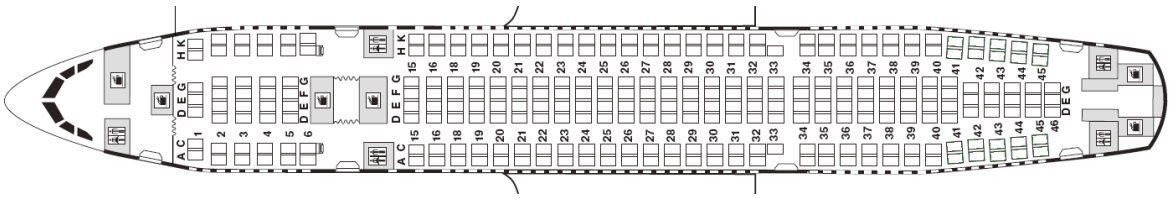

Abbildung 6.2.: Passagierflugzeug - Innenraum

6.2. Umsetzung der grafische Darstellung eines Flugzeugmodells

Generierung eines Flugzeugmodells Die grafische Darstellung eines Flugzeugmodells wurde mit Hilfe der folgenden Abbildung 6.2 eines großen Passagierflugzeugs gemacht. Hierfür wurden die Koordinaten der Außenwände, Innenräume, Notausgänge und Sitze als Wavefront Standard 3D Object Format (OBJ)-Format in eine Datei gespeichert[2]. Mit Hilfe der erstellen Dateien und einem freien OBJ-Parser konnten die Koordinaten in Objekte umgewandelt werden. Diese Objekte beinhalten die grafische Komponente zur Darstellung der statischen Flugzeugkomponenten und deren Positionen.

Initialisierung der Darstellung Beim Start des Observer Agenten werden die OBJ-Objekte also ausgelesen und die grafische Darstellung initialisiert. Des weiteren wird die Darstellung der dynamischen Objekte, wie Pax oder Feuer, während der Initialisierung des Observers festgelegt. Nach diesem Vorgang ist dem Observer bekannt, wie die Objekte gezeichnet werden sollen. An welcher Stelle ein Objekt angezeigt werden soll, kann aber erst ermittelt werden, wenn dem zugehörigen Simulationsobjekt eine Position zugewiesen wurde. Die Verbindung der Darstellung und der Simulationsobjekte geschieht über einen eindeutigen Bezeichner (z. B. „seat1“ oder „seat2“). Diese Verbindung wird im Initialisierungsplan des Environment Agenten hergestellt. Wenn die Simulationsobjekte erstellt werden, muss der entsprechende Bezeichner angegeben werden und außerdem eine Position festgelegt werden. Der Environment Agent hat direkten Zugriff auf die Simulationsengine[3] und kann dadurch die Objekte direkt mit Hilfe der Simulationsengine initialisieren.

Initialisierung der Pax Wenn der Pax Agent initialisiert wird, befragt er als erstes den Environment Agenten nach einem freien Sitzplatz („get_start_position“ Goal). Nachdem er

[2] Die Koordinaten wurden per Hand aus dem Bild ausgelesen. Schöner wäre hier der direkte Zugriff auf Konstruktionsdaten für spätere Versionen

[3] Die SimEnvironment Capability von Jadex V.2 Beta2 wird an dieser Stelle genutzt.

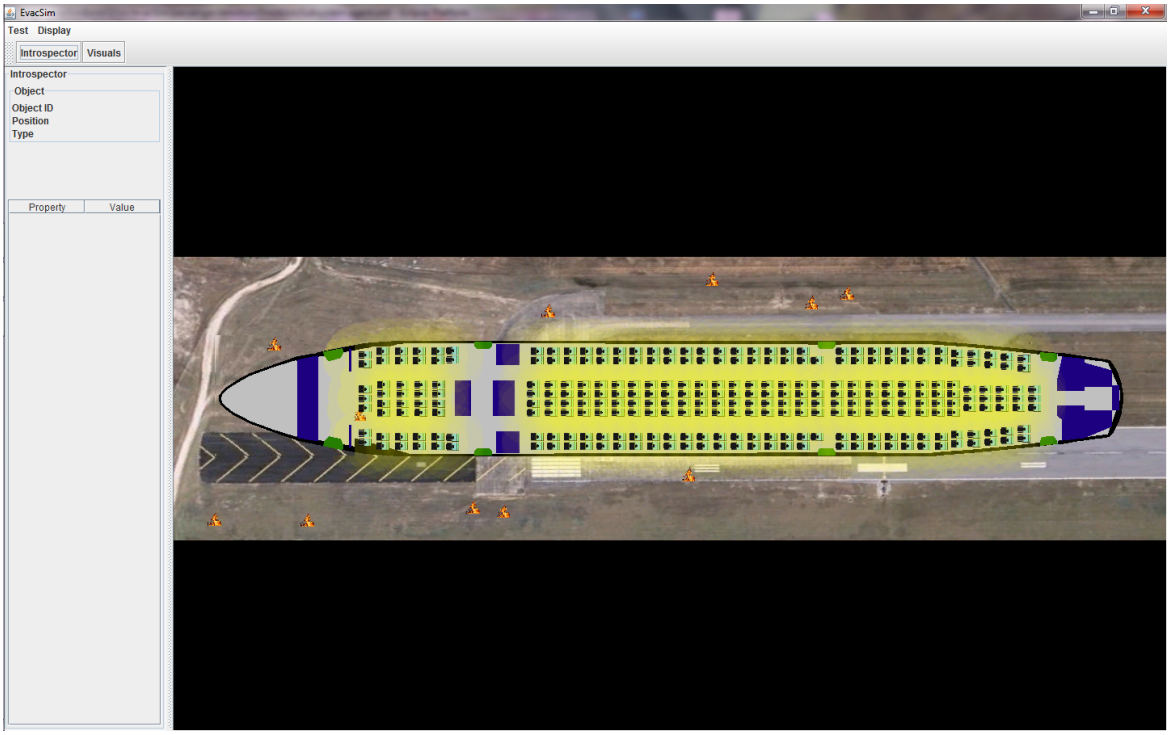

Abbildung 6.3.: Grafische Darstellung des Flugzeuginnenraumes

die Antwort erhalten hat, also eine Startposition bekannt ist, initialisiert er sein zugehöriges Simulationsobjekt[4].

Darstellung Die Darstellung der soeben gestarteten Evakuierungssimulation ist auf der Abbildung 6.3 zu sehen. Neben den Pax, den Umrissen des Flugzeugs und den Innenräumen werden die Notausgänge in Grün dargestellt. Zufällig verteilte Feuer befinden sich ebenfalls auf der Darstellung. Der Sichtbereich der Pax ist die blassgelbe Zone, die jeden Pax kreisförmig umgibt.

