Universität Hamburg
Fakultät für Mathematik,
Informatik und Naturwissenschaften

Verteilte Systeme und Informationssysteme

Baccalaureatsarbeit

Visualisierung der Interaktion in Multiagentensystemen

13. Oktober 2005

Jarig Richter-Peill

0jrichte@informatik.uni-hamburg.de
Studiengang Informatik
Matr.-Nr. 5299806

Betreuer: Prof. Dr. W. Lamersdorf

Bearbeitungszeitraum: 1.9.2005 bis 13.10.2005

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Kurzfassung

In dieser Arbeit wird ein Ansatz zur Visualisierung der Interaktion in Multiagentensystemen vorgestellt. Es wird analysiert, was für Anforderungen an ein solches System zu stellen sind. Neben den – entlang des Referenzmodells der Visualisierung – getroffenen Entwurfsentscheidungen wird der Ansatz einer Architektur für die Realisierung eines Systems zur Visualisierung der Interaktion in Multiagentensystemen vorgestellt. Durch Mockups wird inspiriert, wie die Visualisierung aussehen könnte. Die Visualisierung basiert auf zwei unterschiedlichen – miteinander interagierenden – Perspektiven auf das Multiagentensystem. Die architekturzentrierte Perspektive stellt die Architektur des Systems in den Vordergrund, die interaktionszentrierte Perspektive die Interaktion der Agenten.

2 Einleitung

Diese Arbeit entstand im Zuge des im Sommersemester 2005 stattgefundenen Projektes „Realisierung verteilter Agentensysteme“ des Arbeitsbereichs VSIS der Fakultät für Mathematik, Informatik und Naturwissenschaften an der Universität Hamburg.

Ziel des Projektes war es, eine verteilte Anwendung als Multiagentensystem zu entwickeln. JADE wurde als FIPA konforme Agentenplattform eingesetzt und durch Jadex zu einer BDI-Agentenplattform erweitert. Das Vorgehensmodell im Projekt hat sich an der Prometheus Methode orientiert.

Aus dem Projekt heraus entstand die Motivation zu dieser Arbeit. Ziel dieser Arbeit ist es, zu evaluieren, wie Visualisierung der Interaktion in Multiagentensystemen – mit Blick auf die Plattform Jade – zu realisieren ist.

In der Einleitung wird der Einsatz von Softwarevisualisierung und Visualisierung von Multiagentensystemen im Speziellen motiviert.

Das nächste Kapitel behandelt die theoretischen Grundlagen der Visualisierung im Allgemeinen und der Softwarevisualisierung im Speziellen. Es werden die Modellierung von Softwarevisualisierungssystemen und der Datenfluss in Visualisierungssystemen erläutert. Die Grundlagen von Multiagentensystemen, der FIPA-Kommunikation und der Plattform JADE werden kurz angerissen.

Im folgenden Kapitel wird eine Anforderungsanalyse für den Entwurf eines konkreten Visualisierungssystems für Multiagentensysteme durchgeführt. Der Datenfluss unter den getroffenen Anforderungen sowie der Entwurf werden konkretisiert. Mockups zeigen, wie die Visualisierung aussehen könnte. Aus der bis dahin getroffenen Vorarbeit wird eine Architektur für ein Visualisierungssystem für die Interaktion in Multiagentensystemen entwickelt.

Im letzten Kapitel wird ein Ausblick auf das, was auf diese Arbeit folgen könnte, gegeben.

2.1 Motivation für die Visualisierung von Software

„Der Verstand vermag nichts anzuschauen, und die Sinne nichts zu denken. Nur daraus, dass sie sich vereinigen, kann Erkenntnis entspringen.“(Immanuel Kant, 1781)[Kan] Software ist unsichtbar. Durch die damit verbundene Verbergung der Komplexität der Software sinkt deren Handhabbarkeit. Durch Softwarevisualisierung soll die Software sichtbar gemacht werden – sollen kognitive Bilder im Betrachter erzeugt werden, die das Verstehen fördern [Die03] – und somit die Komplexität handhabbar werden [BE96].

Visualisierung kann als Externalisierung von Gedankenbildern begriffen werden. Sie kann als künstliches Langzeitgedächtnis verwendet werden und zur Kommunikation bei-

tragen. Die Perspektive, die durch Softwarevisualisierung geboten wird, kann neue Denkanstöße geben [PBG98].

Bereits der Quelltext einer Software kann als eine Art der Visualisierung betrachtet werden. Der Vorteil einer nicht textuellen Visualisierung gegenüber einer textuellen liegt in der Möglichkeit begründet, mehrere Dimensionen zur Darstellung von Informationen nutzen zu können, wobei der Informationsgehalt mit jeder weiteren Dimension exponentiell zunimmt [PBG98]. Auch wenn Text nicht 1-dimensional wahrgenommen wird, Layout, Verweise, Kommentare und hin- und herspringen beim Lesen brechen die erste Dimension auf, so ist die Dimensionszahl der nicht textuellen Darstellung dennoch höher und somit auch die transportierbare Informationsdichte größer. Ein jedoch nach Möglichkeit von textueller auf nicht textuelle Visualisierung zu übertragender Effekt ist der, dass Text kaum noch in seiner Repräsentation wahrgenommen wird, sondern nur in seiner Bedeutung. Experimente zeigen, dass textuelle Repräsentationen zum Teil verbreiteten Visualisierungen überlegen sind [Pet95]. Professionalisierung im Lesen von Texten macht diesen Effekt möglich. Es gilt also, bei Visualisierung nicht nur eine gute Darstellung zu treffen, sondern auch die Betrachter im Umgang mit der Visualisierung zu schulen, damit die Präsenz der Syntax in der Wahrnehmung in den Hintergrund tritt. Vermutlich kann aber ein ähnlicher Effekt durch eine geschickte Wahl der Visualisierung erzielt werden, indem die natürliche kognitive Repräsentation von Objekten gewählt und somit die Leistungsfähigkeit des perzeptuellen Systems ausgenützt wird, um die Dekodierung der Syntax zu übernehmen.

Es ist die Hoffnung, dass durch Visualisierung, ganz im Sinne von „Ein Bild sagt mehr als tausend Worte“, der Betrachter der Visualisierung wie der Betrachter eines Gemäldes die Informationen allein durch betrachten in sich aufsaugt. Hier sollte aber die Frage gestellt werden, ob denn jeder Betrachter dieselben tausend Worte in sich aufnimmt [Pet95]. Unbenommen ist wohl, dass Bilder, wie zuvor erwähnt, ein enormes Potenzial als Informationsträger besitzen. Und auch die menschliche Verarbeitungskapazität der visuell dargebotenen Information besitzt großes Potenzial. So ist es uns zum Beispiel ohne weiteres möglich, in einem Megapixelbild einen einzelnen Pixel zu identifizieren [Shn96], gleichsam sich in einer Informationsmenge von über einer Million Informationen zurechtzufinden. Laut [War04] formt das Auge mit dem visuellen Kortex zusammen den Kanal mit der höchsten Bandbreite zum kognitiven System des Menschen. Diesen leistungsfähigen Kanal macht sich die Visualisierung zu Nutze.

Im Maschinenbau, in der Medizin, Physik, Chemie, Architektur und vielen anderen Disziplinen werden von Informatikern entwickelte Systeme zur Visualisierung eingesetzt. Doch in der Informatik selbst wird nur wenig gebrauch von anspruchsvollen Visualisierungssystemen zur Unterstützung des Entwurfs, der Implementierung oder der Wartung von Software gemacht [Die03].

Der Bedarf jedoch besteht. Laut einer E-Mail Umfrage [Die02] unter Forschern in den Bereichen Software Maintenance, Reverse Engineering und Re-Engineering sind 40% der befragten der Meinung, dass Software Visualisierung in ihrem Bereich absolut notwendig ist, 42% sind der Meinung, dass Softwarevisualisierung wichtig aber nicht zwingend sei. Laut [Deu01] würden 40% - 60% der Zeit im Bereich Software Maintenance auf das Verstehen der Software verwendet.

2.2 Motivation für die Visualisierung von Multi Agentensystemen

Multiagentensysteme versprechen die Beherrschung komplexer Anwendungsgebiete, bei denen herkömmliche Technologien an ihre Grenzen stoßen [KNB+03]. Mit der Komplexität des Anwendungsgebietes steigt aber auch die Komplexität der Anwendung. Visualisierung von Multiagentensystemen wird somit schon allein durch die große Komplexität der Systeme motiviert [SN01].

Um ein Multiagentensystem zu verstehen, muss, anders als bei monolithischen Programmen, zum einen das Verhalten eines jeden einzelnen Agenten verstanden werden und zum anderen das Verhalten des Gesamtsystems, das sich aus der Interaktion der Agenten, der von ihnen gebildeten Gesellschaft [KNB+03], ergibt.

Visualisierung von Multiagentensystemen scheint somit wichtiger zu sein als die von monolithischen Systemen [BHS04] und Single-Agent Systemen [NNLC99]. Das Verhalten von Multiagentensystemen scheint sogar komplexer und unvorhersehbarer als das anderer verteilter Systeme zu sein [BHS04], was das Bedürfnis nach einer adäquaten Visualisierung unterstreicht.

3 Grundlagen

In diesem Kapitel werden die Grundlagen für die vorliegende Arbeit vorgestellt. Die Grundlagen gliedern sich dabei in zwei Bereiche. Agentenbezogene Grundlagen und Visualisierungsbezogene Grundlagen. Es wird formuliert, was unter einem Multiagentensystem zu verstehen ist. Die für diese Arbeit relevanten FIPA-Spezifikationen und die Agentenplattform JADE werden vorgestellt. Anschließend werden die Grundlagen der Visualisierung im Allgemeinen, sowie die Grundlagen der Softwarevisualisierung im Speziellen beschrieben. Ein genereller Ansatz zur Modellierung von Softwarevisualisierungssystemen und der Datenfluss in Visualisierungssystemen nach dem Referenzmodell der Visualisierung werden vorgestellt. Geon Diagramme werden als Grundlage der Visualisierung in dieser Arbeit vorgestellt.

3.1 Agenten

3.1.1 Multiagentensysteme

Multiagentensysteme sind verteilte Systeme, deren Komponenten, die Agenten, koordiniert gemeinsam ein gegebenes Problem lösen [LA95]. Nach [Syc98] sind folgende Merkmale charakteristisch für Multiagentensysteme:

- Jeder Agent hat für die Lösung eines Problems unzureichende Möglichkeiten oder Informationen, also eine eingeschränkte Perspektive
- Es gibt keine globale Systemkontrolle
- Die Daten sind dezentralisiert
- Die Berechnung ist asynchron

Agenten in einem Multiagentensystem haben eine beschränkte, lokale Perspektive [NNLC99] auf das Gesamtsystem und kennen den Nutzen des gesamten Systems nicht [RZ94].

Ein Multiagentensystem ist also derart zu verstehen, dass der Gesamtnutzen des Systems dadurch entsteht, dass jeder einzelne Agent egoistisch seine eigenen Ziele verfolgt [RZ94].

Der Einsatz von Multiagentensystemen soll es ermöglichen, komplexe Anwendungsgebiete besser als bisher zu beherrschen, was laut [KNB+03] in zahlreichen Simulationen und Fallstudien belegt wird.

3.1.2 FIPA

Das „IEEE Computer Society Foundation for Intelligent Physical Agents Standards Committee“ (FIPA SC) wurde gegründet, um anerkannte Standards, empfehlenswerte Praxis und Leitfäden für agentenbezogene Technologie zu entwickeln.

Die FIPA Spezifikationen sind in 5 Themengebiete unterteilt: *Applications, Abstract Architecture, Agent Communication, Agent Management* und *Agent Message Transport* [FIP].

Das Themengebiet *Agent Communication* setzt sich mit den Themen *Agent Communication Language Messages* (ACL Nachrichten), *Message exchange interaction protocols, Speech act theory-based communicative acts* und *Content language representations* auseinander.

