Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung AW 2 - SoSe 2009

Gregor Balthasar

JARTS - Jadex plays Realtime Strategy
Ein Framework zur einfachen Integration von Jadex-BDI-Agenten in ein Realtime Strategy Spiel

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

Kurzzusammenfassung

In dieser Arbeit wird aufgezeigt, wie sich das JARTS-Framework in den Kontext anderer, bereits existierender Testumgebungs-Frameworks eingliedert. Dazu werden Themengebiet und Projekt kurz erläutert, daraufhin eine Auswahl ähnlicher Arbeiten herangezogen und mit dem JARTS-Framework verglichen.

1 Einführung in das Themengebiet

Multiagenten-Systeme (MAS) haben in den letzten Jahren eine stetig zunehmende Bedeutung in der Softwareentwicklung erlangt. Damit einher gehen internationale Konferenzen und Workshops (AAMAS[1] als eine der bedeutendsten Konferenzen, jährlich seit 2002). Ein wichtiges Thema, welches sich auch bei den Konferenzen widerspiegelt, sind Testmöglichkeiten für MAS. Eine spezielle Form davon sind Testumgebungen (engl. *testbeds*). Die Zielsetzungen solcher Testumgebungen weichen dabei teilweise deutlich voneinander ab. Generell lassen sich folgende (Haupt-)Ziele identifizieren: (1) Identifikation von Problemen bei der Entwicklung von MAS, (2) Bewertung von Entwicklungsmethodologien und -tools für MAS, (3) Testen von Stabilität und Geschwindigkeit von MAS, (4) Einsatz von Testumgebungen in der Lehre, (5) Bewertung der Glaubwürdigkeit *Künstlicher Intelligenz* (KI) [Behrens u. a. (2008); Livingstone (2006)]. Das JARTS-Projekt (Jadex plays Realtime Strategy) zielt hauptsächlich darauf ab, eine Testumgebung für die Lehre und Forschung zu bieten, versucht aber gleichzeitig auch den Brückenschlag zu den anderen genannten Zielen zu erreichen. Dazu soll es ein Framework bieten, welches Entwicklern die Möglichkeit gibt, ohne jegliche Schnittstellen- oder Netzwerkprotokollkentnisse Jadex-BDI-Agenten für ein bereits bestehendes (evtl. kommerzielles), komplexes Realtime Strategy (RTS) Spiel zu programmieren. Dieses Framework setzt sich einerseits aus einem an den Game-Server eines RTS Spiels angegliederten Modul, welches die Kommunikation zum Server kapselt, und andererseits aus einem auf dem Capability-Konzept [Braubach u. a. (2005)] basierenden Jadex-Modul zusammen, wobei beide Module mittels einer TCP-Verbindung miteinander kommunizieren werden.

In den folgenden Kapiteln werden nun zuerst grundlegende, in dieser Arbeit häufiger vorkommende Begrifflichkeiten geklärt, daraufhin werden einige bereits bestehende Arbeiten ähnlicher oder vergleichbarer Thematik herangezogen und bewertet. Kapitel 4 dient dann der Abgrenzung der vorher betrachteten Arbeiten von der eigenen, um diese in das Themengebiet einordnen zu können. Dazu werden Unterschiede herausgearbeitet und eventuelle Gemeinsamkeiten genannt. Im letzten Kapitel wird alles Besprochene noch einmal zusammengefasst.

[1] s. http://www.conferences.hu/AAMAS2009/introduction.html

2 Grundlegende Begrifflichkeiten

2.1 First Person Shooter (Ego-Shooter)

First Person Shooter (FPS) bezeichnen ein Genre im Bereich der Computerspiele, in dem der Spieler Kontrolle über einen einzelnen Charakter übernimmt, der sich in einer (meist feindlichen) dreidimensionalen, virtuellen Umgebung befindet. Der Spieler steuert diesen Charakter aus der Ich-Perspektive durch die vorgefertigte Umgebung und wird dort entweder mit computerkontrollierten Charakteren (Bots) oder anderen menschlichen Mitspielern konfrontiert, die meist mithilfe von Waffen zu bekämpfen sind.

