Technische Universität Hamburg-Harburg
Institute for Software Systems

Prof. Dr. Sibylle Schupp

Evaluation einer agentenbasierten Modellierung für ein AIS-Überwachungssystem mit optimierter Festmachereinsatzplanung

Diplomarbeit

Alexej Frank

6. November 2009

Technische Universität Hamburg-Harburg

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit von mir selbständig und nur unter Verwendung der aufgeführten Hilfsmittel erstellt wurde.

Hamburg, den 6. November 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

In der heutigen Zeit spielt der Transport der Güter eine wichtige Rolle für die Wirtschaft. Die Globalisierung hat in den letzten Jahren zu einem enormen Anstieg der transportierten Güter geführt. Ein großer Teil der Güter wird dabei in Containern auf dem Wasser transportiert. Die meisten Container werden in Deutschland im Hamburger Hafen umgeschlagen. Der Hamburger Hafen ist nach Rotterdam der zweitgrößte Containerhafen in Europa und verzeichnet über 12.000 Schiffsankünfte im Jahr, was bedeutet, dass ca. jede halbe Stunde ein Seeschiff ankommt. All diese Schiffe müssen an ihren Liegeplätzen fest- und losgemacht werden. Für diese Aufgaben sind die sogenannten Festmacher zuständig. Das hohe und ungleichmäßige Schiffsaufkommen erschwert die Personal- und Einsatzplanung der Festmacher. Das Ziel der Festmacher ist die pünktliche und schnelle Abfertigung aller aufkommenden Schiffe, wobei Wartezeiten beiderseits vermieden werden sollen. Um die Einsatzzentrale, die die Aufgabenverteilung durchführt, zu unterstützen und die Verteilung möglichst automatisch durchzuführen, soll ein Multiagentensystem(MAS) für dieses Problem evaluiert werden.
Eine ähnliche Vorgehensweise wurde im Hafen von Rotterdam für den „Hinterlandtransport[1]“ angewendet. Der in [17] beschriebene Ansatz optimiert die Routen der „Bargen“ einerseits und maximiert die Auslastung der Terminals andererseits.

Der dichtere Schiffsverkehr hat auch zu vielen Schiffskollisionen auf hoher See geführt. Um dieses zu vermeiden wurde das Automatische Identifikationssystem(AIS) entwickelt. Die Schiffe senden dabei ein Signal mit Daten wie Position, Geschwindigkeit und Richtung über Funk. Die Schiffe in der Nähe des Senders können diese Signale empfangen, die Daten auswerten und so Kollisionen vermeiden. Die AIS-Daten ermöglichen die Positionsbestimmung der Schiffe in Echtzeit und können die Grundlage für das Schätzen von Ankünften liefern.

[1] Als Hinterlandtransport wird in diesem Zusammenhang der Transport von Containern, von den Hafenterminals in das Landesinnere verstanden.

Eine weitere Datenquelle bietet die Hamburg Port Authority(HPA), die die Daten über die Schiffsbewegungen im Hafen bereitstellt. Aus diesen Daten können die erwarteten Ankünfte von Schiffen an bestimmten Terminals extrahiert werden.

Im Rahmen dieser Diplomarbeit wird ein Prototyp eines AIS-Überwachungssystems realisiert. Auf dieser Basis wird versucht ein Multiagentensystem zu modellieren und zu realisieren, welches die Einsatzplanung der Festmacher optimieren soll. Im Kapitel 2 wird eine Einführung in das AIS gegeben und der entwickelte Prototyp erläutert. Kapitel 3 beschreibt die Grundlagen für Agenten und Multiagentensysteme. Dann werden im Kapitel 4 die zum Modellieren und Implementieren des Multiagentensystems eingesetzen Tools vorgestellt. Anschließend wird im Kapitel 5 das Modell und die Implementierung des einwickelten Multiagentensystems erläutert. Das Kapitel 6 gibt eine Zusammenfassung und Ausblick.

1.1 Aufgabenstellung/Zielsetzung

Diese Arbeit verfolgt zwei Ziele. Zum einen soll ein Prototyp für ein AIS-Überwachungssystem entwickelt werden. Zum anderen soll ein Multiagentensystem für die Optimierung der Festmachereinsatzplanung modelliert und nach Möglichkeit implementiert werden.

Für die Entwicklung des AIS-Überwachungssystems müssen folgende Aufgaben bearbeitet werden:

- Inbetriebnahme des AIS-Empfängers
- Aufbau einer Empfängerstation, die die Daten des Empfängers über eine serielle Schnittstelle ausliest und über das Internet an den Zentralrechner weiterleitet. Die Empfängerstation soll als eine Client-Anwendung aufgebaut werden. Durch diesem modularen Aufbau soll ermöglicht werden, dass Daten von mehreren Empfängern an dem Zentralrechner gesammelt werden können.
- Aufbau des Zentralrechners. Der Zentralrechner soll als Server in dem System fungieren und Daten von verschiedenen Empfängerstationen sammeln. Die eingetroffenen AIS-Daten, müssen dekodiert, zusammengeführt und gespeichert werden. Außerdem sollen die Daten der HPA ausgelesen und mit den AIS-Daten verknüpft werden. Um die Daten dem Benutzer zugänglich zu machen, soll eine graphische Benutzeroberfläche realisiert werden. In der graphischen Benutzeroberfläche sollen die Positionen der Schiffe in einem Geoinformationssystem(GIS) dargestellt werden. Die Positionen der Schiffe sollen bei Änderungen möglichst in Echtzeit aktualisiert werden. Das GIS soll außerdem Funktionen wie Zoomen, Verschieben der Karte, Filtern bestimmter Schiffsgruppen

und Anzeigen von vorhandener Schiffsinformation per Mausklick bieten. Neben der Darstellung im GIS soll eine tabellarische Darstellung der verfügbaren Daten erfolgen. Der Schwerpunkt liegt dabei auf der Anzeige der HPA-Daten, um die geplanten Schiffsankünfte für den Benutzer übersichtlich zu machen.

Das System muss leicht erweiterbar sein, sodass weitere Datenquellen, Filter und andere Funktionalitäten leicht in das System eingefügt werden können. Leichte Bedienbarkeit ist auch ein wichtiges Kriterium, weil die Benutzer des Systems keine EDV-Fachleute sind. Die Hochverfügbarkeit ist auch ein wichtiger Aspekt.

Um ein Multiagentensystem zu entwickeln, muss zuerst das aktuelle Umfeld und die momentanen Abläufe erfasst werden. Um sich einen Überblick zu verschaffen, ist ein Besuch des Betriebes erforderlich. Als Nächstes muss dann ein Modell erstellt werden, welches die Umgebung und die Abläufe beschreibt. Dieses Modell ist eine möglichst einfache Darstellung des Systems. Es existieren mehrere Vorgehensweisen zur Erstellung von agentenorientierten Softwaremodellen. Für diese Arbeit wurde das TROPOS-Modell(Kap. 4) ausgewählt. Dieses Verfahren unterstützt die Entwicklung eines Multiagentensystems über alle Stufen hinweg. Wenn das Modell erstellt wurde, kann es dann in die konkrete Softwareimplementierung überführt werden. Für die Implementierung wird die JADEX-Plattform(Kap. 4) genutzt. Diese Plattform ist sehr gut geeignet, um ein mit TROPOS entwickeltes Modell umzusetzen.

1.2 Stand der Technik

Im Internet sind einige Produkte zu finden, die Lösungen für AIS-Überwachungssysteme realisieren. So gut wie alle diese Produkte sind browserbasierte Anwendungen wie „Vesseltracker.com“, „marinetraffic.com“ und „digital-seas.com“. Diese Anwendungen bieten oft ähnliche Funktionalitäten, wie das Visualisieren, Filtern, Verfolgen und Speichern von Schiffsdaten. Neben den Browserlösungen existieren auch einige Produkte, die als Desktopanwendung realisiert sind. Viele der Produkte decken nur bestimmte Häfen ab und es gibt wenige Anwendungen, die eine globale Abdeckung bieten. Für die Festmacher ist außerdem besonders wichtig, dass die Daten möglichst aktuell und richtig sind, was nicht viele Anwendungen bereitstellen können. Momentan benutzen die Festmacher eine von „Vesseltracker.com“ angebotene Lösung für die Überwachung und eine andere Anwendung zum Verarbeiten von HPA-Daten und zur Verwaltung der Auftragsvergabe. Die während dieser Arbeit zu entwickelnde Anwendung soll ein Prototyp für eine Komplettlösung werden, die die Überwachung und die Verwaltung realisiert.

Agentenbasierte Systeme werden heutzutage in vielen verschiedenen Bereichen wie Medizin [21], Logistik [7], Verkehr [19] und vielen anderen verwendet. Jedes mal wenn ein System entwickelt werden soll, welches aus mehreren selbstständig agierenden Teilsystemen besteht, ist eine agentenbasierte Lösung ein oft verwendeter

Ansatz. Der bereits erwähnte Ansatz von Albert Douma [17] zeigt, wie ein agentbasierter Ansatz für eine Optimierung der Containertransporte eingesetzt werden kann. Es wurde unter anderem eine agentenbasierte Lösung für die Festmachereinsatzoptimierung gewählt, da das ähnliche Problem von Douma erfolgreich mit einem solchen Ansatz gelöst werden konnte.

Kapitel 2

AIS-Überwachungssystem

2.1 AIS-Grundlagen

2.1.1 Hintergrund

International Maritime Organisation(IMO) ist 1948 von der UN ins Leben gerufen worden, um die rechtlichen Rahmenbedingungen für die internationale Schiffahrt zu schaffen. Die erste Sitzung fand im Jahr 1959 statt. Die Aufgaben betreffen ökologische Anliegen, technische Kooperation, Rechtsangelegenheiten und maritime Sicherheit. [3] Die Organisation hat 169 Mitgliedstaaten und hat ihren Sitz in London. Erst nach dem Untergang der Titanik hat man sich verstärkt Gedanken über die maritime Sicherheit gemacht und eine internationale Kommison einberufen, die internationale Standarts enwickeln sollte. Die erste Version des International Convention for the Safety of Life at Sea (SOLAS) (Internationales Übereinkommen zum Schutz des menschlichen Lebens auf See) ist 1914 veröffentlicht worden. Seit der Gründung der IMO ist die Weiterentwicklung von SOLAS eine ihrer der Hauptaufgaben. 1974 wurde die Letzte Version mit 12 Kapiteln verabschiedet. Diese Version wird bis heute gepflegt und von der IMO verändert.
1997 hat das IMO-Unterkomitee für Navigationssicherheit den Entwurf für einen AIS-Standart verabschiedet. Im selben Jahr wurden von der International Telecommunication Union(ITU) zwei UKW Frequenzen bei 161.975 und 162.025 MHz für AIS reserviert. 1998 wurde AIS in das Kapitel V des SOLAS-Übereinkommens aufgenommen und der Standart *„Technical Characteristics for a Ship-borne Automatic Identification System (AIS) Using Time Division Multiple Access in The Maritime Mobile Band“* verabschiedet.
In 2002 wurde im Kapitel V, Regulierung 19 des SOLAS-Übereinkommens festgelegt, in welchen Schritten AIS auf den Schiffen eingeführt werden soll. [26] Spätestens 2008 sollten demnach folgende Schiffe mit AIS ausgerüstet werden:

- Schiffe über 300 Bruttoraumzahl(BRZ), die in der internationalen Schifffahrt eingesetzt werden.
- Schiffe über 500 BRZ, die nicht in der internationalen Schifffahrt eingesetzt werden.
- Alle Passagierschiffe.

Die Bruttoraumzahl wird wie folgt berechnet:

$$BRZ = K_1 \cdot V \tag{2.1}$$

$$K_1 = 0,2 + 0,02 \log V \tag{2.2}$$

Dabei ist V der Zahlenwert des in Kubikmeter gemessenen Inhalts aller geschlossenen Räume [1].

2.1.2 Technischer Hintergrund

Für die Übertragung der AIS-Signale werden die Frequenzen AIS1(161,975 MHz) und AIS2(162,025 MHz) genutzt. Die Kanäle verfügen über eine Bandbreite von jeweils 25 kHz. Auf hoher See beträgt die Signalreichweite 20-30 Seemeilen. [25]
Die AIS-Stationen sind in zwei Typen (Mobil und Stationär) aufgeteilt. Zu den mobilen Stationen gehören die auf Schiffen stationierte Klasse A und Klasse B Typen, die Stationen in Flugzeugen (SAR) und die Navigationshilfen (AtoN Aids to Navogation). Zu den stationären Stationen gehören Basistationen und Wiederholungsstationen. Die IMO hat 26 Nachrichtentypen definiert, von denen in dieser Arbeit nur die Nachrichten der Klasse A näher betrachtet werden.
Die Schiffsstationen der Klasse A werden auf den Schiffen, die im Kapitel V des SOLAS-Übereinkommens beschrieben werden, installiert. Die Klasse B ist für alle anderen Schiffe vorgesehen.
Da alle Teilnehmer ihre Nachrichten gleichzeitig übertragen möchten, muss die Reihenfolge der Übertragungen geregelt werden. Weil alle Teilnehmer auf den selben Freuqenzen senden, bleibt nur die zeitliche Trennung der Signale. Dazu wird ein Time Division Multiple Access(TDMA) Verfahren verwendet. Bei diesem Verfahren wird die Zeit in Fenster aufgeteilt die für AIS einer vollen Minute entsprechen. Jedes Fenster wird in Zeitschlitze aufgeteilt. Bei dem AIS-TDMA wird jedes Fenster in 2250 Zeitschlitze aufeteilt, sodass jeder Schlitz 0,2666 s lang ist(siehe Abb. 2.1). Dabei kann ein Sender während eines Zeitschlitzes(Timeslot) sein Signal übertragen. Dadurch dass die Teilnehmer über GPS auf eine millionstel Sekunde zeitsynchronisiert sind, verwenden sie die geleiche Aufteilung der Zeitschlitze. Um Überschneidungen zu vermeiden, müssen die Zeitschlitze unter den Telnehmern in Reichweite aufgeteilt werden. Dazu wird das selbst-organisierende Protokoll (SOTDMA) verwendet. Dabei

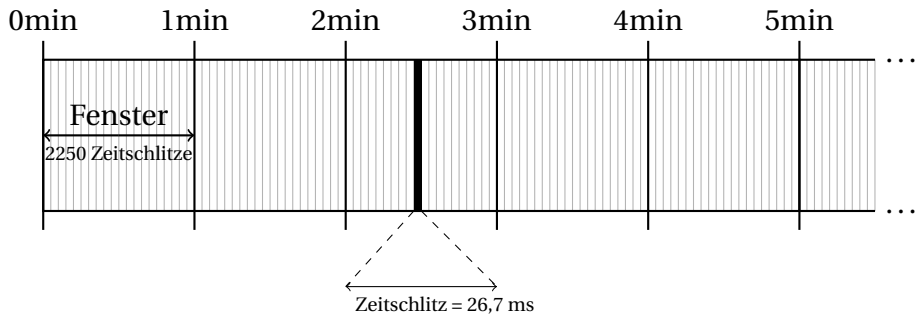


Abbildung 2.1 – *Darstellung des TDMA-Verfahrens bei AIS*

hört ein Sender zuerst eine Minute lang den Kanal ab und erstellt eine Liste mit der Zeitschlitzbelegung. Nach einer Minute sucht sich dann der Sender freie Zeitschlitze aus, die er belegen möchte und reserviert diese, in dem er eine Nachricht während dieser Zeitschlitze sendet. Um die reservierten Zeitschlitze zu behalten, muss ein Sender in jedem Fenster ein Signal während seiner Zeitschlitze senden. [27], [23] Die SOTDMA Nachrichten haben das in der Abb. 2.2 dargestellte Format.

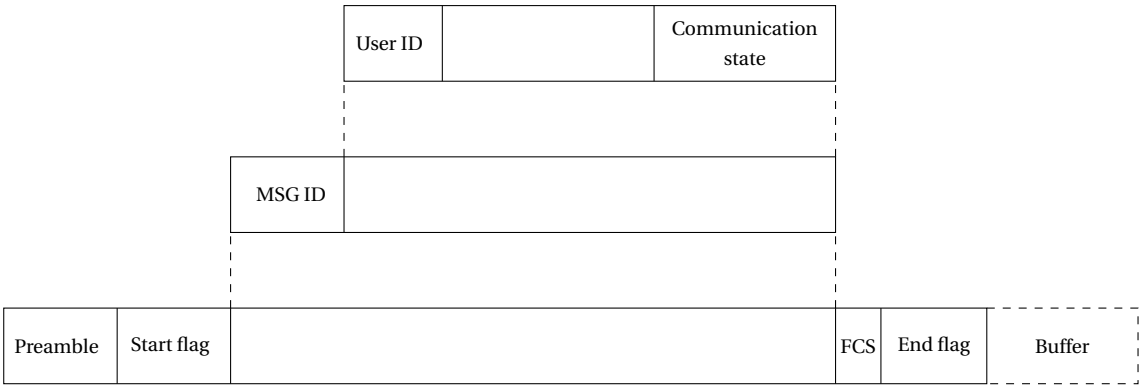


Abbildung 2.2 – *Darstellung einer SOTDMA-Nachricht. FCS - Frame Check Sequence*

Neben dem SOTDMA-Protokoll werden bei AIS weitere vier Protokolle verwendet:

- RATDMA (Random Access Time Division Multiple Access)
- IDTMA (Incremental Time Division Multiple Access)
- FATDMA (Fixed Access Time Division Multiple Access)
- CSTDMA (Carrier Sense Time Division Multiple Access)- für AIS-Klasse B

Tabelle 2.1 – *Sendeintervalle für Klasse A Meldungen*

Geschwindigkeit	Sendeintervall
vor Anker/Festgemacht und nicht schneller als 3 Knoten	3 min
vor Anker/Festgemacht und schneller als 3 Knoten	10 s
0-14 Knoten geradeaus	10 s
0-14 Knoten mit Kursänderung	3,3 s
14-23 Knoten geradeaus	6 s
14-23 Knoten mit Kursänderung	2 s
>23 Knoten geradeaus	2 s
>23 Knoten mit Kursänderung	2 s

Für diese Arbeit wurde ein AIS-Empfänger der Firma Weatherdock AG [6] verwendet. Mithilfe einer Antenne empfängt das Gerät die Signale der beiden AIS-Kanäle. Aus den empfangenen Signalen wird die AIS-Nachricht extrahiert und über eine serielle Schnittstelle weitergegeben. Eine AIS-Nachricht hat die in der Abbildung 2.3 dargestellte Struktur. Die Prüfsumme ist das Ergebnis der Anwendung von XOR auf

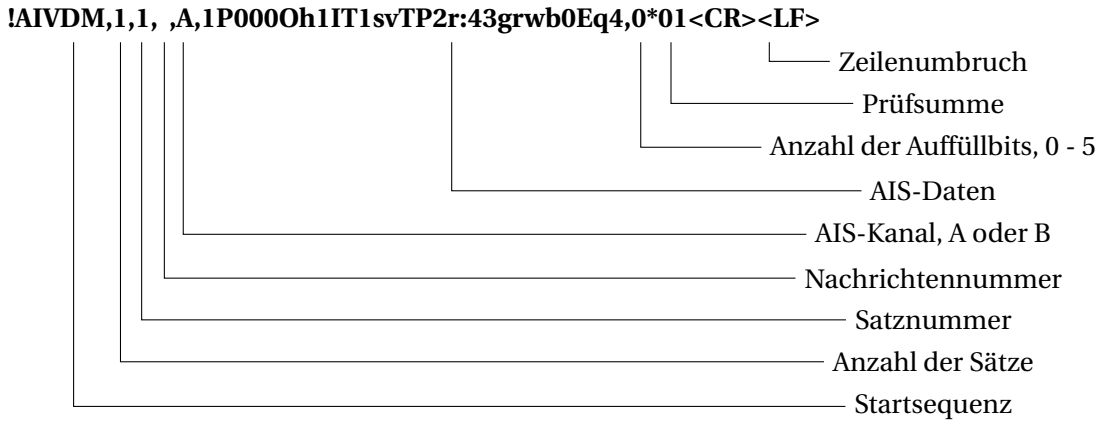


Abbildung 2.3 – *AIS-Satzstruktur. Eine detailierte Beschriebung der Felder ist in [24] zu finden.*

die Zeichen zwischen „!“ und „*“. Die Nachrichten können aus mehreren Teilen bestehen, die nach dem Empfangen zusammengesetzt werden müssen.

Die Schiffsmeldungen sind in zwei Typen aufgeteilt. Das sind zum einen die Meldungen 1,2 und 3, die die dynamischen Informationen enthalten und zum anderen die Meldung 5, die die statischen Informationen enthält. Die dynamischen Meldungen werden in Zeitabsänden abhängig von der Geschwindigkeit übetragen. In der Tabelle 2.1 sind die Zeitabstände dargestellt [27]. Die statische Meldung wird alle 6

Minuten oder wenn sich ein Parameter ändert versendet.

Für die Kodierung der AIS-Daten wird eine 6-bit ASCII Kodierung genutzt. Dabei werden die Daten zu einem binären String zusammengefasst und anschließend in 6-Bit Zeichen aufgeteilt. Zum übertragen werden die 6-Bit Zeichen in 8-Bit ASCII Zeichen umgewandelt. Die 6-Bit Zeichen wurden so ausgewählt, dass sie mit einem einfachen Algorithmus in die 8-Bit ASCII-Zeichen umgewandelt werden können. In der Tabelle 2.2 sind die Ausgewählten Zeichen dargestellt [24]. Nach dem Empfangen, müssen die Nachrichten dekodiert werden. Dazu müssen alle empfangenen Bytes in eine 6-Bit Repräsentation umgerechnet werden. Die empfangenen Bytes werden dabei als Zahlen behandelt. Von jeder Zahl wird am Anfang 48 abgezogen. Wenn die Zahl danach größer als 40 ist, wird nochmal 8 abgezogen. So werden alle empfangenen 8-Bit ASCII Zeichen in eine 6-Bit Repräsentation umgerechnet. Die umgerechneten 6-Bit Zeichen werden dann zu einem Bitstring zusamengesetzt und entsprechend des Nachrichtentyps zerlegt und weiterverarbeitet.
Es werden zwei verschiedene 6-Bit ASCII Darstellungen benutzt. Die bereits beschriebene Darstellung wird bei der Übertragung der Nachrichten genutzt. Die zweite Darstellung wird benutzt, um den in der Nachricht übertragenen Text zu dekodieren. Die zweite Konvertierungstabelle ist in der Tabelle 2.3 dargestellt [27].

2.1.2.1 Dynamische Nachrichten

In diesem Abschnitt ist die Struktur der Nachrichten vom Typ 1,2 und 3 erläutert. Die Nachrichten von diesem Typ werden in einem AIS-Satz übertragen.