Die Spezifikation der abstrakten Syntax von FIPA ACL Nachrichten besagt, dass die Struktur einer konformen Nachricht aus einem oder mehreren Parametern besteht. Der einzige zwingende Parameter ist *Performative*, sämtliche anderen sind optional. Mögliche Parameter sind: *Performative, Sender, Receiver, Reply-to, Content, Language, Encoding, Ontology, Protocol, Conversation-id, Reply-with, In-reply-to, Reply-by* und beliebige weitere nicht FIPA spezifizierte Parameter, gekennzeichnet durch den Begin mit *X-* [FIP02a].

Agenten verfolgen nach der FIPA Spezifikation ihre eigenen Ziele durch die Kommunikation mit anderen Agenten. Agenten wählen *Communicative acts* anhand des erwarteten Ergebnisses in Bezug zu ihren Zielen. *Speech act theory-based communicative acts* spezifizieren ein Sammlung von FIPA konformen *Communicative acts*, die in einer ACL Nachricht im Parameter *Performative* gesetzt werden können. Um FIPA konforme Agenten zu entwickeln, muss zwingend der *Communicative act Not-understood* implementiert werden. Alle anderen *Communicative acts* sind optional. Wird jedoch ein weiterer *Communicative act* implementiert, so muss er in Übereinstimmung mit der in den FIPA Spezifikationen definierten Semantik des *Communicative act* implementiert werden, da der Agent ansonsten seine FIPA-konformität verliert. Standardisierte *Communicative acts* sind: *Accept proposal, Agree, Cancel, Call for Proposal, Confirm, Disconfirm, Failure, Inform, Inform If, Inform Ref, Not understood, Propagate, Propose, Proxy, Query if, Query ref, Refuse, Reject proposal, Request, Request when, Request whenever* und *Subscribe*. Als Beispiel sei hier der *Communicative act Request* erläutert. Ein Agent sendet eine Nachricht mit Performative *Request*, wenn er den Empfänger dazu anhalten will, eine Aktion, zum Beispiel einen weiteren *communicative act*, durchzuführen. Der Inhalt der Nachricht sollte die nachgefragte Aktion in einer für den Empfänger verständlichen Sprache beschreiben. [FIP02c]

„Message exchange interaction protocols“ sind vorgefertigte Interaktionsprotokolle [FIP]. Die folgenden FIPA Interaktionsprotokolle sind spezifiziert: *Request, Query, Request When, Contract Net, Iterated Contract Net, English Auction, Dutch Auction, Brokering, Recruiting, Subscribe* und *Propose*. Als einfaches Beispiel sei hier kurz das Interaktionsprotokoll *Request* erwähnt. Das *FIPA Request Interaction Protocol*[FIP02d] ermöglicht es, dass ein Agent einen anderen zu einer Aktion auffordert. Dazu sendet der auffordernde Agent *request* an den aufgeforderten Agent. Dieser entscheidet, ob er die

Aufforderung annimmt und sendet seinerseits ein *refuse* oder, falls bei Einverständnis notwendig, ein *agree* zurück. Wurde die Aufforderung angenommen, so sendet der aufgeforderte Agent *failure*, *inform-done* oder *inform-result*, je nachdem ob die Aufforderung fehlschlug oder durchgeführt wurde, an den auffordernden Agenten. Durch *inform-result* wird zusätzlich das Resultat der geforderten Aktion mitgeteilt.

Content language representations behandelt verschiedene Arten von Syntax um den Inhalt einer Nachricht zu beschreiben. Es wird hier nicht weiter darauf eingegangen.

Das Themengebiet *Agent Message Transport* spezifiziert den Transport der ACL Nachrichten und die Repräsentation derselben. Die diesem Gebiet zugeordnete *FIPA Agent Message Transport Service Specification* spezifiziert den Nachrichtentransport für Agenten auf FIPA Agentenplattformen. Das Referenzmodell für den Agenten Nachrichten Transport beinhält drei Ebenen. Das *Message Transport Protocol* (MTP) ist für den physikalischen Transfer der Nachrichten zwischen zwei *Agent Communication Channels* (ACCs) zuständig. Der Message Transport Service (MTS) wird den Agenten einer Agentenplattform von derselben als Service zum Transport von FIPA ACL Nachrichten, auch inter-plattform, zur Verfügung gestellt. Die ACL repräsentiert den Payload der von MTS und MTP beförderten Nachrichten. Eine nach dieser Spezifikation transportierte Nachricht besteht aus einer in einen Umschlag, den so genannten *Message Envelope* gesteckten ACL Nachricht. Ein *Message Envelope* enthält mindestens die Parameter *To*, *From*, *Date* und *Acl-representation* [FIP02b].

3.1.3 JADE

Das Java Agent Development Framework (JADE)[JAD] ist ein Framework für die Entwicklung von FIPA konformen Multiagentensystemen. Es besteht aus einem Package zur Entwicklung von Java Agenten und einer FIPA konformen Agentenplattform. FIPA konforme Agentenplattform bedeutet im Wesentlichen die Existenz eines Agent Management Systems (AMS), eines Directory Facilitator (DF) und eines Message Transport Systems auch genannt Agent Communication Channel (ACC). Eine JADE Agentenplattform kann auf mehrere Hosts verteilt werden. JADE ermöglicht neben der Implementierung von Agenten auch die Implementierung von Tool Agenten, die Methoden zum Zugriff auf das AMS anbieten und somit eine Vielzahl an Analysemöglichkeiten zur Verfügung stellen. Eine Schnittstelle zur Kommunikation mit Java Applikationen wird ebenfalls geboten.

3.2 Visualisierung

3.2.1 Visualisierung allgemein

Visualisierung ist die Darstellung abstrakter Sachverhalte, von Informationen im Allgemeinen. Die Definition wandelte sich im Laufe der Zeit von „constructing a visual image in the mind“ zu „a graphical representation of data or concepts“ [War04]. Im Kern geht es aber darum, die kognitiven Fähigkeiten des Verstandes mit externen Mitteln zu verstärken. Auf der einen Seite ist da der Menschen, der über das visuelle System so-

viel an Informationen aufnehmen und vorverarbeiten kann wie über kein anderes seiner sensorischen Systeme. Auf der anderen Seite die Rechenleistung und die Informationsvielfalt unserer Computersysteme und Netze. Der Nutzen der Visualisierung liegt nun darin, dass eine große Bandbreite an Informationen vom Menschen schnell interpretiert werden kann, sofern die Visualisierung adäquat ist. Visualisierung bietet laut [War04] die folgenden Vorteile:

- Unterstützung des Verstehens gewaltiger Informationsmengen
- Wahrnehmung von emergenten Eigenschaften, die nicht vorauszusehen sind
- Schnelles Entdecken von Fehlern in den visualisierten Daten, wodurch die Qualitätskontrolle unterstützt wird
- Förderung des Verständnisses des Charakters der Daten auf und zwischen allen Abstraktionsebenen
- Förderung der Hypothesengenerierung

Nach [War04] besteht der Prozess der Visualisierung aus 4 Phasen:

- Sammeln und speichern der Daten
- Transformieren der Daten in eine für den Menschen verständliche Form
- Darstellen der Daten
- Perzeption und kognitive Verarbeitung der Daten durch den Menschen

3.2.2 Softwarevisualisierung

Softwarevisualisierung ist die Visualisierung verschiedener Aspekte von Software. Es werden Computerprogramme, zugehörige Dokumentationen und Daten grafisch dargestellt, um die mentale Repräsentation, der Funktionsweise der Software, des Betrachters zu verbessern [Die02]. In [PBS98] wird Softwarevisualisierung wie folgt definiert: Softwarevisualisierung bedient sich der Mittel der Typographie, des Grafik Designs, der Animation und der Kinematographie in Verbindung mit moderner Human-Computer-Interaction und Computergrafik, um sowohl das Verständnis, als auch den effektiven Einsatz von Software zu unterstützen.

Nach [BE96] sind grundsätzlich drei Eigenschaften von Software visualisierbar:

- Die Struktur der Software,
- das Laufzeitverhalten der Software
- und der Code selbst.

Die Struktur der Software lässt sich beispielsweise durch gerichtete Graphen, die die Beziehungen der Komponenten untereinander angeben, darstellen. Das Laufzeitverhalten der Software lässt sich zum Beispiel durch Animationen darstellen und der Code einer Software durch Pretty Printer visualisieren.

Nach [PBS98] kann das Gebiet der Softwarevisualisierung in folgende Bereiche gegliedert werden:

- Algorithmen Visualisierung: Visualisierung von Software auf höherer Abstraktionsebene
  - Statische Algorithmen Visualisierung
  - Algorithmen Animation
- Programm Visualisierung
  - Statische Code Visualisierung
  - Statische Daten Visualisierung
  - Daten Animation
  - Code Animation

3.2.3 Modellierung von Softwarevisualisierungssystemen

Da es im Bereich der Visualisierung keine allumfassende Lösung gibt [Pet95] und von Anwendern kleine Tools gegenüber allumfassenden Lösungen bevorzugt werden [HGSG90], muss zunächst evaluiert werden, welche Art von Softwarevisualisierung unterstützt werden soll. Soll mehr unterstützt werden als durch ein einzelnes Tool geleistet werden kann, so bietet es sich an, wie in [NNLC99], einen Toolkit zusammenzustellen, der die Handhabbarkeit durch ein gemeinsames Look and Feel erleichtert.

Softwarevisualisierungssysteme lassen sich in 5 Dimensionen charakterisieren[MFM03][MMC02]

- Aufgabe: Zweck, dem die Visualisierung dient.
- Zielgruppe: Personengruppe, die von der Visualisierung profitiert.
- Ziel: Aspekte der Software, die visualisiert werden.
- Medium: Ort der Darstellung.
- Repräsentation: Art und Weise der Darstellung.

Am Anfang der Überlegung steht die Frage, welchem Zweck die Visualisierung dienen und an wen sie sich wenden soll. Hieraus ergibt sich die Perspektive, unter der die Software visualisiert werden soll. Unter den Beschränkungen des verfügbaren Mediums lässt sich dann eine adäquate Darstellungsweise wählen.

Aus der HCI ist bekannt, dass Systeme nach den Tasks der Benutzer designed werden sollen [PBG98]. Es ist somit grundlegend bei der Gestaltung zu berücksichtigen, welche Aufgaben unterstützt werden sollen, wie sie unterstützt werden können, aber auch wie

das entwickelte Tool die Aufgaben der Anwender verändert [PBG98]. Welche Aufgaben aber sind es, die durch ein Softwarevisualisierungssystem unterstützt werden sollen? Generell kann gesagt werden, dass es Zweck der Softwarevisualisierung ist, beim Betrachter mentale Bilder auszulösen und dadurch das Verständnis für die visualisierte Software zu erhöhen [Die03]. Es ist wichtig, die visualisierten Informationen möglichst gut auf die individuellen Bedürfnisse der unterschiedlichen Tasks anzupassen, da eine Visualisierung nur dann Wert hat, wenn sie dem Betrachter die Information zeigt, die er für seinen Task benötigt [Pet95].

Gründe der Softwarevisualisierung können zum Beispiel die folgenden sein:

- Die Unterstützung des Softwareentwicklungsprozesses. Durch die Visualisierung der Software zur Entwicklungszeit kann ein allen Entwicklern gemeinsames mentales Bild der Software geschaffen, sowie der Überblick über das gesamte System gewährleistet werden. Durch die Visualisierung der Software kann die Kommunikation zwischen den Entwicklern und das Verständnis der Software verbessert werden.

- Testen der Software. Das Testen der Software kann mit Hilfe von Visualisierung durch visuelle Bestätigung aufgestellter Hypothesen unterstützt werden.

- Durch die Visualisierung kann auf effiziente Weise das Verstehen von Algorithmen, Architekturen, Entwurfsmustern u.a. unterstützt werden. Softwarevisualisierung bietet sich damit für die Lehre an.