2.2 Realtime Strategy Game (Echtzeitstrategie-Spiel)

Realtime Strategy Spiele (RTS) titulieren ein Computerspiel-Genre, in dem der Spieler das Kommando über eine virtuelle Armee übernimmt. Das Spielgeschehen wird meist aus der isometrischen Perspektive entweder zwei- oder dreidimensional dargestellt. Der Spieler kann Einheiten anwählen, zu Gruppen zusammenfassen, sie an bestimmte Positionen schicken und gegnerische Einheiten angreifen lassen. Häufig gehen mit den eigentlichen Gefechten noch der Aufbau einer Basis, Forschung (beides um z.B. weitere, spezialisierte Einheitentypen bauen zu können) und Ressourcenmanagement einher (bestimmte Ressourcen müssen kontinuierlich gefunden und abgebaut werden, um damit den Bau von Einheiten zu ermöglichen).

3 Vorstellung und Bewertung vergleichbarer Arbeiten

In diesem Kapitel wird eine Auswahl mit dem JARTS Projekt vergleichbarer Arbeiten vorgestellt und einer kurzen Bewertung unterzogen. Diese Auswahl beansprucht keine Vollständigkeit. Am Ende des Kapitels werden die vorgestellten Arbeiten in einem Fazit miteinander verglichen und es wird versucht ein *State of the Art* im Forschungsfeld Testumgebungen für verteilte Künstliche Intelligenz zu formulieren.

3.1 FlexBot

FlexBot ist ein vom *Computer Science Department* der *Northwestern University* entwickeltes Framework zur Integration von selbstentwickelten Bots in das kommerzielle Computerspiel Half-Life. Half-Life ist im First Person Shooter Genre angesiedelt und wurde im Jahr 1998 veröffentlicht [Valve].

Kurzbeschreibung

Das FlexBot-Framework bietet eine DLL, die zur Laufzeit in das Computerspiel Half-Life eingebunden wird. Diese DLL kann weitere *behavior*-DLLs einbinden, in denen dann die eigentliche KI des Bots enthalten ist und die daraufhin mit der FlexBot-DLL kommunizieren [Khoo u. a. (2002)]. Somit können Bots in jeder Programmiersprache entwickelt werden, mit der es möglich ist Windows-DLLs zu erzeugen. Das Framework bietet ein festes, unveränderliches Set von Sensoren und Effektoren (*get_vision_distance(float angle, int range)* ist z.B. ein Sensor, *do_rotation(int speed)* ein Effektor), die dem Bot-Entwickler zur Verfügung stehen [Dunham].

Bewertung

Da aufgrund der weiterhin bestehenden Kommerzialität des dem FlexBot-Framework zugrundeliegenden Computerspiels Half-Life ein Test des Frameworks nicht ohne weiteres möglich war, kann hier nur kurz das grundsätzliche Konzept bewertet werden.

Die Idee, die Bots über DLLs zu einem Teil des Game-Servers werden, bzw. diesen die Bot-KI dynamisch zur Laufzeit einbinden zu lassen, birgt sowohl Vor- als auch Nachteile. Als Vorteil kann man bewerten, dass damit die Bots eng mit dem Game-Server gekoppelt und somit einem typischen kommerziellen Szenario sehr nah sind. Ein Nachteil ist, dass man bei den nutzbaren Programmiersprachen eingeschränkt ist auf solche, die es erlauben, Windows-DLLs zu erzeugen. Damit fallen z.B. spezielle Multiagentensprachen weg und für Java-basierte Multiagentenplattformen wird die Entwicklung zumindest sehr aufwändig (Entwicklung von DLLs in einer anderen Programmiersprache (C, C++, C#, etc.), Einbindung in Java durch *Java Native Interface* (JNI) Invocation).

Allerdings ist es für Forschung und Lehre, sowie die anderen oben benannten Ziele für Testumgebungen irrelevant, ob die Bots eine enge Kopplung zum Game-Server haben oder nicht.

3.2 GameBots - Unreal Tournament (1/2003/2004)

GameBots ist ein in Kollaboration des *Information Sciences Institutes* der *University of Southern California* mit dem *Computer Science Department* der *Carnegie Mellon University* entstandenes Framework zur Integration von Bots in das kommerzielle First Person Shooter Spiel *Unreal Tournament* aus dem Jahr 1999 [EpicGames]. Basierend auf dem gleichen Konzept wurde das Framework vom *Rochester Institute of Technology* für die Nachfolge-Versionen des Spiels (UT 2003/2004) portiert, die sowohl eine realistischere Grafik, als auch Physik bieten.