Tabelle 2.4 – *Nachrichtentyp 1,2 und 3*

Parameter	Bits	Beschreibung
Nachricht ID	6	Nummer der Nachricht. In diesem Fall 1,2 oder 3.
Wiederholungs-indikator	2	Zeigt an wie oft die Nachricht wiederholt wurde.
Benutzer ID	30	MMSI(Maritime Mobile Service Identity) Nummer
Navigationsstatus	4	0=Motorangetrieben, 1= vor Anker, 2=ohne Kommando, 3=eingeschränkte Manövrierfähigkeit, 4= Eingeschränkt durch Tiefgang, 5= festgemacht, 6=auf Grund, 7 = beim Fischen, 8=segelnd, 9-14 = für später reserviert, 15 = nicht definiert = default

Fortsetzung Nachrichtentyp 1,2 und 3

Parameter	Bits	Beschreibung
Drehrate Rate of turn ROT_{AIS}	8	0 bis +126 = Rechtsdrehung bis zu 708° pro Minute oder höher 0 bis -126 = Linksdrehung bis zu 708° pro Minute oder höher. Die Werte zwischen 0 und 708° sind wie folgt kodiert: $$ROT_{AIS} = 4,733 \cdot \sqrt{ROT_{sensor}} \text{Grad pro Minute} \quad (2.3)$$ ROT_{sensor} ist die Drehrate, die von einem Messgerät geliefert wird. ROT_{AIS} wird zur nächsten Ganzzahl gerundet. +127 = Rechtsdrehung mit mehr als 5° pro 30 s(Kein Messgerät vorhanden) -127 = Linksdrehung mit mehr als 5° pro 30 s(Kein Messgerät vorhanden) -128 zeigt an, dass keine Rotationsinformation vorhanden ist. Drehrate darf nicht von der Kursinformation abgeleitet werden.
Speed over Ground SOG	10	Geschwindigkeit über Grund in Schritten von 1/10 Knoten.(0-102,2 Knoten) 1023 = nicht vorhanden. 1022 = 102,2 Knoten oder höher.
Positionsgenauigkeit	1	1=hoch(<10 m) 0=niedrig(>10 m)
Longitude	28	Geographische Länge in 1/10000 min (±180°, Ost = positiv, West = negativ) 2-er Kompliment bilden. 1° = 60 min. 181 = nicht vorhanden =default.
Latitude	27	Geographische Breite in 1/10000 min (±90°, Nord = positiv, Süd = negativ) 2-er Kompliment bilden. 1° = 60 min. 91 = nicht vorhanden =default.
Course over Ground COG	12	Kurs über Grund in 1/10 Grad. 3600 = nicht verfügbar = default. 3601-4095 sollen ungenutzt bleiben.
True heading	9	Kielrichtung (0-359°) 511 = nicht verfügbar =default
Zeitstempel	6	UTC Sekunde, in der die Nachricht erzeugt wurde.
Spezieller Manöverindikator	2	0= nicht verfügbar = default 1 = mit keinem speziellen Manöver beschäftigt 2= mit einem speziellen Manöver beschäftigt.

Fortsetzung Nachrichtentyp 1,2 und 3

Parameter	Bits	Beschreibung
Frei	3	Nicht benutzt.
RAIM-Flag	1	Receiver automomous integrity monitoring Flag des elektronischen Standortbestimmungsgerätes. 0=RAIM wird nicht benutzt = default 1=RAIM wird benutzt
Kommunikations-status	19	Wird bei SOTDMA verwendet
Summe Bits	168	Benötigt ein Zeitschlitz

2.1.2.2 Statische Nachricht

In diesem Abschnitt wird die Struktur der Nachricht vom Typ 5 erläutert. Die Nachrichten diesen Types werden oft in mehreren AIS-Sätzen übertragen.

Tabelle 2.5 – *Nachrichtentyp 5*

Parameter	Bits	Beschreibung
Nachricht ID	6	Nummer der Nachricht. In diesem Fall 5.
Wiederholungs-indikator	2	Zeigt, an wie oft die Nachricht wiederholt wurde.
Benutzer ID	30	MMSI(Maritime Mobile Service Identity) Nummer
Indikator für die AIS Version	2	0= Station ist konform zu ITU-R M.1371-1 1= Station ist konform zu ITU-R M.1371-3 2,3 = für zukünftige Versionen
IMO Nummer	30	Eine einzigartige Nummer zur Identifikation eines Schiffes. 1-999999999; 0=nicht vorhanden = default
Rufzeichen	42	Sieben 6-Bit ACII Zeichen, @@@@@@@ = nicht vorhanden=default
Name	120	Maximal 20 6-Bit ASCII Zeichen „@@@@@@@@@@@@@@@@@@@@“ = nicht vorhanden = default
Schiffstyp	8	0 = nicht vorhanden = default 1-99 = zu finden in §3.3.2 von [27] 100-199 = für regionalen Gebrauch 200-255 = für die Zukunft reserviert
Schiffsdimension	30	Siehe Abb. **??**

Fortsetzung Nachrichtentyp 5

Parameter	Bits	Beschreibung
Typ des Positionsbestimmugsgerätes	4	0 = undefiniert = default 1 = GPS 2 = GLONASS 3 = kombiniert GPS/GLONASS 4 = Loran-C 5 = Chayka 6 = integriertes Navigationssystem 7 = vermessen(surveyed) 8 = Galleo 9-15 = nicht benutzt
ETA	20	Voraussichtliche Ankunftszeit (Estimated Time of Arrival) MMDDHHMM UTC Bits 19-16: Monat 1-12; 0 = nicht vorhanden Bits 15-11: Tag 1-31; 0 = nicht vorhanden Bits 10-6: Stunde 0-23; 24 = nicht vorhanden Bits 5-0: Minute 0-59; 60 = nicht vorhanden
Aktueller Tiefgang	8	In 1/10 m, 255 = 22.5 m oder größer 0 = nicht vorhanden = default
Ziel	120	Maximal 20 6-Bit ASCII Zeichen „@@@@@@@@@@@@@@@@@@@@“ = nicht vorhanden
DTE	1	Data terminal equipment[1] vorhanden (0 = vorhanden, 1 = nicht vorhanden)
Frei	1	Nicht benutzt.
Summe Bits	424	Benötigt zwei Zeitschlitze.

2.2 Implementierung

In diesem Kapitel wird die Realisierung des AIS-Überwachungssystems beschrieben. Das entwickelte System ist ein Prototyp zur Überwachung des Schiffsverkehrs im Bereich des Hamburger Hafens. Um eine größere Fläche abdecken zu können, müssen mehrere Empfängersationen aufgebaut werden, die die gesammelten Daten an den Zentralrechner weiterleiten. Um das realisieren zu können und das System skalierbar zu machen, wurde das System in die in der Abb 2.5 dargestellten Komponenten auf-

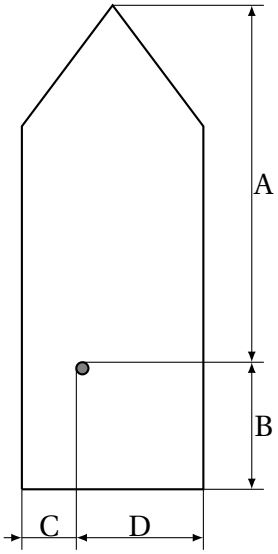
[1]Der Datenterminal indikator signalisiert, ob die sendende Station über die minimale Austattung(Tastatur und Anzeige) verfügt.

Tabelle 2.2 – *6-Bit Darstellung bei der Kodierung*

8-Bit ASCII HEX = binär	Gültige Zeichen	6-Bit Darstellung	8-Bit ASCII HEX = binär	Gültige Zeichen	6-Bit Darstellung
30 = 00110000	0	000000	50 = 01010000	P	100000
31 = 00110001	1	000001	51 = 01010001	Q	100001
32 = 00110010	2	000010	52 = 01010010	R	100010
33 = 00110011	3	000011	53 = 01010011	S	100011
34 = 00110100	4	000100	54 = 01010100	T	100100
35 = 00110101	5	000101	55 = 01010101	U	100101
36 = 00110110	6	000110	56 = 01010110	V	100110
37 = 00110111	7	000111	57 = 01010111	W	100111
38 = 00111000	8	001000	60 = 01100000	’	101000
39 = 00111001	9	001001	61 = 01100001	a	101001
3A = 00111010	:	001010	62 = 01100010	b	101010
3B = 00111011	;	001011	63 = 01100011	c	101011
3C = 00111100	<	001100	64 = 01100100	d	101100
3D = 00111101	=	001101	65 = 01100101	e	101101
3E = 00111110	>	001110	66 = 01100110	f	101110
3F = 00111111	?	001111	67 = 01100111	g	101111
40 = 01000000	@	010000	68 = 01101000	h	110000
41 = 01000001	A	010001	69 = 01101001	i	110001
42 = 01000010	B	010010	6A = 01101010	j	110010
43 = 01000011	C	010011	6B = 01101011	k	110011
44 = 01000100	D	010100	6C = 01101100	l	110100
45 = 01000101	E	010101	6D = 01101101	m	110101
46 = 01000110	F	010110	6E = 01101110	n	110110
47 = 01000111	G	010111	6F = 01101111	o	110111
48 = 01001000	H	011000	70 = 01110000	p	111000
49 = 01001001	I	011001	71 = 01110001	q	111001
4A = 01001010	J	011010	72 = 01110010	r	111010
4B = 01001011	K	011011	73 = 01110011	s	111011
4C = 01001100	L	011100	74 = 01110100	t	111100
4D = 01001101	M	011101	75 = 01110101	u	111101
4E = 01001110	N	011110	76 = 01110110	v	111110
4F = 01001111	O	011111	77 = 01110111	w	111111

Tabelle 2.3 – *6-Bit ASCII Darstellung zum dekodieren des übertragenden Textes*

6-Bit ASCII			**Standart ASCII**		**6-Bit ASCII**			**Standart ASCII**	
Chr	**Dec**	**Binary**	**Dec**	**Binary**	**Chr**	**Dec**	**Binary**	**Dec**	**Binary**
@	0	00 0000	64	0100 0000	!	33	10 0001	33	0010 0001
A	1	00 0001	65	0100 0001	„	34	10 0010	34	0010 0010
B	2	00 0010	66	0100 0010	#	35	10 0011	35	0010 0011
C	3	00 0011	67	0100 0011	$	36	10 0100	36	0010 0100
D	4	00 0100	68	0100 0100	%	37	10 0101	37	0010 0101
E	5	00 0101	69	0100 0101	&	38	10 0110	38	0010 0110
F	6	00 0110	70	0100 0110	‘	39	10 0111	39	0010 0111
G	7	00 0111	71	0100 0111	(	40	10 1000	40	0010 1000
H	8	00 1000	72	0100 1000	)	41	10 1001	41	0010 1001
I	9	00 1001	73	0100 1001	*	42	10 1010	42	0010 1010
J	10	00 1010	74	0100 1010	+	43	10 1011	43	0010 1011
K	11	00 1011	75	0100 1011	,	44	10 1100	44	0010 1100
L	12	00 1100	76	0100 1100	-	45	10 1101	45	0010 1101
M	13	00 1101	77	0100 1101	.	46	10 1110	46	0010 1110
N	14	00 1110	78	0100 1110	/	47	10 1111	47	0010 1111
O	15	00 1111	79	0100 1111	0	48	11 0000	48	0011 0000
P	16	01 0000	80	0101 0000	1	49	11 0001	49	0011 0001
Q	17	01 0001	81	0101 0001	2	50	11 0010	50	0011 0010
R	18	01 0010	82	0101 0010	3	51	11 0011	51	0011 0011
S	19	01 0011	83	0101 0011	4	52	11 0100	52	0011 0100
T	20	01 0100	84	0101 0100	5	53	11 0101	53	0011 0101
U	21	01 0101	85	0101 0101	6	54	11 0110	54	0011 0110
V	22	01 0110	86	0101 0110	7	55	11 0111	55	0011 0111
W	23	01 0111	87	0101 0111	8	56	11 1000	56	0011 1000
X	24	01 1000	88	0101 1000	9	57	11 1001	57	0011 1001
Y	25	01 1001	89	0101 1001	:	58	11 1010	58	0011 1010
Z	26	01 1010	90	0101 1010	;	59	11 1011	59	0011 1011
[	27	01 1011	91	0101 1011	<	60	11 1100	60	0011 1100
\	28	01 1100	92	0101 1100	=	61	11 1101	61	0011 1101
]	29	01 1101	93	0101 1101	>	62	11 1110	62	0011 1110
^	30	01 1110	94	0101 1110	?	63	11 1111	63	0011 1111
-	31	01 1111	95	0101 1111					
Space	32	10 0000	32	0010 0000					



	Bits	Bitfelder	Abstand
A	9	Bit 21 - Bit 29	0-511 m; 511=511 m oder länger
B	9	Bit 12- bit 20	0-511 m; 511=511 m oder länger
C	6	Bit 6 - Bit 11	0-63 m; 63 = 63 m oder länger
D	6	Bit 0 - Bit 5	0-63 m; 63 = 63 m oder länger

Abbildung 2.4 – *Schiffsmaße*

geteilt. Die Implementierung der Softwarebestandteile der Komponenten erfolgte in JAVA.

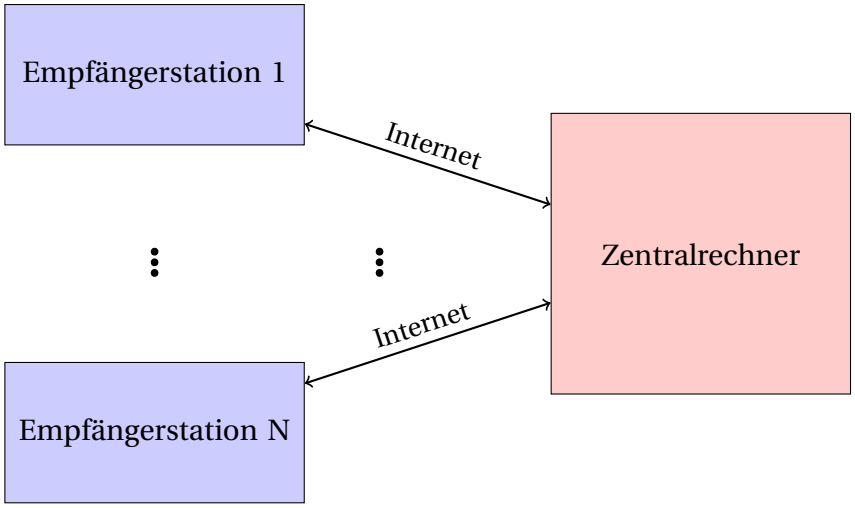


Abbildung 2.5 – *Systemübersicht*

2.2.1 Empfängerstation

Die Aufgabe der Empfängerstation ist das Empfangen der AIS-Signale und deren Weiterleitung an den Zentralrechner. Die Empfängerstation besteht aus einem AIS-Empfänger

mit einer UKW-Antenne und einem internetfähigen Rechner (siehe Abb. 2.6).

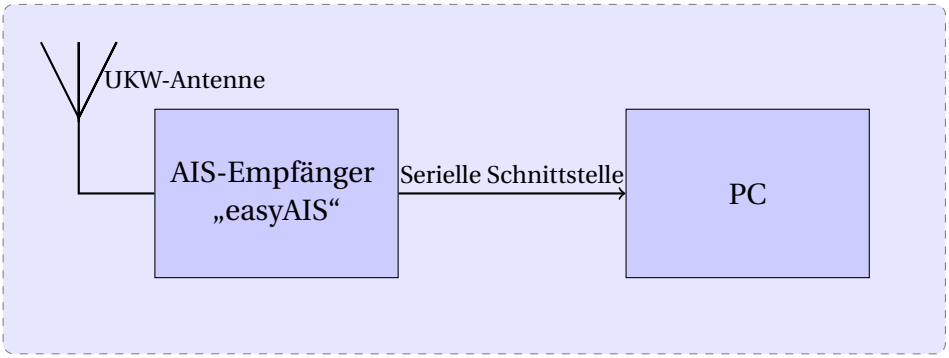


Abbildung 2.6 – *Empfängerstation*

Der eingesetze „easyAIS“ Empfänger empfängt Signale, die auf den beiden AIS-Frequenzen gesendet werden. Die Signale, die als gültige Nachrichten identifiziert werden, werden über eine serielle Schnittstelle weitergeleitet.
Der PC, der mit dem Empfänger verbunden ist, kann diese Nachrichten empfangen. Dazu müssen die folgenden Einstellungen der seriellen Schnittstelle vorgenommen werden:

- Baudrate[2] = 38400
- Wortlänge = 8-Bit; Anzahl Stopbits = 1;Parität = N

Für den PC wurde ein Programm geschrieben, welches die Funktionalität des Empfängermoduls realisiert. Um die serielle Schnittstelle zu überwachen und auszulesen, wurde die Open Source Bibliothek RXTX verwendet. Diese native Bibliothek stellt Klassen zur Verfügung, um mit den parallelen und seriellen Schnittstellen zu kommunizieren.
Für die Kommunikation mit dem AIS-Empfänger wird die ausgewählte Schnittstelle geöffnet und es werden die beschrieben Einstelungen vorgenommen. Dann wird die Schnittstelle überwacht und es wird ein Event ausgelöst, wenn Daten an der Schnittstelle anliegen. Die Daten werden dann Byteweise ausgelesen und in ein Puffer geschrieben bis das Ende der Nachricht erreicht wurde.
Die empfangene Nachricht wird nun ausgewertet. Es wird zuerst die Startsequenz überprüft und alle Nachrichten, die nicht mit „!AIVDM“ starten, werden verworfen. Als Nächstes wird die Prüfsumme berechnet und mit der übertragenen Prüfsumme verglichen. Wenn die beiden Prüfsummen übereinstimmen, wird die Nachricht weiterverarbeitet. Es wird dann die Anzahl der AIS-Sätze und der Nachrichtentyp ausgelesen. Die Daten der Nachrichten werden in die ursprüngliche 6-Bit Form transformiert(sieht Tab. 2.2) und die mehrteiligen Nachrichten werden zusammengesetzt.

[2]Mit Baud wird die Schrittgeschwindigkeit einer seriellen Übertragung bezeichnet.

Danach werden die Daten an den Zentralrechner übertragen. Dazu werden der Nachrichtentyp, die Datengröße und anschließend die Daten übertragen.
Der PC ist als Client mit dem Zentralrechner verbunden. Die Daten werden mithilfe des TCP/IP Protokolls übertragen. Wenn keine Verbindung zu dem Zentralrechner besteht, wird solange versucht die Verbindung aufzubauen, bis es erfolgreich ist.

2.2.2 Zentralrechner

Der Zentralrechner hat die Aufgabe die Daten von den Empfängerstationen und der HPA zu sammeln, zu verwalten und zu visualisieren. Dafür wurde eine JAVA-Anwendung realisiert.

Schiffsdatenverwaltung

Die Hauptklasse dieser Anwendung beihnhaltet die Verwaltung und die Visualisierung der Daten. Die Daten eines Schiffes werden in einem Objekt der Klasse „ShipInfo“ gespeichert. Diese Klasse beihnhaltet alle schiffsrelevanten Parameter. Alle Schiffe werden in zwei Typen aufgeteilt:

- Komplette Schiffe - mindestenst die statische Information oder HPA-Daten vorhanden.
- Unvollständige Schiffe - nur dynamische Information vorhanden.

Die Unterscheidung wird vorgenommen, weil die Daten aus den beiden Quellen über verschiedene Schlüssel einem Schiff zugeordnet werden. Die kompletten Schiffe werden über die IMO-Nummer identifiziert, die unvollständigen über die MMSI-Nummer. Die Schiffe werden in zwei verschiedenen Hashtabellen (*„completeShipMap“ und „incompleteShipMap“*) gespeichert. Die statischen Nachrichten ermöglichen die Verbindung zwischen den beiden Schiffstypen, weil in denen die IMO- und MMSI-Nummer enthalten ist. Diese Verbindung wird durch die Hashtabelle *„mmsiToImoMap“* repräsentiert. Um die Hashtabellen mit Daten zu füllen, wurden Klassen erzeugt, die diese Daten von den Quellen abholen und in die Hashtabellen schreiben.
Für die Verwaltung und Darstellung der HPA-Daten, wurden zusätzlich verschiedene Tabellen erzeugt, die auch gefüllt werden müssen.

AIS-Daten

Um die Daten von den Empfängerstationen zu sammeln wurde eine Server-Klasse definiert, die eine Socketverbidung generiert und auf Verbindungen der Clients wartet. Wenn sich ein Client mit dem Server verbunden hat, wird für diesen Client ein

Thread erzeugt, in dem die Daten dieses Clients empfangen werden, solange die Verbindung besteht. Durch den Threadaufbau können die Daten von mehreren Clients parallel empfangen werden. Die Client-Klasse ist von der Klasse „Observable“ abgeleitet und wird von der Hauptklasse beobachtet.

Die Daten werden nach dem Empfang mit einem Zeitstempel versehen und mit den gesendeten Parametern (Nachrichtentyp und Größe) verglichen. Danach werden die Daten je nach Typ ausgewertet(siehe Tabellen 2.4 und 2.5) und in die Hashtabellen geschrieben. Dafür werden folgende Schritte ausgeführt:

- MMSI-Nummer extrahieren.
- Wenn das Schiff in „completeShipMap“ gefunden wurde:
  - Das gefundene Schiff auswählen
  - Empfangene Nachricht mit der letzten im Schiff gespeicherten Nachricht vergeleichen. Wenn gleich, abbrechen.
- Wenn das Schiff in „incompleteShipMap“ gefunden wurde:
  - Das gefundene Schiff auswählen
  - Empfangene Nachricht mit der letzten im Schiff gespeicherten Nachricht vergeleichen. Wenn gleich, abbrechen.
- Wenn das Schiff in keiner Hashtabelle gefunden wurde:
  - Neues Schiff erzeugen.
  - Neues Schiff in die Hashtabelle einfügen(Typ 1,2,3: „incompleteShipMap“, Typ 5 : „completeShipMap“)
- Nachricht auswerten und die Schiffsdaten aktualisieren.
- Die Hauptklasse über Änderungen informieren.

HPA-Daten

Die HPA-Daten beinhalten zusätzliche Informationen über Schiffsbewegungen im Hamburger Hafen. Diese Daten werden über einen kostenpflichtigen Webservice in Form einer XML-Datei zur Verfügung gestellt. Da keine Beschreibung dieser Daten bereitgestellt wird, musste der Ihnhalt zuerst interpretiert werden.
Aus den Daten wurden vier wichtige Bewegungstypen der Schiffe extrahiert:

- „+“- Das Schiff ist unterwegs in den Hafen.

- „- “- Das Schiff verlässt den Hafen.
- „0“- Das Schiff wechselt das Terminal innerhalb des Hafens.
- „H“ - Das Schiff ist im Hafen festgemacht.