6.3. Steuerung der Physic Engine durch das Environment

Die Simulationsumgebung von Jadex bietet ein Interface für Vorgänge an, die zu jeder Runde der Simulation ausgeführt werden sollen (siehe Jadex Simulations Engine 4.2). Dieses Interface wird durch den „PhysicProcess“ implementiert, um die Bewegung der Pax mit Hilfe der

[4]Die SimAgent Capability von Jadex V.2 Beta2 wird an dieser Stelle genutzt. „sim_create_object“Goal

Physic Engine umzusetzen. Der folgende Pseudocode algorithm 3 stellt die Funktionsweise dar:

Algorithm 3 PhysicProcess.execute(DeltaTime)

Require: In der Physic Engine existiert ein Object, für jeden Pax

```
for Jeden Pax do
    Berechne die Kraft die durch die Bewegungsziele des Pax ausgelöst wurden
    Aktualisiere die Kraft, die auf das Physik Objekt des Pax wirkt
end for
Physic Engine berechne einen Schritt in Abhängigkeit von DeltaTime
Während der Berechnung aufgetretene Kollisionen werden gespeichert
for Jeden Pax do
    Aktualisiere die Position und Beschleunigung des Pax-SimObjects
end for
if Kollisionen aufgetreten then
    for Jede Kollision do
        Benachrichtige beteiligten Pax über Kollision
    end for
end if
```

Es wird also in jeder Simulationsrunde eine Synchronisation der Pax mit der Physic Engine in Bezug auf die Beschleunigung und Position ausgeführt. Zusätzlich werden die von der Physic Engine berechneten Kollisionen aufgefangen und gespeichert. Das geschieht über einen „MyCollisionEventListener“ und einen Proxy. Der „CollisionEventProxy“ bereitet die „CollisionEvents“ der PhysicEngine auf und wandelt sie in ein „MyCollisionEvent“-Objekte um. Der Proxy wurde eingeführt, um eine lose Kopplung zu erreichen, indem die internen Strukturen der PhysicEngine vor dem Rest der Anwendung versteckt werden.

6.4. Umsetzung der Bewegung der Pax

In diesem Abschnitt wird die Umsetzung der Bewegung der Pax dargestellt.

addIndexEntry(Bewegungsablauf) In Abhängigkeit des momentanen Motivs wird ein Bewegungsziel generiert. Das Goal „MoveToDestination“ wird genutzt, um ein seek und arrive Verhalten (siehe MovementBehaviors 4.6) zu einer bestimmten Position zu generieren. Der „MoveToDestinationPlan“ initialisiert einen Task (Task wurden in 4.2 erklärt) „SeekAndArriveTask“. Der Task beschleunigt den Pax und prüft in jeder Simulationsrunde, ob das

angesteuerte Ziel in der näheren Umgebung ist. Die nähere Umgebung wird durch die Geschwindigkeit des Pax bestimmt. Je schneller der Pax, umso weiterreichender ist die nähere Umgebung. Befindet sich das Ziel in diesem Bereich, wird die Beschleunigung je nach Abstand zum Ziel reduziert (asymptotische Annäherung). Bei der Unterschreitung einer Toleranz zum Zielpunkt wird die Geschwindigkeit auf 0 gesetzt und ein „TargetReachedEvent“ ausgelöst, das wieder vom MoveToPositionPlan erwartet wird, und als erfolgreiche Umsetzung des Plans gewertet werden kann. Sollte während dieses Vorgangs eine Kollision aufgetreten sein, wird der Vorgang unterbrochen und der Pax direkt gestoppt. Danach kann eine neue Bewegungsrichtung bestimmt werden.

Bewegungsziel und Verhaltensweisen Das Explore Goal sorgt dafür, dass das nächste Bewegungsziel entweder zufällig oder gezielt bestimmt wird. Zufällig wird das Ziel bestimmt, wenn keine Informationen über den nächsten Notausgang vorliegen oder wenn sich kein Richtungssignal im Sichtradius befindet. Gezielt kann das Ziel bestimmt werden, wenn der Pax ein Richtungssignal „sieht“. Dann wird sich der Pax direkt zu diesem Signal bewegen, um die Information vom Signal zu erhalten, in welcher Richtung der nächste Ausgang liegt. Ist diese Information vorhanden, wird der sich der Pax in diese Richtung bewegen.

Das zweite umgesetzte Verhalten ist das „Flight“ Verhalten. Ist Physical Integrity das aktuelle Motiv, wird in eine Richtung gelaufen, die vom Schmerzverursacher wegführt. Ist die Furcht aus einem anderen Motivator entstanden, wird sich zufällig bewegt, was eine Art Panik darstellt.

Wichtig in Bezug auf die Verhaltensmodulation ist noch, dass bei jeder erfolgreichen Zielansteuerung („TargetReachedEvent“) die Certainty gesteigert wird, und bei jedem Fehlversuch (z. B. CollisionEvent) die Certainty reduziert wird.

6.5. Zusammenfassung

In diesem Kapitel wurde gezeigt, wie die emotionale Komponente umgesetzt wurde. Von besonderer Bedeutung ist das Meta Goal zur Motivbestimmung im Emotional System Agent. Des weiteren wurde beschrieben, wie die grafische Umsetzung der Simulation erfolgte. Das Flugzeug wird als OBJ-Format eingelesen und mit Hilfe der Observer Modularisierungskonzept von Jadex und anderen Agentenplattformen (Capability) die grafische Darstellung initialisiert. Daraufhin wurde dargestellt, wie die Synchronisation der Physik Engine mit dem Environment umgesetzt wurde. Als letzter Punkt wurde die Umsetzung der Bewegungen der Pax mit Hilfe der Steering Behavior „seek“ und „arrive“ erläutert. Außerdem wurde die Wegfindung mit Hilfe der Explore- und Flight-Verhaltensweisen dargestellt.

7. Test und Bewertung

Das strukturierte Testen für Multiagentensysteme (MAS) ist momentan Gegenstand der Forschung und ein Standard ist noch nicht festgelegt.

Testen in Agentensystemen ist wegen der Autonomie der Agenten ein kompliziertes Unterfangen. Das hauptsächliche Testen, während der Entwicklung, lief auf das ständige Starten der Simulation und das Betrachten der Veränderungen im Ablauf der Simulation hinaus. Dieses Verfahren ist sehr zeitaufwändig, da gewartet werden muss, bis der zu testende Zustand eingetroffen ist. Die Vorgehensweise eines solchen Testablaufs wird im Abschnitt 7.1 beispielhaft beschrieben.

Auch ist die Umsetzung von Testfällen recht kompliziert. Abstrakte Verhaltensweisen, wie z. B. *verbrenne Dich am Feuer und bekomme das Bedürfnis der Schmerzvermeidung und fliehe daher für einen gewissen Zeitraum,* sind schwer in ein Testprogramm zu fassen. Für solche Testfälle ist wohl die visuelle Überprüfung des Geschehens die einfachste Möglichkeit.