- Dokumentieren der Software. Heute ist das, was die XP Methode „the source code is the design“ formalisiert, in vielen Projekten De Facto Standard [Deu01], mit all seinen Vorzügen aber auch Gefahren, die die Nicht-Dokumentation des Designs mit sich bringt. Es ist also eine durchaus sinnvolle Möglichkeit, die Dokumentation des Designs durch Visualisierung zu generieren.

- Debugging, also das Verstehen des Programms, generieren und evaluieren von Hypothesen bezüglich der Problemstellung, Lösen des Problems und anschließendes Testen des Programms [HGSG90], kann durch Visualisierung unterstützt werden.

- Reverse Engineering, dass Analysieren eines Systems, um seine Komponenten und deren Beziehungen untereinander zu identifizieren und in anderer, eventuell abstrakterer, Form darzustellen [Die02], ist eine weitere mögliche Aufgabe der Softwarevisualisierung.

- Auch das mit dem Reverse Engineering verwandte und auf ihm basierende Re-Engineering, also das Abändern und Neuimplementieren von Systemen [Die02], ist eine weitere mögliche Aufgabe der Softwarevisualisierung.

- Im Kundenkontakt oder bei der Kommunikation mit dem Management kann Softwarevisualisierung auch sinnvoll eingesetzt werden, um auf hohem Abstraktionsniveau die Funktionsweise der Software zu kommunizieren.

Zielgruppen der Softwarevisualisierung sind zum Beispiel unter anderem Studenten und Lernende, Entwickler, Projektleiter und Kunden.

Mögliche Ziele der Visualisierung sind zum Beispiel die Architektur und das Design der Software, Algorithmen, Source Code, Daten, Ausführungs- und Traceinformationen (abstrakt oder konkret), Metriken, Dokumentation und die Evolution der Software.

Bei der Wahl der Medien, durch die die Softwarevisualisierung stattfindet, gibt es leider einen Interessenkonflikt. Auf der einen Seite steht die Präsentation durch teure und exotische, aber dafür sehr effiziente Systeme, wie VR-Umgebungen und auf der anderen Seite die Präsentation durch weit verbreitete, also relativ günstige Systeme wie hochauflösende Bildschirme und Papier.

Einige Möglichkeiten der Repräsentation sind:

- Das Highlighten von Quellcode, um die rein textuelle Repräsentation (1-dimensional) um weitere Informationsdimensionen zu bereichern und somit den Quellcode schneller verständlich zu machen.
- UML Diagramme, als verbreitete und mächtige Methode zur Repräsentation und Kommunikation von Software. UML Diagramme haben zwar eine schnell schreibbare, jedoch nicht allzu leicht lesbare Notation.
- Geon Diagramme [ITW01] als Repräsentationsform beliebiger Grafen. Im Gegensatz zu der Standardnotation der UML sind Geon Diagramme schwer zu schreiben aber dafür leicht zu lesen.
- „Line Representation“ und „Pixel Representation“ [BE96] zur schnellen Übersicht über sehr große Informationsmengen geeignet. Informationen werden als Linie oder Pixel auf dem Bildschirm dargestellt.
- Jacobson's Interaktionsdiagramme als weit verbreitete Notation der Interaktion zwischen Objekten [JCJO92].
- „Execution Pattern“ als in [PLVW98] vorgeschlagene Repräsentation für Traces in Objektorientierten Programmen.

3.2.4 Datenfluss in Visualiserungssystemen

Nach dem Referenzmodell für Visualisierung [MMC02] ist der Vorgang der Visualisierung als eine Folge von Abbildungen zu verstehen, in die der Benutzer eingreifen kann. Die an der Quelle anfallenden Informationen werden in ein für die Visualisierungssoftware handhabbares Datenformat transformiert und eventuell mit zusätzlichen Informationen, z.B. einem Zeitstempel, angereichert. Die Daten werden dann auf die visuellen Strukturen abgebildet und dabei eventuell gefiltert oder verdichtet. Zuletzt, soweit es das Referenzmodell betrifft, werden die Visuellen Strukturen dargestellt. Werden die Grenzen des Softwaresystems überschritten, so geht die Folge der Abbildungen weiter (siehe Abbildung 3.1). Der Mensch nimmt die Darstellung perzeptuell wahr, perzeptuelle Daten, und verarbeitet sie zu kognitiven Repräsentationen. Die Überschreitung der Grenzen

des Softwaresystems ist wichtig, um auf die Relevanz einer adäquaten Darstellung für den Wert der Visualisierung explizit hinzuweisen.

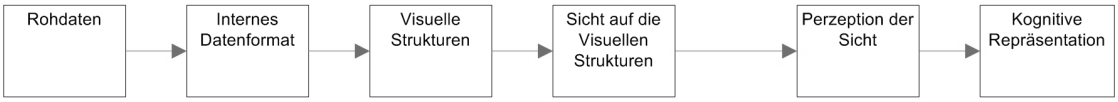

Abbildung 3.1: Am Referenzmodell [MMC02] orientiertes Modell der Transformation der Daten beim Vorgang der Visualisierung

Ausgehend von den an der Quelle anfallenden Daten und den Theorien über kognitive Repräsentationen und Perzeption kann nun angefangen werden, die „Abbildungsstrategie“ aufzubauen.

3.2.5 Geon Diagramme

Mit Blick auf eine möglichst schnelle kognitive Aufnahme von Visualisierungen sind die Kognitionswissenschaften zu rate zu ziehen. Theorien zum Objekterkennen lassen sich in zwei Kategorien einteilen. Bild- und Strukturbasiert [ITW01]. Biederman entwickelte einen strukturbasierten Ansatz zur Erklärung der Objekterkennung. Nach diesem Ansatz setzen sich alle Objekte aus einem Alphabet von 36 Primitiven (siehe Abbildung 3.2), so genannten Geons (Geometric Items) zusammen [ITW01]. Geons sind einfache dreidimensionale Objekte. Grundprinzip des Erkennens komplexer dreidimensionaler Objekte ist es, die Dekomposition des komplexen Objektes in seine Primitive und deren Relationen zueinander durchzuführen. Ergebnis der Dekomposition ist die „Geon Structioral Description“ (GSD) bestehend aus den Geons, ihren Attributen (Farbe, Skalierung etc.) und den Beziehungen der Geons untereinander. Die GSD ist dafür verantwortlich, dass ein Objekt aus allen Perspektiven als das Gleiche erkannt wird, da die Dekomposition des Objektes immer zu der gleichen GSD führt [IW00].

Da diese Theorie nun besagt, dass der Mensch ein ausgeprägtes System zur Verarbeitung von Geons besitzt, sollte es die Effizienz von Diagrammen wesentlich steigern, wenn dieses System genutzt, sprich die GSD zu Repräsentationszwecken verwendet wird [IW00]. In [ITW01] werden folgende Regeln zur Erstellung von Geon Diagrammen aufgestellt:

- Geons als 3D-Primitive

  - G1: repräsentiere die Hauptsystemteile (Entities) als simple Geons
  - G2: repräsentiere Verbindungen zwischen Entities als Verbindung zwischen Geons
  - G3: repräsentiere kleine Subkomponenten als Geon Anhänge an Geons
  - G4: schattiere Geons um ihre 3D Form zu betonen
  - G5: repräsentiere sekundäre Attribute von Entities und Beziehungen über Farbe und Textur und Symbole auf der Oberfläche der Geons

- Gutes 2D Layout
  - L1: Alle Geons sollten in der gewählten Perspektive sichtbar sein
  - L2: Layout überwiegend in der orthogonalen Ebene zu Blickrichtung
  - L3: Verbindungen zwischen Geons klar erkennbar machen

- Semantik Regeln
  - SM1: Gleichheit und Generalisierung: Gleiche Form für Elemente gleicher Art
  - SM2: Geon A auf B suggeriert A abhängig B (Gravitizität)
  - SM3: Einschluss: A enthält B, syntaktisch: interne Komponente verbunden mit selben Geon außerhalb
  - SM4: Um multiple Assoziationen darzustellen: Serie von Verbindungen
  - SM5: Stärke einer Verbindung: Je dicker desto stärker
  - SM6: Sequenz: Geons in einer Reihe angeordnet werden zur Metapher für Kette von Operationen oder andere Lineare Struktur
  - SM7: Symmetrie: Symmetrische Informationsstrukturen werden symmetrisch dargestellt
  - SM8: Zentral und peripher: ist eine Komponente von zentraler Bedeutung für die Struktur wird sie zentral positioniert (und ihre Verbindungen)
  - SM9: Size: Größere Komponenten in Gebiete abbilden

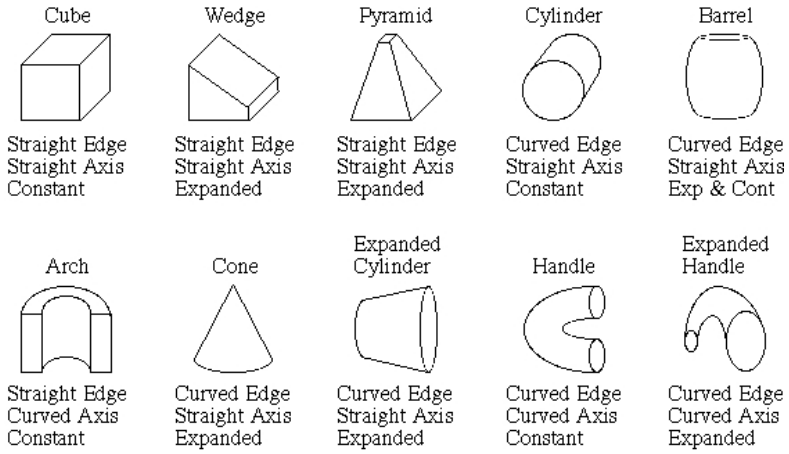

Abbildung 3.2: Auswahl aus dem Alphabet der 36 Geons. Geons sind die Primitive der Dekomposition von komplexen Objekten (Bildquelle: [Kir01])

4 Visualisierungssystem für die Interaktion in Multiagentensystemen

In diesem Kapitel werden zunächst die Anforderungen an ein System zur Visualisierung der Interaktion in Multiagentensystemen evaluiert. Nach der Anforderungsanalyse folgen Überlegungen für den Entwurf eines konkreten Systems. Mockups sollen inspirieren, wie die grafische Umsetzung der Visualisierung aussehen könnte. Anschließend wird eine auf den Anforderungen und den Entwurfsentscheidungen basierende Architektur für ein System zur Visualisierung der Interaktion in Multiagentensystemen vorgestellt.

4.1 Anforderungsanalyse

Um eine große Verbreitung zu erreichen, soll das System auf Standard Computersystemen laufen und eine möglichst große Anpassbarkeit und Erweiterbarkeit durch die Benutzer erlauben. Zu diesem Zweck soll die Architektur aus austauschbaren Komponenten bestehen und an den Schnittstellen nach Möglichkeit auf Standards zurückgreifen. Auch sollte bei externen Schnittstellen und Exportformaten nach Möglichkeit auf Standards zurückgegriffen werden. Die Komponenten der Architektur lassen sich an dem Datenfluss des Referenzmodells für Visualisierung (siehe Kapitel 3.2.4) orientieren.

Den Gedanken zur Modellierung in Kapitel 3.2.3 folgend wird nun evaluiert, wie ein System zur Visualisierung der Interaktion in Multiagentensystemen zu modellieren ist. Dazu wird sich zunächst mit den einzelnen Dimensionen von Softwarevisualisierung im Lichte der Agententechnik auseinander gesetzt.

4.1.1 Aufgabe

Zuerst wird der Zweck der Visualisierung erörtert. Ziel der Konstruktion eines Systems zur Visualisierung von Multiagentensystemen ist es in diesem Fall, die Verständlichkeit von Multiagentensystemen insbesondere zur Entwicklungszeit zu erhöhen und die Kommunizierbarkeit des Multiagentensystems zu steigern. Da insbesondere die Interaktion der Agenten untereinander für die Komplexität von Multiagentensystemen verantwortlich ist und gegenüber monolithischen Systemen eine weitere Fehlerklasse produziert, sollte diese im Mittelpunkt der Betrachtung stehen.