Es können mehrere Einträge für ein Schiff existieren, die die bekannten Bewegungen eines Schiffes repräsentieren. Es wurde der Thread „HPA-Table“ erzeugt, der die Aufgabe hat die Daten in bestimmten Abständen zu aktualisieren und sie in Tabellen und Hashtabellen zu schreiben, die von der Hauptklasse bereitgestellt werden. Es werden folgende Schritte ausgeführt:

- Abfragen des Webservices. Es wird eine XML-Datei erzeugt.
- XML-Tags in eine Liste schreiben.
- Eine temporäre Hashtabelle für Schiffe erzeugen „tempShipMap“.
- Für jeden XML-Tag:
  - IMO-Nummer auslesen
  - Prüfen, ob das Schiff mit der ausgelesenen IMO-Nummer bereits in „tempShipMap“ enthalten ist.
  - Wenn das Schiff neu ist, einen Eintrag in „tempShipMap“ erzeugen.
  - Bewegungstyp prüfen und ggf. neuen Auftrag erzeugen und dem Schiff hinzufügen.
  - Alle Attribute in die dafür vorgesehene Tabelle als Zeile einfügen.
- Die „tempShipMap“ wird nun mit der „completeShipMap“ abgeglichen.
- Die Tabellen aktualisieren.

Visualisierung

Für die Visualisierung und Bedienung der Applikation wurde eine graphische Benutzeroberfläche realisiert. Ein wichtiger Bestandteil der Benutzeroberfläche ist die Darstellung der empfangenen AIS-Signale. Die Signale werden in der Funktion „update“, die zum Beobachter-Muster gehört, visualisiert. Als Grundlage der Darstellung dient eine Karte, die von OpenStreetMap[3] bezogen wird. Für die Einbindung der Karte in die Benutzeroberfläche wird die Swingx-Bibliothek [5] verwendet. Die Bibliothek biete die Möglichkeit Objekte auf der Karte mittels „Painter“ darzustellen. Die Schiffe werden maßstabgetreu und farblich unterschiedlich dargestellt.
Eine genauere Beschreibung der graphischen Benutzeroberfläche ist im Anhang zu finden.

[3] OpenStreetMap ist eine frei verfügbare Weltkarte, die von Benutzern erstellt wird. (siehe [4])

Kapitel 3

Multiagentensysteme

In diesem Kapitel wird eine Einführung in Multiagentensysteme gegeben. Multiagentensysteme sind Systeme, die aus mehreren interagierenden Agenten bestehen [42]. Multiagentensysteme werden heutzutage in vielen verschiedenen Bereichen wie Gesundheitswesen, Finanzwesen, Informationsbeschaffung, Logistik und vielen Anderen eingesetzt. Es ist eine junge aber mittlerweile weit verbreitete Disziplin der Informatik.

3.1 Was sind Agenten?

Wenn man sich mit dem Thema beschäftigt, ist es wichtig eine Definition für den Agenten zu haben. In der Literatur existieren mehrere verschiedene Definitionen des Begriffes Agent. In [30] werden einige dieser Definitionen aufgelistet. Dazu gehören unter anderem die folgenden Definitionen:

- **Russel und Norvig:**(1995) *„Als Agent kann alles bezeichnet werden, das seine Umgebung durch Sensoren wahrnimmt und mittels Effektoren in dieser Umwelt handelt“*. [38]
- **Maes:**(1995) *„Automome Agenten sind Computersysteme, die in einer komplexen, dynamischen Umgebung selbstständig wahrnehmen und agieren und dabei verschiedene Ziele oder Aufgaben bewältigen, für die sie entwickelt wurden“* [30], [28]
- **Hayes-Roth:**(1995) *„Intelligente Agenten führen kontinuierlich drei verschiedene Funktionen aus: die Wahrnehmung dynamischer Bedingungen in der Umgebung, Handlungen zur Beinflussung der Bedingungen in der Umgebung und logisches Denken, um Wahrnehmungen zu interpretieren, Probleme zu lösen, Schlussfolgerungen zu ziehen und Handlungen zu bestimmen.“* [30], [22]

- **Wooldridge**:(2001) *„ Ein Agent ist ein Computersystem, welches sich in einer Umgebung befindet und welches in der Lage ist, autonome Handlungen in dieser Umgebung auszuführen, um Designziele zu erfüllen“* [42]

In den Definitionen spielen Begriffe wie „Umgebung“, „Handlung“ und „Wahrnehmung“ eine zentrale Rolle und verbinden die Definitionen miteinander. Eine etwas ausführlichere Definition stammt von Wooldridge und Jennings 1995:
„ Ein Agent ist ein auf Hardware oder (häufiger) auf Software basierendes Computersystem, das über die folgenden Eigenschaften verfügt:

- ***Autonomie**: Agenten operieren ohne direkte menschliche oder andere Einwirkungen und haben eine gewisse Kontrolle über ihre Handlungen und ihren inneren Zustand.*
- ***Soziale Fähigkeiten**: Agenten stehen in einer Beziehung mit anderen Agenten (und vielleicht mit Menschen) mittels einer Art Agentensprache.*
- ***Reaktivität**: Agenten nehmen ihre Umwelt wahr (welche eine physikalische Welt, ein Benutzer über eine grafische Benutzeroberfläche, eine Reihe anderer Agenten, das Internet oder eine Kombination dieser Möglichkeiten sein kann) und reagieren rechtzeitig auf Veränderungen.*
- ***Proaktivität**: Agenten reagieren nicht nur auf ihre Umgebung, sondern sind auch zu zielgerichteten Verhalten fähig, indem sie Iniativen ergreifen.“ [43]*

Diese Fähigkeiten beschreiben laut Wooldridge einen intelligenten Agenten.
Oft werden Agenten mit Objekten der objektorientierten Programmierung verwechselt, weil die Objekte durch eigene Funktionen und Variablen, die von anderen Objekten aufgerufen werden können, einen Grad an Selbstständigkeit aufweisen. Die Objekte können kontrollieren, ob ihre Funktionen und Variablen öffentlich(public- für andere Objekten sichtbar) oder privat (private- nur innerhalb des Objektes sichtbar) sind, nicht aber wer und wann auf die öffentlichen Parameter zugreift. Damit ein Agent eine Aktion ausführt, weil ein anderer Agent es von ihm möchte, muss er eine Anfrage des anderen Agenten erhalten. Wenn der Agent eine Anfrage erhält, kann er selbst entscheiden, wie er darauf reagiert und ob er die gewollte Aktion ausführt oder nicht. Man kann folgende drei Unterscheidungskriterien festlegen(aus [42]):

- Agenten verkörpern eine strengere Auffassung von Autonomie als Objekte und sie können insbesondere für sich entscheiden, ob sie eine Aktion auf Anfrage eines anderen Agenten ausführen oder nicht.
- Agenten sind im Stande zum flexiblen(reaktiv, proaktiv, sozial) Verhalten wogegeb das Standartmodell der Objekte keine Aussagen über solche Verhaltenstypen macht.

- Ein Multiagentensystem ist an sich nebenläufig(multi-threaded) und es wird davon ausgegangen, dass jeder Agent über mindestens einen Thread verfügt.

3.2 BDI-Agenten

Die Belief-Desire-Intention(BDI) Architektur hat ihre Ursprünge in den philosophischen Theorien von Bratman [9], [10]. Rao und Georgeff [37], [36] haben dann eine formale Beschreibung dieser Architektur entwickelt.
Die Architektur besteht aus den Komponenten Überzeugung(belief), Ziel(desire) und Absicht(intention). Diese können wie folgt beschrieben werden:

- **Überzeugung(belief):** Die Überzeugung ist das Wissen über den Zustand der Umwelt, welches aufgrund von Wahrnehmungen ständig aktualisiert wird. Die Komponente kann durch eine Variable, eine Datenbank oder eine andere Datenstruktur repräsentiert werden.

- **Ziel(desire):** Das Ziel ist die Komponente, welche die zu erreichenden Ziele eines Agenten beschreibt. Die Ziele können auf verscheidene Weisen generiert werden und sind notwendig, um den Agenten zum Handeln zu bewegen.

- **Absicht(intention):** Absicht beschreibt die ausgewählten Aktionen, mit denen ein Ziel erreicht werden soll. Die Komponente besitzt eine Bibliothek mit Plänen(Sequenz von Aktionen), aus denen Pläne ausgesucht werden, die zum erreichen eines Ziels führen können.

Die von Rao und Georgeff entworfene Architektur beihnhaltet die drei Datenstrukturen von Überzeugung, Ziel und Absicht und ergänzt diese mit einer Warteschlange von Ereignissen. Auf den Datenstrukturen sind Abfrage- und Aktualisierungsoperationen erlaubt. Es werden sowohl externe wie auch interne Ereignisse wahrgenommen. Es wird auch angenommen, dass die Ereignisse atomar sind und sofort nach dem Auftreten erkannt werden. Die Aktionen am Ausgang werde auch als atomar angenommen. Der Ablauf wird durch die Interpereterschleife in Alg. 1 beschrieben. Zuerst werden die Optionen mithilfe des „Options-generator“s aus der Ereigniswarteschlange ausgelesen. Dann wählt der „Deliberator“ einige Optionen aus und fügt sie zu der Struktur „Absichten“ hinzu. Wenn eine Absicht eine Aktion ausführen möchte, wird diese vom Agenten ausgeführt. Alle externen Ereignisse, die in dieser Zeit passieren, werden zu der Warteschlange hinzugefügt. Die internen Ereignisse werden sofort hinzugefügt. Nun aktualisiert der Agent die Absicht- und die Zielstrukturen, indem alle erreichten Ziele und ausgeführten Absichten verworfen werden. Auch die unereichbaren Ziele und unausführbaren Absichten werden verworfen.
Das Wissen repräsentiert den momentanen Zustand der Umwelt, der sich über Zeit

```
initialisierung();
repeat
    options ← options_generator(Warteschlange) ;
    selected_options← deliberate(options);
    update_intentions(selected_options);
    execute();
    get_new_external_events();
    drop_successful_attitudes();
    drop_imposible_attitudes();
until true ;
```

Algorithmus 1 : Kernschleife des Interpreters eines BDI-Agenten

ändern kann. Es werden nur einfache Literale ohne Disjunktion und Implikation verwendet.
Die Hilfsmittel, die zum Erreichen von Zielen zur Verfügung stehen, werden als Pläne bezeichnet und können als eine Form von Wissen angesehen werden. Jeder Plan hat einen Rumpf(body), der eine Abfoge von Aktionen oder Unterzielen enthält, mit deren Hilfe der Plan erfoglreich abgeschlossen werden kann. Damit ein Plan augewählt werden kann, müssen zwei Bedingungen erfüllt sein:

- Aufrufbedingung(invocation condition): Diese Bedingung repräsentiert das Auslöserevent, welches nötig ist, um den Plan aufzurufen.
- Vorbedingung(precondition): Diese Bedingung beschreibt die Umstände, die für die Ausführung des Plans notwendig sind.

Jede Absicht, die aus bestimmten Plänen besteht, wird mithilfe eines Stacks von hierarchisch zugehörigen Plänen realisiert. Es können Stacks für mehrere Absichten parallel aufgebaut sein. [36]
Es existieren mehrere Frameworks wie dMArs [16], JACK [14] und JADEX(siehe Kap. 4.2), die diese Architektur realisieren.

3.3 Interaktion und Kooperation zwischen Agenten

In einem Multiagentensystem agieren mehrere Agenten mit möglicherweise unterschiedlichen Zielen in der gleichen Umgebung. Das hat zur Folge, dass die Agenten sich gegenseitig durch ihre Handlungen beinflussen. Diese dynamische Beziehung, die aus einer Reihe von Aktionen entsteht, deren Konsequenzen das zukünftige Verhalten der Agenten beienflussen, wird Interaktion gennant. Die Agenten können mithilfe von Ereignissen Kontakt zueinander aufnehmen. Dieser Kontakt kann direkt oder über die Umgebung erfolgen. Für eine Interaktion werden laut Ferber [18] folgende Voraussetzungen benötigt:

- Eine Menge von Agenten, die agieren und/oder kommunizieren können.
- Situationen, in denen sich Agenten begegnen: Zusammenarbeit, Bewegung von Fahrzeugen unter Beachtung des Kollisionsrisikos, Nutzung begrenzter Ressourcen, die Regulierung des Gruppenzusammenhalts.
- Dynamische Elemente, die lokale und zeitlich bergrenzte Beziehungen zwischen Agenten erlauben: Kommunikation, Einfluss, Kräftefelder im Hinblick auf die Anziehung und Abstoßung von Agenten, usw.
- Ein gewisser Spielraum in der Beziehung zwischen Agenten, das es ihnen nicht nur ermöglicht eine Beziehung zu erhalten, sondern sich auch wieder von ihr zu lösen. Dieses ist Voraussetzung für individuelle Autonomie. Wären Agenten vollständig durch einen festen Zusammenhang an einander gebunden, dann würde ihre Interaktion starr und sie agierten nicht länger im eigentlichen Sinn des Wortes, noch besäßen sie die für unseren Agentenbegriff als notwendig angesehene Autonomie.

In diesem Zusammenhang wird auch der Begriff der Interaktionssituation eingeführt und bezeichnet eine zusammengehörende Menge von Verhaltensweisen, die aus der Gruppierung von Agenten entstehen, die agieren müssen, um ihre Ziele zu erreichen, und dabei zugleich ihre mehr oder weniger begrenzten Ressourcen und Fähigkeiten zu beachten haben.

Die drei Elemente Ziele, Ressourcen und Fähigkeiten der Agenten können zu verschiedenen Interaktionssituationen führen. Diese Interaktionsarten werden in [18] erläutert.

Wenn Agenten ein gemeinsames Ziel haben und versuchen einander zu helfen und eventuelle Konflikte zu lösen, spricht man von Kooperation. Ferber definiert zwei Bedingungen für eine Kooperation, von denen eine erfüllt sein muss:

- Das Hinzufügen eines neuen Agenten erlaubt es, die Performanz einer Gruppe in messbarer Weise zu erhöhen.
- Die Aktionen der Agenten dienen dazu, potenzielle oder aktuelle Konflikte zu vermeiden oder aufzulösen.

Für die Realisierung der Kooperationen werden in [18] sechs Methoden beschrieben:

- **Gruppierung und Vervielfältigung:** Diese Methode hat die Aufgabe, die Agenten physisch anzunähern. Das kann durch Bildung einer Einheit im Raum oder eines Netzwerkes sein. Durch die Gruppenbildung können Agenten ihre Performance und die Stabilität des Systems erhöhen. In der Tierwelt werden auch Gruppen gebildet, um eine Jagt durchzuführen oder nach Futter zu suchen, was auf die Softwareagenten übertragen werden kann.

- **Kommunikation:** Die Kommunikation ist ein sehr wichtiger Teil einer erfolgreichen Kooperation. Die Agenten können so Informationen und Wissen austauschen, Aufgaben verteilen und die Ausführung koordinieren. Der Austausch der Information kann über Nachrichten erfolgen.

- **Spezielisierung:** Die Agenten können sich im Laufe der Zeit füe eine bestimmte Aufgabe spiezialisieren, wodurch sie diese Aufgabe effizienter lösen können. Ein Agent kann von Anfang an spezialisiert sein oder sich diese Eigenschaft nach und nach durch lernen aneignen.

- **Zusammenarbeit durch gemeinsamen Zugriff auf Aufträge und Ressourcen:** Die Zusammenarbeit erfolgt, wenn die Agenten eine gemeinsame Aufgabe haben. Dazu werden Methoden verwendet, die den Agenten ermöglichen, Aufgaben, Informationen und Ressourcen untereinander aufzuteilen. Es muss entschieden werden, welcher Agent welche Aufgabe übernimmt. Die Aufgaben werden mithilfe von Angebots- und Nachfragemechanismen verteilt. Es werden dabei zentralisierte und verteilte Ansätze unterschieden. Bei den zentralisierten Ansätzen übernimmt ein Koordinationsagent das Angebot und die Nachfrage und sorgt dann füe eine optimale Verteilung, wogegen können bei den verteilten Ansätzen alle Agenten gleichzeitig ein Angebot abgeben und nachfragen. In Kap. 3.4 wird eine Protokollstruktur für einen verteilten Ansatz beschrieben.

- **Koordination von Aktionen:** Neben den produktiven Aktionen führen Agenten auch unproduktive Aktionen aus, die aber dazu dienen die Bedingungen für produktive Aktionen zu verbessern. Diese Aktionen sind notwendig, um das Verhalten der Agenten in Zeit und Raum so zu ordnen, dass die Performanz des Systems durch bessere Leistungen oder durch eine Reduzierung von Konflikten verbessert wird.

- **Konfliktlösung durch Schlichten und Verhandeln:** Schlichtungen und Verhandlungen werden bei Multiagentensystemen eingesetzt um Konflikte zu lösen und dadurch sicherzustellen, dass die Konflikte nicht ausarten und die Performanz des Systems nicht beinträchtigt wird. Um solche Konflikte zu vermeiden, werden Verhaltensregeln definiert. Man versucht die Auflösung der Konflikte den Agenten zu überlassen, ohne eine Schlichtungsinstanz einzusetzen.

3.4 Kontraktnetz

Ein einfacher und oft eingesetzter Mechanismus zur Verteilung von Aufgaben in einem verteilten System ist das Kontraktnetz. Dieses Verfahren wurde 1979 von R.G.

Smith vorgeschlagen und orientiert sich an dem Verlauf für das Schließen von Verträgen auf dem Markt. Im Kontraktnetz befinden sich ein Manager, der die Aufgaben anbietet und mehrere Bieter, die diese Aufgabe bekommen wollen. Es werden vier Schritte ausgeführt [18]:

1. Zuerst wird die Anfrage formuliert. Der Manager sendet die Beschreibung der Aufgabe entweder an alle ihm geeignet erscheinenden Agenten oder an alle im System verfügbaren Agenten.

2. Im Zweiten Schritt prüfen die Bieteragenten die Aufgabenbeschreibung und erzeugen ggf. ein Angebot, welches sie an den Manager zurücksenden.

3. Der Manager erhält und bewertet die Angebote und schließt den Kontrakt mit dem Bieter, der das beste Angebot vorgelegt hat.

4. Im vierten Schritt bestätigt der Bieter, der für sein Angebot den Zusschlag erhalten hat und damit zum Kontraktor geworden ist, eine Nachricht an den Manager, um diesem mitzuteilen, dass er die betreffende Aufgabe bearbeiten wird und sich verpflichtet das auch zu tun. Alternativ informiert er den Manager darüber, dass er nicht mehr für die Bearbeitung der Aufgabe zur Verfügung steht. Der Manager muss dann die abgegebenen Gebote erneut prüfen (Schritt 3) und zu einer anderen Zuweisung kommen.

Für die Implementierung werden in [18] folgende Nachrichtentypen vorgeschlagen:

- **RequestForBids(T,X)** Aufforderung des Managers X an einen potenziellen Bieter, bei Interesse ein Angebot zur Berarbeitung der Aufgabe T vorzulegen.

- **Propose(T,0)** positive Antwort des Bieteragenten Y auf eine Nachricht RequestForBids in Bezug auf Aufgabe T. Der Bieter sendet dabei das Angebot 0 an den Manager Zurück, da es neben den Daten auch den Namen des Bieters enthält.

- **NotInterested(T,Y)** Negative Antwort eins Anbieters an den Manager. Auf diese Weise erfährt der Manager, dass Y die Aufgabe T nicht ausführen wird.

- **Award(T,C,X)** Nachricht vom Manager X an einen Bieter, mit der er diesen davon in Kenntnis setzt, dass mit ihm der Vertrag C in Bezug auf die Bearbeitung der Aufgabe T geschlossen wird.

- **Accept(T,Y)** Bestätigung des Bieters Y, den angebotenen Vertrag zu erfüllen.

- **Refuse(T,Y)** Ablehnende Antwort eines Bieters Y, um den Manager darüber zu informieren, dass er die ihm zur Bearbeitung übergebene Aufgabe nicht ausführen kann.

In vielen Systemen kann davon Ausgegangen werden, dass der ausgewählte Bieter den Auftrag erfüllen wird. Das würde die letzten beiden Nachrichten überflüssig machen und die Menge an verschickten Nachrichten im Netz reduzieren.
Ein weiterer wichtiger Aspekt ist die Sprache, die die Agenten zum Kommunizieren gebrauchen. Es muss festgelegt sein, wie der Ihnhalt einer Nachricht aufgebaut ist und dieser Aufbau muss für alle Agenten gleich sein. Smith [39] hat auch ein Vorschlag für eine Kontraktsprache gemacht. Dazu hat er Nachrichtendefinitionen entwickelt, die die Beschreibung der Aufgabe, Kontraktnummer, benötigte Fähigkeiten, Form der Angebote und Ablaufzeitpunkt beihnhaltet. Unter der Form der Angebote, versteht man die Kriterien, die von den Bietern zum Erstellen der Gebote herangezogen werden sollen. Der Systemdesigner hat die Aufgabe, Bewertungsfunktionen festzulegen, die die Berechnung eines Gebotes ermöglichen.
Eine Aufgabe bei der Realisierung des Kontraktnetzes, die gelöst werden muss, ist das Erreichen von Bietern. Das bedeutet, dass der Manager wissen muss, welche potenziellen Bieter existieren und wie man diese erreicht. Wenn man ein kleines Netzwerk von Agenten hat, in dem der Manager eine Liste aller Agenten zur Verfügung hat, kann dieses Problem vernachlässigt werden.
Es ist auch sinnvoll das Warten auf Gebote zeitlich zu begrenzen, weil es vorkommen kann, dass ein Agent ausfällt und kein Gebot abgibt. Das würde dazu führen, dass der Manager eine unbestimmte Zeit Wartet und das System dadurch blockiert. Wenn man eine Zeitschranke einsetzt, muss eine Zeit gewählt werden, die nicht zu kurz(Gebote gehen verloren) aber auch nicht zu lang(unnötiges Warten) ist. Dazu können adaptive Verfahren eingesetzt werden, die die Wartezeit optimieren.

Kapitel 4

Werkzeuge für die Agentenrealisierung

In diesem Kapitel werden die Werkzeuge beschrieben, die benutzt werden, um Agentensysteme zu modellieren und zu implementieren.

4.1 TROPOS

TROPOS [15], [40], [20], [13] ist eine Modellierungsmethode, die die Enwicklung von agenetenorintierter Software erleichtern soll. Es wurde als ein gemeinsames Projekt der Univesitäten Toronto (Kanada), Louvain (Belgien), Pernambuco (Brasilien), Trento (Italien) und ITC-Irst (Italien) entwickelt. Die Methode ist an den Grundgedanken der BDI-Agenten angelehnt. Aus der *i** Methode von Eric Yu [44] wurden Konzepte „actor“ (Akteur), „goal“ (Ziel), „task“ (Aufgabe), „resource“(Ressourcen) und „social dependency“ (Soziale Abhängigkeit) übernommen. Die *i** Methode wurde entwickelt, um die frühe Anforderungsanalyse für das System und dessen Umgebung zu modellieren.
Die TROPOS-Methode unterstützt die folgenden fünf Phasen der Softwareentwicklung:

- **Early Requirements** Während der frühen Anforderungsanalyse beschäftigt sich mit dem Verständnis der organisatorischen Struktur, in der das zu entwickelnde System funktionieren soll. Es werden alle Beteligten aus der Umgebung als Akteure, die Ziele haben und Abhängigekeiten zum Deligieren von Zielen, Plänen und Ressourcen aufweisen, modelliert.