Kurzbeschreibung

Das Framework bietet ein mit dem Unreal Tournament-Server verbundenes Modul, welches die Kommunikation mit dem Server kapselt. Die Programmierung von Bots findet davon separiert und plattformunabhängig statt; die selbstentwickelten Bots müssen eine TCP-Verbindung zum GameBots-Server aufbauen, über die sie mit synchronen (Datenstrom im 1/10s Takt) und asynchronen Nachrichten Sensordaten in einem vorher definierten Protokoll erhalten. Effektor-Nachrichten werden asynchron an den GameBots-Server gesendet und unterliegen ebenfalls einem vordefinierten Protokoll. Es bestehen Client-Implementationen für Java und SOAR, die ihrerseits den Aufbau dieser Nachrichten kapseln [Adobbati u. a. (2001).

Beispiele für protokollkonforme Nachrichten:

- INV {Id DM-Test.MinigunAmmoPickup4} {Location 2525.00,-1341.00,-116.00} {Reachable True} {Class Xweapons.MinigunAmmoPickup}
- RUNTO {Location 1080.7336812597903,-345.18735030270204,-78.15}

Bewertung

Durch den Einsatz der GameBots für Unreal Tournament 2003 im Rahmen der Wahlpflichtveranstaltung *Einführung in die Agentenorientierte Softwareentwicklung* geleitet von Prof. Dr. Wolfgang Renz an der HAW Hamburg (SoSe 2007) konnten praktische Erfahrungen mit diesem Framework gewonnen werden. Die vorhandene Implementation eines Java-Clients vereinfachte die Integration in die Java-basierte Multiagentenplattform Jadex, so dass ohne großen Aufwand mit der Entwicklung der Bot-Agenten begonnen werden konnte. Leider ließen sich die in den Ausarbeitungen versprochenen Features im Praxis-Test nicht vollständig nutzen. Das Fehlen von bestimmten Sensordaten z.B. machte eine autonome Navigation der Agenten in der Umgebung unmöglich. Ebenfalls erzielte die Nutzung gewisser Effektoren nicht den gewünschten Erfolg.

Positiv ist der plattformunabhängige Ansatz zu bewerten; die Trennung in ein servernahes Modul und in ein Client-Modul, wobei die Module miteinander per TCP kommunizieren, macht die Entwicklung auch für exotischere KI- bzw. Multiagentensprachen möglich.

3.3 AgentContest 2008 (2009)

Die AgentContest-Reihe ist eine von 2005 an jährlich im Rahmen des PROMAS-Workshops (bis 2006 im Rahmen der CLIMA Konferenz) von einem Team rund um Prof. Dr. Jürgen Dix[2]

[2] Professor an der Universität Clausthal, Lehrstuhl für Computational Intelligence

durchgeführte Wettbewerbs-Serie, bei der verschiedene Teams aus der ganzen Welt ihre MAS in einem bestimmten, vorgegebenen Szenario gegeneinander antreten lassen [Behrens u. a. (2008)].

Kurzbeschreibung

Der AgentContest 2008 (und auch der kommende Contest 2009) bietet eine frei verfügbare Java-basierte Server-Implementation eines videospielähnlichen Szenarios. Bei diesem Szenario geht es darum, dass zwei aus jeweils sechs eigenständigen Agenten bestehende Teams versuchen, auf einer zweidimensionalen, in Gitter-Parzellen eingeteilten Umgebung virtuelle Kühe in ein dem Team zugeordnetes Gatter zu treiben. Jeder der sechs Agenten eines Teams hält eigenständig eine TCP-Verbindung zum Game-Server. Ein vereinbartes Protokoll definiert dabei die XML-basierte Kommunikation zwischen Client und Server. Die Simulation eines Szenarios erfolgt rundenbasiert, zu Anfang jeder Runde sendet der Server eine *Request-Action*-Anfrage an den Agenten-Client, die zusätzlich alle Sensordaten des jeweiligen Agenten enthält. Der Agent muss dann in einem vorgegebenen Zeitrahmen eine *Action*-Nachricht zurücksenden, wobei ihm als Aktor nur die Bewegung in eine von acht Richtungen zur Verfügung steht. Sowohl in der Server-Implementation von 2008, als auch in der von 2009 können Felder nicht begehbar von den beweglichen Einheiten sein, diese werden dann als *Obstacles* bezeichnet und bringen die Notwendigkeit der Navigation in das Szenario ein.