Es soll ermöglicht werden, bei Ausführung der Software und post mortem die Interaktion zwischen Agenten und Agentengruppen durch Nachrichten, Konversationen und Protokolle sowohl aus der Perspektive der Architektur als auch aus Perspektive der Interaktion zu visualisieren.

Da die zugrunde liegende Motivation des Benutzers für eine Visualisierung während der Entwicklung hochgradig individuell ist, sollte die Form der Visualisierung stark individualisierbar sein. So sollte es möglich sein, Hypothesen des Benutzers über die Interaktion der Agenten explizit zu formulieren und durch das Visualisierungssystem überprüfen und darstellen zu lassen, um die persönliche Strategie der Softwareexploration [HGSG90] zu unterstützen. Als Beispiel für eine solche Hypothese sei hier die Annahme, dass zwei Agenten niemals miteinander kommunizieren genannt.

Um das Verständnis des visualisierten Softwaresystems noch zu verstärken soll das Visualisierungssystem hochgradig interaktiv sein. Layouts sollen vom Benutzer veränderbar sein, Elemente zu Gruppen abstrahiert werden können und Informationsdimensionen auf Visualisierungsdimensionen möglichst frei zuweisbar sein. Um die Exploration der Visualisierung zu unterstützen, soll das Markieren von beliebigen Stellen in der Visualisierung möglich sein, die Visualisierung frei beweglich und zoombar – verbunden mit semantischem Zoom – sein.

4.1.2 Zielgruppe

Die Personengruppe, die in erster Linie von der Visualisierung profitieren soll, ist die der (studentische) Entwickler und der Projektleitung. Es sollte also einerseits ein hohes Abstraktionsniveau erreicht werden, um auf hohem Abstraktionsniveau über die Software zu kommunizieren, andererseits aber auch das kleinste Detail verfügbar sein, um die Entwicklung der Software zu unterstützen.

4.1.3 Ziel

Im Zentrum der Betrachtung steht die Interaktion der Agenten und Agentengruppen durch Nachrichtenaustausch, Konversation und Protokolle. Es ist sowohl der dynamische Aspekt der Interaktion darzustellen als auch der statische innerhalb einer Architekturbeschreibung des Systems. Es soll also zunächst zwei Perspektiven auf das Softwaresystem geben: eine architekturzentrierte Perspektive und eine interaktionszentrierte Perspektive.

Die architekturzentrierte Perspektive soll die Beziehung der Agenten in den Vordergrund stellen und nur mit zweiter Priorität die einzelnen ausgetauschten Nachrichten darstellen.

Die interaktionszentrierte Perspektive soll Aspekte der Architektur vernachlässigen und sich ganz der Aufgabe widmen, die ausgetauschten Nachrichten in ihrer zeitlichen Abfolge und den verschiedensten Aspekten darzustellen.

Durch Interaktion zwischen der architekturzentrierten und der interaktionszentrierten Perspektiven soll ein kohärentes Gesamtbild der Interaktion des zu visualisierenden Multiagentensystems entstehen.

4.1.4 Medium

Als Medium kommen, so eine weite Verbreitung des Systems gewünscht ist, hochauflösende Monitore und Beamer, sowie Papier in Frage. Mit Blick auf die Darstellung auf Papier

ist zu berücksichtigen, dass meist nur schwarz-weiß gedruckt wird. Es gilt also der Farbdimension in der Visualisierung keine zu prominente Rolle zuzuordnen, sofern sich die Perspektive für den Ausdruck eignet.

4.1.5 Repräsentation

Die Repräsentation sollte möglichst schnell interpretierbar sein und möglichst viele Informationen auf einmal übermitteln. Irrelevante Informationen sollten nicht dargestellt werden, da sie die relevanten Informationen verdecken.

Im Fall der architekturzentrierten Perspektive bieten sich hier Geon Diagramme an, da sie mit Blick auf schnelle Perzeption entwickelt worden sind. Um vom höchsten Abstraktionsniveau bis ins kleinste Detail Informationen zu erhalten wird grafisches Zoomen in der Kombination mit semantischem Zoomen verwendet.

Im Fall der interaktionszentrierten Perspektive gibt es eine Vielzahl darzustellender Entitäten, die Nachrichten. Um die Masse an Informationen in den Griff zu bekommen, wird eine von der Idee der „Line Representation“ und „Pixel Represenation“ [BE96] inspirierte Darstellung verwendet. Verbunden mit semantischem Zoom sind selbst hier die Vorteile der Geon Diagramme verwendbar.

Um die manuelle Exploration der Software zu unterstützen, muss die benötigte Information sichtbar sein. Idealerweise sollte nur die notwendige Information sichtbar sein. Jedenfalls sollte es vermieden werden, dass die notwendige Information durch irrelevante Information verdeckt wird. Hierzu ist zu berücksichtigen, dass hervorgehobene Information die restlichen Informationen verdeckt. Weiter ist zu berücksichtigen, dass grafische Darstellung ohne Bedeutung trotzdem als Information interpretiert werden kann und somit zur Verwirrung des Betrachters beiträgt. Der Benutzer verfügt für gewöhnlich über Wissen, mit Hilfe dessen er unterscheiden kann was relevant ist und was nicht [Die02]. Um die manuelle Wahl dessen, was notwendige Information und was irrelevant ist, zu ermöglichen, sollte der Benutzer bis zu einem gewissen Grad selber formulieren können, welche Informationsdimension wie auf welche Darstellungsdimension abgebildet wird. Des Weiteren sollte Filterung und Abstraktion durch Vorgaben des Benutzers ermöglicht werden. Durch freie Manipulation des Layouts, zum Beispiel bei der architekturzentrierten Visualisierung, ist der Benutzer in der Lage, die Visualisierung seinen Bedürfnissen maximal anzupassen. Da insbesondere das Anpassen des Layouts, aber auch die anderen Konfigurationsmaßnahmen, sehr zeitaufwändig sein können, muss es möglich sein, Abbildungsvorschriften, Abstraktionen, Hypothesen und Layout zu speichern. Es sollte zum einen jede Konfiguration einzeln, aber auch die gesamte Konfiguration der Visualisierung als eine Einheit speicherbar sein. Insbesondere fördert dies auch den Wiedererkennungswert einer folgenden Visualisierung desselben Systems. Jegliche Maßnahme des Benutzers zur Anpassung der Visualisierung soll von ihm auch wieder rückgängig gemacht werden können, damit Hemmungen vor Experimenten mit der Visualisierung abgebaut werden und die Exploration der Visualisierung gefördert wird.

Um die Masse an relevanter Information in den Griff zu bekommen, ist es einerseits notwendig eine möglichst schnell interpretierbare Repräsentationsform zu wählen und andererseits eine Repräsentationsform, die möglichst viele Informationen auf einmal

darstellt. Um die Navigation in der Visualisierung zu ermöglichen, soll stufenloses grafisches und semantisches Zoomen und Verschieben ermöglicht werden. Es soll also möglich sein, von einer komplett auf dem Monitor darstellbaren Visualisierung bis hinunter auf das kleinste Informationsdetail zu zoomen. Um den Kontext während des Zoomens und Verschiebens nicht zu verlieren, gibt es unter anderem die Möglichkeiten, die Visualisierung in Fischaugenoptik oder einer hyperbolischen Perspektive vorzunehmen [HMM00], was jedoch große Probleme beim Layout der Visualisierung nach sich zieht. Deshalb ist die Möglichkeit, eine zweite Perspektive als Übersichtskarte zu verwenden vorzuziehen. Auf der Übersichtskarte soll der momentan betrachtete Bereich markiert werden. Ergänzt werden kann diese Methode durch Hinweise auf den umgebenden Kontext am Visualisierungsrand. Um möglichst schnell die gesuchte Information zu finden, sollte es eine Suchfunktion geben. Die Suchparameter sollten speicherbar sein, damit komplexe Suchanfragen schnell wiederholt durchführbar sind. Um die Navigation weiter zu erleichtern, sollte es möglich sein, an beliebiger Stelle der Visualisierung Notizen oder simple Markierungen zu platzieren.

4.2 Entwurf

Die hier getroffenen Entwurfsentscheidungen bewegen sich entlang des im Referenzmodell der Visualisierung beschriebenen Datenflusses.

4.2.1 Quelle

Datenquelle eines Systems zur Visualisierung der Interaktion in Multiagentensystemen sind, da die Interaktion der Agenten visualisiert werden soll, Traces von Programmabläufen in Multiagentensystemen, also von verteilten Systeme.

Das Problem, das bei der Erstellung von Traces besteht, ist, dass bei jeder Messung der gemessene Gegenstand, hier das Agentensystem, beeinflusst wird. So werden durch das den Trace produzierende Programm Ressourcen der das Agentensystem verwaltenden Plattform verbraucht. Insbesondere kommt es dann zu einer Beeinflussung des Verhaltens, wenn das Nachrichtensystem der Agentenplattform von dem den Trace erzeugenden System verwendet wird. So ist als sicher anzunehmen, dass der Trace, der durch Messungen gewonnen wird, nicht das Programmverhalten widerspiegelt, dass ohne Messungen aufgetreten wäre.

Um diesen Einfluss möglichst gering zu halten, sollte die Messung minimalinvasiv vorgenommen werden. Dies bedeutet, dass nach Möglichkeit vorhandene Debugging Schnittstellen verwendet werden sollten und dass auf jedem Rechensystem, das Agenten verwaltet, lediglich die Daten gesammelt werden, nicht aber verarbeitet werden sollten. Stattdessen sollte ein dedizierter Rechnerknoten eingerichtet werden, der sämtliche Traces sammelt und aufbereitet. Um die Kapazitäten des Systems schon an dieser Stelle zu entlasten, sollte es nach Möglichkeit dem Benutzer freigestellt sein, bereits durch die Nutzung der Schnittstellen am Agentensystem die Tracedaten zu filtern.

Als potenzielle Informationsquellen wird die Menge der Agentenplattformen auf die Menge der FIPA konformen Agentenplattformen beschränkt. Die gewonnenen Daten

beschreiben also FIPA konform die anfallenden Nachrichten.

Da nun von den FIPA Spezifikationen ausgegangen werden kann, können die möglichen Informationsdimensionen gut beschrieben werden. Mögliche Informationsdimensionen die an der Datenquelle anfallen sind basierend auf der FIPA Spezifikation der ACL Nachrichten [FIP02a]:

- *Performative*: Typ des *Communicative act*, bezeichnet den Typ des *Communicative act* der Nachricht
- *Sender*: Teilnehmer der Kommunikation, bezeichnet den Sender der Nachricht
- *Receiver*: Teilnehmer der Kommunikation, bezeichnet den oder die Empfänger der Nachricht
- *Reply-to*: Teilnehmer der Kommunikation, bezeichnet den Agenten, zu dem folgende Nachrichten des Konversationsthreads geschickt werden sollen
- *Content*: Inhalt der Nachricht, bezeichnet den Inhalt der Nachricht
- *Language*: Beschreibung des Inhalts, bezeichnet die Sprache in der der Inhalt der Nachricht verfasst ist
- *Encoding*: Beschreibung des Inhalts, bezeichnet die Kodierung der Sprache des Inhalts
- *Ontology*: Beschreibung des Inhalts, bezeichnet die für den Inhalt verwendete Ontologie und gibt somit dem Inhalt Bedeutung
- *Protocol*: Kontrolle der Konversation, bezeichnet das Interaktionsprotokoll in dessen Zuge diese Nachricht versendet wird. Jeder von Null verschiedene Wert weist die Nachricht als zu einer Konversation zugehörig aus, womit folgende Regeln befolgt werden müssen:
  - Der Initiator des Protokolls muss dem *Conversation-id* Parameter einen von Null verschiedenen Wert zuweisen
  - Jede Antwort auf die Nachricht im Zuge des Protokolls soll den initialen Wert für den *Conversation-id* Parameter erhalten
  - Der timeout Wert des *Reply-by* Parameters bezeichnet den Zeitintervall in dem der sendende Agent spätestens eine Antwort konform des Protokolls erwartet
- *Conversation-id*: Kontrolle der Konversation, bezeichnet die Konversation zu der die Nachricht gehört
- *Reply-with*: Kontrolle der Konversation, bezeichnet Ausdruck der bei Antwort verwendet werden soll, damit die Antwort eindeutig dieser Nachricht zugeordnet werden kann

- *In-reply-to*: Kontrolle der Konversation, Ausdruck der die Nachricht eindeutig als Antwort auf eine vorhergehende Nachricht auszeichnet
- *Reply-by*: Kontrolle der Konversation bezeichnet den Zeitintervall bis zu dem der sendende Agent auf Antwort wartet
- sowie beliebige weitere nicht FIPA spezifizierte Parameter, gekennzeichnet durch den Begin mit *X-*

Des Weiteren werden von der Datenquelle, also dem Multiagentensystem, noch Informationen dazu, welche Agenten existieren und wann eine Nachricht, nach lokaler Zeit des Agenten, versendet bzw. empfangen wurde, benötigt. Zu jeder Nachricht gibt es also zwei oder mehr Ereignisse, das Ereignis des Versendens und Ereignisse des Empfangens.