Anschließend folgt die Auseinandersetzung mit den im Kapitel 3 (S.37) aufgestellten fachlichen und technischen Anforderungen und den tatsächlich umgesetzten Funktionalitäten. Bewertet werden wird zum einen das aufgestellte Design und zum anderen der finale Prototyp (Abschnitt 7.3). Die Nutzung der BDI-Funktionen und des Prometheus Vorgehensmodell werden analysiert. Schlussendlich wird das PSI-Modell hinsichtlich des Nutzens für die Simulation von Menschen in Entfluchtungssituationen beurteilt (Abschnitt 7.6).

7.1. Testvorgehen während der Umsetzung einer Anforderung

Am Beispiel der Anforderung *Reaktion auf Kollision* 3.1 wird ein typischer Testablauf beschrieben.

- Als erstes wird, anhand des Designs, in der ADF eine Nachricht implementiert (simulation_event) . Diese Nachricht ist der Trigger für einen Plan (siehe Ausschnitt aus der

```
<plan name="collision_event_plan">
   <body class="CollisionEventPlan" />
   <trigger>
       <internalevent ref="simulation_event">
          <match>$event.getParameter("type").getValue().equals("collision")
          </match>
       </internalevent>
   </trigger>
</plan
```

Abbildung 7.1.: Ausschnitt aus der ADF zur Definition des „collision_event_plan

ADF 7.1). Der Plan, wenn aktiviert, löst den zugehörigen Java „CollisionEventPlan“ aus, der nur eine Nachricht auf der Konsole ausgibt.

- Der zugehörige Plan beinhaltet erstmal nur eine Zeile Code:

  ```
  getLogger().fine("'CollisionEventPlan successfully startet"');
  ```

- Nun wird die Jadex-Umgebung[1] gestartet. Als erstes ist wichtig, dass die manipulierte ADF von Jadex fehlerfrei eingelesen werden kann. Wenn nicht, muss die ADF überprüft werden.
- Dann wird die Simulation mit wenigen Pax gestartet und solange gewartet, bis eine Kollision offensichtlich aufgetreten ist.
- Wenn nun auf der Konsole, die im Plan definierte Ausgabe erfolgt ist, kann der Test als erfolgreich angesehen werden und im nächsten Schritt der Plan mit Funktionalität gefüllt werden. Wichtig hierbei ist, dass diese Tests regelmäßig stattfinden damit evtl. Fehlerquellen frühzeitig erkannt werden.
- Fand keine Ausgabe statt, muss die Fehlerquelle gesucht und der Test erneut durchgeführt werden, bis der Fehler behoben wurde.
- Die sukzessive Implementierung der Anforderung ist erfolgreich, wenn die benötigten Funktionen der Anforderung erfüllt werden.

Um das Testen der Kollisionen zu vereinfachen wurde im finalen Prototypen zusätzlich eine grafische Darstellung der Kollisionen implementiert, siehe kleiner roter Punkt auf der Abbildung 7.1.

[1] Um Jadex zu starten, wird die Klasse „StandalonePlatform“ ausgeführt.

Abbildung 7.2.: Kollision vor dem Ausgang

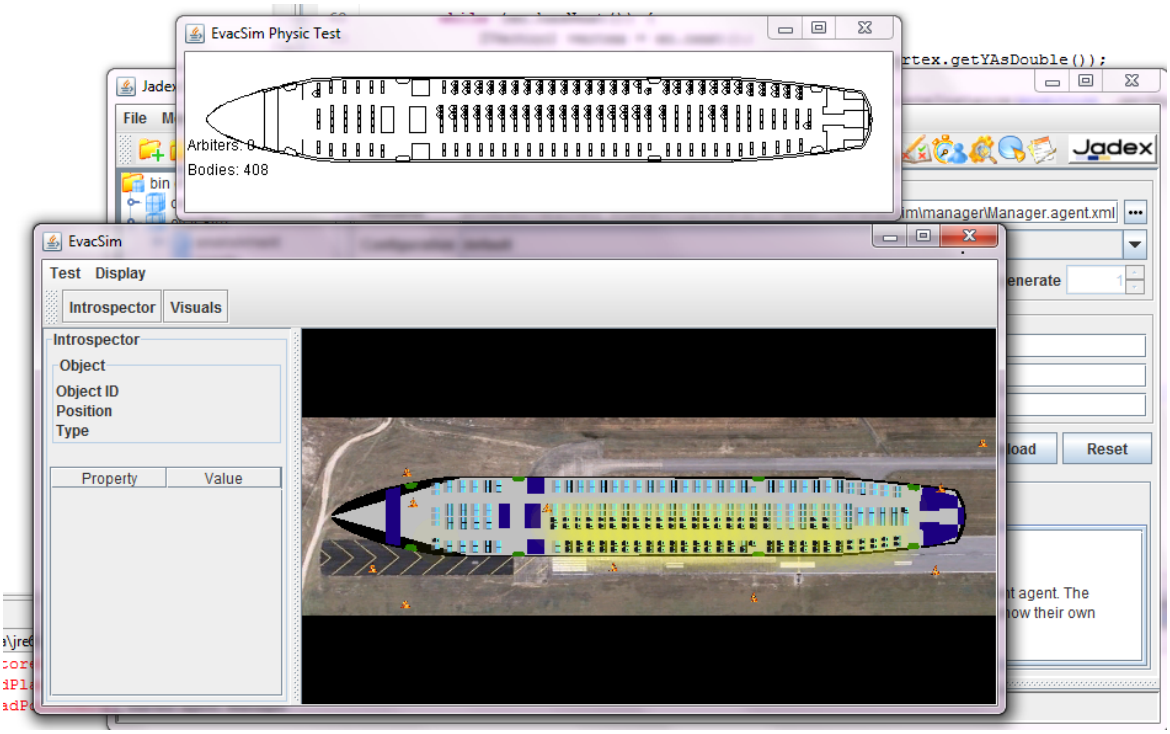

Abbildung 7.3.: Test der Physic Engine

7.2. Weitere Test während der Entwicklung

Eine Vielzahl von weiteren Tests wurde während der Entwicklung vorgenommen. Die Funktionalität der Objektorientierten (OO) Anteile der Arbeit wurde mit JUnit[2]-Tests teilweise abgedeckt; z. B. wurden Junit-Test für den OBJ-Import6.2 geschrieben. Damit wurde aufgedeckt, dass an dieser Stelle vermehrt eine falsche Anzahl von Objekten generiert wurde und der Fehler konnte lokalisiert und verbessert werden

Um die Synchronität der Physic Engine mit der Simulationsumgebung zu überprüfen, wurde eine grafische Test Ausgabe der Physic Engine implementiert (siehe 7.3).

[2]Junit ist die Standardmäßig genutzte Testumgebung für Java www.junit.org.