- **Late Requirements** In der späten Anforderungsanalyse wird das Modell mit dem zu entwickelnden System erweitert. Das System wird auch als Akteur dargestellt und die Abhängigkeiten zu den anderen Akteuren repräsentieren die funktionalen und nichtfunktionalen Anforderungen an das System.

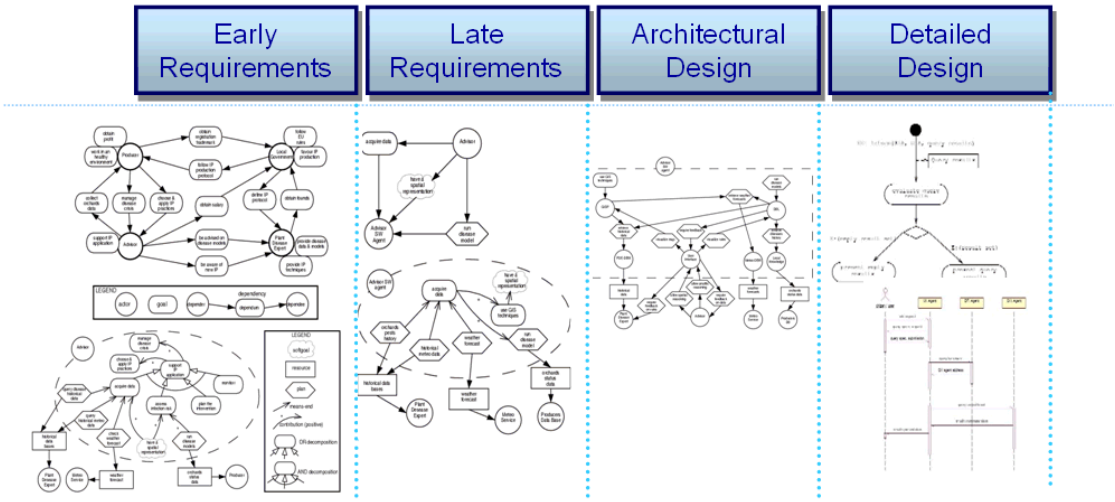


Abbildung 4.1 – *TROPOS-Phasen der Softwareentwicklung. Bild aus [8]*

- **Architectural Design** Das Architekturdesign beschäftigt sich mit der globalen Struktur des Systems mit den Teilsystemen. Die Teilsysteme werden auch als Akteure dargestellt. Die Daten- und Kontrollverbindungen zwischen den Teilsystemen werden durch Abhängigkeiten repräsentiert. Es ist auch erlaubt, die Teilsysteme auf Softwareagenten abzubilden.

- **Detailed Design** In diesem Schritt werden das Verhalten der Agenten und deren Kommunikation detailiert beschrieben. Dazu können Darstellungen von Protokollen, UML-Diagramme und sonstige Diagramme verwendet werden.

- **Implementation** Die letzte Phase der Agentenentsicklung ist die Implementierung. In dieser Phase wird das Modell aus den ersten vier Schritten auf eine Agentenplattform übertragen. Die in der Literatur am häufigsten erwähnten Plattformen sind JACK [14] und JADEX(siehe Kap. 4.2). JADEX ist auch die Plattform, die für die Implementierung der Agenten in dieser Arbeit eingesetzt wird.

Für die Modellierung werden folgende Begriffe verwendet:

- **Actor(Akteur)**Ein Akteur repräsentiert einen physischen Agenten oder einen Softwareagenten sowie eine Rolle oder Position mit Zielen und Absichten. Eine Rolle repräsentiert ein bestimmtes Verhalten in einem bestimmten Kontext, wie „Käufer“ und „Verkäufer“. Ein Agent kann mehrere Rollen darstellen, die man dann als Position bezeichnet. Die Rollen „Käufer“ und „Verkäufer“können zum Beispiel zu der Position „Kaufmann“ zusammengefasst werden.

- **Goal(Ziel)** Unter Zielen versteht man die strategischen Interessen eines Agenten. Bei TROPOS werden zwei Typen von Zielen „Hard goal“ (hartes Ziel) und „Softgoal“ (schwaches Ziel) unterschieden. Bei einem harten Ziel kann geprüft werden, ob es erreicht ist oder nicht, was bei einem schwachen Ziel nicht möglich ist. Zum Beispiel wäre das Ziel „100€ verdienen“ ein hartes Ziel und das Ziel „glücklich sein“ ein schwaches. Schwache Ziele werden vorallem verwendet, um nichtfunktionale Anforderungen zu modellieren.

- **Plan** Ein Plan ist die Darstellung des Weges zum Ziel.

- **Ressource** Eine Ressource ist ein physiches oder informationelles Objekt.

- **Dependency (Abhängigkeit)** Eine Abhängigkeit zwischen zwei Akteuren zeigt an, dass der eine von dem anderen zwecks Erreichen eines Ziels, Ausfürung eines Plans oder Bereitstellung einer Ressource abhängt. Der erste Agent wird als *„depender“* und der zweite als *„dependee“* bezeichnet. Das Objekt, welches zur Abhängigkeit führt, wird *„dependum“* genannt. Die Abhängigkeit zwischen Agenten, erlaubt ihnen Ziele zu erreichen, die sie sonst vielleicht nicht erreichen könnten. Ein Agent ist aber auch darauf angewiesen, dass ein Anderer einen von ihm angeforderten Plan ausführt, das angeforderte Ziel erreicht oder die angeforderte Ressource liefert.

- **Capability (Fähigkeit)** Die Fähigkeit repräsentiert das Können eines Agenten, einen Plan zu definieren, auszuwählen und auszuführen, um ein Ziel unter Berücksichtigung des Zustandes der Umwelt und bestimmter Events zu erreichen.

- **Belief (Überzeugung)** Die Überzeugung repräsentiert das Wissen des Agenten über die Umwelt.

Das Gesamtmodell kann in folgende Untermodelle zerlegt werden:

- **Akteurmodellierung** Die Akteurmodellierung besteht daraus, alle Akteure aus der Umgebung und die Akteure und Agenten des Systems zu bestimmen. Bei der frühen Anforderungsanalyse werden die Akteure modelliert, die in der Umgebung aktiv sind und ihre Ziele verfogen. Bei der späten Anforderungsanalyse wird das System als Akteur modelliert und dann bei dem Architekturdesign in Untersysteme zerlegt. Bei der detailierten Designanalyse werden die Agenten abhängig von der verwendeten Plattform spezifiziert und schließlich bei der Implementierung als Code modelliert.

- **Abhängigkeitenmodellierung** beinhaltet die Bestimmung der Abhängigkeiten zwischen den Agenten. In der frühen Anforderungsanalyse werden die Abhän-

gigkeiten zwischen den Akteuren in der Umgebung modelliert. Nach der Zielmodellierung die weiter unten beschrieben wird, können weitere Abhängigkeiten entstehen. Bei der späten Anforderungsanalyse werden die Abhängigkeiten des zu implementierenden Systems analysiert. Beim Architekturdesign wird der Kontrollfluß durch Abhängigkeiten zwischen den Untersystemen dargestellt und bildet die Grundlage für die spätere Fähigkeitenmodellierung.

- **Zielmodellierung** analysiert die Ziele der Akteure aus deren Sicht und benutzt dafür drei folgende Techniken: Zweck-Mittel-Analyse (means-end analysis), Beitragsanalyse (contribution analysis) und UND/ODER-Zerlegung (AND/OR decomposition). Die Zweck-Mittel-Analyse bestimmt Unterziele, Pläne, Ressourcen und schwache Ziele, die zum Erreichen des Ziels verwendet werden können. Unter Beitrag versteht man in diesem Zusammenhang den positiven oder negativen Einfluss eines Ziels auf das untersuchte Ziel. Das wird häufig bei den schwachen Zielen verwendet und wird durch die qualitative Metrik (- -, -, +, ++) dargestellt. Die UND/ODER-Zerlegung erlaubt die Zerlegung eines Ziels in Teilziele, von denen entweder alle(AND) oder eines(ODER) erfüllt werden müssen. Die Zielanalyse wird bei der frühen und späten Anforderungsanalyse und beim Architekturdesign verwendet.

- **Planmodelierung** Bei der Modellierung der Pläne werden die selben Techniken wie bei der Zielanalyse verwendet.

- **Fähigkeitenmodellierung**. Die Modellierung der Fähigkeiten beginnt in der Phase des Architekturdesigns, wenn alle Akteure spezifiziert wurden. Es werden die Fähigkeiten der einzelnen Agenten analysiert und modelliert. In der detailierten Designanalyse werden die Fähigkeiten weiter spezifiziert und anschließend in die ausgewählte Plattform übertragen.

Für die Darstellung des Modelle werden verscheidene Typen von Diagrammen verwendet. Für die Akteur- und Abhängigkeitsmodellierung verwendet man das Akteurdiagramm(actor diagram) und für die Ziel- und Planmodellierung wird das Zieldiagramm verwendet. In diesen Diagrammen werden die in der Abb. 4.2 dargestellten Symbole für die verschiedenen Begriffe verwendet. Beispiele der Diagramme sind in Kap. 5 dargestellt.

Um die Modellierung zu erleichtern, wurde ein Eclipse Plug-in namens TAOM4E[1] [8], [31], [41] entwickelt. Das Tool stellt eine graphische Oberfläche bereit, die das Erstellen von TROPOS-Diagrammen per Drag-and-Drop ermöglicht. Es werden alle Phasen der Softwareentwicklung unterstützt, nur bei dem detailierten Design gibt es keine symbolische Unterstützung zum Erstellen von Diagrammen. Es ist ein Export

[1] Tool for Agent Oriented visual Modeling for the Eclipse platform

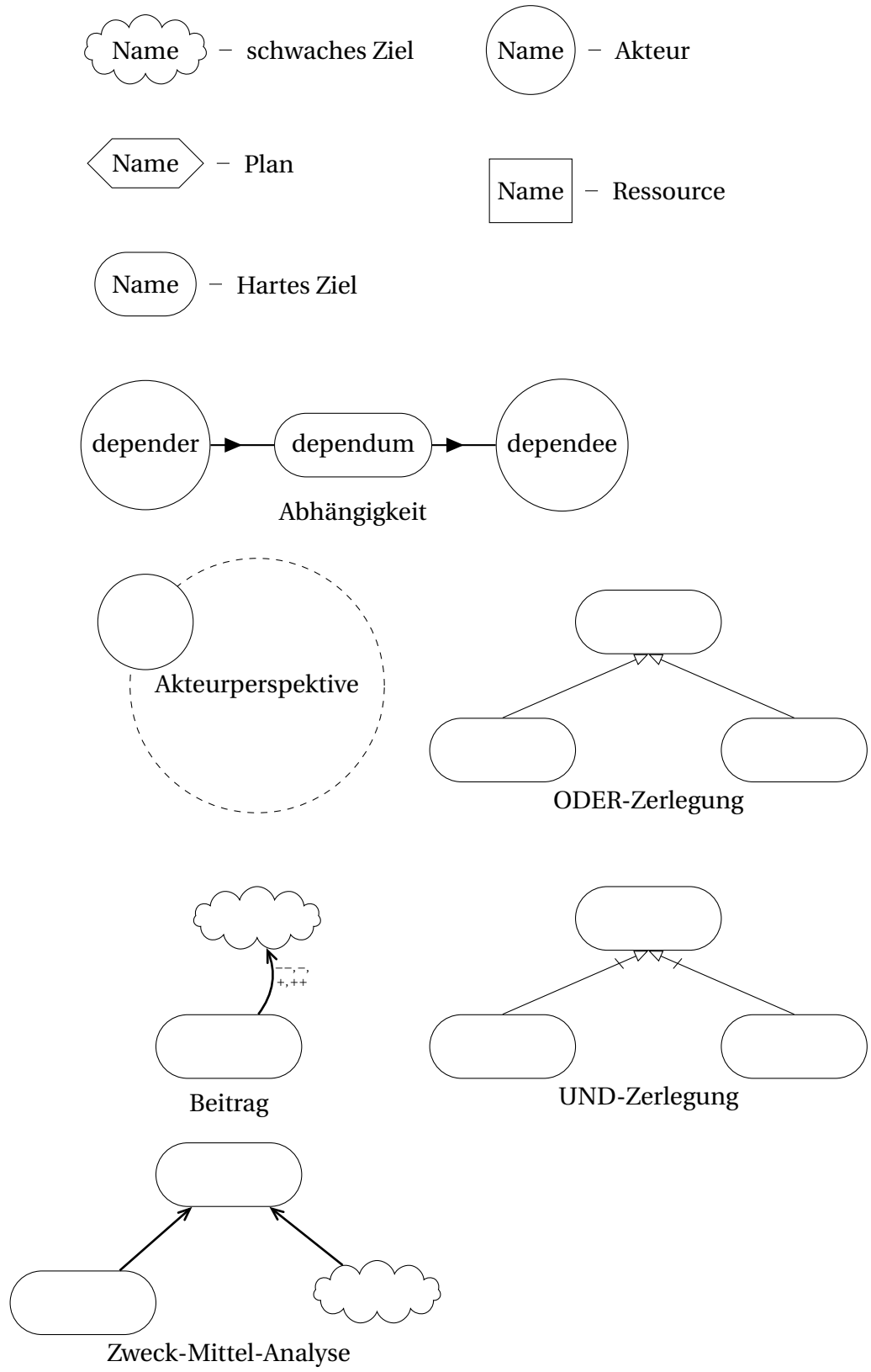


Abbildung 4.2 – *Modellierungssymbole der TROPOS-Methode*

der Diagaramme als Bild möglich. In der Abb. 4.3 ist die graphische Benutzeroberfläche des TAOM4E-Plug-in abgebildet. Später wurde TAOM4E um das Tool t2x [29]

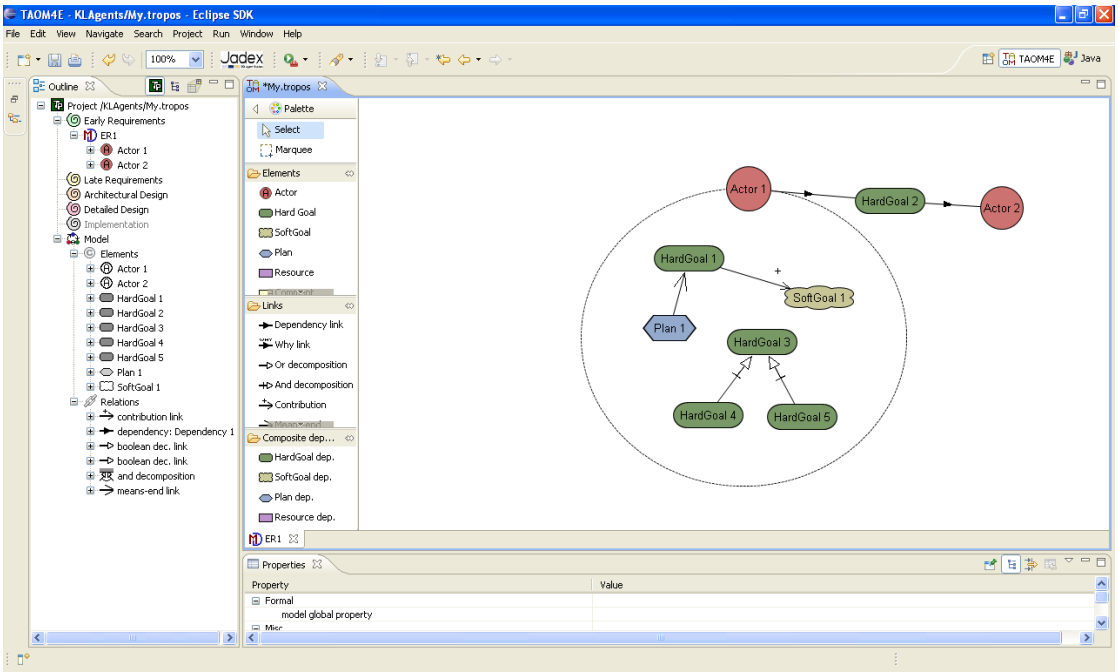


Abbildung 4.3 – *TAOM4E-Benutzeroberfläche in Eclipse*

erweitert. Dieses Tool bietet die Möglichkeit die modellierten Agenten als JADEX-Codegerüst zu exportieren. So wird eine direkte Abbildung vom Modell zum Quellcode möglich und man kann Änderungen des Modells sofort in der Implemtierung sehen.

4.2 JADEX

JADEX ist eine Plattform zum Enwickeln von BDI-konformen Agentensystemen, welche an der Universität Hamburg entwickelt wurde [35], [12], [34], [11]. JADEX wurde so konzepiert, dass es als Erweiterung von bestehenden Agentenplattformen wie JADE dient, die keine Unterstützung für die BDI-Architektur besitzen.
Die generelle Architektur des JADEX-Systems ist in Abb. 4.4 dargestellt. Der Agent ist eine abgeschottete Einheit, die nach Außen nur durch Nachrichten kommuniziert. Das Innere des Agenten besteht aus der Raktions- und Deliberationseinheit (Practical reasoning interpreter) und den Fähigkeiten (Capabilitiy) eines Agenten. Die Deliberationseinheit bestimmt die Menge von Zielen, die als erreichbar angesehen werden. Die Reaktionseinheit (Means-end reasoning) fängt alle internen und externen

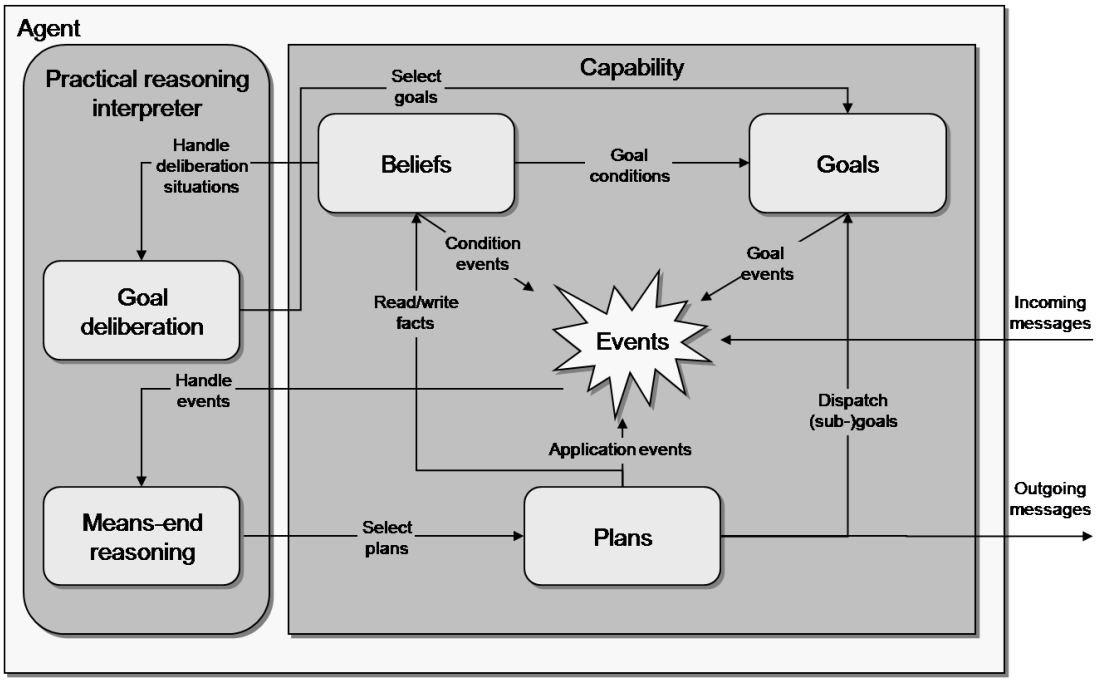


Abbildung 4.4 – *Jadex-Architektur. Aus [35]*

Events ab und wählt Pläne aus, die diese Events behandeln und dabei weitere Events auslösen können, Überzeugungen lesen und verändern, neue Ziele erzeugen und Nachrichten verschicken. Die Reaktions- und Deliberationseinheit ist für alle Agenten identisch. Ziele, Überzeugungen und Pläne sind für jeden Agenten speziell definiert.
Für die Implementierung wird eine Kombination aus XML-basierten Agentendefinition und JAVA-Klassen verwendet.(siehe Abb. 4.5) Dieses Agent Definition File (ADF)

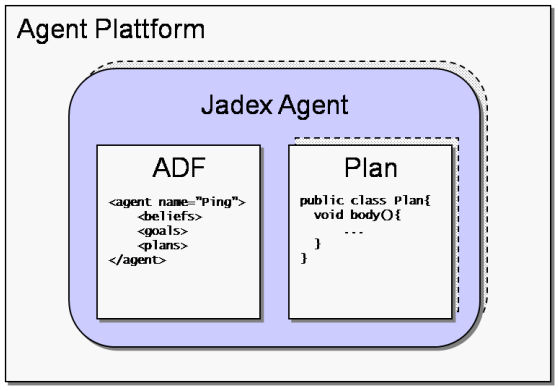


Abbildung 4.5 – *Agentenspezifikation in JADEX. Aus [35]*

(Listing 4.2) enthält Verweise auf JAVA-Klassen, und die Beschreibung der Zieltypen, Überzeugungen und Pläne. Im Folgeneden werden die einezelnen Konzepte der JADEX-Plattform erläutert.