4.2.2 Aufbereitete Daten

Die durch den Trace gewonnenen Daten werden an zentraler Stelle in ein für das Visualisierungsystem einfach handhabbares Format gebracht und um weitere Informationen, zum Beispiel einen „globalen“ Zeitstempel, erweitert. Wird beabsichtigt die gewonnenen Daten durch Datamining Technologien zu analysieren, so liegt es auf der Hand, die Daten an dieser Stelle in ein geeignetes Datenbankformat zu bringen.

Da hier von Multiagentensystemen die Rede ist, geht es um verteilte Systeme. Kein Teil des Systems hat allumfassende Informationen über den gesamten Systemzustand. Da, für den Fall das eine der Visualisierungsdimensionen den zeitlichen Ablauf beinhalten soll, aber so etwas wie ein Gesamtüberblick gebraucht wird, müssen die verfügbaren Informationen an zentraler Stelle gesammelt werden. Jede Agentenplattform liefert korrekte Tracedaten. Gäbe es eine globale Uhr, so könnten diese Daten schon zur Laufzeit zu einem konsistenten Gesamtbild zusammengefügt werden. Da aber keine globale Uhr existiert, bleiben nur zwei mögliche Alternativen. Die erste ist ein konsistentes Gesamtbild post mortem, wodurch jedoch die Interaktionsmöglichkeit eingeschränkt wird. Die zweite ist ein best effort Bild zur Laufzeit. Da beide Ansätze ihre Vor- und Nachteile haben, ist es sinnvoll, beide zu verfolgen.

Für den Ansatz der post mortem Visualisierung bietet sich die Metapher des Videorekorders an. Es sollte dem Nutzer möglich sein, den Trace aus der Laufzeitvisualisierung heraus aufzuzeichnen, ihn an jede beliebige Stelle zu spulen, ihn in unterschiedlichen Geschwindigkeiten abzuspielen, ihn zu stoppen und durchzusteppen.

Im Falle der Laufzeitvisualisierung sollte der Trace ein- und ausschaltbar sein. Debugging macht es zu dem erforderlich, dass eine große Zahl von Traces erzeugt und untereinander verglichen werden kann [Die02].

Die Informationsdimensionen werden also um die Dimensionen einer globalen Zeit und somit eindeutig beschreibbaren Absende- und Empfangszeitpunkten erweitert.

4.2.3 Visualisierungsdaten

Visualisierungsdaten sind die Daten, die visualisiert werden sollen. Das bedeutet, dass durch Datamining Technologien, Filtern und Verdichten der Daten genau das, was den

Betrachter interessiert, aus der Masse an Informationen herauskristallisiert wird. Zudem ist anzumerken, dass, falls die aufbereiteten Daten in einem relationalen Format gespeichert wurden, die Informationen wiederum in neue Struktur gebracht werden müssen, um die Visualisierung der Interaktion des Multiagentensystems zu ermöglichen.

Zwei Kriterien sind für visuelle Metaphern von entscheidender Bedeutung: Ausdrucksfähigkeit und Effektivität [MFM03]. Ausdruckfähigkeit bezieht sich dabei auf die Möglichkeit, alles das darzustellen, was dargestellt werden soll. Effektivität bezieht sich auf die Effektivität der Metapher hinsichtlich der Informationsrepräsentation. Hierbei lässt sich unterscheiden: Effektivität bezüglich der perzeptuell wahrnehmbaren Information, Effektivität unter ästhetischen Aspekten und Effektivität bezüglich Optimierung.

Da eine ganze Menge von Informationsdimensionen zu visualisieren ist, werden mehrere Perspektiven zur Visualisierung verwendet. Zwei grundlegend verschiedene Perspektiven sollen in Interaktion miteinander zum Einsatz kommen. Zusätzlich sollen Techniken des Filterns, Suchens, Abstrahierens und der benutzerdefinierten Abbildung zwischen Informations- und Darstellungsdimension verwendet werden, um möglichst viele verschiedene Perspektiven generieren zu können.

Zum einen wird eine architekturzentrierte Perspektive mit Agenten als Entitäten und Verbindungen, bzw. Relationen, zwischen den Agenten, die die Kommunikationsaktivität widerspiegeln verwendet. Zum anderen wird eine interaktionszentrierte Perspektive mit Nachrichtenereignissen als Entitäten, die in zeitlicher Abfolge und wählbaren Kriterien geordnet werden verwendet.

In der architekturzentrierten Perspektive sind die Agenten die darzustellenden Entitäten und die ausgetauschten Nachrichten die darzustellenden Beziehungen zwischen den Entitäten. Eine Nachricht ist dabei entweder versendet und empfangen, oder nur versendet aber noch nicht empfangen. Durch Gruppieren von Agenten zu Agentengruppen werden Agentenentitäten zu einer Gruppenentität zusammengefasst. Durch Filtern können Entitäten oder Beziehungen entfernt oder mit Attributen, die sie als durch den speziellen Filter ausgefiltert auszeichnen, versehen werden. Dies soll ermöglichen, dass sich einzelne Filter rückgängig machen lassen ohne alle Filter von neuem berechnen zu müssen. Andererseits muss so bei jedem neuen Filter die gesamte Knoten- und Kantenmenge von neuem gefiltert werden. Durch Suchfunktionen werden Entitäten und ihre Beziehungen mit einem Attribut als von der speziellen Suchanfrage aufgefunden markiert.

In der interaktionszentrierten Perspektive werden die Nachrichtenereignisse – Senden und Empfangen – repräsentiert. Die zeitliche Abfolge wird dabei auf eine kausale Ordnung reduziert, welche der Perspektive zugrunde liegt. Filtern und Suchen wird wie in der architekturzentrierten Perspektive gehandhabt. In der interaktionszentrierten Perspektive soll es insbesondere möglich sein, die Ereignisse nach beliebigen Kategorien der Informationsdimensionen zu gruppieren.

Die den beiden Perspektiven zugrunde liegenden Modelle können zu einem integriert werden, in dem sämtliche Informationsdimensionen der aufbereiteten Daten erhalten und um Attribute für Ergebnisse von Filtern und Suchen, sowie Verdichtung etc. erweitert werden. Dies hat insbesondere den Vorteil, dass eine Interaktion der beiden Perspektiven erleichtert wird, da sie das gleiche Modell darstellen.

4.2.4 Visualisierung

Die geeignete Wahl der Repräsentation ist ein entscheidender Schritt auf dem Weg von den ursprünglichen Daten über den Monitor bis hin zu der kognitiven Interpretation durch den Betrachter.

Die zunächst nahe liegende Wahl, UML- oder UML ähnliche Notationen zur Repräsentation zu verwenden, schließlich handelt es sich hierbei um die am weitesten verbreitete Notation [ITW01], würde viel von dem Potenzial grafischer Notation verschenken. Die Repräsentation der UML wurde scheinbar recht willkürlich gewählt und nur Experten können sie leicht lesen [ITW01].

Eine unter Beachtung der Forschungsergebnisse in den Kognitionswissenschaften entwickelte und somit für kognitive Verarbeitung optimierte Darstellung von Diagrammen ist die Notation der Geon Diagramme (siehe Kapitel 3.2.5).

Grundsätzlich ist bei der Gestaltung der Visualisierung zu entscheiden, welche Informationsdimension auf welche Darstellungsdimension abgebildet wird.

Es ist dabei zu beachten, dass nicht jede Informationsdimension auf jede Darstellungsdimension abgebildet werden kann, da zwei Klassen von Dimensionen zu unterscheiden sind: geordnete und kategorische. Zudem ist es zu vermeiden, zwei Informationsdimensionen auf eine Darstellungsdimension abzubilden, da dann in der Repräsentation beide Informationsdimensionen verloren sind. Um die Zahl der Informationstragenden Dimensionen eines Bildes oder einer Animation auf mehr als 3 bzw. 4 zu erhöhen, können zum Beispiel zusätzliche bedeutungstragende Dimensionen wie Konturstärke oder Größe für geordnete und Textur, Farbe, Orientierung oder Form für kategorische Dimensionen verwendet werden [PBG98].

Bei der Verwendung von Geon Diagrammen stehen uns somit unter anderem die folgenden bedeutungstragenden Merkmale zur Verfügung: die Gestalt der Geons, die Größe der Geons, die Farbe der Geons, die Textur der Geons, Verbindungen zwischen Geons, die Orientierung der Geons, die Gestalt der Verbindung zwischen Geons, die Dicke der Verbindung, die Farbe der Verbindung zwischen Geons, die Textur der Verbindung zwischen Geons, die Zahl der Verbindungen zwischen Geons, der Einschluss eines Geon in einem Geon, die Reihenanordnung von Geons als Metapher für Kausalfolge, Blinken, Strahlen, Transparenz, die Position auf der X-Achse, die Position auf der Y-Achse, die Position auf der Z-Achse und die Zeit.

Welche dieser bedeutungstragenden Dimensionen für die einzelnen Perspektiven verwendbar sind und wie die Informationsdimensionen auf sie abzubilden sind, gilt es im Einzelnen zu analysieren.

Um dem Benutzer bei der Präsentation einen maximalen Verständniserfolg zu bringen ist es notwendig, dem Benutzer eine größtmögliche Interaktivität mit den präsentierten Daten zu ermöglichen.

Der Benutzer sollte die sekundäre Notation, also das Layout, verändern können, sofern dies keine formal festgelegte Bedeutung trägt. Die Abbildung der Informationsdimensionen auf Darstellungsdimensionen sollte bis zu einem gewissen Grad auch vom Benutzer wählbar sein. So sollte der Benutzer im Falle von Geon Diagrammen wählen können, welche Informationsdimension auf die Darstellungsdimension Farbe abgebildet wird.

Zoomen durch die Visualisierung sollte angesichts der großen Informationsmenge auf jeden Fall möglich sein. Die gleichzeitige Verwendung eines semantischen Zooms mit einem grafischen ermöglicht es dem Benutzer, von einer sehr abstrakten Sicht mit niedrigem Detaillierungsgrad in eine viel konkretere Sicht mit hohem Detaillierungsgrad zu zoomen. Um dabei den Kontext nicht zu verlieren, soll ein kleines Fenster, gleichsam eine Übersichtskarte, die Position des betrachteten Ausschnitts der Visualisierung in selbiger anzeigen.

Das Einfügen von Marken und Notizen soll an beliebiger Stelle möglich sein, damit der Benutzer sich später schneller orientieren und zurechtfinden kann.

Die architekturzentrierte Perspektive soll starken Gebrauch von den Vorteilen der Geon Diagramme machen. In erster Linie sollen hier die Agenten als Individuen und die zwischen ihnen ausgetauschten Nachrichten dargestellt werden. Idee ist es, jeden Agenten bzw. jede Agentengruppe als Geon Primitiv darzustellen (siehe Abbildung 4.1). Hierdurch wird eine intuitive Unterscheidung und Wiedererkennung der Agenten gefördert. Vom Benutzer zu einer Gruppe zusammengefassten Agenten werden als Geonprimitive im grafischen Einschluss eines weiteren Geonprimitiv, der Gruppe, dargestellt. Repräsentiert wird die Verbindungsstärke ganz im Sinne der Geon Diagramme durch dicke der Verbindung, kann aber auch zusätzlich durch andere vom Nutzer frei wählbare Darstellungsdimensionen visualisiert werden. Verbindungen sind hier im Allgemeinen Transparent darzustellen.