Interessant an der Abbildung ist die Umkehrung von Oben und Unten in der Darstellung der Physic Engine, im Gegensatz zur Simulationsdarstellung. Dieser Effekt hat jedoch keinen Einfluss auf die Bewegung der Pax. Betrachtet man die Nase des Flugzeuges, bemerkt man, dass bei der Simulationsdarstellung, die konkave Innenseite der Nase nicht richtig dargestellt wird. Anhand der Physic Engine sieht man jedoch, dass die Elemente richtig modelliert wurden. Der Fehler liegt daran, dass OpenGl ohne weiteres keine konkaven Polygone darstellen kann[3].

7.3. Bewertung der Umsetzung der Anforderungen

Im Folgenden wird die Umsetzung der aufgestellten fachlichen Anforderungen im Kapitel 3 (S.37) in Bezug auf das vorgestellte Design und in Bezug auf den entwickelten finalen Prototypen bewertet.

7.3.1. Fachliche Anforderung des emotional gesteuerten Fluchtverhaltens

Die größte Herausforderung bei den aufgestellten Anforderungen war die Umsetzung, des emotional gesteuerten Fluchtverhaltens.

Design Hier konnte eine Architektur entworfen werden, die die emotionale Steuerung des Pax in einen eigenständigen Agenten legt. Der Emotionale Agent steuert das Verhalten des Pax auf Basis der PSI-Bedürfnisse (5.1). Der Pax Agent dient in diesem Design als Schnittstelle zur Außenwelt für den Emotionalen Agenten. Er beinhaltet die Wahrnehmung und Vorverarbeitung der Umweltsituation für den Emotionalen Agenten. Zusätzlich verfügt er über die Methoden, um mit der Umwelt zu agieren. Man könnte sagen:“Der Pax Agent ist der Körper und der emotionale Agent sein Gehirn“. Dieser Vergleich trifft die gewählte Architektur wohl am treffensten. Dieses Design ermöglicht emotional gesteuertes Fluchtverhalten und setzt somit die Anforderung erfolgreich um. Außerdem ist durch die Kopplung über ein Kommunikationsprotokoll eine klare Schnittstelle vorhanden, die einen Austausch des Emotionalen Agenten ermöglicht, wenn z. B. ein Agent entwickelt werden soll, der die OCC-Theorie umsetzt. Diese Umsetzung würde auch einen interessanten Vergleich der beiden Theorien ermöglichen.

[3] Jedes konkave Polygon kann durch Teslation in mehrere konvexe Polygone umgerechnet werden. Konvexe Polygone sind bei der Schnittpunktberechnung wesentlich performanter.

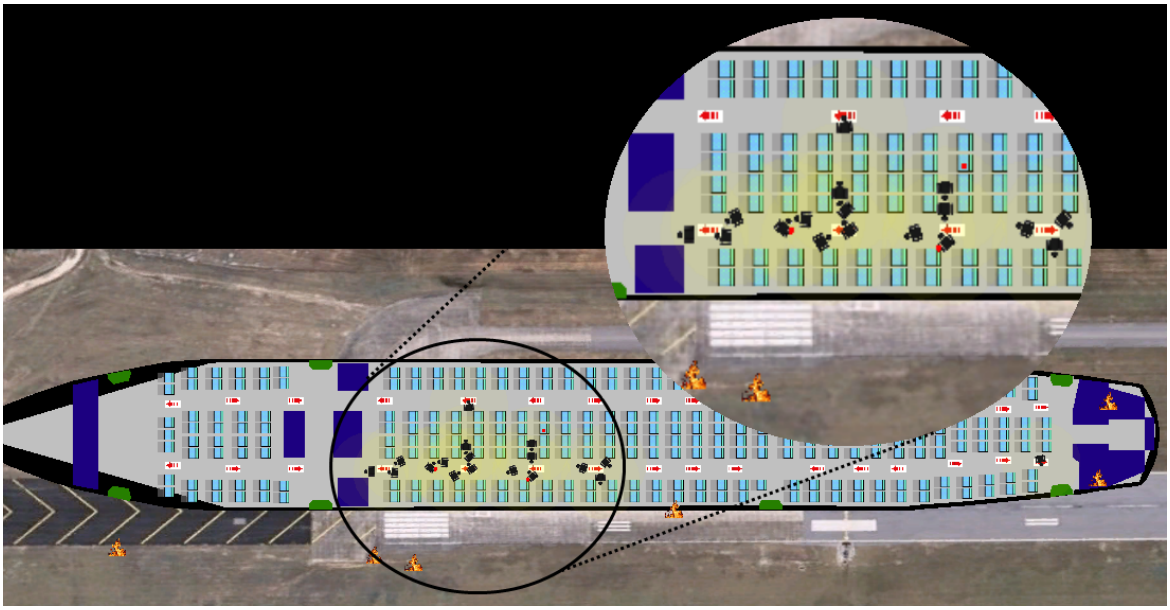

Abbildung 7.4.: Fluchtverhalten der Pax

Prototyp Im finalen Prototypen wurde das vorgeschlagenen Design grundsätzlich umgesetzt. Eine Abweichung ist, dass der Pax Agent direkt die emotionalen Bedürfnisse des Emotionalen Agenten über Nachrichten manipulieren kann.

Im Pax Agent wurden die Verhaltensweisen Flucht(„flee“) vor Gefahren und Erkunden(„explore“) der Umgebung erfolgreich umgesetzt.

Die Abbildung 7.4 zeigt einen Simulationslauf mit 20 Pax, die sich in die Richtung der Ausgänge bewegen und Kollisionssituation sind zu erkennen. Das gesamt Verhalten der Pax ist aber noch sehr einfach, da es noch viele Punkte gibt, an denen noch mehr Aufwand investiert werden müsste.

Bei der Umsetzung der PSI-Theorie, war es besonders schwierig, Ereignissen, die dem Pax widerfahren sind, die richtige Gewichtung in Bezug auf die Modulation der Bedürfnisse zu geben, so dass der Pax vernünftig zwischen den Verhaltensweisen wechselt. Um hier ein besseres Ergebnis zu erreichen, müsste wesentlich mehr Aufwand und Zeit in die Kalibrierung der Bedürfnismodulationen gesteckt werden.

Des weiteren wurden nicht alle Verhaltensweisen implementiert, die durch das entwickelte emotionale System hätten ausgelöst werden können. Die Entwicklung des Emotionalen Agenten und des Pax Agenten hätte daher paralleler betrieben werden müssen.

Diese Anforderung wurde durch den Prototypen insofern umgesetzt, dass gezeigt wurde, dass der Wechsel zwischen Verhaltensweisen mit Hilfe der PSI-Theorie möglich ist, und so-

mit ein Emotional gesteuertes Verhalten erzeugt wird. Die Tendenz auch komplexeres Verhalten, durch die emotionale Steuerung zu erzeugen ist erkennbar und mit einem zeitlichen Mehraufwand auch erreichbar.