Listing 4.1 – *Agent Definition File(ADF)*

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
       http://jadex.sourceforge.net/jadex-0.96.xsd"

  name="AgentName"
  package="packageName">
  <beliefs> ... </beliefs>
  <goals> ... </goals>
  <plans> ... </plans>
  <messages> ... </messages>
  ...
</agent>
```

Überzeugungen (Beliefs)

Die Überzeugungen bei JADEX werden durch beliebige JAVA-Objekte abgebildet und repräsentieren analog zur BDI-Architektur das Wissen des Agenten über die Umwelt und andere Agenten. Die Objekte werden als Fakten(beliefs) oder Sets von Fakten(belief sets) gespeichert. JADEX stellt die Funktionalität bereit, mit der man aus den verschiedenen Klassen auf die Objekte zugreifen kann. Man kann Veränderungen der Fakten auch im ADF vornehmen, in dem man über die Systemvariable „$beliefbase“ auf die einzelnen Fakten zugreift. Die Fakten werden von der Reaktionseinheit überwacht, die Events auslösen kann, falls sich Fakten ändern.

Listing 4.2 – *Agent Definition File(ADF)*

```
<beliefs>
  <belief name="auto" class="Auto">
    <fact>new Auto()</fact>
  </belief>

  <beliefset name="automarken" class="String">
    <fact>"BMW"</fact>
    <fact>"AUDI"</fact>
    <fact>"VW"</fact>
```

```
  </beliefset>

  <belief name="autofarbe" class="Farbe">
    <fact> $beliefbase.auto.getColor()</fact>
  </belief>
</beliefs>
```

Ziele(Goals)

Durch Ziele können bei JADEX Zustände definiert werden, die angestrebt werden, ohne konkrete Aktionen festzulegen. Die zum Erreichen des Ziels notwendigen Pläne werden währen der Laufzeit für jedes Ziel ausgewählt. Dabei können abhängig von den Überzeugungen verschiedene Planfolgen zum gewünschten Erfolg führen. Die Ziele eines Agenten können aktiv oder inaktiv sein, abhängig davon, ob Voraussetzungen für das Erfüllen des Ziels gegeben sind oder nicht. Es existieren vier Typen von Zielen:

- **Performgoal:** Dieser Zieltyp wird eingesetzt, wenn bestimmte Aktionen ausgeführt werden müssen. Das Ergebnis der Aktionen wird dabei außer Acht gelassen. Ein solches Ziel kann zum Beispiel das Patrouillieren eines Gebietes sein (Listing 4.3).

  Listing 4.3 – *Performgoal*

  ```
  <!-- Dieses Ziel ist solange Aktiv, bis die Nacht vorbei ist.
       -->
  <performgoal name="patroullieren" retry="true" exclude="never
      ">
    <contextcondition>
      $beliefbase.istNacht
    </contextcondition>
  </performgoal>
  ```

- **Achievegoal:** Dieses Ziel wird eingesetzt, um einen bestimmten Zustand der Umwelt zu erreichen. Achievegoals besitzen Attribute wie <targetcondition> und <failurecondition>, die festlegen, wann das Ziel als erfüllt bzw. als fehlgeschlagen angesehen wird.So ein Ziel kann zum Beispiel das Reparieren eines Autos sein (Listing 4.4).

  Listing 4.4 – *Achievegoal*

  ```
  <!-- Dieses Ziel soll versuchen, ein Auto zu reparieren -->
  ```

```
<achievegoal name="autoReparieren">
  <targetcondition>
    $beliefbase.auto.istRepariert()
  </targetcondition>
  <failurecondition>
    $beliefbase.auto.istIrreparabel()
  </failurecondition>
</achievegoal>
```

- **Querygoal:** Dieser Zieltyp ist ähnlich zu dem Achievegoal. Das Ziel wird verwendet, um bestimmte Informationen zu bekommen. Das Ziel ist erfüllt, wenn eine Information bereitsteht. So ein Ziel kann zum Beispiel das Finden einer bestimmten Adresse sein (Listening 4.5).

Listing 4.5 – *Querygoal*

```
<!-- Dieses Ziel soll versuchen, eine Adresse zum gegebenen
    Namen im Adressbuch zu finden. Das Ziel ist erreicht, wenn
     der Ausgabeparameter != null ist. -->
  <querygoal name="adresseFinden">
    <!-- Eingabeparameter -->
    <parameter name="name" class="String"/>
    <!-- Ausgabeparameter. -->
    <parameter name="adresse" class="Adresse" direction="out"
       >
          <value> $beliefbase.adressbuch.adresseFinden($goal.
             name) </value>
    </parameter>
</querygoal>
```

- **Maintaingoal:** Um bestimmte Bedingungen zu überwachen, wird der Typ Maintaingoal eingesetzt. Wenn diese Bedingung nicht mehr erfüllt ist, wird das Ziel aktiviert und es werden Pläne ausgeführt, um die Bedingung wieder zu erfüllen. Ein Maintangoal kann zum Beispiel zum Überwachen von Temperatur eingesetzt werden (Listening 4.6).

Listing 4.6 – *Maintaingoal*

```
<!-- Dieses Ziel Überacht die Temperatur und wird Aktiviert
    wenn diese einen Wert über 50 erreicht -->
<maintaingoal name="temperaturZuHoch">
  <maintaincondition>
```

```
        $beliefbase.temperatur &gt; 50 <!-- &gt; <=> '>'-->
      </maintaincondition>
    </maintaingoal>
```

Pläne (Plans)

Pläne repräsentieren die Möglichkeiten eines Agenten zum Erreichen von Zielen und zum Reagieren auf Ereignisse. Jeder Plan besteht aus dem Plankopf(Head) und dem Plankörper(Body). Der Plankopf wird in dem ADF definiert und enthält den Verweis auf den Plankörper, die Ziele und Events, für die der Plan eingesetzt werden kann, sowie die Bedingungen, die zum Ausführen des Plans erfüllt sein müssen. Es können auch Variablenbindungen spezifiziert werden. Der Plankörper hingegen ist eine Ableitung der von JADEX bereitgestellten JAVA-Klasse „Plan“, die die eigentliche Funktionalität des Planes definiert. Dort werden in der Funktion „body()“ alle Aktionen aufgelistet, die zur Ausführung des Plans benötigt werden. Es können auch weitere Funktionen implementiert werden, die Aktionen definieren, die beim Fehlschlagen, Abbruch oder erfolgreicher Ausführung ausgeführt werden. Im Listing 4.7 ist die Struktur eines Plankopfes und im Listing 4.8 ist die Implementierung des Plankörpers dargestellt.

Listing 4.7 – *Plankopf im ADF*

```
<!-- Planname -->
<plan name="Name">
<!-- Verweis auf den Plankörper. -->
  <body>new Plankörper()<body/>
  <!-- Verweis auf das Ziel -->
  <trigger>
    <goal ref="Zielname"/>
  </trigger>
  <!-- Bedingung, die währen der Ausführung erfüllt werden muss.
     Wenn die Bedingung nicht erfüllt ist, wird der Plan
     abgebrochen. -->
  <contextcondition> Bedingung </contextcondition>
</plan>
```

Listing 4.8 – *Plankörper JAVA-Klasse*

```
/** Plankörper */
public class Plankörper extends jadex.runtime.Plan {

        public void body() {
```

```
        // Aktionen, die für die Ausführung des Plans nötig sind.
        }

        public void passed() {
        // Optionale Aktionen bei erfolgreicher Planausführung
        }
        public void failed() {
        // Optionale Aktionen beim Fehlschlagen des Planes
        }
        public void aborted() {
        // Optionale Aktionen beim Abbruch des Plans
        }
}
```

Fähigkeiten(Capabilities)

Für komplexe Agentensysteme stellt JADEX „Capabilities“(Fähigkeiten) bereit, mit deren Hilfe die Modularität und die Wiederverwendbarkeit von Quellcode verbessert werden soll. Die „Capabilities“ repräsentieren ein Modul, in dem bestimmte Funktionalitäten, Überzeugungen, Ziele und Pläne enthalten sind. Für die Definition der „Capabilities“ wird eine XML-Datei, die ähnlich dem ADF ist, verwendet. In dem ADF eines Agenten können die „Capabilities“ dann eingebunden werden.

Listing 4.9 – *Capability*

```
<capability xmlns="http://jadex.sourceforge.net/jadex"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://jadex.sourceforge.net/jadex
  http://jadex.sourceforge.net/jadex-0.96.xsd"