Kurz erwähnt werden sollte noch, dass das AgentContest-Framework Ähnlichkeiten zu einigen, häufig mit Multiagentenplattformen ausgelieferten Testumgebungen aufweist (z.B. zu der Hunter-Prey Testumgebung von Jadex) und hier als Stellvertreter dieser Gruppe ausgewählt wurde. Die Auswahl erfolgte aufgrund der Plattformunabhängigkeit des AgentContest-Frameworks, welche bei den anderen erwähnten Testumgebungen nicht bzw. nur bedingt gegeben ist.

Bewertung

Die eigene Teilnahme am AgentContest 2008 ermöglicht hier sowohl eine kurze Bewertung technischer und fachlicher Aspekte, als auch eine Bewertung bezüglich des Erreichens der für den Contest gesetzten Ziele (Identifizierung von Schlüsselproblemen, Bewertung von Methodologien und Tools zur Erstellung von MAS, Anregen der Forschung) [Balthasar u. a. (2008), Behrens u. a. (2008)].

Durch die über TCP gehaltene Verbindung und das in XML definierte Nachrichtenprotokoll besteht ein Interface, das völlige Plattformunabhängigkeit ermöglicht. Kritikpunkt auf fachlicher Ebene ist lediglich, dass der Algorithmus, anhand dessen die virtuellen Kühe ihre Bewegung

durchführen, in der Server-Implementation von 2008 nicht wie in der Szenariobeschreibung festgelegt funktioniert.

Die Zielsetzung des Contests wurde (zumindest aus der Sicht der eigenen Arbeit) erfüllt, es ließen sich sowohl Schlüsselprobleme erkennen, als auch Unstimmigkeiten und fehlende Übertragbarkeit auf das eingesetzte MAS in der genutzten Methodologie [Balthasar (2008)]. Da die Implementation des Contest-Servers frei verfügbar und somit auch über den Contest hinaus verwendbar ist, ist auch ein Einsatz in der Lehre durchaus vorstellbar.

3.4 Fazit

Alle in diesem Kapitel vorgestellten Testumgebungs-Frameworks haben eine Gemeinsamkeit: Sie bieten eine virtuelle Umgebung mit der sich zu testende Systeme verbinden können. Dafür muss bei zwei von drei Frameworks eine TCP-Verbindung aufgebaut werden, über die dann Sensor- und Effektordaten mittels eines vorher festgelegten Protokolls ausgetauscht werden. Dadurch besteht nur eine sehr lose Kopplung und eine damit einhergehende hohe Plattformunabhängigkeit. Ein Framework (FlexBot) nutzt Windows-DLLs und eine enge Kopplung zum Game-Server, was den Entwicklungsaufwand für z.B. Java-basierte Plattformen erhöht.

Weiterhin zeigt sich, dass es zwei grundsätzlich verschiedene Ansätze für die gebotene Umgebung und somit zwei Gruppen von Frameworks gibt: Zwei der Frameworks setzen auf die Server-Implementationen kommerzieller Computerspiele aus dem First Person Shooter Genre und werden im Folgenden als „Bot-Frameworks“ bezeichnet; das dritte Framework setzt auf die eigene Implementation eines einfachen zweidimensionalen Szenarios und wird im Weiteren auch stellvertretend für diese Gruppe Realtime Strategy Spiel-ähnlicher Testumgebungen genannt.

Der Ansatz der Bot-Frameworks, die Umgebung nicht selbst zu implementieren, scheint einen gewissen Vorteil zu bieten, da dadurch der Aufwand, eine komplexe, konsistente Umgebung zu erschaffen, entfällt. So zeigten sich im Test z.B. die erwähnten Probleme mit den Umgebungsalgorithmen der Server-Implementation der AgentContest 2008 Testumgebung, trotz des relativ simpel gehaltenen Szenarios. Ein weiterer Unterschied zwischen beiden Gruppen besteht darin, dass in den Bot-Frameworks eine Mensch-Agent-Interaktion möglich ist, die das AgentContest-Framework nicht bietet.