7.3.2. Fachliche Anforderung der Navigation im Evakuierungsgegenstand

Die Umsetzung der Navigation wurde in der Analyse, siehe 4.6 (S.53) mittels der emotionalen Verhaltenssteuerung und Steering Behavior festgelegt. Hierbei wurden die Behavior Seek/Arrive/Flee, Obstacle Avoidance und Distance als benötigt eingestuft. Außerdem wurde erkannt, dass Richtungssignale in die Umgebung aufgenommen werden sollen, an denen sich der Pax orientieren kann und ihm die Richtung zum Ausgang bekannt ist.

Design Das Design legte die Umsetzung der Steering Behavior durch eine Capability im Pax Agenten fest, siehe 5.2.1. Diese Capability sollte die Steering Behavior kapseln und über Aktionsnachrichten die Beschleunigungsveränderungen an das Environment kommunizieren. Im Design wurden die Capability nicht im Detail erläutert. Daher bietet das Design eine große Freiheit bezüglich der Umsetzung, um auch andere Wegfindungs Capabilities einbinden zu können. Dieses Design ermöglicht die Navigation im Simulationsgegenstand, macht aber keine detaillierten Angaben darüber, wie es verwirklicht werden soll.

Prototyp Das „Explore“-Goal im Prototypen ist für die Umsetzung der Wegfindung zum Ausgang gedacht. Dieses Goal löst die Suche nach Richtungssignalen, die zum Notausgang zeigen, aus. Wenn eine Richtung zum Ausgang bekannt ist, wird das Seek Steering Behavior aktiv. Im Prototypen wurde die Umsetzung der Steering Behavior auf Seek/Arrive/Flee beschränkt. Damit läuft der Pax zwar oft gegen Hindernisse, korrigiert dann aber die Bewegungsrichtung (Abschnitt 7.3.3). Somit ist die Anforderung der Navigation erfüllt, könnte aber noch verbessert werden, indem mehr Verhaltensweisen implementiert und die restlichen Steering Behavior umgesetzt werden.

7.3.3. Fachliche Anforderung der sinnvollen Reaktion auf Kollisionen

Der Pax soll auf Kollisionen mit Gegenständen und mit anderen Pax sinnvoll reagieren.

Design Das Design ermöglicht die Behandlung von Kollisionen. Es soll mittels Nachrichten zwischen den Pax eine Konfliktlösung für Kollisionen erarbeitet werden.

Fachliche Anforderung	Design	Finaler Prototyp
Navigation in der Simulationsumgebung	möglich	umgesetzt
Sinnvolle Reaktion auf Kollisionen	möglich	umgesetzt
Emotional gesteuertes Fluchtverhalten	möglich	umgesetzt
Simulation von Gefahren	möglich	feuer umgesetzt
Verschiedene Emotionstypen	möglich	nicht umgesetzt
Flugzeugcrew organisiert Evakuierung	möglich	nicht umgesetzt
Verschiedene Anfangskonfigurationen	möglich	Anzahl der Pax, Anzahl der Feuer

Tabelle 7.1.: Bewertung der Umsetzung der fachliche Anforderungen

Prototyp Im Prototypen wurde die vorgeschlagenen Architektur umgesetzt und die Reaktion auf Kollisionen findet statt. Der Pax wird für einen kurzen Moment angehalten und ein neues Bewegungsziel berechnet.

Die Konfliktlösung von Kollisionen unter Pax ist mitunter kompliziert, z. B. ist eine Kollision die in den Rücken des anderen Pax geht, nicht unbedingt durch solch einen Algorithmus zu lösen, der einem anderen Pax den Vortritt lässt. Eine Anpassung der Bewegungsgeschwindigkeit des hinteren Pax wäre angemessener. Dieses Verhalten könnte ein durch ein weiteres Steering Behavior umgesetzt werden. Außerdem sollten die Konfliktlösung bereits initiiert werden, bevor eine Kollision unter den Pax stattfindet. In Anbetracht der zeitlichen Begrenztheit der Arbeit wurde auf die komplexe Umsetzung verzichtet.

7.3.4. Fachliche Anforderungen - Conditional und Optional

Die konditionalen und optionalen fachlichen Anforderungen wurden, im Design (Kapitel 5 (S.58)) vollständig berücksichtigt. Der finale Prototyp setzt die Simulation von Gefahren in Form eines Feuers um.

7.3.5. Zusammenfassung der Bewertung des Designs und des Prototypen

Die Tabelle zeigt eine Auflistung der fachlichen Anforderungen und die Bewertung bezüglich der Umsetzung für das Design und den finalen Prototypen.

Technische Anforderung	Finaler Prototyp
Multiagentenplattform	einwandfrei
Kommunikation zwischen den Agenten	einwandfrei
Grafische Aufbereitung des Simulationsgeschehens	einwandfrei
Physik Berechnungen	einwandfrei, bei vielen Objekten langsam
Simulationsumgebung	kleine Probleme

Tabelle 7.2.: Bewertung der Umsetzung der technische Anforderungen

7.3.6. Bewertung der Technischen Anforderungen

Die Technischen Anforderungen wurden im finalen Prototypen alle umgesetzt. Kleinere Schwierigkeiten gab es bei der Verwendung der Simulationsumgebung der Jadex Version2 Beta2. Hier war die Festlegung von komplexen Bedingungen für Goals sehr unübersichtlich und die Dokumentation schlecht. Das Problem wurde durch Verwendung anderer Goaltypen aber umgangen. Außerdem waren kleinere Darstellungsfehler zu verzeichnen. So verschieben sich die Pax, wenn die Simulation mit Java2D läuft, um ein Stück nach unten.

Die Tabelle 7.2 zeigt nochmal eine Zusammenfassung der technischen Anforderungen.

7.4. Bewertung der Nutzung von Prometheus als Vorgehensmodell

Das Vorgehensmodell von Prometheus ist in allen Bereichen der Entwicklung erfolgreich genutzt worden. Es hat sich als wertvolles Hilfsmittel erwiesen, um einen Überblick über das System zu erhalten und es zu beschreiben. Leider konnte die Fülle an generierten Diagrammen nicht vollständig in den Designteil dieser Arbeit integriert werden. Das hätte den Rahmen gesprengt. Ein Manko des PDT ist, das beim detaillierten Design der Agenten die visuelle Verbindung der Plans mit den Goals fehlt. Semantisch kann die Verbindung hergestellt werden, wird aber optisch nicht dargestellt. Ein weiteres Manko sind die regelmäßigen Abstürze des Prometheus Design Tool, sowie etwas fehlerhafte Generierung von Bildern der Diagramme. Interessant bei dem Vorgehen von Prometheus ist, dass der ständige Wechsel zwischen den verschiedenen Phasen der Entwicklung, siehe Abschnitt 2.1.4, zu einem stetigen Qualitätsfortschritt der gesamten Architektur führt. Eine Änderung in einem Diagramm führt durch dieses Wechsel oft zu verschiedensten Änderungen in anderen Diagrammen. Hierbei muss darauf geachtet werden, dass die Konsistenz des Gesamtsystems erhalten bleibt. Zusammenfassend scheint Prometheus, trotz kleinerer Mängel, ein geeignetes Vorgehensmodell für die Entwicklung einer Evakuierungssimulation mit Jadex zu sein.