Die Interaktion zwischen Agenten soll durch Verbindungen zwischen den Geons dargestellt werden. Die zeitliche Abfolge der ausgetauschten Nachrichten soll in der architekturzentrierten Perspektive auf die Zeit selbst abgebildet werden und somit eine Animation des Geschehens im Multiagentensystem darstellen. Bevor die Verarbeitung der Tracedaten beginnt, sind also zunächst nur die Agenten und Agentengruppen als Geon Primitive sichtbar. Werden die ersten Nachrichten ausgetauscht, kommt es zu Verbindungen der Geons. Durch die Verbindungsstärke kann beispielsweise die Zahl der ausgetauschten Nachrichten, aber auch die Zahl der durchgeführten Konversationen oder eine andere frei vom Benutzer gewählte Gewichtung dargestellt werden. Ist in der Timeline der Zeitpunkt eines Absendeereignisses eingetroffen, so wird in der betroffenen Verbindung eine weitere nicht Transparente Verbindung dargestellt. Die erste Hälfte der neuen Verbindung wird nun ausgehend von dem den Sendenden Agenten repräsentierenden Geon dargestellt, während die zweite Hälfte zunächst unsichtbar bleibt. Wird die Nachricht empfangen, so wird die zweite Hälfte der Verbindung ebenfalls dargestellt. Wird nach Empfang das nächste Sendeereignis registriert, so wird die Verbindung, die das konkrete Sende- und Empfangsereignis repräsentierte, wieder gelöscht. Es gibt also eine kumulierende Darstellung der Interaktion in der architekturzentrierten Perspektive und gleichzeitig eine explizite.

Auf der niedrigsten semantischen Zoom ebene sind nur die Geon Primitive, ohne Transparenz, und ihre Verbindungen sichtbar. Durch Suche gefundene Agenten, Nachrichten, Konversationen, Performatives oder sonstige Informationen sollen durch Blinken der entsprechenden Geon Primitive und Verbindungen hervorgehoben werden. Da Blinken eine auf Papier nicht sichtbare Informationsdimension darstellt, ist zusätzlich eine weitere Repräsentation wie zum Beispiel das Verstärken der Konturen zu wählen. Der Benut-

zer soll auch hier die Möglichkeit haben, die Darstellungsdimension frei zuzuordnen. Durch Filtern sollen die entsprechenden Geon Primitive oder Verbindungen wahlweise verschwinden oder transparent dargestellt werden. Das Layout soll durch simples Drag und Drop jederzeit durch den Benutzer veränderbar sein. Durch Markieren von Geon Primitiven kann der Benutzer Agenten oder Agentengruppen zu einer Gruppe abstrahieren, die von da an als einzelnes Geon Primitiv dargestellt werden, bis der semantische Zoom die Darstellung der Elemente der Gruppe durch Geon Primitive im Einschluss der transparenten Darstellung der Gruppe darstellt. Durch Anklicken der Primitive und der Verbindungen kann das jeweilige Element ausgewählt und zusätzlich textuelle Information präsentiert werden.

Vorteil dieser Darstellung sollte sein, dass sich eine schnelle Übersicht über die Architektur des Agentensystems sowie die statischen Aspekte der Interaktion darstellen lässt. Durch die Möglichkeiten des Highlightens und Ausblendens durch Suche und Filtern, sowie der Abstraktion, kann der Benutzer schnell die für ihn relevanten Informationen erhalten. Um die zeitliche Abfolge der Interaktion sinnvoll und auf einen Blick darzustellen, bedarf es jedoch einer weiteren Perspektive. Der interaktionszentrierten Perspektive.

Die interaktionszentrierte Perspektive (siehe Abbildung 4.2) ist wie folgt aufgebaut. Die darzustellenden Elemente sind hier die Ereignisse des Absendens einer Nachricht, sowie deren Empfang. In der X-Achse wird die zeitliche Abfolge der Ereignisse dargestellt. Da eine Nachricht einen Sende- und einen Empfangszeitpunkt besitzt, überlappen die Zeiträume der Nachrichtenübermittlung sich unter Umständen. Durch das Trennen der Nachrichtenübermittlung in Sende- und Empfangsereignis können die Ereignisse als momentan angenommen werden und so in eine temporale Reihenfolge gebracht werden.

Die Ereignisse selbst werden bei niedrigstem semantischem Zoom als Geons (nach Zugehörigkeit des Ereignisses zum Agenten), dann als kleines Kästchen und auf höchster semantischer Zoomebene schließlich nur noch als Punkt dargestellt. Bis jetzt gibt es also eine Darstellung von Nachrichtenereignissen in ihrer zeitlichen Abfolge.

Weiter soll es nun möglich sein, Nachrichten zu gruppieren. Mit Gruppieren sei hier das grafische Absetzen in der Y-Achse gemeint. Gruppiert werden soll nach den unterschiedlichsten Kriterien, wie Sender und Empfänger, Performative, Konversation und beliebigen anderen Kriterien, die zur Verfügung stehen.

Durch Suchanfragen soll es möglich sein, Nachrichten zu highlighten oder wahlweise die anderen Nachrichten in den Hintergrund treten zu lassen. Durch Filtern sollen Nachrichten entfernt oder ausgegraut werden können.

Da in komplexen Systemen eine große Menge an Nachrichtenereignissen anfallen wird, muss die Auswahl der Darstellungsdimensionen eingeschränkt werden, da manche Dimension in der Darstellung der Masse verloren gehen würde. Es stehen folgende Dimensionen zur Darstellung zur Verfügung: X-Achse, Y-Achse, Farbe, Blinken. Als feste, unveränderliche Abbildungen werden hier nur die Abbildung der Zeitinformation auf die X-Achse, sowie die Materialisation eines Ereignisses als Punkt, Kästchen oder Geon auf der X-Achse bezüglich seines zeitlichen Auftretens angenommen.

Durch die Kombination der architekturzentrierten Perspektive und der interaktionszentrierten Perspektive ergeben sich nun weitere Möglichkeiten. So soll durch Wahl eines einzelnen Nachrichtenereignisses in der interaktionszentrierten Perspektive ähnlich einer

Suche in der architekturzentrierten Perspektive dort die entsprechende Verbindung von Geons highlighten, zu der das gewählte Nachrichtenereignis gehört. Andersherum soll es möglich sein durch die Wahl einer Verbindung in der architekturzentrierten Perspektive die dazugehörenden Nachrichtenereignisse in der interaktionszentrierten Perspektive zu highlighten, oder durch die Wahl eines Geon Primitiv in der architekturzentrierten Perspektive jedes Nachrichtenereignis, an dem der Agent beteiligt ist, gehighlightet werden. Die Darstellung der architekturzentrierten Perspektive könnte in der höchsten semantischen Zoom Ansicht durch grafischen Einschluss die zu dem jeweiligen Geon bzw. der jeweiligen Verbindung gehörenden Interaktionsperspektiven anzeigen. Mit zugehöriger Interaktionsperspektive ist hier wahlweise die gesamte Interaktionsperspektive mit gehighlighteten Elementen oder eine gefilterte Interaktionsperspektive gemeint.

4.3 Architektur

Die hier vorgestellte Architektur besteht aus 4 Komponenten: dem AgentplatformAdapter, dem Tracer, dem Visualizer und dem DataAdapter.

Aufgabe des AgentplatformAdapters ist es, eine systeminterne Standardschnittstelle zum Erhalt von Traces beliebiger Agentenplattformen zu spezifizieren, so dass der Rest des Systems von der Agentenplattform unabhängig ist. Der AgentplatformAdapter realisiert somit den Teil des Systems, der die Quelldaten empfängt und auf ein systeminternes Format abbildet. Auf jedem Knoten eines verteilten Systems auf dem eine Agentenplattform lokalisiert ist, wird auch ein AgentplattformAdapter platziert.

Der Tracer ist die zentrale Sammelstelle für sämtliche Tracedaten und liegt nach Möglichkeit auf einem separaten Knoten. Im gesamten System gibt es nur einen Tracer. Die von den AgentplatformAdapter auf systeminternes Format gebrachten Quelldaten werden vom Tracer empfangen und um zusätzliche Informationen angereichert. Der Tracer bietet Methoden zum Navigieren im Trace, zum Filtern, Verdichten und Data Mining der Tracedaten an. Er produziert somit die Visualisierungsdaten.

Der Visualizer empfängt die Visualisierungsdaten vom Tracer und stellt sie dar. Es können mehrere unterschiedliche Visualizer gleichzeitig denselben Trace visualisieren.

Der DataAdapter bildet die Persistenzschicht des Systems und entkoppelt das System von der Technik der Datenhaltung.

Der Datenfluss im System ist in Abbildung 4.3 zu sehen, die Benutzt Beziehungen in Abbildung reffig:benutzt.

4.3.1 AgentplatformAdapter

Der AgentplatformAdapter verwendet die jeweiligen Möglichkeiten der Agentenplattform, um an Tracedaten zu gelangen. Der AgentplatformAdapter bietet eine GUI, auf der zu sehen ist, welche Agenten getraced werden.

Im Falle von Jade bietet es sich hierzu an, einen von ToolAgent erbenden Agenten, hier TraceAgent genannt, zu verwenden. Der TraceAgent kommuniziert über die von JADE angebotene Schnittstelle zu Java Programmen mit dem AgentplatformAdapter, der hier in der Realisierung für die JADE Plattform JADEAdapter genannt wird.

Der JADEAdapter meldet sich als Beobachter für die Tracedaten des TraceAgents an und sendet ihm über die Schnittstelle Nachrichten zur Konfiguration und Steuerung. Bei vom TraceAgent registrierten relevanten Ereignissen wird der JADEAdapter informiert, woraufhin dieser die anfallenden Quelldaten in das systeminterne Format bringt und seinerseits den Tracer informiert.

4.3.2 Tracer

Der Tracer verfügt über ein internes Modell des Multiagentensystems. Dieses Modell repräsentiert die aufbereiteten Daten. Der Tracer meldet sich als Beobachter an sämtlichen AgentPlatformAdaptern an. Die AgentPlatformAdapter werden vom Tracer aus gesteuert und konfiguriert. Vom Tracer aus lässt sich per GUI konfigurieren, welche Agenten getraced werden sollen.

Erhält der Tracer die Benachrichtigung über ein relevantes Ereignis, so reichert er die erhaltenen Daten um Attribute wie zum Beispiel globalen Zeitstempel und Agentenplattform an und fügt sie in das Modell der aufbereiteten Daten ein.

Der Tracer bietet über seine GUI die Möglichkeit zur Aufnahme eines Traces, verbunden mit dem Speichern über den DataAdapter, so wie das Laden und abspielen eines zuvor aufgezeichneten Traces. Beim Laden und Speichern über den DataAdapter wird jeweils das Modell der aufbereiteten Daten, also der eigentliche Trace, geladen oder gespeichert, sowie auch auf Wunsch der Visualizer jedes Modell der Visualisierungsdaten, damit die Einstellungen der Visualizer nicht verloren gehen.

Der Tracer hat für jeden angemeldeten Visualizer ein eigenes Visualisierungsdatenmodell. Jedes dieser Modelle besitzt eine Referenz auf das Modell der aufbereiteten Daten und ergänzt es um die von jedem Visualizer individuell vorgenommenen Anpassungen wie zum Beispiel Filterungen, Verdichtungen, Auswahlen etc.