Bezüglich der Aktualität der vorgestellten Frameworks muss vermerkt werden, dass die beiden Bot-Frameworks ihre letzten Updates jeweils im Jahr 2002 erhielten, sie bestehen seitdem unverändert und werden hauptsächlich im Bereich der Lehre eingesetzt. Das AgentContest-Framework wird jährlich aktualisiert und das Szenario erweitert, wobei die grundlegenden Eigenschaften allerdings erhalten bleiben. Ein *State of the Art* lässt sich somit relativ schwer

definieren, einige Arbeiten zum Thema Testumgebungen mit computerspielbasierten bzw. -ähnlichen Umgebungen aus Konferenzen dieses Jahres stehen allerdings noch aus. Wird hinzugezogen, dass noch weitere, den hier vorgestellten Arbeiten sehr ähnliche Bot-Frameworks existieren, von denen keines zeitlich neuer ist, ließe sich folgende These aufstellen: „Die bestehenden Frameworks reichen trotz ihres Alters aus, um die für sie gesetzten Ziele immer noch zu erfüllen und der Aufwand Frameworks für modernere Computerspiele zu erstellen im Vergleich zum daraus resultierenden Nutzen wäre zu hoch“.

4 Abgrenzung zur eigenen Arbeit

Mit der abschließenden These des letzten Kapitels stellt sich natürlich die Frage: Warum dennoch ein weiteres Framework entwickeln? Aus diesem Grund soll an dieser Stelle erklärt werden, was das JARTS-Framework anders machen soll als die bestehenden Frameworks, wo szenariobedingte Unterschiede liegen, wo eine unterschiedliche Zielsetzung besteht und wo gegebenenfalls Gemeinsamkeiten auftreten.

4.1 Szenariobedingte Unterschiede

Zwei der vorgestellten Arbeiten (FlexBot und GameBots) bieten Frameworks zur Integration von verteilter KI in First Person Shooter Spiele. Wie bereits beschrieben soll JARTS ein Framework für ein Realtime Strategy Spiel bieten. Hier besteht ein genereller Unterschied zwischen den beiden Spiel-Genres (s. Kapitel 2). Daraus gefolgert entstehen völlig andere Anforderungen an die KI (bzw. das Multiagentensystem), von denen im Folgenden die gravierendsten gegenübergestellt werden:

Realtime Strategy	**First Person Shooter**
Kontrolle einer große Zahl von Einheiten	Kontrolle einer einzelnen Einheit
Navigation geschieht serverseitig	Navigation sehr wichtige Anforderung
Globale Strategie in mehreren Bereichen nötig	Taktische Aspekte
Drei große Anforderungsbereiche: Ressourcenmanagement, Basis- und Einheitenbau, Leitung des Kampfgeschehens	Laufen und Schießen, evtl. Flagge erobern (modusabhängig)

Tabelle 1: gravierende Unterschiede RTS / FPS

Eine größere Ähnlichkeit besteht zwischen den Szenarien der AgentContest-Serie und denen eines Realtime Strategy Spiels. Beide bieten eine in Parzellen aufgeteilte, zweidimensionale[3] Umgebung, in der mehrere Einheiten gesteuert werden müssen, die kooperativ ein Ziel erreichen sollen. Allerdings fallen beim AgentContest-Szenario der Bau von Einheiten, sowie das Ressourcenmanagement weg. Auch gibt es im Vergleich zu einem RTS Spiel sehr wenige Effektoren und eine geringe Anzahl von 6-10 Einheiten pro Team. Zusätzlich zu den genannten Unterschieden kommt noch ein für die in Kapitel 4.2 beschriebenen, unterschiedlichen Zielsetzungen wichtiger Aspekt hinzu: Die AgentContest-Umgebung ist im Gegensatz zu RTS Spielen rundenbasiert.

Das Realtime Strategy Szenario von JARTS bietet noch einen weiteren interessanten Aspekt, der den vorgestellten Bot-Frameworks, sowie dem AgentContest-Framework szenariobedingt fehlt: Das Testen von Möglichkeiten zur Spieler-Unterstützung. RTS Spiele können durchaus so umfangreich und komplex sein, dass es für einen einzelnen Spieler schwierig wird, sie vollständig zu kontrollieren. Wenn die Architektur des fertigen JARTS-Frameworks es erlaubt, könnte unter dessen Zuhilfenahme ein Multiagentensystem zur Spieler-Unterstützung entwickelt werden, welches den Spieler in bestimmten Aspekten des Spiels unterstützen könnte (z.B. Ressourcenmanagement, Rückzug von angegriffenen Einheiten, etc.).