  name="MyCapability"
  package="mypackage">
  <beliefs> ... </beliefs>
  <goals> ... </goals>
  <plans> ... </plans>
  ...
</capability>
```

Im JADEX-Userguide [33] ist eine detailierte Beschreibung der JADEX-Plattform gegeben. Um die Agenten zu starten und zu überwachen stellt JADEX eine Reihe von Tools bereit. Das zentrale Tool ist das „Control Center“, welches die Möglichkeit bie-

tet die vorhandenen JADEX-Plug-ins zu starten. Zum Laden, Starten und Löschen von Agenten wird der „Starter“ verwendet. In der Abb. 4.6 ist das „Control Center“ mit ausgeführtem „Starter“ dargestellt. Ein weiteres Plug-in ist der „Tracer“, mit des-

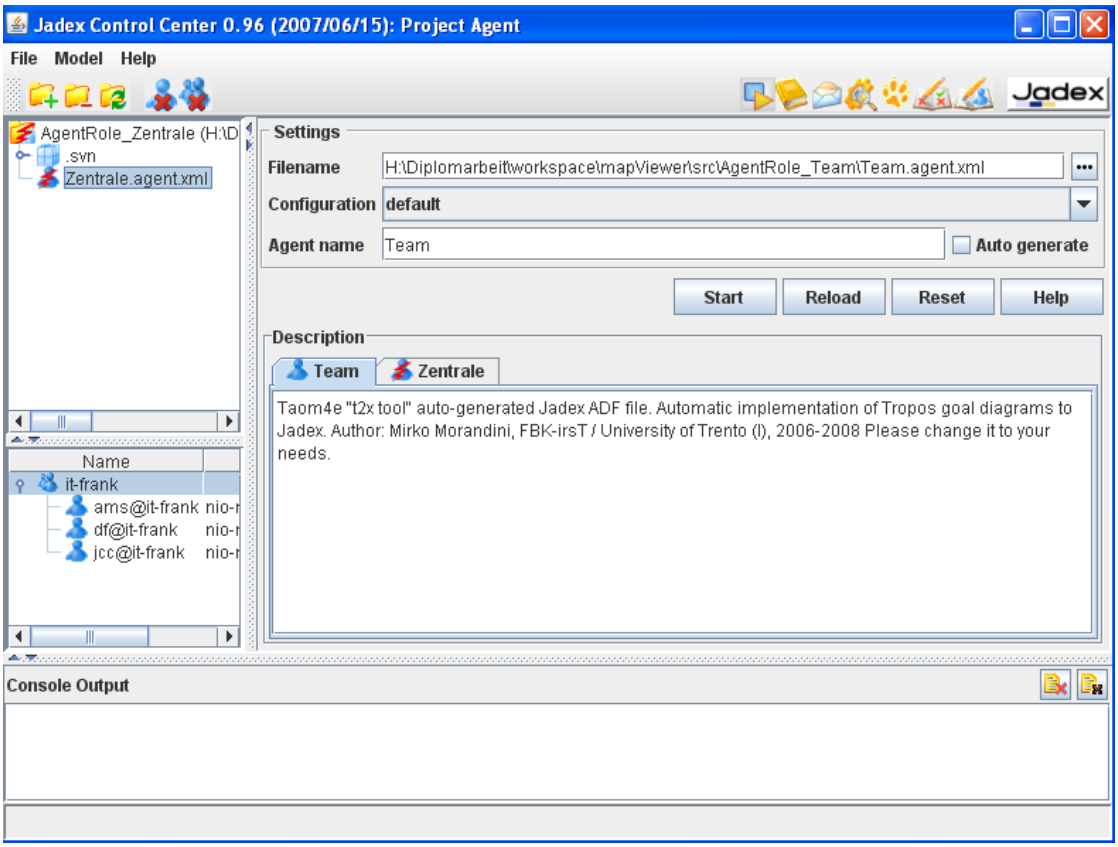


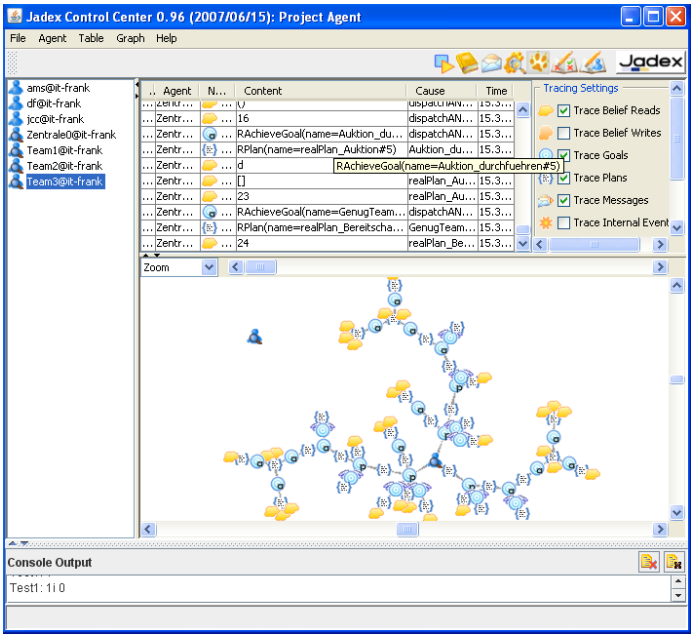
Abbildung 4.6 – *Control Center mit ausgeführtem Starter*

sen Hilfe es möglich ist, das dynamische Verhalten der Agenten zu überwachen. Es werden je nach Auswahl alle Abläufe des Agenten in einer Tabelle oder einem Graphen dargestellt. Zu den Abläufen gehören zum Beispiel das Lesen/Schreiben einer Überzeugung, das Ausführen eines Plans oder Versenden/Empfangen einer Nachricht. Dieses Tool kann gut dazu verwendet werden, die Handlungen eines Agenten zu verstehen und Fehler zu finden. Um den inneren Zustand eines Agenten zu überwachen wird das „Introspector“ Tool genutzt. Es bietet die Möglichkeit die Pläne, Ziele und Überzeugungen eines Agenten zu überwachen und während der Laufzeit zu ändern. Ausserdem wird von dem Tool ein Debugger bereitgestellt, der eine schrittweise Ausführung der Agenten erlaubt. Die beiden Plug-ins sind in der Abb. 4.7 dargestellt.

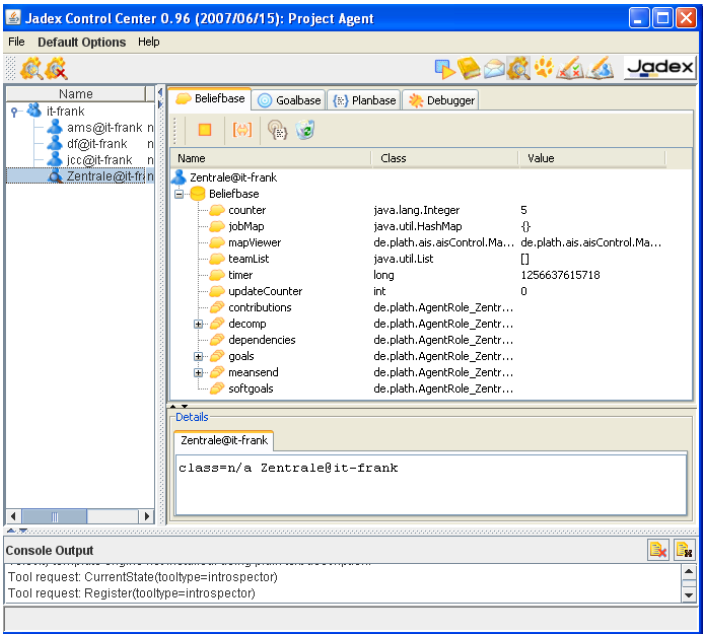
Weitere Plug-ins sind der „DF(Directory Facilitator) Browser“ und das „Conversation Center“. Der „DF Browser“ Visualisiert die aktuellen Regestrirungen von Ser-

vices bei dem DF-Agenten. Der DF-Agent ist ein JADEX-Agent, der beim Starten der Plattform ausgeführt wird, und für andere Agenten die Möglichkeit bietet, ihre Services bei ihm anzumelden. Die Agenten haben somit die Möglichkeit die Services anderer Agenten zu finden und zu nutzen. Das „Convesation Center“ bietet dem Nutzer die Möglichkeit, Nachrichten zu erstellen, diese an einen aktiven Agenten zu senden und die Antwort des Agenten zu lesen.

Ene genauere Beschreibung der Tools und deren Funktionalität ist in dem JADEX-Tool Guide [32] zu finden.

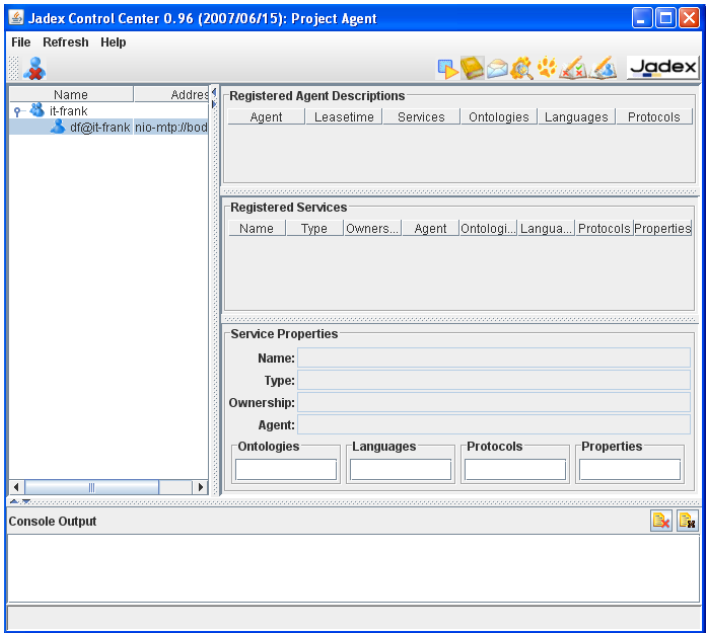


(a) Tracer

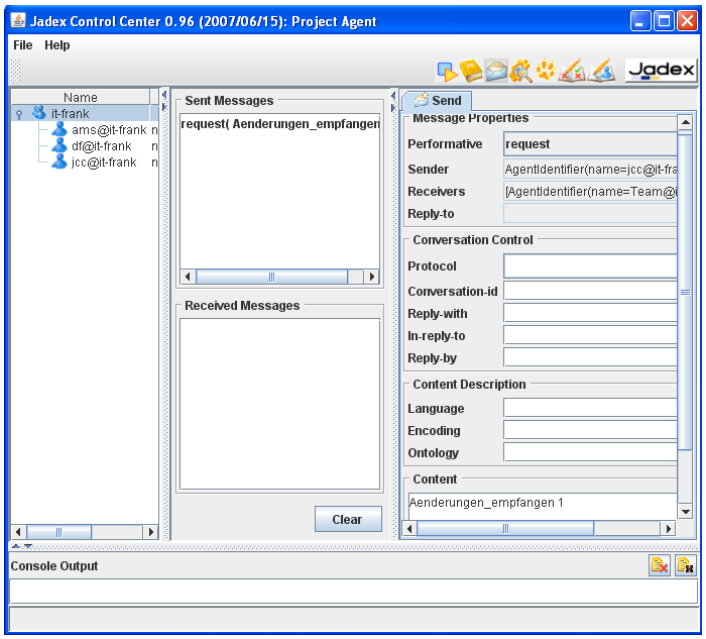


(b) Introspector

Abbildung 4.7 – *Tracer(a) und Introspector(b)*



(a) DF Browser



(b) Conversation Center

Abbildung 4.8 – *DF Browser und Conversation Center*

Kapitel 5

Aufbau der Agenten

In diesem Kapitel wird die Modellierung und Realisierung der Agenten für die Festmachereinsatzplanung erläutert. Es wird zuerst die Umgebung der Agenten beschrieben. Danach werden die Agenten erläutert. Zu jedem Agent wird zuerst die reale Situation beschrieben, die den momentanen Ablauf der Einsatzplanung wiedergibt. Die reale Welt wird dann vereinfacht mithilfe der TROPOS-Methode modelliert. Es werden zuerst die frühen Anforderungen (early requirements) modelliert. Als nächstes werden die späten Anforderungen (late requirements) erstellt. Danach wird das Architekturdesign (architectural design) des AIS-Überwachungssystems beschrieben. Es folgen dann das detaillierte Design (detailed design) und die Implementierung.

5.1 Reale Situation

5.1.1 Umgebung

Der Festmacherbetrieb ist im Hamburger Hafen angesiedelt. Der Aufgabenbereich der Festmacher beschränkt sich auf die über 30 Hafenbecken des Hafens. Durch langjährige Erfahrung kennen sich die Fesmacher bestens in dem Hafen aus. Sie kennen alle Wasserwege und Strassenrouten, sodass sie selbst entscheiden können, welchen Weg sie nehmen. Jeder Fahrer hat dabei seine eigenen Vorlieben für Routen. Die Routenauswahl ist ein wichtiger und komplizierter Punkt bei der Realisierung des Agentenansatztes. Für die automatische Routenerstellung müssen alle vorhandenen Wege innerhalb des Hafens modelliert werden, was zum Beispiel mit herkömmlichen Routenplanern nicht immer möglich ist, weil nicht alle Wege auf den Karten verzeichnet sind. Die Wasserwege müssen auch modelliert werden. Dazu ist eine vollständige Wasserkarte des Hafens erforderlich und die Hafenverkehrsordnung ist auch zu beachten. Die Wasserstände der Elbe können auch eine Rolle bei der Wahl der Route spielen und müssten daher auch beachtet werden. In diesem Umfeld agieren die

Zentrale und die Einsatzteams. Die Einsatzteams bekommen von der Zentrale Aufträge, die sie dann ausführen müssen. Die Kommunikation zwischen Zentrale und Teams erfolgt über Funk. Die Zentrale hat keine Möglichkeit die Position der Teamfahrzeuge zu beobachten.

5.1.2 Zentrale

In der Zentrale werden die Aufträge verwaltet und verteilt. Die Zentrale ist immer mit mindestens einem Koordinator besetzt, der die Vergabe und Ausführung überwacht. Er erstellt Aufträge und bestimmt die Teams, die den Auftrag ausführen sollen. Die Ausführung der Aufträge wird von den Teams an die Zentrale gemeldet. Der Koordinator hat mehrere Informationsquellen, die ihm bei der Erstellung der Aufträge zur Verfügung stehen. Zum einen sind es die Daten der HPA, die die erwarteten Ankünfte beinhalten. Als weitere Quelle dient das Faxgerät. Darüber bekommt die Zentrale Informationen über neue Aufträge oder Änderungen der alten Aufträge. Es kommen auch Aufträge über das Telefon an. Diese sind meist kurzfristige Aufträge für Schiffe, die den Liegeplatz innerhalb des Hafen ändern wollen oder den Hafen verlassen wollen. Als weitere Unterstützung haben die Koordinatoren ein AIS-Überwachungssystem, welches ihnen die aktuelle Position des Schiffes anzeigt und somit die Möglichkeit gibt, die Ankunftszeit abzuschätzen. Außerdem stellt das AIS-Überwachungssystem weitere Informationen über das Schiff bereit, die für den Koordinator nützlich sein können. Die Information darüber, wie viele Boote und Festmacher am Ufer eingesetzt werden sollen, bezieht der Koordinator entweder aus den Aufzeichnungen über frühere Aufträge an dem Liegeplatz, oder er richtet sich nach seinen Erfahrungswerten. Wenn der Koordinator feststellt, dass kein freies Team zur Ausführung eines Auftrages bereitsteht, hat er die Möglichkeit Teams aus der Bereitschaft zu holen oder einen Konkurrenten zu beauftragen. Es wird angenommen, dass es immer möglich ist, Teams von der Konkurrenz zu bekommen. Insgesamt kann man den Arbeitsablauf eines Koordinators in folgende Schritte unterteilen:

- Informationsquellen beobachten.
- Aus vorhandenen Informationen Aufträge erstellen.
- Für den Auftrag passende Teams auswählen.
- Wenn keine freien Teams vorhanden, Bereitschaft anrufen. Wenn Bereitschaftsruf nicht erfolgreich war, Konkurrenz beauftragen.
- Meldung der Teams entgegennehmen.
- Teams über ihre nächsten Aufträge informieren.

5.1.3 Team

Das Team besteht aus einem oder mehreren Festmachern. Jedes Team verfügt über ein Auto oder ein Boot, um sich im Hafen zu bewegen. Jedes Fahrzeug ist mit einem Funkgerät ausgestattet. Ein Team bekommt von der Zentrale Informationen zum Auftrag, der von dem Team als nächstes ausgeführt werden soll. Wenn das Team den Auftrag angenommen hat, wird dieser anschließend ausgeführt. Die Ausführung wird über Funk an die Zentrale gemeldet. So weiß die Zentrale, wann ein Auftrag begonnen hat und wann ein Team wieder zur Verfügung steht. Ein Team kann auch mit anderen Teams über Funk Kontakt aufnehmen. Während der Schichten können die Teams Pausen machen oder das Fahrzeug betanken. Es kann auch vorkommen, dass ein Fahrzeug ausfällt und das Team für einen bestimmten Zeitraum nicht mehr einsetzbar ist.

5.2 Frühe Anforderungen(Early Requirements)

Für die Modellierung des Systems werden folgende Annahmen getroffen:

- Zuerst werden die Daten der HPA als einzige und zuverlässige Quelle für Aufträge angesehen. Später können dann auch weitere Quellen miteinbezogen werden.
- Die Zentrale kennt nur Teams, die sich vorher angemeldet haben.
- Es werden nur bestimmte Liegaplatzhäfen betrachtet.
- Es werden alle extrahierten Aufträge als eigene Aufträge angenommen.
- Alle Agenten des Systems verfolgen ein gemeinsames Ziel, möglichst alle Aufträge zu erfüllen.
- Es können Aufträge unter den Teams eines Unternehmens getauscht werden.

5.2.1 Umgebungsmodell

Das Umegbungsmodell definiert die Rahmenbedingungen für die in der Umgebung agierenden Agenten. In diesem Abschnitt werden die Vereinfachungen der Umgebung dargestellt. Die Umgebung wird durch 15 ausgewählte Liegeplatzhäfen repräsentiert (Abb. 5.1). Diese Liegeplatzhäfen sind die wichtigsten im Hamburger Hafen und werden oft angesteuert. Im Modell wird der Hafen durch zwei Graphen dargestellt. Zum einen ist es der Graph mit den Wasserverbindungen G_{wasser} und zum anderen der Graph mit den Straßenverbindungen G_{str}. G_{wasser} ist ein gewichteter un-

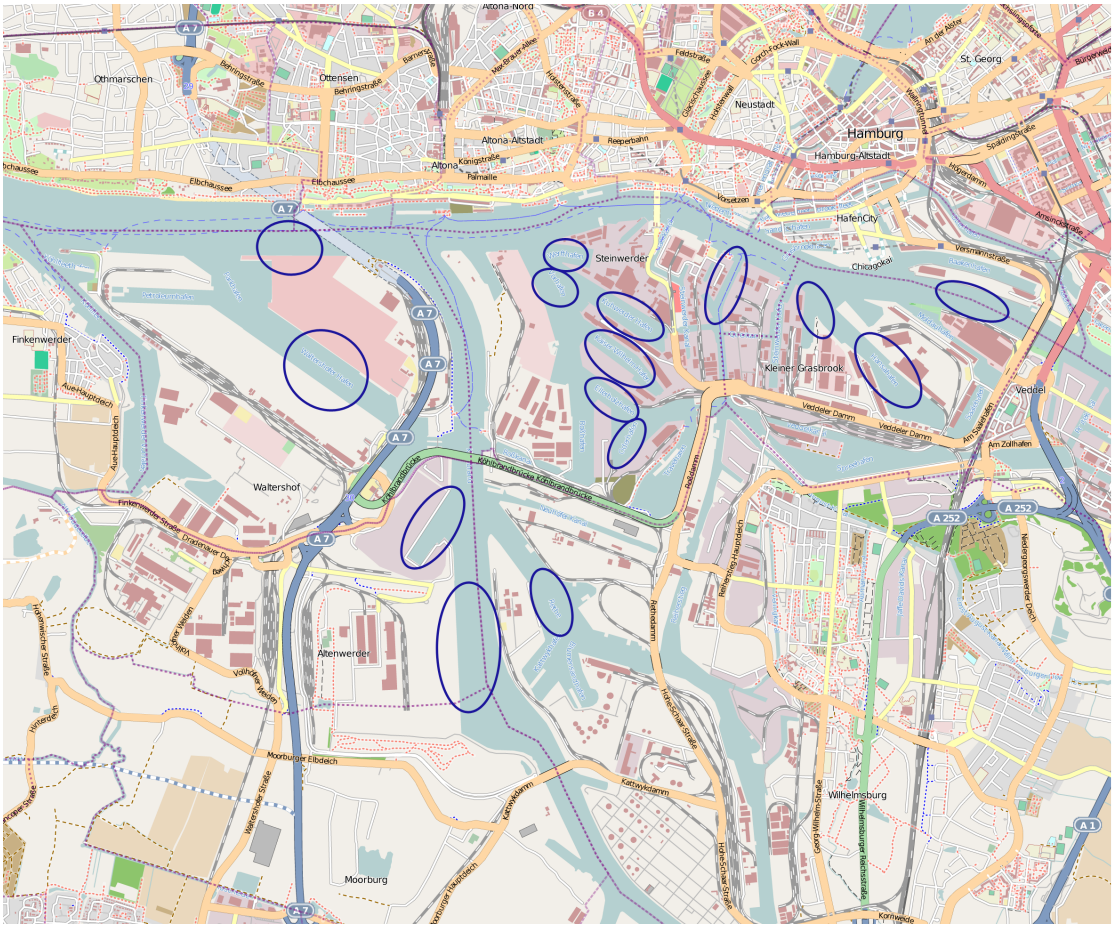


Abbildung 5.1 – *Ausgewählte Liegeplatzhäfen. Karte von OpenstreetMap.*

gerichteter Graph. Die Knoten sind die ausgewählten Häfen. Die Kanten repräsentieren die möglichen Verbindungen und die Gewichte die Entfernung zwischen den Knoten. Analog ist G_{str} aufgebaut. Die Gewichte in den beiden Graphen sind ungefähre Werte, die mithilfe einer Karte und eines Routenplaners ermittelt wurden. Die Kantenstruktur der Graphen ist auch vereinfacht worden, um die Komplexität zu verringern. Die beiden Graphen sind in den Abb. 5.2 und 5.3 dargestellt.

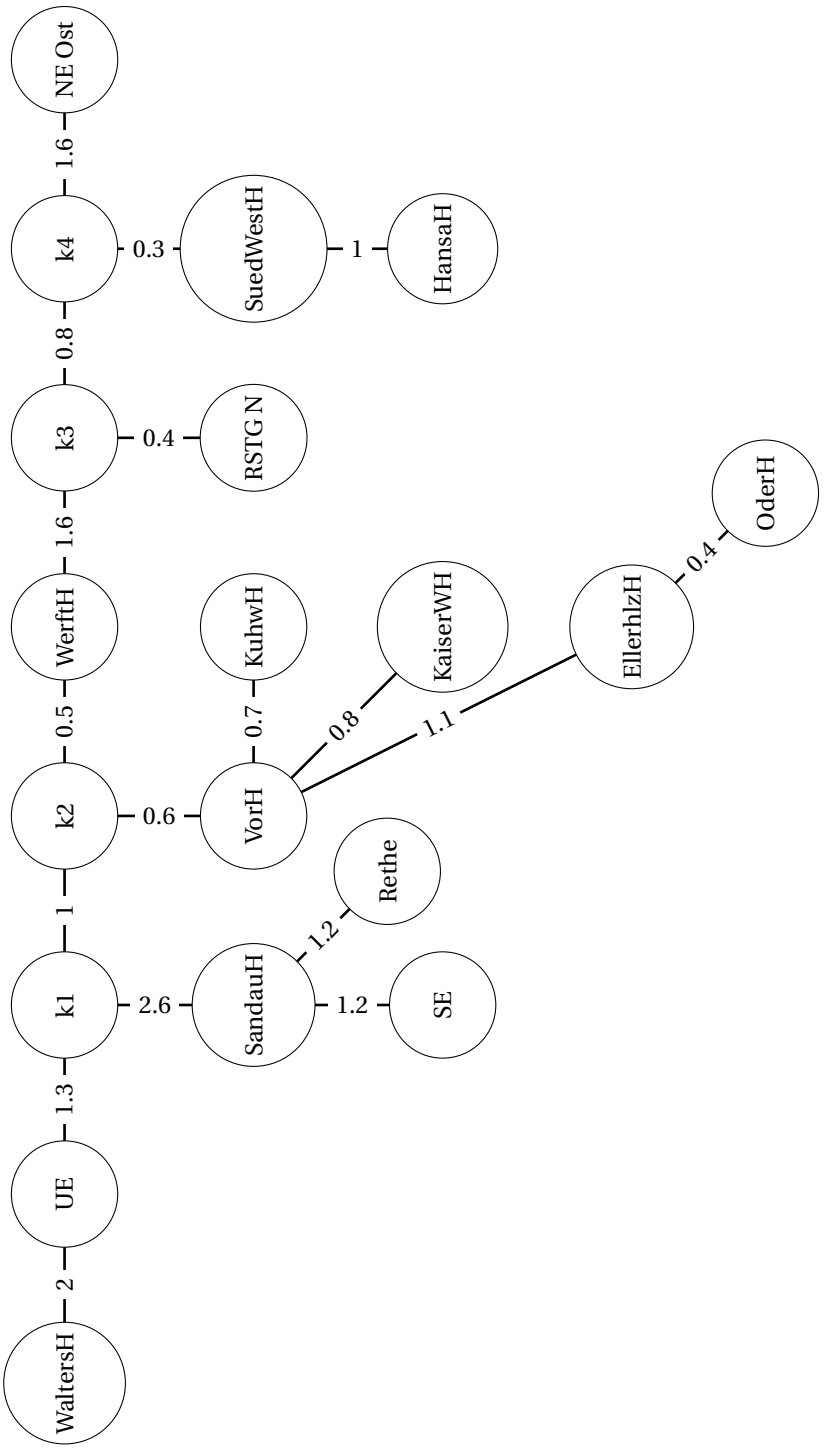


Abbildung 5.2 – *Graph G_{wasser} der ausgewählten Teilhäfen. Die Gewichte repräsentieren Entfernungen auf dem Wasser zwischen den Teilhäfen in km. k1 ... k4 sind Kreuzungsknoten*

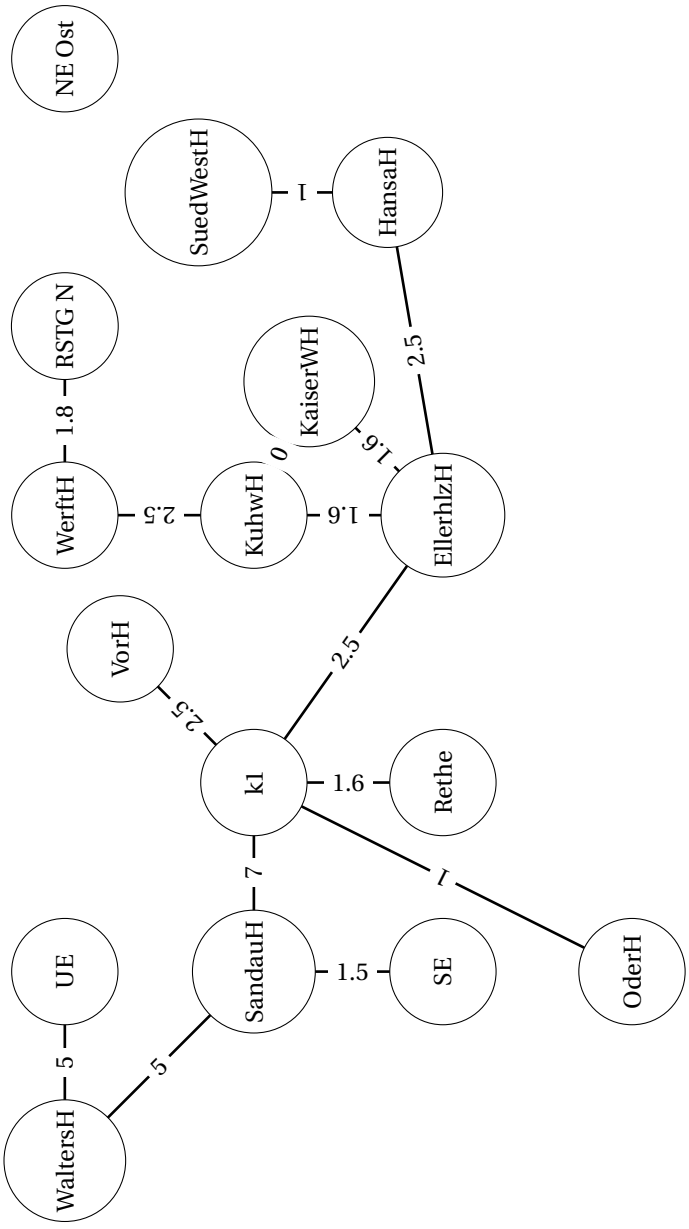


Abbildung 5.3 – *Graph G_{str} der ausgewählten Teilhäfen. Die Gewichte repräsentieren Entfernungen auf dem Land zwischen den Teilhäfen in km. k1 ist ein Kreuzungsknoten. NE Ost ist mit dem Auto nicht erreichbar.*

5.2.2 Actor Diagram

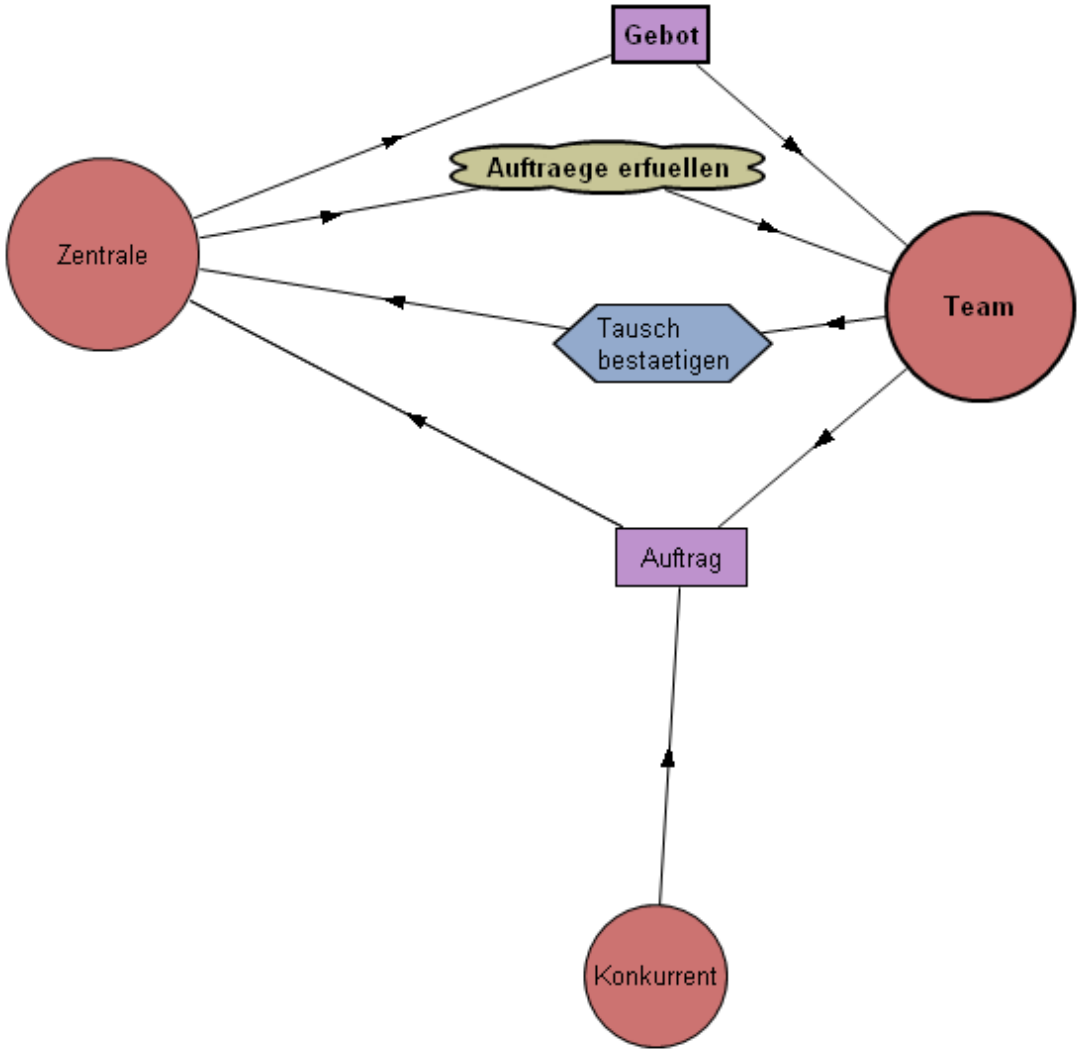


Abbildung 5.4 – *TAOM4E Actor diagram*

Im ersten Schritt der Analyse wird das „Actor diagram“ erstellt. Dazu werden alle Rollen, die im Umfeld des zu entwickelnden System agieren, gegenübergestellt. Im Umfeld des zu modellierenden Festmacherbetriebes wurden die Rollen Zentrale, Team und Konkurrent identifiziert. Diese Rollen und deren Beziehung untereinander sind in der Abb. 5.4 dargestellt. Die beiden wichtigsten Rollen sind Zentrale und Team. Die Zentrale möchte Gebote von den Teams empfangen und delegiert das schwache Ziel „Auftraege erfuellen“ an die Teams, was durch die Abhängigkeitsbezie-

hung dargestellt ist. Die Teams ihrerseits delegieren den Plan „Tausch bestaetigen“ an die Zentrale. Durch die Ressourcenabhängigkeit mit der Ressource „Auftrag“ wird veranschaulicht, dass Teams und Konkurrenten Aufträge von der Zentrale erhalten wollen.

5.2.3 Zielanalyse des Zentraleagenten

Um aus der realen Beschreibung der Aufgaben der Zentrale einen Softwareagenten zu entwickeln, wird ein Modell der Zentrale erstellt. Dabei wird das TROPOS Verfahren zum modellieren eingesetzt. In den frühen Anforderungen werden die Ziele und Pläne des Zentrale-Agenten definiert. In Abb. 5.5 sieht man den Agenten mit seinen Zielen und Plänen. Das erste Ziel ist „Tausch verwalten“. Dieses Ziel soll erreicht wer-

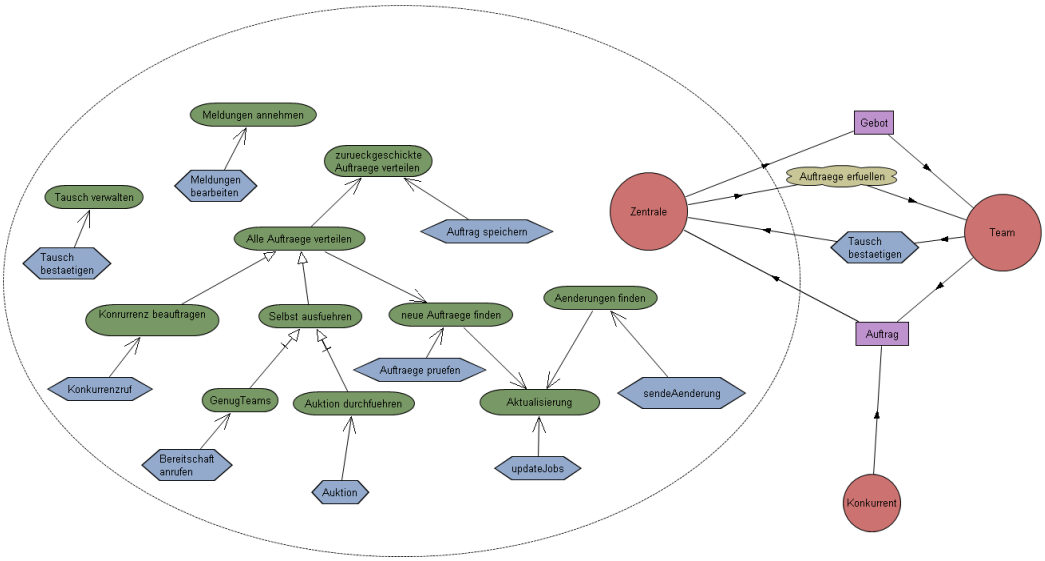


Abbildung 5.5 – *TAOM4E Zieldiagramm des Agenten „Zentrale“*

den, wenn eine Nachricht mit einer Tauschanfrage eintrifft. Um das Ziel zu erreichen muss der Tausch überprüft werden. Nach der Überprüfung durch die Software oder den Benutzer, wird eine Antwort an das anfragende Team geschickt. Diese Antwort kann entweder eine Bestätigung oder eine Ablehnung sein.
Ein weiteres Ziel ist „Meldungen annehmen“. Dieses Ziel wird durch eine Nachricht mit einer Meldung aktiviert. Mithilfe dieser Meldungen behält die Zentrale Überblick über die sich im Einsatz befindlichen Teams und die Ausführung der Aufträge.
Die Zentrale bietet auch die Möglichkeit, Aufträge von den Teams zurückzunehmen. Die entsprechende Nachricht aktiviert das Ziel „zurueckgeschickte Auftraege verteilen“. Nach der Aktivierung des Ziels wird der Plan „Auftrag speichern“ ausgeführt, der den zurückgegebenen Auftrag als nicht vergeben speichert. Danach wird das Ziel „Al-

le Auftraege verteilen“ aktiviert. Das Ziel wird erreicht, wenn eines oder-verknüpften Unterziele „Konkurrenz beauftragen“ oder „Selbst ausfuehren“ erreicht wird. Deswegen ist bei der Implementierung darauf zu achten, dass das Ziel „Selbst ausfuehren“ als erstes aktiviert wird. Nur in dem Fall, dass „Selbst ausfuehren“ nicht erreicht werden kann, soll „Konkurrenz beauftragen“ aktiviert werden.