4.3.3 Visualizer

Der Visualizer ist für die Darstellung der Visualisierung zuständig. Jeder Visualizer enthält ebenfalls ein internes Modell des zu visualisierenden Multiagentensystems, das View Modell. Dieses Modell referenziert seinerseits das im Tracer vorgehaltene Visualisierungsmodell. Letztlich wird also das Visualisierungsmodell um zusätzliche Informationen, die die Darstellung betreffen, eines jeden Visualizer angereichert. Über die GUI getroffene Änderungen der Visualisierung (z.B. Layout, Abbildung der Informationsdimensionen auf Darstellungsdimensionen, gesetzte Marken und Notizen) werden im View Modell des Visualizers vermerkt. Der DataAdapter wird zum Speichern des View Modells des Visualizers verwendet, damit die getroffenen Anpassungen nicht verloren gehen.

4.3.4 DataAdapter

Der DataAdapter soll das Persistenzsystem kapseln. Dadurch wird gewährleistet, dass das System bei einem Wechsel des Persistenzsystems bis auf den DataAdapter nicht geändert werden muss. Der DataAdapter bietet Methoden zum Laden und Speichern

der internen Modelle des Tracers und des Visualizers an. Hierdurch werden Traces, Visualisierungskonfigurationen und View Konfigurationen persistent.

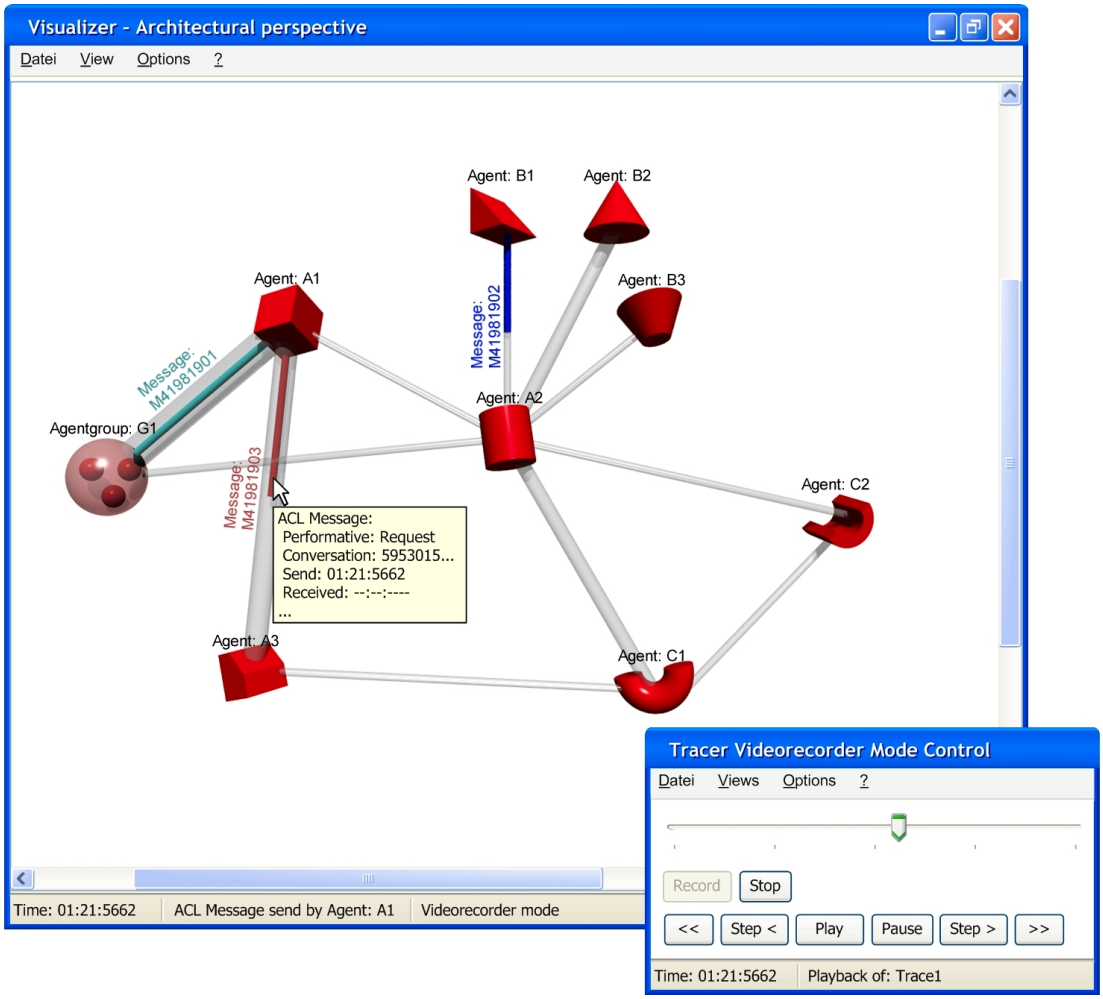

Abbildung 4.1: Mockup der Visualisierung der architekturzentrierten Perspektive. Ein Agent aus der Agentengruppe G1 hat eine Nachricht an den Agenten A1 gesendet. Bevor diese Nachricht empfangen wurde, hat der Agent B1 eine Nachricht an Agent A2 und der Agent A1 eine Nachricht an Agent A3 gesendet. Das Mockup zeigt den Moment, in dem der Agent A1 die Nachricht empfangen hat, die Nachrichten an A3 und A2 aber noch nicht empfangen worden sind.

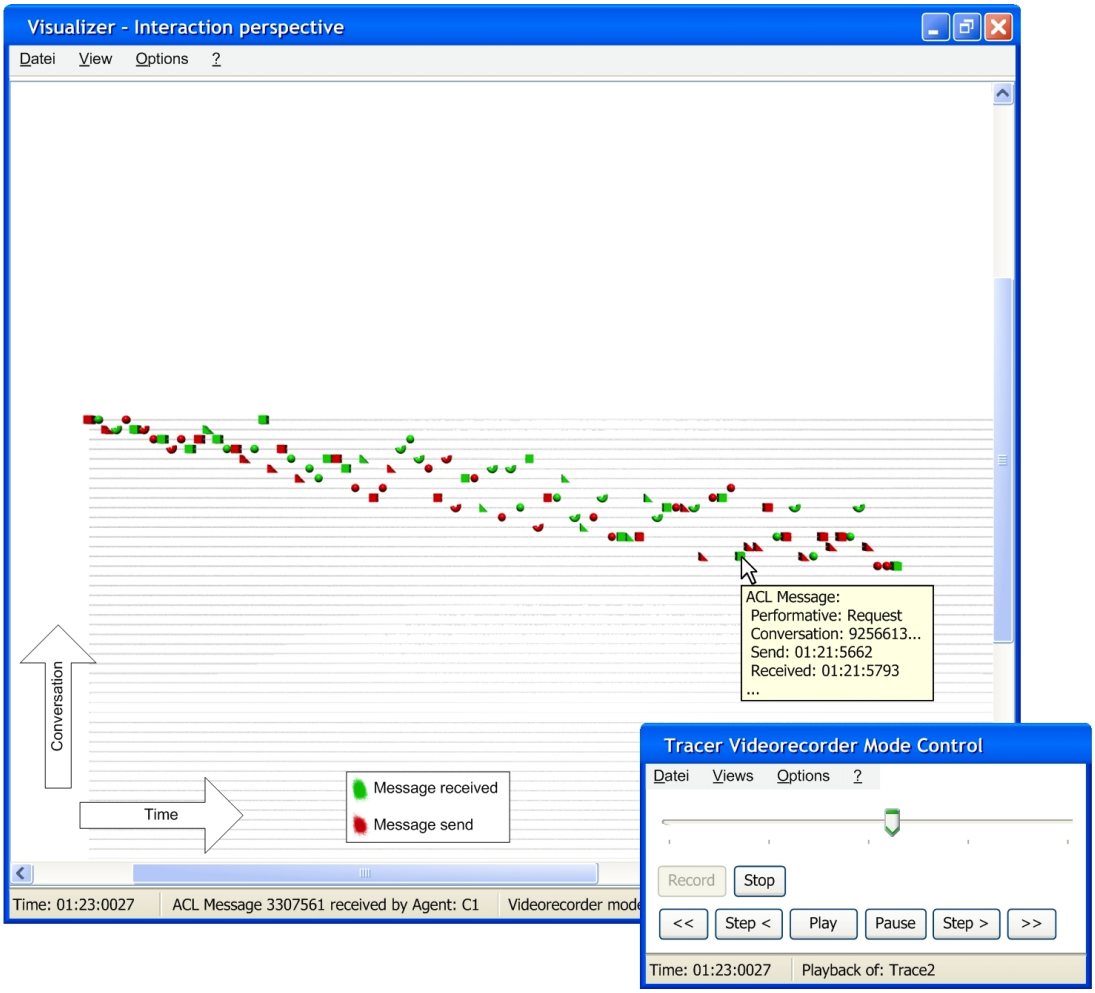

Abbildung 4.2: Mockup der Visualisierung der interaktionszentrierten Perspektive. Das Mockup zeigt eine Folge von Sende- und Empfangsereignissen. In der X-Achse ist der Verlauf der Zeit dargestellt. In der Y-Achse wurde die Gruppierung nach Konversationen gewählt. Die Farben drücken aus, ob es sich um ein Sende- oder Empfangsereignis handelt. Durch die Form der Geons wird die Zugehörigkeit der Ereignisse zu den Agenten ausgedrückt.

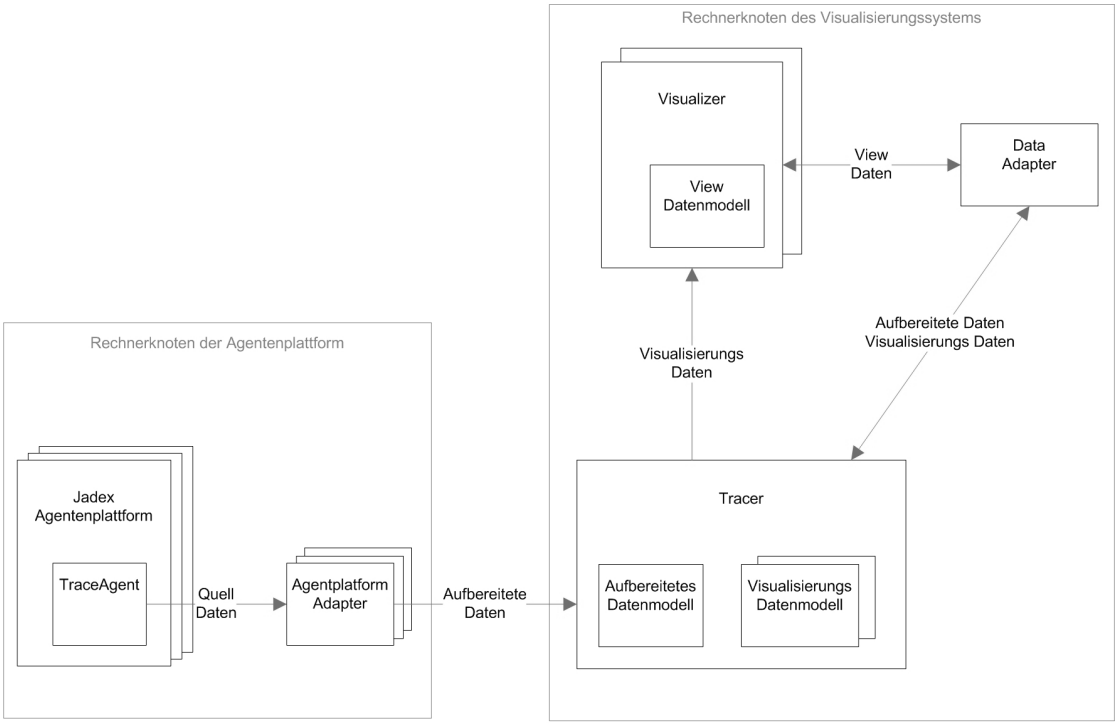

Abbildung 4.3: Architektur des Visualisierungssystems für die Interaktion in Multiagentensystemen: Datenfluss im System