4.2 Unterschiede in der Zielsetzung

Die Zielsetzung von JARTS wurde schon in der Einführung kurz erwähnt und soll an dieser Stelle im Vergleich zu den anderen Frameworks noch etwas ausführlicher beleuchtet werden.

Zuerst zu der Zielsetzung der beiden Bot-Frameworks. Diese bieten Testumgebungen für künstliche Intelligenz, aber nicht explizit für Multiagentensysteme. Somit ist deren Zielsetzung auch nicht vollständig auf die in der Einführung aufgezeigten generalisierten Zielsetzungen für MAS-Testumgebungen abbildbar (die Ziele (1)-(3)[4] fallen z.B. ganzheitlich weg). Haupt-Zielsetzung der beiden Frameworks ist der Einsatz in der Lehre, wo sie auch für MAS zum Einsatz kommen können (wobei das FlexBot-Framework durch den DLL-Ansatz viel Anpassungsaufwand für Java-basierte Multiagenten-Plattformen erfordern würde). Dabei zielen sie darauf ab, mit interessanten Umgebungen das Interesse an der Entwicklung von KI anzuregen. Ein weiteres Ziel entsteht durch die gute Beobachtbarkeit, sowie die mögliche Mensch-KI(Agent)-Interaktion dieser Umgebungen, welche das Forschungsfeld „Glaubwürdigkeit Künstlicher Intelligenz“ ausnutzen kann, um Studien anhand dieser Umgebungen vorzunehmen.

[3] RTS Spiele, die eine dreidimensionale Engine beinhalten, nutzen diese meist nur zur realistischeren Darstellung, darunter liegt eine zweidimensionale Abstraktion

[4] s. Kapitel 1

Das direkt auf MAS zugeschnittene AgentContest-Framework hingegen deckt in seiner Zielsetzung die Ziele (1)-(3)[5] ab. Allerdings besteht eine Einschränkung bei (2): Testen der Stabilität von MAS ist durchaus gewährleistet, doch Geschwindigkeits-Tests sind durch die rundenbasierte Konzeption und das großzügig gewählte Timeout nur bedingt sinnvoll. Der Einsatz in der Lehre wird von den Erstellern des Frameworks zwar nicht direkt adressiert, wäre aber ebenfalls durchaus denkbar.

Auffällig ist bei dieser Betrachtung, dass die Zielsetzung dieser zwei unterschiedlichen Framework-Gruppen jeweils sehr spezifisch ist. An dieser Stelle kommt das JARTS-Projekt ins Spiel, welches die Zielsetzungen beider Framework-Gruppen großteils vereinen soll. Hauptsächlich für den Einsatz in der Lehre ausgelegt, wird es durch die Konzeption speziell für MAS die Möglichkeit bieten, generelle Probleme bei der Entwicklung dieser Systeme zu identifizieren, sowie Tools und Methodologien zu testen und zu bewerten. Durch den Echtzeitaspekt entsteht weiterhin die Möglichkeit, MAS auf Geschwindigkeit und Stabilität zu testen. Die gute Beobachtbarkeit (bei fast allen RTS Spielen besteht die Möglichkeit, ein Spiel als „Observer“ zu beobachten oder sich zumindest eine Wiederholung davon anzuschauen) ermöglicht zudem auch Tests bezüglich der Glaubwürdigkeit von KI.

4.3 Gemeinsamkeiten in der Architektur

Konzeptionell bestehen viele Gemeinsamkeiten mit der Architektur des GameBots-Frameworks, welches genauso wie bei JARTS geplant auf ein schon bestehendes Videospiel aufsetzt und ein dem Game-Server angegliedertes Modul bietet, das die Kommunikation zum Server kapselt. Ebenfalls wie beim GameBots-Framework soll dieses Modul Clients die Möglichkeit geben, sich per TCP und einem festgelegten Nachrichtenformat damit zu verbinden. Da die endgültige Auswahl des dem JARTS-Framework zugrundeliegenden RTS Spiels zu diesem frühen Zeitpunkt noch nicht getroffen ist, sind natürlich noch Abweichungen in der finalen Architektur möglich.