7.5. Bewertung der Umsetzung von BDI-Funktionen

Es wurden viele der BDI-Fähigkeiten von Jadex genutzt, wie verschiedene Goaltypen und sogar ein Meta-Goal zur Bestimmung des aktuellen Motivs, sowie Messageevents und Kommunikationsakte nach dem FIPA Standard und Deliberation. Mehr Verwendung hätte das Konzept der Capabilities finden können; z. B. hätten die unterschiedlichen Verhaltensweisen des Pax Agenten in eine Capability [4] ausgelagert werden können.

7.6. Bewertung der PSI-Theorie für Evakuierungssimulationen

Die PSI-Theorie bietet ein umfangreiches und interessantes Konzept für die Simulation von lebensnahem Verhalten. Die Vielfalt der Verhaltensweisen, die mit PSI modelliert werden können, sind es gerade, die in den bestehenden **EvacSims!** (**EvacSims!**) noch nicht umgesetzt wurden und die die Ergebnisse der Simulationen noch näher an die Realität bringen könnten. Dennoch, die Möglichkeiten des PSI-Konzept übersteigen die Anforderungen einer Evakuierungssimulation. Daher erscheint die Entwicklung einer Evakuierungssimulation nur mit einer abgewandelten Form der PSI-Theorie sinnvoll. Außerdem ist es wegen der ineinander verwobenen Bestandteile der PSI-Theorie schwer, eine iterative Vorgehensweise umzusetzen. Es bedarf einer gewissen Grundkomplexität, bevor angefangen werden kann ein Verhalten zu erzeugen. Ist diese Hürde aber genommen, ist die Integration neuer Verhaltensweisen relativ einfach.

Die Verwendung der Bewertungstheorie von OCC (siehe 2.3.1 (S.27)) wäre für diese Arbeit wohl wesentlich einfacher gewesen, da in dieser Theorie direkt auf Ereignisse eine Emotion ausgelöst wird. Das daraus resultierende Verhalten wäre nach Meinung des Autors aber relativ statisch und vorhersehbar. Die PSI-Theorie hat demgegenüber, gerade durch ihre Komplexität und den dennoch intuitiven Aufbau, eine gewisse Finesse. Der Anreiz besteht somit für den interessierten Informatiker, sich gerade dieser Komplexität anzunehmen.

7.7. Fazit

Die Umsetzung einer Evakuierungssimulation mit Hilfe von Agententechnologie und emotionalen Theorien ist ein sehr umfangreiches Projekt und beinhaltet viele verschiedene Aspekte und Grundlagen, die erstmal erarbeitet und verstanden werden müssen. Die Breite der Arbeit

[4] Capability: (Modularisierungskonzept von Jadex und anderen Agentenplattformen)

und die zeitliche Begrenzung haben es verhindert, dass im finalen Prototypen in die Tiefe gegangen werde konnte. Die fachlichen Anforderungen konnten daher nur auf konzeptioneller Ebene mit dem Prototypen überprüft werden. Es konnte ein Einblick in die Entwicklung und das Design einer Evakuierungssimulation gegeben werden und das emotionale Modell der PSI-Theorie in diese Simulation integriert werden.

Auch die Entscheidung für und die Auswahl der Physic Engine ist positiv zu benennen, da so die einfache Handhabung der Kollisionserkennung und weiche Bewegungen der Pax ermöglicht wurde. Die Agentenplattform Jadex hat hervorragend funktioniert und bei der Entwicklung der Agenten sehr gut unterstützt. Besonders positiv ist die Möglichkeit der zeitlichen Steuerung (ab Jadex Version 2 Beta2) des Simulationsablaufes aufgefallen.

Auch die vorgegebenen Capabilities für die Kommunikation von Agenten konnten vielfältig genutzt werden. Die Performanz der Simulation auf dem Testsystems[5] war bei 5 - 10 Pax gut. Kamen weitere Pax hinzu, wurde die Simulation immer langsamer. Weist man der Java Virtual Machine 512MB RAM[6] zu, können mit dem Testsystem alle knapp 300 Plätze des ausgewählten Flugzeuginnenraums besetzt werden. Bei dieser Anzahl der Pax verzögert sich die Ausführung der Simulation im Gegensatz zur Realzeit um den Faktor 100-500. Hier ist also zu überlegen, ob es nicht performanter und speicherschonender ist, den Emotionalen Agenten als Capability in den Pax zu integrieren. Diese Maßnahme würde den Speicherbedarf ungefähr halbieren. Eine weiter Möglichkeit ist die Verteilung der Simulation auf verschiedene Computer. Da es sich hier um ein Agentensystem handelt, ist diese Anforderung mit relativ kleinem Aufwand zu bewerkstelligen.

[5] Getestet wurde auf einem ©Athlon64 3700+ mit 2GB RAM und ©Windows Vista.
[6] Das erreicht man mit dem Parameter -Xmx512M.

8. Zusammenfassung und Ausblick

8.1. Zusammenfassung

Die interessante Grundidee dieser Arbeit ist, wie in der Einleitung beschrieben, die Steuerung von menschlichem Verhalten durch emotionale Systeme. Diese Idee wurde aufgegriffen und versucht, sie am Beispiel einer Evakuierungssimulation umzusetzen. Nach einer ausführlichen Einleitung in die Themengebiete Multiagentensysteme, Evakuierungssimulationen und Emotionale Systeme wurde eine fachliche und technische Analyse vorgenommen. Darin Technologien für die Umsetzung der Anforderungen, wie z. B. eine Physic Engine und verschiedene Steering Behavior, vorgestellt, die geeigneten ausgewählt und die Umsetzbarkeit in Jadex anhand eines technischen Prototypen gezeigt. Die Nutzung der PSI-Theorie für das emotionale System wurde aufgrund der in ihr enthaltenen Verhaltenssteuerung entschieden.

Danach wurde anhand des Designs ein Vorschlag zur Architektur des Gesamtsystems gemacht und mit Prometheus Diagrammen dargestellt. Von besonderer Bedeutung ist die Trennung zwischen Pax Agent und Emotional System Agent, die die Austauschbarkeit der Pax und des emotionalen Systems ermöglicht. Die praktische Umsetzbarkeit des Designs wurde in der Realisation eines finalen Prototypen bewiesen und die Funktionsweise der implementierten Anforderungen beschrieben. Des Weiteren wurde das Testvorgehen während der Entwicklung anhand eines Beispiels erläutert.

In der Bewertung zeigte sich, dass die Komplexität der PSI-Theorie und die zeitliche Begrenzung der Arbeit ein Hindernis für eine genauere Umsetzung der angestrebten Verhaltensregulierung waren. Auch das sukzessive Implementieren von neuen Funktionen, in Verbindung mit visuellen Tests, nahm sehr viel Zeit in Anspruch. Es sind definitiv Standards nötig, um eine testgetriebene Entwicklung der Agenten zu vereinfachen und zu beschleunigen. Das Vorgehensmodell Prometheus hat die Umsetzung des Multiagentensystems erleichtert. Die Standards für Agenten-Methodologien sollten aber um weitere Konzepte, wie z. B. die Modellierungsmöglichkeit des Konzeptes der Deliberation, erweitert werden.