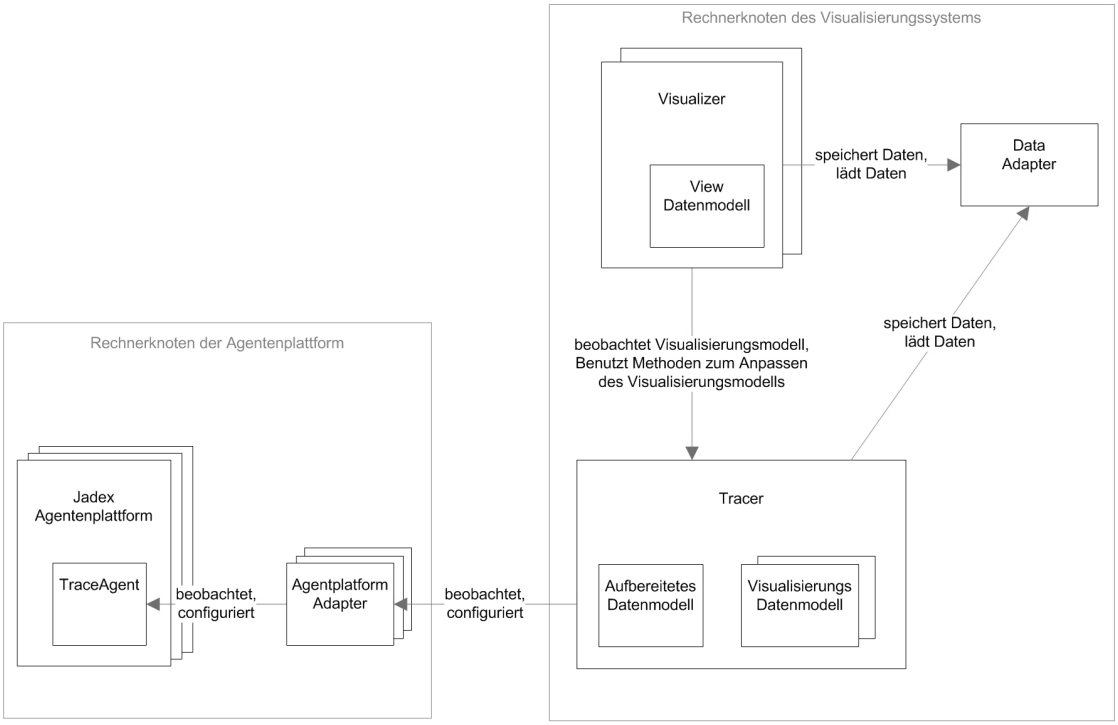

Abbildung 4.4: Architektur des Visualisierungssystems für die Interaktion in Multiagentensystemen: Benutzt Beziehungen im System

5 Zusammenfassung und Ausblick

In dieser Arbeit wurden Ideen für die Realisierung eines Visualisierungssystems der Interaktion in Multiagentensystemen entwickelt. Durch die Anlehnung an die – auf kognitionswisschenschaftlichen Erkenntnissen basierenden – Geon Diagramme und die Interaktionsmöglichkeiten des Benutzers mit dem System, sollte ein nach diesen Ideen implementiertes System das Verstehen von Multiagentensystemen und somit auch die Entwicklung von Multiagentensystemen erleichtern.

Die prototypische Realisierung des vorgestellten Systems gilt es nun umzusetzen. Zunächst sollte sich die Implementierung auf das Kernsystem mit den zwei vorgestellten Perspektiven auf Multiagentensysteme und die Anpassung an die Agentenplattform JADE konzentrieren. Das System sollte, aufgrund der modularen Konzeption, leicht an andere Agentenplattformen anpassbar sein und leicht die Implementierung neuer Perspektiven auf die Interaktion in Multiagentensystemen ermöglichen. Eine Entwicklung von einem Einzelplatzwerkzeug hin zu einem Werkzeug für Collaborative Work wäre wünschenswert.

Die prognostizierten Vorzüge des in dieser Arbeit vorgestellten Visualisierungssystems gilt es mit Hilfe des Prototypen empirisch zu belegen. Das veränderte Verhalten der Benutzer, zum Beispiel von Entwicklern, aufgrund der Anwendung des Prototypen, gilt es zu analysieren.

Literaturverzeichnis

[BE96] Ball, Thomas ; Eick, Stephen G.: Software Visualization in the Large. In: *IEEE Computer* 29 (1996), Nr. 4, S. 33–43

[BHS04] Botía, Juan A. ; Hernansaez, Juan M. ; Skarmeta, Fernando G.: Towards an Approach for Debugging MAS Through the Analysis of ACL Messages. In: *Multiagent System Technologies, Second German Conference, MATES 2004, Erfurt, Germany, September 29-30, 2004, Proceedings*, 2004, S. 301–312

[Deu01] van Deursen, Arie: Program comprehension risks and opportunities in extreme programming / A. van Deursen / Centrum voor Wiskunde en Informatica, Software Engineering. Amsterdam : Centrum voor Wiskunde en Informatica, 2001 (SEN R0110). – Forschungsbericht. – 10 S. – ISSN 1386–369X

[Die02] Diehl, Stephan (Hrsg.): *Software Visualization, International Seminar Dagstuhl Castle, Germany, May 20-25, 2001, Revised Lectures*. Bd. *2269*. Springer, 2002 (Lecture Notes in Computer Science). – ISBN 3–540–43323–6

[Die03] Diehl, S.: Softwarevisualisierung. In: *Informatik Lexikon der Gesellschaft für Informatik e.V.* (2003). – `http://www.gi-ev.de/service/informatiklexikon/informatiklexikon-detailansicht/meldung/79/` (5. Okt 2005)

[FIP] FIPA. *Internetseite*. `http://www.fipa.org` (5. Okt 2005)

[FIP02a] FIPA. *FIPA ACL Message Structure Specification*. SC00061G. 1996-2002

[FIP02b] FIPA. *FIPA Agent Message Transport Service Specification*. SC00067F. 1996-2002

[FIP02c] FIPA. *FIPA Communicative Act Library Specification*. SC00037J. 1996-2002

[FIP02d] FIPA. *FIPA Request Interaction Protocol Specification*. SC00026H. 1996-2002

[HGSG90] Hoc, J.-M. (Hrsg.) ; Green, T.R.G. (Hrsg.) ; Samurcay, R. (Hrsg.) ; Gilmore, D.J. (Hrsg.): *Psychology of Programming*. London : Academic Press, 1990

[HMM00] Herman, Ivan ; Melançon, Guy ; Marshall, M. S.: Graph Visualization and Navigation in Information Visualization: A Survey. In: *IEEE Transactions on Visualization and Computer Graphics* 6 (2000), Nr. 1, S. 24–43

[ITW01] Irani, Pourang ; Tingley, Maureen ; Ware, Colin: Using Perceptual Syntax to Enhance Semantic Content in Diagrams. In: *IEEE Comput. Graph. Appl.* 21 (2001), Nr. 5, S. 76–85. – ISSN 0272–1716

[IW00] Irani, Pourang ; Ware, Colin: Diagrams based on structural object perception. In: *AVI '00: Proceedings of the working conference on Advanced visual interfaces*. New York, NY, USA : ACM Press, 2000. – ISBN 1–58113–252–2, S. 61–67

[JAD] JADE. *Internetseite*. `http://jade.tilab.com` (5. Okt 2005)

[JCJO92] Jacobson, I. ; Christerson, M. ; Jonsson, P. ; Overgaard, G.: *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992

[Kan] Kant, Immanuel ; Heidemann, Ingeborg (Hrsg.): *Kritik der reinen Vernunft*. 2. Ditzingen : Reclam. – auch unter: `http://gutenberg.spiegel.de/kant/krvb/krvb.htm`. – ISBN 3150064619

[Kir01] Kirkpatrick, Kimberly: Object Recognition. In: Cook, Dr. Robert G. (Hrsg.): *Avian Visual Cognition*. Dr. Robert G. Cook, 2001. – Cyberbook: `http://www.pigeon.psy.tufts.edu/avc/toc.htm` (10. Okt 2005)

[KNB+03] Krempels, Karl-Heinz ; Nimis, Jens ; Braubach, Lars ; Pokahr, Alexander ; Herrler, Rainer ; Scholz, T.: Entwicklung intelligenter Multi-Multiagentensysteme - Werkzeugunterstützung, Lösungen und offene Fragen. In: Dittrich, K. (Hrsg.) ; König, W. (Hrsg.) ; Oberweis, A. (Hrsg.) ; Rannenberg, K. (Hrsg.) ; Wahlster, W. (Hrsg.): *Informatik 2003 - 33. Jahrestagung der GI* Gesellschaft für Informatik e.V., Köllen Druck+Verlag GmbH Bonn, 9 2003, S. 31–46

[LA95] Liedekerke, Marc H. V. ; Avouris, Nicholas M.: Debugging multi-agent systems. In: *Information and Software Technology Journal* 37 (1995), February, Nr. 2, S. 103–112

[MFM03] Marcus, Andrian ; Feng, Louis ; Maletic, Jonathan I.: 3D representations for software visualization. In: *SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization*. New York, NY, USA : ACM Press, 2003. – ISBN 1–58113–642–0, S. 27–ff

[MMC02] Maletic, Jonathan I. ; Marcus, Andrian ; Collard, Michael L.: ATask Oriented View of Software Visualization. In: *VISSOFT '02: Proceedings of the 1st International Workshop on Visualizing Software for Understanding*

and Analysis. Washington, DC, USA : IEEE Computer Society, 2002. – ISBN 0–7695–1662–9, S. 32

[NNLC99] Ndumu, Divine T. ; Nwana, Hyacinth S. ; Lee, Lyndon C. ; Collis, Jaron C.: Visualising and debugging distributed multi-agent systems. In: *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*. New York, NY, USA : ACM Press, 1999. – ISBN 1–58113–066–X, S. 326–333

[PBG98] Petre, Marian ; Blackwell, Alan ; Green, Thomas: Cognitive Questions in Software Visualization. In: Stasko, John (Hrsg.) ; Domingue, John (Hrsg.) ; Brown, Marc H. (Hrsg.): *Software Visualization: Programming as a Multimedia Experience*. The MIT Press, 1998. – ISBN 0262193957, S. 453–480

[PBS98] Price, Blaine ; Baecker, Ronald ; Small, Ian: An Introduction to Software Visualization. In: Stasko, John (Hrsg.) ; Domingue, John (Hrsg.) ; Brown, Marc H. (Hrsg.): *Software Visualization: Programming as a Multimedia Experience*. The MIT Press, 1998. – ISBN 0262193957, S. 3–27

[Pet95] Petre, Marian: Why looking isn't always seeing: readership skills and graphical programming. In: *Commun. ACM* 38 (1995), Nr. 6, S. 33–44. – ISSN 0001–0782

[PLVW98] Pauw, Wim D. ; Lorenz, David ; Vlissides, John ; Wegman, Mark: Execution Patterns in Object-Oriented Visualization. In: *Proceedings Conference on Object-Oriented Technologies and Systems (COOTS '98)*, USENIX, 1998, S. 219–234

[RZ94] Rosenschein, Jeffrey S. ; Zlotkin, Gilad: Consenting Agents: Designing Conventions for Automated Negotiation. In: *AI Magazine* 15 (1994), Fall, Nr. 3, S. 29–46. – Also reprinted in *Readings in Agents*, edited by Michael N. Huhns and Munindar P. Singh, Morgan Kaufmann Publishers, San Francisco, California, 1997, pages 353–370.

[Shn96] Shneiderman, Ben: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In: *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*. Washington, DC, USA : IEEE Computer Society, 1996. – ISBN 0–8186–7508–X, S. 336

[SN01] Schroeder, Michael ; Noy, Penny: Multi-agent visualisation based on multivariate data. In: *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*. New York, NY, USA : ACM Press, 2001. – ISBN 1–58113–326–X, S. 85–91

[Syc98] Sycara, Katia P.: Multiagent Systems. In: *AI Magazine* 19 (1998), Nr. 2, S. 79–92

[War04] Ware, Colin: *Information visualization: perception for design*. Second. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2004. – ISBN 1–55860–511–8