Um das Ziel „Selbst ausfuehren“ zu erreichen, müssen die beiden und-verknüpften Unterziele „GenugTeams“ und „Auktion durchfuehren“ erreicht werden. Zuerst soll die Auktion durchgeführt werden. Wenn bei der Auktion festgestellt wird, dass es nicht genug Teams für diesen Auftrag gibt, werden die unbesetzten Plätze mit „Bereitschaft“ markiert. Danach wird versucht das Ziel „GenugTeams“ zu erreichen. Dabei wird überprüft, ob es Aufträge gibt, die mit „Bereitschaft“ markierte Plätze haben. Wenn das der Fall ist, wird versucht, Teams aus der Bereitschaft zu aktivieren. Wenn der Bereitschaftsruf nicht möglich ist, kann das Ziel „GenugTeams“ nicht erfüllt werden und somit bleibt auch das Ziel „Selbst ausfuehren“ nicht erreicht. Es wird also versucht, das Ziel „Konkurrenz beauftragen“ aktiviert und es wird versucht, den Auftrag mit Teams von der Konkurrenz zu vervollständigen.
Ein weiteres wichtiges Ziel ist „Aktualisierung“. Dieses Ziel soll dafür sorgen, dass die Auftragsliste immer aktuell bleibt. Bei Aktivierung dieses Ziels wird der Plan „updateJobs“ ausgeführt. Danach sollen die beiden Unterziele „Aenderungen finden“ und „neue Auftraege finden“ aktiviert werden. Das Ziel „neue Auftraege finden“ wiederum aktiviert das Ziel „Alle Auftraege verteilen“ in dem Fall, dass neue Aufträge gefunden wurden.

5.2.4 Zielanalyse des Teamagenten

Das mit TAOM4E erstellte TROPOS Modell des Teamagenten ist in Abb. 5.6 dargestellt. Der Teamagent ist einfacher aufgebaut als der Zentrale-Agent. Von der Zentrale empfängt der Teamagent Auftragsangebote. Eine Nachricht mit dem Angebot aktiviert das Ziel „Angebote empfangen“. Um das Ziel zu erreichen, muss der Agent an der Auktion teilnehmen. Der Plan „An Auktion teilnehmen“ realisiert die Verhandlungen mit der Zentrale.
Die Teams haben auch die Möglichkeit Aufträge untereinander zu tauschen, was in der realen Situation nicht gemacht wird. Diese Möglichkeit soll den Teams mehr Selbständigkeit geben. Die Zentrale soll dadurch entlastet werden. Die Tauschanfragen aktivieren das Ziel „Tauschanfragen empfangen“, was dazu führt, dass der Plan „Tauschanfrage bearbeiten“ ausgeführt wird. Dieser Plan führt Verhandlungen mit dem anfragenden Team durch.
Von der Zentrale empfängt das Team Nachrichten mit geänderten Aufträgen. Diese Nachrichten aktivieren das Ziel „Aenderungen empfangen“. Es werden folglich die Ziele „Auftraege konsistent halten“ und „Inkonsistente Auftraege abgeben“ aktiviert. Das Ziel „Inkonsistente Auftraege abgeben“ aktiviert dann die Ausführung des Plans

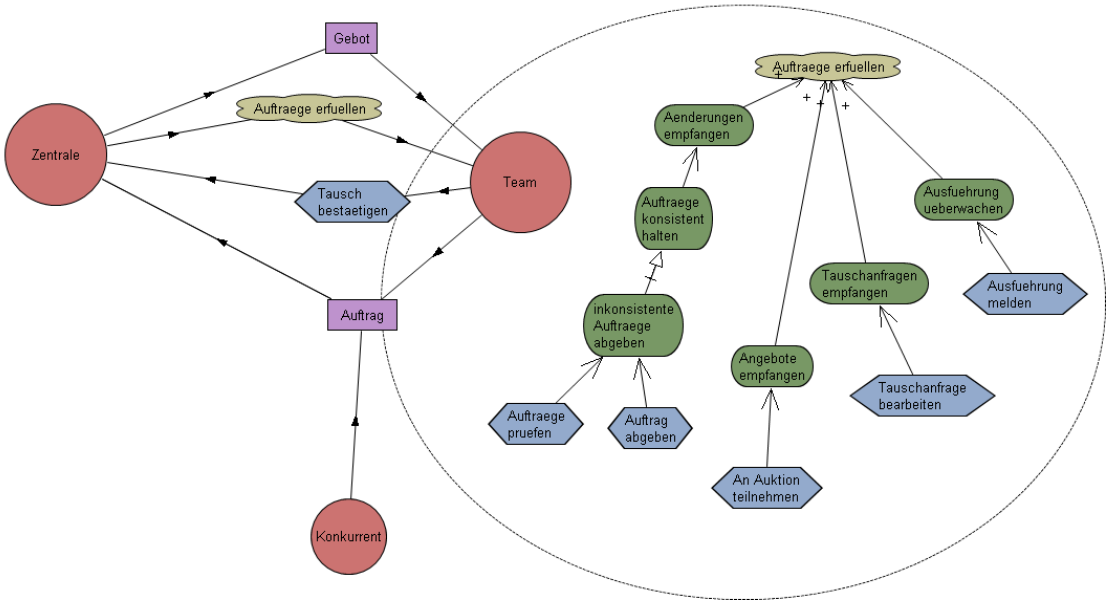


Abbildung 5.6 – *TAOM4E Zieldiagramm des Agenten „Team“*

„Auftraege pruefen“. Wenn dieser Plan erfolgreich ausgeführt wird ist das Ziel erreicht. Anderenfalls wird der Plan „Auftraege abegeben“ aktiviert und versucht die inkonsistenten Aufträge abzugeben. Wenn kein bekannter Agent den Auftrag übernehmen will, wird dieser an die Zentrale abgegeben.
Die Ausführung der Aufträge soll an die Zentrale gemeldet werden. Dazu existiert das Ziel „Ausfuerung ueberwachen“, welches mithilfe des Plans „Ausfuehrung melden“ erreicht werden kann.
Die Ziele „Ausfuerung ueberwachen“, „Aenderungen empfangen“, „Tauschanfragen empfangen“und „Angebote empfangen“ tragen alle positiv zum schwachen Ziel(soft goal) „Auftraege erfuellen“ bei.

5.3 Späte Anforderungen (Late Requirements)

In den späten Anforderungen wird das Modell um das zu entwickelnde System erweitert. Das AIS-Überwachungssystem wird hier in das Modell eingebunden. Das System liefert die Daten für die Zentrale und visualisiert diese in einer graphischen Benutzeroberfläche. Über die Benutzeroberfläche können die Benutzer auch die für den Ablauf nötigen Eingaben tätigen. Das Diagramm in Abb. 5.7 ist eine Erweiterung des Diagramms aus den frühen Anforderungen(Early Requirements). Das AIS-Überwachungssystem wird nur von der Zentrale als Informationsquelle eingesetzt. Die Zentrale stellt Anforderungen in Form von Zielabhängigkeiten an das System.

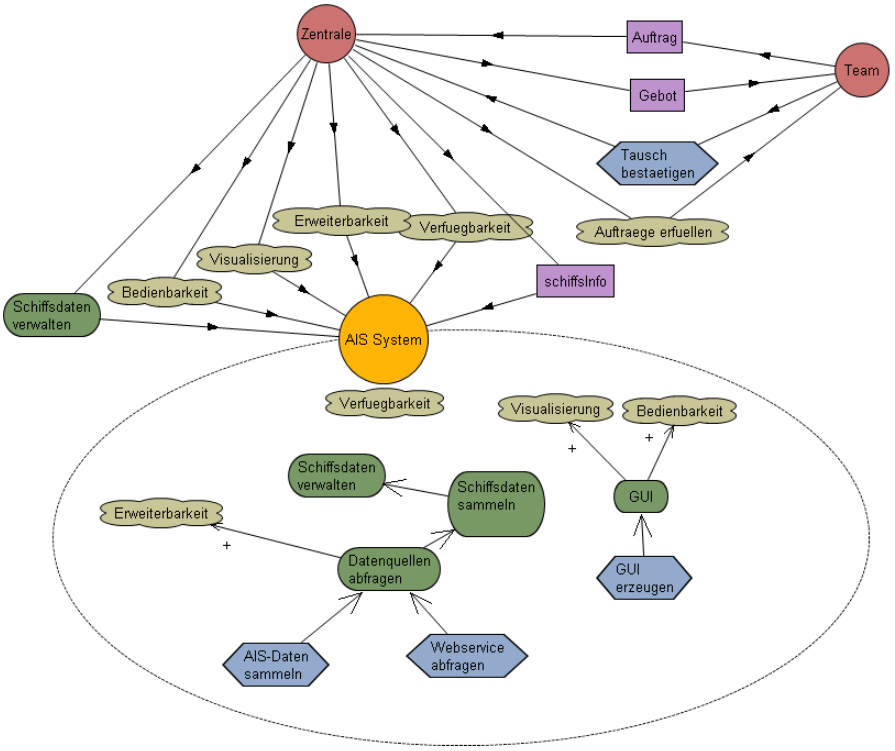


Abbildung 5.7 – *TAOM4E Späte Anforderungen*

Die schwachen Ziele „Visualisierung“, „Bedienbarkeit“, „Erweiterbarkeit“ und „Verfügbarkeit“ repräsentieren die nicht funktionalen Anforderungen der Zentrale an das System. Das harte Ziel „Schiffsdaten verwalten“ fordert von dem System, dass die Schiffsdaten aufbereitet, aktualisiert und der Zentrale zur Verfügung gestellt werden. Um die Ziele „Visualisierung“ und „Bedienbarkeit“ zu erfüllen, wird eine graphische Benutzeroberfläche(GUI) eingesetzt. Dazu wurde das Ziel „GUI“ definiert, welches mit dem Plan „GUI erzeugen“ erreicht wird.
Das Ziel „Schiffsdaten verwalten“ aktiviert das Ziel „Schiffsdaten sammeln“, welches das Zusammenführen und Bereitstellen der verfügbaren Daten repräsentiert. Um dieses zu erreichen, müssen die verschiedenen Datenquellen abgefragt werden. Dazu wird das Ziel „Datenquellen abfragen“ aktiviert. Mithilfe der Pläne „AIS-Daten sammeln“ und „Webservice abfragen“ werden die verfügbaren Daten gesammelt. Dadurch, dass verschiedene Datenquellen in das System integriert werden können, wird das schwache Ziel „Erweiterbarkeit“ positiv beeinflusst.

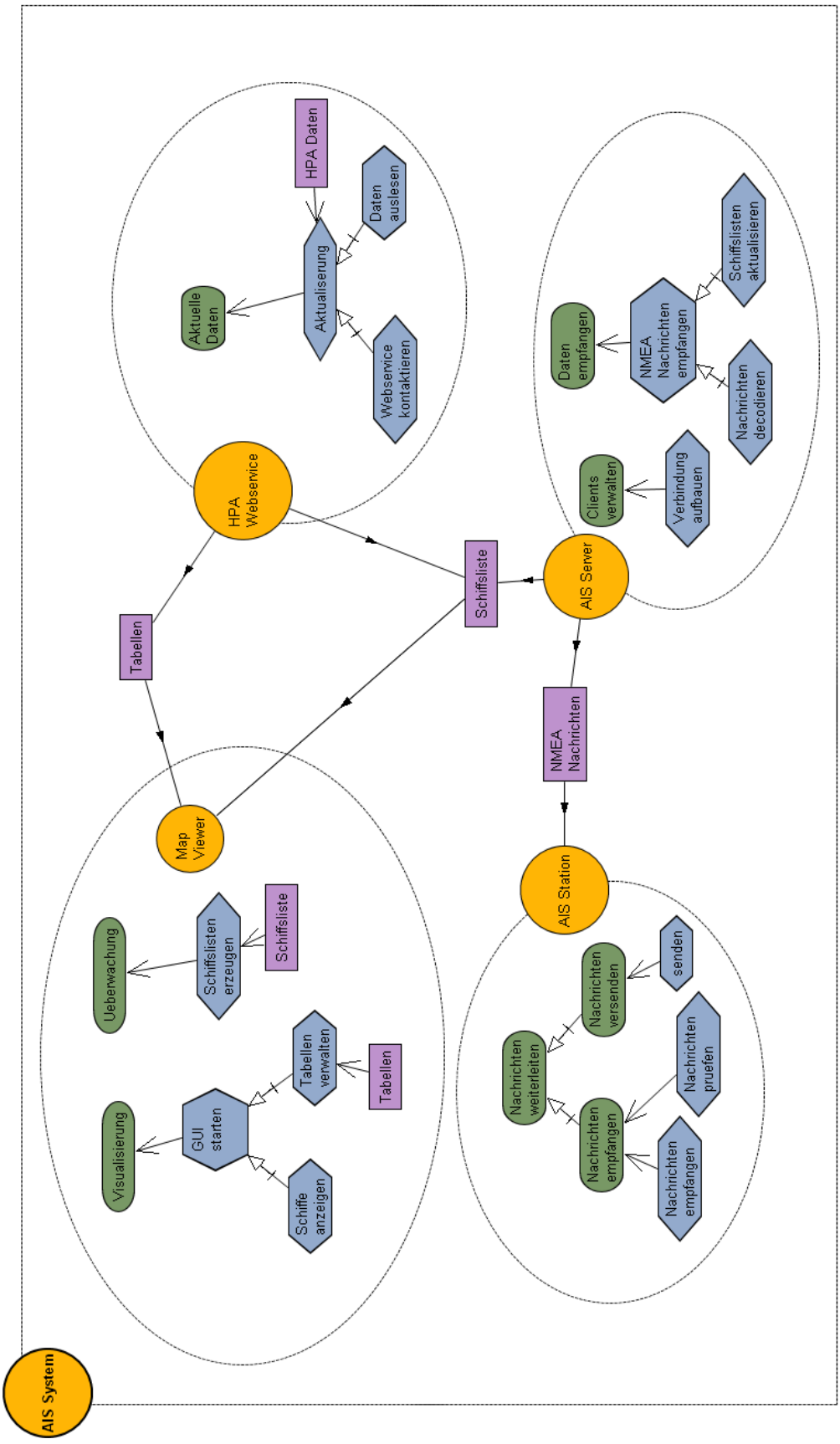


Abbildung 5.8 – *TAOM4E Architekturdesign Diagramm*

5.4 Architekturdesign (Architectural Design)

Im Architekturdesign wird das zu entwickelnde System in wichtige Bestandteile zerlegt und in dem Architekturdesign Diagramm dargestellt. Für das AIS-Überwachungssystem entstand so das in Abb. 5.8 dargestellte Diagramm. Das AIS-Überwachungssystem besteht aus vier wichtigen Bestandteilen:

- Map Viewer
- HPA Webservice
- AIS Server
- AIS Client

Die Funktionalität des Systems wurde schon im Kapitel 2 erläutert. Hier wird das AIS-System aus der Sicht der Agentenmodellierung betrachtet.

5.5 Detailliertes Design (Detailed Design) und Implementierung

In diesem Abschnitt wird das detailierte Design der zu implementierten Agenten durchgeführt und hinweise für die Implementierung gegeben. Aus dem TROPOS Modell, das mit TAOM4E erstellt wurde, lässt sich mit dem t2x-Tool Quellcode für die JADEX-Plattform erzeugen. Das Tool erzeugt das ADF-File und entsprechende JAVA-Klassen, die Ziele, Pläne und deren Abhängigkeiten repräsentieren. Die Aufgabe ist nun, diese erzeugten Files so zu ergänzen, dass die gewünschte Funktionalität erreicht wird.
Die Vergabe der Aufträge erfolgt nach einem Prinzip, das von dem Kontraktnetz abgeleitet wurde.

5.5.1 Zentrale

Meldungen Empfangen

Wie im Modell beschrieben, hat die Zentrale die Möglichkeit, Meldungen von den Teams zu empfangen. Eine Meldungsnachricht besteht aus einer Zeichenkette, die zwei durch Leerzeichen getrennte Teile enthält. Der erste Teil bestimmt den Nachrichtentyp. Eine Meldung beginnt mit den Zeichen „Meldungen_annehmen“. Es sind vier Meldungen erlaubt, die wie folgt definiert sind:

- Start**JobID* - Diese Meldung besagt, dass der Auftrag *JobID* gestartet wurde.

- End**JobID* - Diese Meldung besagt, dass der Auftrag *JobID* beendet wurde.
- SignUp**TeamID* - Diese Meldung wird zum Anmelden eines Teams bei der Zentrale verwendet.
- SignOff**TeamID* - Mit dieser Meldung meldet sich ein Team bei der Zentrale ab.

Wenn eine „Start*“ oder „End*“ Meldung eingetroffen ist, wird die aktuelle Zeit in den Auftrag mit der gesendeten JobID eingetragen. Die Aufträge werden im Beliefbase in einer Hashtabelle gespeichert und referenzieren die Objekte des AIS-Überwachungssystems. Die angemeldeten Teams werden in einer Liste im Beliefbase gespeichert.

Startplan(Initial Plan)

In der Konfiguration des Agenten kann man einen Plan festlegen, der bei der Initialisierung ausgeführt wird. Dieser Plan wird „initial Plan“ genannt. Der Zentrale-Agent erzeugt in seinem Startplan einen Thread mit dem AIS-Überwachungssystem.

Zurückgeschickte Aufträge annehmen

Wenn die Zentrale einen Auftrag zurückbekommt, wird zuerst der Plan „Auftrag speichern“ ausgeführt. In dem Plan werden die zurückgeschickten Aufträge ausgewertet und das Team, welches den Auftrag abgegeben hat, wird aus dem Auftrag gestrichen. Der Auftrag wird als nicht vergeben markiert. Das Ziel darf mit dem Plan nicht erreicht werden, damit das Ziel „Alle Auftraege verteilen“ aktiviert wird. Dazu wird im Plan „Auftrag speichern“ eine Exception geworfen. Die Implementierung des Ziels „Alle Auftraege verteilen“ wird im Abschnitt 5.5.1 beschrieben.

Aufträge verteilen

Wenn das Ziel „Alle Auftraege verteilen“ aktiviert wird, wird versucht das Ziel durch Aktivieren eines der Unterziele zu erreichen. Das Unterziel „Selbst ausfuehren“ wird als erstes aktiviert, weil dieses Ziel vorteilhafter für den Agenten ist.
In dem Plan „Auktion“ ist die Vergabe der Aufträge implementiert. Zuerst wird die Auftragsliste nach nicht vergebenen Aufträgen durchsucht. Wenn ein solcher Auftrag gefunden wurde, wird er ausgewertet. Bei der Auswertung wird überprüft, wie viele Personen und Fahrzeuge benötigt werden. Danach wird der Auftrag an alle Teams gesendet, die noch nicht an diesem Auftrag beteiligt sind. Nach dem Versenden der Angebote, wird auf die Antworten aller Teams gewartet. Wenn alle Antworten eingetroffen sind, werden die benötigten Teams ausgewählt. Das sind die Teams mit den besten Geboten. Die Gebote der Teams werden dazu nach der Größe aufsteigend sortiert. Dann wird die Liste der Gebote der Reihe nach durchgegangen und die Teams

ausgewählt, bis die nötige Anzahl von Personen erreicht ist. Die Gebote der Teams mit einem Boot werden dabei getrennt von den Fahrzeugteams betrachtet. Das Protokoll ist in Abb. 5.9 dargestellt. Falls es nicht genug Teams gibt, die den Auftrag erfüllen können, werden die freien Plätze mit „Bereitschaft“ markiert. In einer Schleife werden so alle noch nicht vergebenen Aufträge in der Liste verteilt. Die Berechnung der Gebote und die Struktur der Gebotsnachricht ist im Abschnitt 5.5.2 beschrieben. Nach dem Verteilen wird der Plan beendet.
Jetzt wird das Ziel „GenugTeams“ aktiviert. In dem Plan „Bereitschaft anrufen“ wird

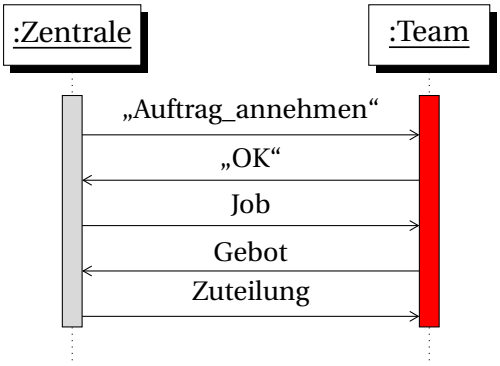


Abbildung 5.9 – *Auktionsprotokoll*

die Auftragsliste nach Aufträgen durchsucht, die mit „Bereitschaft“ markiert sind. Der Bereitschaftsruf kann nicht automatisch erfolgen. Aus diesem Grund wird der Bediener aufgefordert, den Bereitschaftsruf zu tätigen. Das System wartet dann auf die Rückmeldung des Benutzers. Falls für alle Aufträge genug Teams aus der Bereitschaft gerufen werden konnten, ist das Ziel „Selbst ausfuehren“ erreicht. Das führt wiederum dazu, dass das Ziel „Alle Auftraege verteilen“ erreicht wird, ohne dass das Ziel „Konkurrenz beauftragen“ aktiviert wird. Wenn der Benutzer zurückmeldet, dass der Bereitschaftsruf gescheitert ist, kann das Ziel „Selbst ausfuehren“ nicht erreicht werden. In diesem Fall wird das Ziel „Konkurrenz beauftragen“ aktiviert und der entsprechende Plan „Konkurrenzruf“ ausgeführt. Da dieser Plan noch nicht automatisiert ist, wird auch in diesem Fall der Benutzer aufgefordert den Konkurrenzruf zu tätigen. Nach dem der Koordinator, die Konkurrenz beauftragt hat, trägt er dieses in die Jobbeschreibung ein. Der Agent weiß, dass der Auftrag vergeben ist und er erreicht das Ziel „Alle Auftraege verteilen“. Momentan wird der Fall, dass der Konkurrenzruf nicht möglich ist, nicht betrachtet.

Aktualisierung

Um die Aktualisierung der Auftragsliste durchzuführen, muss das Ziel „Aktualisierung“ aktiviert werden. Damit das automatisch passiert, wurde ein Timer eingebaut,

der alle 60 Sekunden eine Aktualisierung aktiviert. Das Ziel „Aktualisierung“ wurde vom Typ *Achieve Goal* zum Typ *Perform Goal* geändert. Das hat zu Folge, dass das Ziel nur dann nicht erreicht wird, wenn alle Unterziele und Pläne nicht erfüllt werden. Das Ziel kann erreicht werden, wenn der Plan „updateJobs“ erfolgreich ist, das Ziel „Aenderungen finden“ oder „neue Auftraege finden“ erreicht wird. Dabei ist darauf zu achten, dass die Aktivierungen in der richtigen Reihenfolge ausgeführt werden. Zuerst wird der Plan „updateJobs“ aktiviert. Dieser holt sich die aktuellen Daten von dem AIS-Überwachungssystem. Am Ende des Plans wird eine Exception geworfen, damit das Oberziel weiter aktiv bleibt. Die Aufträge werden in dem Belief „jobMap“ vom Typ HashMap gespeichert.
Als nächstes wird das Ziel „Aktualisierungen finden“ aktiviert. Dieses startet den Plan „sendeAenderung“. Der Plan durchsucht die bekannten Aufträge darauf, ob sie als geändert markiert wurden. Wenn geänderte Aufträge gefunden wurden, werden die Teams, die an diesem Auftrag beteiligt sind, benachrichtigt. Dazu wird eine Nachricht vom Typ „request“ an die Teams geschickt. Der Inhalt der Nachricht wird in 5.5.2 spezifiziert. Nach dem Versenden der Änderungen wird im Rückgabewert „result“ der String „ok“ zurückgegeben. Wenn der Plan „ok“ zurückgibt, wird das Ziel nicht erreicht, weil dieser Wert als Fehlerbedingung in der Zielbeschreibung definiert ist. Wie nach der Ausführung des „updateJob“ Plans, bleibt das Ziel „Aktualisierung“ immer noch aktiv.
Als letztes wird das Ziel „neue Auftraege finden“ aktiviert. nach der Aktivierung wird der Plan „Auftraege pruefen“ ausgeführt. Dieser Plan durchsucht die Aufragsliste auf nicht vergebene Aufträge. Wenn nicht vergebene Aufträge gefunden werden, wird eine Exception geworfen, was dem Ziel signalisieren soll, dass das Ziel „Alle auftraege vergeben“ aktiviert werden kann.

Tausch verwalten