Trotzdem konnte gezeigt werden, dass die PSI-Theorie mit BDI Agenten umsetzbar ist und für die Verhaltenssteuerung von Personen in Evakuierungssimulationen genutzt werden kann. Positiv zu bewerten, ist auch die Entscheidung für die unterschiedlichen Steering

Behavior, die in Kombination mit der Physic Engine eine realistische Bewegungssteuerung in einer kontinuierlichen Umgebung ermöglichen.

8.2. Ausblick

Aufbauend auf dieser Arbeit und dem finalen Prototypen könnte eine professionelle Evakuierungssimulation entwickelt werden, die Emotionale Konzepte als Grundlage der Verhaltenssteuerung der Personen hat. Eine Vielzahl an Verbesserungen und Erweiterungen sind möglich; z. B. die Verwendung einer dreidimensionale Darstellung eines Flugzeuges oder eines anderen Evakuierungsgegenstandes mit einer gesonderten Engine für physikalische Berechnungen, die auch Feuer und Rauch unterstützt. Um eine höhere Berechnungsgeschwindigkeit des Simulationsvorganges zu erreichen, wäre eine Verteilung des Systems auf mehrere Rechner eine Option, die umgesetzt werden sollte.

Die Anzahl der Verhaltensweisen könnte für zukünftige Versionen des Programms erweitert werden, so dass die Simulation auch für weitere Anwendungsgebiete genutzt werden kann, wie z. B. Crowd-Behavior Simulation.

Vielleicht könnte auch eine systematische tabellarische Aufstellung aller Bedürfnismodulationen die Programmierung der PSI-Verhaltenssteuerung vereinfachen. Auch die Integration weiterer emotionaler Konzepte wäre, um einen Vergleich mit dem PSI-System herzustellen, denkbar. Praktisch könnte an dieser Stelle ein Baukasten für das Emotionale-System sein.

Das Konzept der Steuerung von Verhalten über Bedürfnisse ist eine interessante Herangehensweise für unvorhersehbare Domänen. Vielleicht wäre es sogar möglich, die Planung der Abläufe in einer komplexen Produktionsanlage durch eine emotionale Steuerung zu unterstützen. Man könnte sich hierbei Vorstellen, dass beispielsweise das Bedürfnis des Hungers den Nachschub von Rohmaterialien regelt und das Bestimmtheitsbedürfnis bei jedem fertigen Produkt steigt und bei Fehlern im Prozessablauf sinkt. Auch die Verwendung für die automatische Skalierung von Webanwendungen wäre weiterhin denkbar.

Die hauptsächliche Herausforderung besteht darin, in Bezug auf die Anwendungsdomäne die richtigen Analogien für die Bedürfnisse und ihre Zu- und Abnahme zu finden.

Literaturverzeichnis

[Fipa] : *http://www.fipa.org/*. – URL http://www.fipa.org/specs/fipa00023/SC00023K.pdf

[Bartneck 2002] : *Integration the OCC Model of Emotions in Embodied Characters. Proceedings of theWorkshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges, Melbourne.* 2002

[EASA 2003] : *EASA ED Decision 2003/02/RM CS-25*. 2003

[Duden2009 2009] *Duden - Deutsches Universalwörterbuch*. Bibliographisches Institut & F. A. Brockhaus AG, 2009, 2009

[Azaiez u. a. 2007] AZAIEZ, Selma ; AZOUAOUI, Ouahiba ; CARVALHO, Tomaz de ; FRANÇA, Jussara de ; CAO, Yang ; DAVALOS, Sergio ; ESFAHANIPOUR, Akbar ; ESMAHI, Larbi ; HAMDANI, Tarek M. ; HASHEMI, Shohreh ; HU, Yen-Hung ; HUGET, Marc-Philippe ; KLEYLE, Robert ; LI, Long-Zhuang ; LIN, Fuhua ; OMER, Kursheed ; PIECZYNSKA, Agnieszka ; QIAN, Yi ; SIRISAENGTAKSIN, Ongard ; TANIGAW, Utako ; WANG, Hongxue ; XIAO, Zhigang ; XIE, Menchun ; YANG, Chunsheng ; LIN, Hong (Hrsg.): *Architectural Design of Multi-Agent Systems: Technologies and Techniques*. 2007

[Bratman u. a. 1988] BRATMAN, Michael E. ; ISRAEL, David J. ; POLLACK, Martha E.: *Plans And Resource-Bounded Practical Reasoning*. 1988

[Braubach u. a. 2005] BRAUBACH, Lars ; POKAHR, Alexander ; MOLDT, Daniel ; LAMERSDORF, Winfried: *Goal Representation for BDI Agent Systems*. 2005

[Damasio 2001] DAMASIO, A. R.: *Descartes Irrtum. DTV, München.* 2001

[Detje 1999] DETJE, F.: *Handeln erklären. Vergleich von Theorien menschlichen Handels und Denkens.* Wiesbaden: Deutscher Universitätsverlag., 1999

[Dörner 2003] DÖRNER, D.: *Autonomie. In Christaller, T. und Wehner, J., editors, Autonome Maschinen.* Westdeutscher Verlag., 2003

[Dörner u. a. 2003] DÖRNER, D. (Hrsg.) ; DETJE, F. (Hrsg.) ; SCHAUB, H. (Hrsg.): *The Mathematics of Emotions. Proceedings of the Fifth International Conference on Cognitive Modeling (ICCM 2003)*. Bd. *75-80)*. Bamberg : Universitätsverlag., 2003

[Ferber 1999] FERBER, Jacques: *Multi-Agent Systems : An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, 1999

[Galea u. a. 2006] GALEA, E. R. ; FINNEY, K. M. ; DIXON, A. J. P. ; SIDDIQUI, A. ; COONEY, D. P.: A Database to Record Human Experience of Evacuation in Aviation Accidents The Aircraft Accident Statistics and Knowledge Database (AASK). In: *CAA PAPER 2006/01* (2006), S. 76

[Galea u. a. 2008] GALEA, E.R. ; SHIELDS, J. ; CANTER, D. ; BOYCE, K. ; DAY, R. ; HULSE, L. ; SIDDIQUI, A ; SUMMERFIELD, L. ; MARSELLE, M. ; GREENALL, P.: Methodologies employed in the collection, retrieval and storage of human factors information derived from first hand accounts of survivors of the WTC disaster of 11 September 2001. In: *Journal of Applied Fire Sciences* (2008)