5 Zusammenfassung

Nachdem wie zu Anfang dieser Arbeit angekündigt nun eine Einordnung von JARTS in den Kontext anderer Arbeiten desselben Themengebiets erfolgt ist, sollen an dieser Stelle noch einmal kurz die wichtigsten Ergebnisse zusammengefasst werden:

- Die Unterschiede in den Anforderungen, die FPS und RTS Spiele an ein MAS haben, wurden aufgezeigt. (Abgrenzung zur Bot-Framework-Gruppe)

[5] s. Kapitel 1

- Der Unterschied zu RTS Spiel-ähnlichen Umgebungen ohne hohe Komplexität und Echtzeitanforderung wurde dargestellt. (Abgrenzung zur AgentContest-Framework-Gruppe)
- Das JARTS-Projekt versucht, eine Vereinigung der spezifischen Zielsetzungen existenter Frameworks zu erreichen. (Einordnung von JARTS ins Themengebiet)

Abschließend ist zu sagen, dass JARTS, mit dem Ansatz ein Framework speziell zur erleichterten Erstellung (ohne spezifische Protokoll- und/oder Schnittstellenkenntnisse) eines Multiagentensystems für ein bestehendes, komplexes Realtime Strategy Spiel zu bieten, etwas wirklich Neues ist und sich somit mit einer ambitionierten Zielsetzung in die bestehenden Testumgebungs-Frameworks eingliedert.

Literatur

[Adobbati u. a. 2001] ADOBBATI, Rogelio ; MARSHALL, Andrew N. ; SCHOLER, Andrew ; TEJADA, Sheila: Gamebots: A 3D Virtual World Test-Bed For Multi-Agent Research. In: *Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, 2001

[Balthasar 2008] BALTHASAR, Gregor: *Agentenbasierte Schwarmsteuerung in einer kompetitiven, dynamischen Umgebung*, HAW Hamburg, Bachelorthesis, 2008. – URL http://opus.haw-hamburg.de/volltexte/2008/634/pdf/ba_thesis_Gregor_Balthasar.pdf. – Zugriffsdatum: 2009.06.14

[Balthasar u. a. 2008] BALTHASAR, Gregor ; SUDEIKAT, Jan ; RENZ, Wolfgang: On coordinating of artificial cowboys: Using jadex to implement herding agents. Erscheint in: *Programming Multi-Agent Systems, 6th International Workshop, ProMAS 2008, Revised and Selected Papers*, Springer, 2008

[Behrens u. a. 2008] BEHRENS, Tristan M. ; DASTANI, Mehdi ; DIX, Jürgen ; NOVÁK, Peter: Agent Contest Competition: 4th Edition. In: *Postproceedings of Sixth International Workshop on Programming Multi-Agent Systems, ProMAS'08, LNAI*, Springer, 2008

[Braubach u. a. 2005] BRAUBACH, Lars ; POKAHR, Alexander ; LAMERSDORF, Winfried: Extending the Capability Concept for Flexible BDI Agent Modularization. In: *Proc. of PROMAS-2005*, 7 2005

[Dunham] DUNHAM, Greg: *FlexBot Beta 1 Manual*. – URL http://www.cs.northwestern.edu/~gdunham/flexbot/manual/index.htm. – Zugriffsdatum: 2009.06.13

[EpicGames] EPICGAMES: *EpicGames Homepage*. – URL http://www.epicgames.com/. – Zugriffsdatum: 2009.06.21

[Khoo u. a. 2002] KHOO, Aaron ; HUNICKE, Robin ; DUNHAM, Greg ; TRIENENS, Nick ; VAN, Muon: FlexBot, Groo, Patton and Hamlet: research using computer games as a platform. In: *Eighteenth national conference on Artificial intelligence*. Menlo Park, CA, USA : American Association for Artificial Intelligence, 2002, S. 1002–1002. – ISBN 0-262-51129-0

[Livingstone 2006] LIVINGSTONE, Daniel: Turing's test and believable AI in games. In: *Comput. Entertain.* 4 (2006), Nr. 1, S. 6. – ISSN 1544-3574

[Valve] VALVE: *Valve Homepage*. – URL http://www.valvesoftware.com/. – Zugriffsdatum: 2009.06.13