[Ghasem-Aghaee u. a. 2008] GHASEM-AGHAEE, Nasser (Hrsg.) ; FATAHI, Somayeh (Hrsg.) ; ÖREN, Tuncer I. (Hrsg.): *Agents with Personality and Emotional Filters for an E-learning Environment*. 2008

[Hannon 2003] HANNON, Charles J. (Hrsg.): *Emotion-based Control Mechanisms for Agent Systems*. 2003

[Hartmann 1996] HARTMANN, Stephan: *The World as a Process: Simulations in the Natural and Social Sciences*. 1996

[Henderson-Sellers und Giorgini 2005] HENDERSON-SELLERS, Brian ; GIORGINI, Paolo: *Agent-Oriented Methodologies*. IGI Publishing, 2005. – 413 S. – URL http://www.igi-pub.com/books/details.asp?id=4931

[Herrler 2007] HERRLER, Rainer: *Agentenbasierte Simulation zur Ablaufoptimierung in Krankenhäusern und anderen verteilten, dynamischen Umgebungen*, JuliusMaximilian-sUniversität Würzburg, Dissertation, 2007

[Hirth u. a. 2008] HIRTH, Jochen (Hrsg.) ; BRAUN, Tim (Hrsg.) ; BERNS, Karsten (Hrsg.) ; University of Kaiserslautern Germany (Veranst.): *Emotion Based Control Architecture for Robotics Applications*. 2008

[Jiang 2007] JIANG, Hong: *From rational to emotional agents*. Columbia, SC, USA, Dissertation, 2007. – Adviser-Vidal, Jose M.

[Johnson 2005] JOHNSON, C.W.: *Applying the Lessons of the Attack on the World Trade Center, 11th September 2001, to the Design and Use of Interactive Evacuation Simulations*. 2005

[Klügl 2006] KLÜGL, Franziska: Multiagentensimulation. In: *Informatik Spektrum* (2006), S. 5

[Klüpfel 2003] KLÜPFEL, Hubert L.: *A Cellular Automaton Model for Crowd Movement and Egress Simulation*, Von der Fakultät 4 Naturwissenschaften der Universität DuisburgEssen Standort Duisburg, Dissertation, 2003

[Lämmel und Nagel 2008] LÄMMEL, Gregor ; NAGEL, Kai: *Multi agent based large-scale evacuation simulation*. 08 2008

[Ortony u. a. 1988] ORTONY, A. ; CLORE, G. ; COLLINS, A. a.: *The Cognitive Structure of Emotions.* New York : Cambidge University Press., 1988

[Padgham 2002] PADGHAM, Lin (Hrsg.): *Prometheus: A Methodology for Developing Intelligent Agents*. 2002

[Padgham und Winikoff 2004] PADGHAM, Lin ; WINIKOFF, Michael: The Prometheus Methodology. (2004)

[Rübenstrunk 1998] RÜBENSTRUNK, G.: *Emotionale Computer Computermodelle von Emotionen und ihre Bedeutung für die emotionspsychologische Forschung*. Internet http://www.ruebenstrunk.de/emeocomp/INHALT.HTM., 1998

[Renz und Sudeikat 2007] RENZ, Wolfgang ; SUDEIKAT, Jan: *Vorlesungsunterlagen Agentenorientierte Software Entwicklung SoSe2007 HAW-Hamburg*. 2007

[Reynolds 1999] REYNOLDS, Craig W. (Hrsg.) ; Sony Computer Entertainment America (Veranst.): *Steering Behaviors For Autonomous Characters*. Bd. *21*. Sony Computer Entertainment America, 1999

[Russel und NORVIG 2003] RUSSEL, Stuart J. ; NORVIG, Peter: *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003 (ISBN 0-13-080302-2)

[Sargent 2004] SARGENT, Robert G. (Hrsg.) ; Department of Electrical Engineering and Computer Science L.C. Smith College of Engineering and Computer Science Syracuse University Syracuse, NY 13244, U.S.A (Veranst.): *VALIDATION AND VERIFICATION OF SIMULATION MODELS*. 2004

[Schmidt 2008] SCHMIDT, Bernd: *Die Modellierung menschlichen Verhaltens Das PECS-Referenzmodell*. 2008

[Schneider 2005] SCHNEIDER, Gordon B.: *Agenten und unsere Emotionen Ein Vergleich von Dörners PSI-Theorie mit der Emotionstheorie von Ortony, Clore und Collins*, Diplomarbeit, 2005

[Sudeikat 2004] SUDEIKAT, Jan: *Betrachtung und Auswahl der Methoden zur Entwicklung von Agentensystemen*, Hochschule f¨ur Angewandte Wissenschaften Hamburg, Diplomarbeit, 2004

[Wooldridge und Jennings 1995] WOOLDRIDGE, M ; JENNINGS, N. R.: Intelligent Agents: Theory and Practice. (1995). – URL http://www.csc.liv.ac.uk/~mjw/pubs/ker95.pdf

[Wooldridge 2002] WOOLDRIDGE, Michael: *An introduction to multiagent systems. Wiley,*. 2002

[Zapf 2007] ZAPF, Dr. M.: *Glaube, Wünsche, Absichten: Entwurf intelligenter Agenten Vorlesungsunterlagen zu Verteilt-kooperative Informationsverarbeitung*. Universität Dortmund. 2007

A. Glossar

Capability Modularisierungskonzept von Jadex und anderen Agentenplattformen

Deliberation Entscheidungsprozess welches Ziel in Abhängigkeit zur momentanen Situation verfolgt wird

Means-End-Analyse Auswahl einer Aktion oder einer Folge von Aktionen die den „Abstand“ zwischen dem angestrebten Ziel und der momentanen Situation verringert

Methodologie Methodologie: (Duden2009, 2009) Lehre, Theorie der wissenschaftlichen Methoden

Practical Reasoning praktische/rationale Entscheidungsfindung

Validierung Das Validieren ist eine Prüftätigkeit, bei der Produkte, Protokolle oder Dokumente in Bezug auf ihre Spezifikationen geprüft werden.

Verifikation Verifizierung oder Verifikation (von lat. veritas, Wahrheit und facere, machen) ist der Nachweis, dass ein vermuteter oder behaupteter Sachverhalt wahr ist.

Pax Begriff für Passagier(e) im Fluggewerbe

B. Abkürzungsverzeichnis

ADF	Agent Definition File
AOSE	Agentenorientierte Software Entwicklung
BDI	Belief Desire Intention
EASA	Europäische Agentur für Flugsicherheit
EBDI	Emotional Belief Desire Intentional
EvacSim	Evakuierungssimulation/Evacuation Simulation
FIPA	Foundation for Intelligent Physical Agents
IFAAMAS	International Foundation of Autonomous Agents and Multi-Agent Systems
KI	Künstliche Intelligenz
MAS	Multiagentensystem
OBJ	Wavefront Standard 3D Object Format
OCC	Ortony, Clore und Collins
OO	Objektorientiert
PDT	Prometheus Design Tool

Index

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 6. Juli 2009	
Ort, Datum	Unterschrift