Wenn die Teams einen Tausch bestätigen möchten, senden sie eine Bestätigungsanfrage an die Zentrale. Diese Anfrage ist vom Typ String und ist wie folgt aufgebaut:

„Tausch_verwalten *JobID*AktuellesTeamID*NeuesTeamID*“

Das kurze Protokoll ist in der Abb. 5.10 dargestellt.

5.5.2 Team

Startplan (initial plan) und Schlußplan (end plan)

Wenn der Teamagent gestartet wird, wird der Startplan ausgeführt. Dieser Plan sorgt dafür, dass der Agent bei der Zentrale angemeldet ist. Die Anmeldung geschieht, indem der eine Meldung an die Zentrale geschickt wird. Der Inhalt dieser Meldung ist

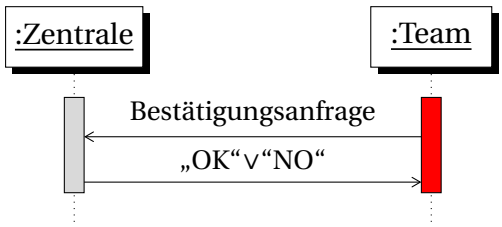


Abbildung 5.10 – *Tauschbestätigung*

bereits im Abschnitt 5.5.1 beschrieben. Im Startplan wird davon ausgegangen, dass der Zentrale-Agent gestartet wurde, bevor ein Teamagent gestartet wird. Der Name des Zentrale-Agenten bleibt unverändert, damit er immer erreicht werden kann. Außerdem wird das GUI gestartet, welches dem Team die Überwachung der Aufträge und Eingabe der Start- und Endzeiten ermöglicht.

Beim Beenden des Teamagenten wird der Schlußplan ausgeführt. In diesem Plan meldet sich der Agent von der Zentrale ab. Die Abmeldung erfolgt mithilfe einer Nachricht analog zur Anmeldung.

Änderungen empfangen

Eine Änderungsnachricht der Zentrale aktiviert das Ziel „Aenderungen empfangen“. Dieses Ziel aktiviert wiederum das Ziel „Auftraege konsistent halten“, welches das Unterziel „inkonsistente Auftraege abgeben“ aktiviert. Die Übermittlung der Änderungen erfolgt nach dem in Abb. 5.11 dargestellten Protokoll und wird im Plan „Auftraege pruefen“ implementiert.

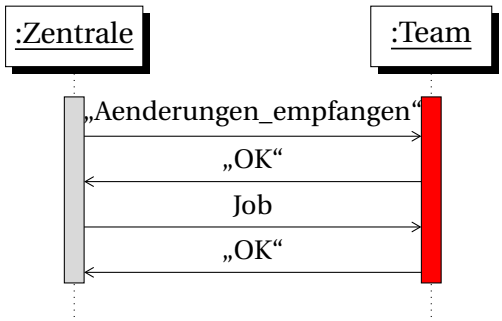


Abbildung 5.11 – *Änderungsprotokoll*

Dieser Plan empfängt die Änderungen und prüft anschließend die Konsistenz der Aufträge. Sind die Aufträge nach der Änderung nicht mehr konsistent, wird eine Exception geworfen und danach der Plan „Auftrag abgeben“ ausgeführt. Dieser Plan implementiert die Verhandlungen mit einem anderen Team. Wenn ein Team gefunden wurde, welches den Auftrag übernehmen würde, wird die Zentrale über die

Tauschabsicht informiert(siehe 5.5.1).
Wenn der Auftrag nicht abgegeben werden konnte, wird der Auftrag an die Zentrale zurückgegeben (siehe Abschnitt 5.5.1). Dazu wird eine Nachricht mit dem Inhalt „Auftrag_zurueck *TeamID***AuftragID*“ an die Zentrale gesendet.

Angebote empfangen

Ein Team kann an Auktionen der Zentrale Teilnehmen, wenn eine Nachricht der Zentrale das Ziel „Angebote empfangen“ aktiviert. Durch die Aktivierung des Ziels wird der Plan „An Auktion teilnehmen“ ausgeführt. In dem Plan, wird das Angebot analysiert und es wird ein Gebot berechnet. Eine einfache Möglichkeit ein Gebot zu berechnen, ist die Berechnung der Fahrzeiten zwischen den Aufträgen. Wenn ein neuer Auftrag ankommt, wird er mit der bestehenden Auftragsliste verglichen. Es werden die Aufträge gesucht, die vor „AuftragVor“ und nach „AuftragNach“ dem neuen Auftrag stattfinden. Das Gebot ist also:

Gebot = Fahrzeit(AuftragVor,AuftragNeu) + Fahrzeit(AuftragNeu,AuftragNach)

Es wird auch überprüft, dass der neue Auftrag zwischen den vorhandenen Aufträgen platziert werden kann. Das berechnete Gebot wird an die Zentrale gesendet und dann wartet der Agent auf die Antwort(siehe Protokoll Abb. 5.9). Die Nachricht mit dem Gebot ist wie folgt aufgebaut: „Bid_*TeamID***Gebot*# *Teamgröße*$*Fahrzeugtyp*“.

Tauschanfragen empfangen

Wenn das Ziel „Tauschanfragen empfangen“ aktiviert wird, wird der Plan „Tauschanfrage bearbeiten“ ausgeführt. Der Plan implementiert zur Kommunikation das Protokoll aus Abb. 5.12. Der angebotene Auftrag wird darauf überprüft, ob er ausführbar

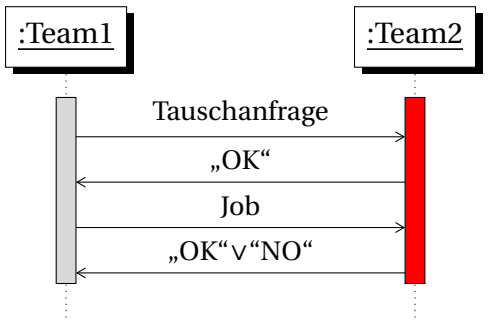


Abbildung 5.12 – *Tauschprotokoll*

ist. Anschließend wird eine Antwort an das anfragende Team gesendet.

Ausführung überwachen

Das Ziel „Ausfuehrung ueberwachen“ wird intern durch den Benutzer aktiviert. Ein Teammitglied muss dazu manuell das Starten und das Beenden eines Auftrages eingeben. Es werden die Start- und Endmeldungen aus 5.5.1 benutzt.

5.6 Modellerweiterungen

Das hier vorgestellte Modell ist eine vereinfachte Darstellung der realen Situation. Dieses Modell kann durch Erweiterungen verbessert werden, um die realen Verhältnisse besser darzustellen. Als Erweiterung kann zum Beispiel das Teamfahrzeug genauer modelliert werden. Die Teams können auch eigene Routenplaner benutzen, um die Entfernungen und Fahrzeiten auszurechnen. Auch die Gebotsfunktion kann mit weiteren Faktoren erweitert werden. Durch den Einsatz von GPS in Fahrzeugen und AIS auf Booten könnte eine Überwachung der Teams durch die Zentrale möglich sein.
Eine weitere Stufe der Modellerweiterung könnte die Auftragsvergabe unternehmensübergreifend geschehen. Das bedeutet, dass ein Festmacherbetrieb auch Teams anderer Betriebe an der Auktion telnehmen lassen kann. Auch die Teams verschiedener Betriebe könnten dann Aufträge untereinander austauschen oder verkaufen.

Kapitel 6

Zusammenfassung und Ausblick

In dieser Arbeit wurde zuerst ein Prototyp für ein AIS-Überwachungssystem entwickelt, der eine Überwachung des Schiffsverkehrs ermöglicht. Es können mehrere AIS-Empfänger eingesetzt werden, um ein größeres Gebiet abdecken zu können. Das entwickelte System soll demnächst bei einem Festmacherbetrieb installiert werden.
Da das AIS-System von mehreren Institutionen entwickelt wurde, hat sich die Suche nach Informationsquellen schwierig gestaltet. Durch die Verwendung von OpenStreetMap entfallen Kosten für die Beschaffung von Kartenmaterial und es wurde dadurch möglich, das System als eine Desktopanwendung zu realisieren. Es wurden mittlerweile AIS-Empfänger auf erdnahen Satelliten installiert, sodass eine weltweite Überwachung sehr viel einfacher realisiert werden kann.(siehe. [2])

Das Einsetzten eines Multiagentensystems zum automatisieren der Auftragsvergabe hat sich meiner Meinung nach als erfolgversprechend erwiesen. Durch die Verselbstständigung der Teams und deren Funktion als Agenten wird die Zentrale entlastet, weil sie unter anderem nicht mehr alle Informationen über die Teams verwalten muss und die Teams in die Verteilung der Aufträge miteinbezogen werden. Mithilfe der verwendeten Werkzeuge konnte ein Modell erstellt werden, welches eine vereinfachte Darstellung der Abläufe bietet. Das Modell erfasst alle wichtigen Abläufe der realen Welt. Im weiteren kann das Modell weiterentwickelt werden, um die reale Situation noch genauer beschreiben zu können.
TROPOS hat sich als ein recht einfaches Verfahren der Modellierung erwiesen. Dadurch dass nur wenige Modellierungssymbole verwendet werden, die in allen Entwicklungsstadien verwendet werden, ist das Verfahren schnell zu lernen. Mit TAOM4E ist das Modell leicht zu erstellen und in Quellcode überführbar. Ein Nachteil der Codegenerierung ist, dass die erzeugten Dateien nach einem erneuten Export überschrieben werden. Wenn man also ein Modell erstellt, es exportiert, die Klassen verändert und dann feststellt, dass das Modell verbessert werden muss, müssen die Änderungen in die neu erzeugten Dateien übertragen werden. Eine Modelländerung in einem späten Implementierungsstadium ist daher mit viel Aufwand verbunden.

Die Codegenerierung erzeugt viele Hilfsklassen, deren Funktionalität erst verstanden werden muss, bevor man die Implementierung fortsetzt. Es werden viele Ausgaben und Nachrichten von diesen generierten Klassen erzeugt, die sich oft als störend erweisen.
JADEX hat sich als ein mächtiges Werkzeug erwiesen. Der Aufbau eines Agenten gewöhnungsbedürftig, weil man sowohl eine XML-Datei sowie JAVA-Klassen verwendet, um einen Agenten zu implementieren. Es ist ziemlich Zeitaufwendig, alle Konzepte, die JADEX bietet, zu verstehen. Die Benutzeranleitung bietet eine umfangreiche und verständliche Beschreibung der einzelnen Konzepte, sodass man immer eine Lösung für ein entstandenes Problem findet. Die von JADEX bereitgestellten Tools erleichtern die Implementierung und ermöglichen eine Fehlersuche auf verschiedenen Ebenen.

Für den realen Einsatz des Multiagentensystems, müsste das Modell sowie die Implementierung optimiert werden. Der nächste Schritt könnte der Aufbau einer Simulationsumgebung sein, um die Reaktionen von Agenten auf verschiedene Ereignisse zu untersuchen. Im Rahmen dieser Arbeit war das nicht möglich gewesen, weil die Daten für eine Simulation nicht bereitgestellt wurden.

Anhang A

Benutzeroberfläche des AIS-Überwachungssystems

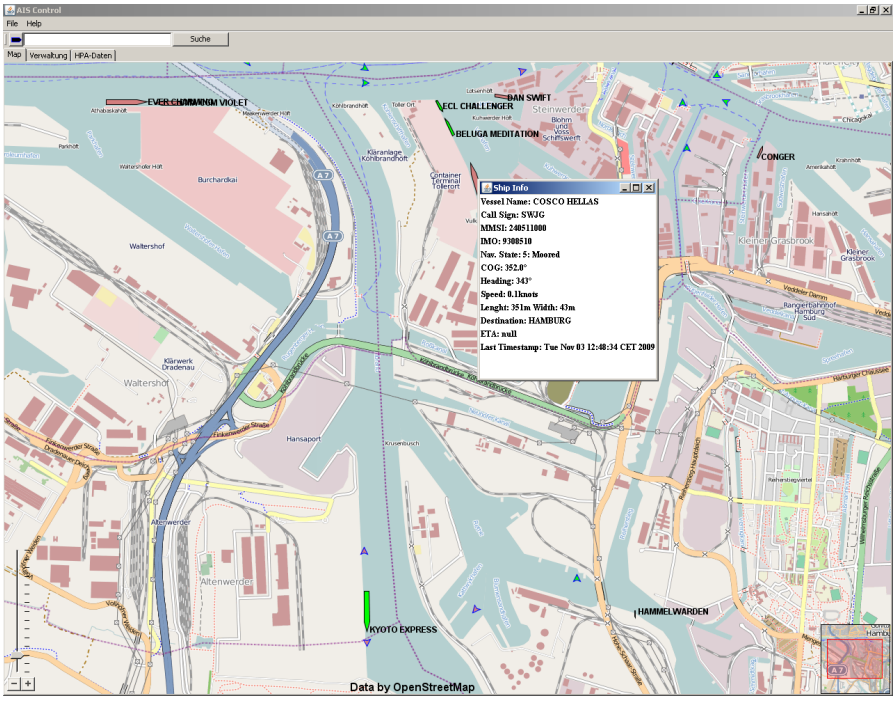


Abbildung A.1 – *AIS-Überwachungssystem GUI 1*

In den Abbildungen A.1, A.2und A.3 ist die graphische Benutzeroberfläche des AIS-Überwachungssystems dargestellt. In der Menüleiste kann man im Untermenü „File“ das Programm beenden und Proxy-Einstellungen für die Internetverbindung einstellen. Im Untermenü „Help“ kann die Version der Anwendung abgelesen werden.

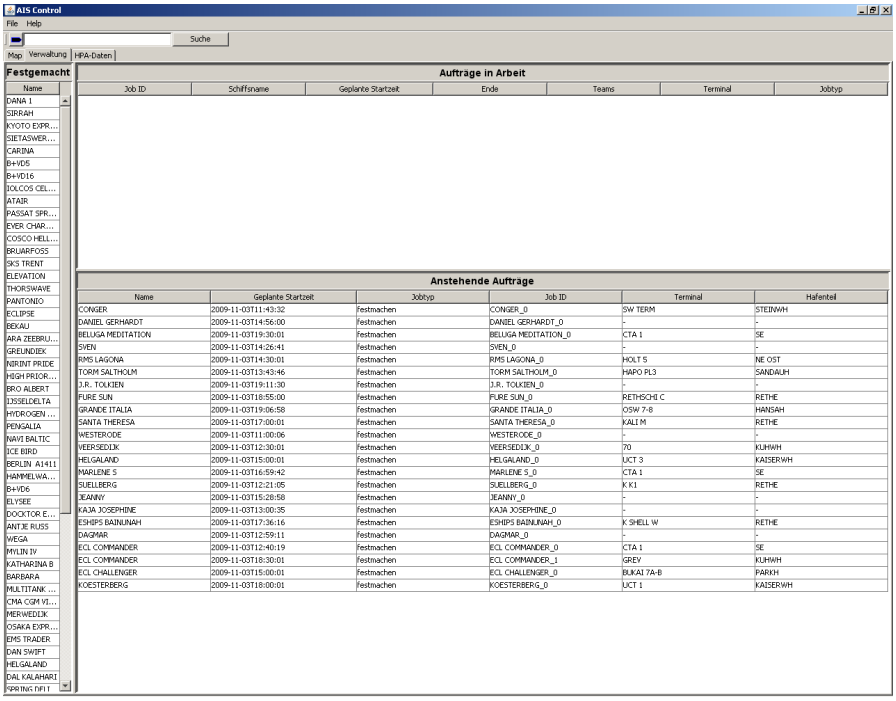


Abbildung A.2 – *AIS-Überwachungssystem GUI 2*

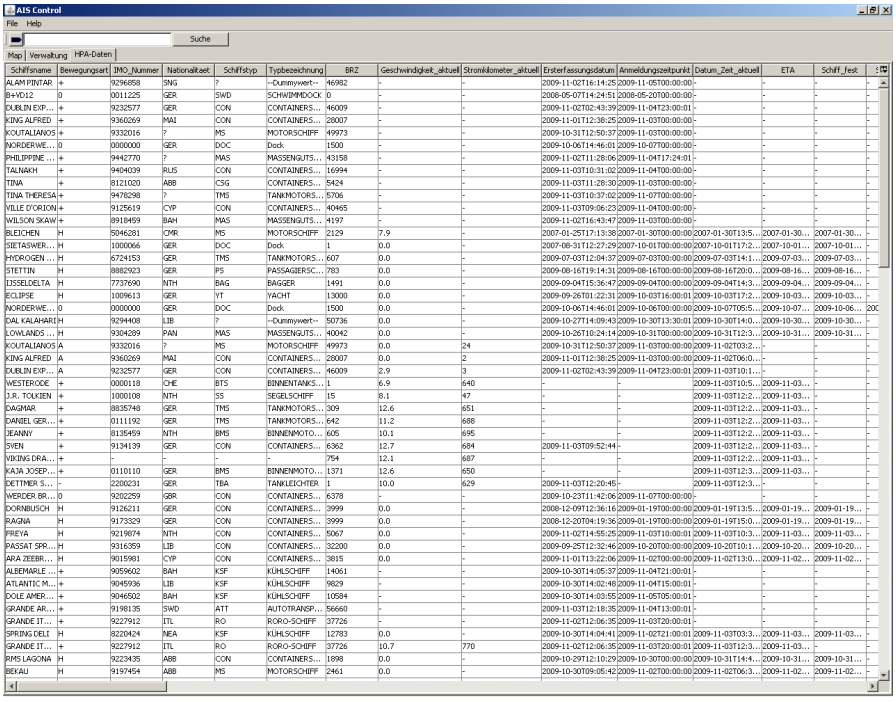


Abbildung A.3 – *AIS-Überwachungssystem GUI 3*

Die Toolbar stellt momentan zwei Funktionen zur Verfügung. Mit dem Button links können alle festgemachten Schiffe auf der Karte ausgeblendet werden. Die weitere Funktionalität ist das Suchen von Schiffen. Dazu muss der Name des Schiffes eingegeben und dann der Button „Suche“ gedrückt werden. Wenn ein Schiff mit dem eingegebenen Namen existiert, wird die Karte in der Position des gesuchten Schiffes zentriert.
Darunter befindet sich ein Panel mit drei Registerkarten. Die erste heisst „Map“ und enthält die Karte mit den überwachten Schiffen. Die Karte kann verschoben und vergrößert werden. Durch das Klicken auf ein Schiff wird ein Fenster mit einigen wichtigen Schiffsinformationen angezeigt. Die grünen Schiffe stellen die fahrenden Schiffe dar und die roten die festgemachten.
Die zweite Registerkarten enthält drei Tabellen. Die linke Tabelle listet alle festgemachten Schiffe. Die rechte obere Tabelle enthält alle vergebenen Aufträge und die untere alle noch nicht vergebenen Aufträge.
Die letzte Registerkarte enthält die Tabelle mit den Daten der HPA.

Literaturverzeichnis

[1] *Internationales Schiffsvermessungs-Übereinkommen von 1969*.

[2] http://www.orbcomm.de, 11 2009.

[3] *International Maritime Organisation*. `http://www.imo.org`, 10 2009.

[4] *OpenStreetMap*. `http://www.openstreetmap.org`, 10 2009.

[5] *Swingx Internetseite*. http://swinglabs.org/, 11 2009.

[6] *Weatherdock AG*. http://easyais.de, 10 2009.

[7] BEHRENS, C., BECKER M., J. D. GEHRKE, D. PETERS und R. LAUR: *Wireless Sensor Networks as an Enabler for Cooperating Logistic Processes*. In: *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN'06)*, Seiten 85–86, New York, USA, 2006. ACM.

[8] BERTOLINI, DAVIDE, ALIAKSEI NOVIKAU, ANGELO SUSI und ANNA PERINI: *TAOM4E: an Eclipse ready tool for Agent-Oriented Modeling. Issue on the development process*. Technischer Bericht, ITC-irst. Italien, 2005.

[9] BRATMAN, M. E.: *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[10] BRATMAN, MICHAEL E., DAVID J. ISRAEL und MARTHA E. POLLACK: *Plans And Resource-Bounded Practical Reasoning*, 1988.

[11] BRAUBACH, LARS, ALEXANDER POKAHR und WINFRIED LAMERSDORF: *Jadex: A Short Overview*. In: *Main Conference Net.ObjectDays 2004*, Seiten 195–207, 9 2004.

[12] BRAUBACH, LARS, ALEXANDER POKAHR und WINFRIED LAMERSDORF: *Jadex: A BDI Agent System Combining Middleware and Reasoning*. In: R. UNLAND, M. CALISTI, M. KLUSCH (Herausgeber): *Software Agent-Based Applications, Platforms and Development Kits*, Seiten 143–168. Birkhäuser-Verlag, 9 2005. Book chapter.

[13] BRESCIANI, P., P. GIORGINI, F. GIUNCHIGLIA, J. MYLOPOULOS und A. PERINI: *Tropos: An Agent-Oriented Software Development Methodology.* Autonomous Agents and Multi-Agent Systems, 8(3):203–236, 2004.

[14] BUSETTA, PAOLO, RALPH RÖNNQUIST, ANDEW HODGSON und ANDREW LUCAS: *JACK Intelligent Agents: Components for Intelligent Agents in Java.* AgentLink Newsletter, 2, January 1999.

[15] CASTRO, JAELSON, MANUEL KOLP und JOHN MYLOPOULOS: *Towards requirements-driven information systems engineering: the Tropos project.* Inf. Syst., 27(6):365–389, September 2002.

[16] D'INVERNO, MARK, MICHAEL LUCK, MICHAEL GEORGEFF, DAVID KINNY und MICHAEL WOOLDRIDGE: *The dMARS Architecture: A Specification of the Distributed Multi-Agent Reasoning System.* Autonomous Agents and Multi-Agent Systems, 9(1-2):5–53, 2004.

[17] DOUMA, ALBERT MENNO: *Aligning the operations of barges and terminals through distributed planning.* Doktorarbeit, Enschede, December 2008.

[18] FERBER, JACQUES: *Multiagentensysteme. Eine Einführung in die Verteilte Künstliche Intelligenz.* Addison-Wesley, 2001.

[19] GERSHENSON, CARLOS: *Self-Organizing Traffic Lights.* COMPLEX SYSTEMS, 16:29, 2004.

[20] GIUNCHIGLIA, FAUSTO, JOHN MYLOPOULOS und ANNA PERINI: *The Tropos Software Development Methodology: Processes, Models and Diagrams.* Seiten 162–173. 2003.

[21] GONZÁLEZ-VÉLEZ, HORACIO, MARIOLA MIER, MARGARIDA JULIÀ-SAPÉ, THEODOROS N. ARVANITIS, JUAN MIGUEL GARCÍA-GÓMEZ, MONTSERRAT ROBLES, PAUL H. LEWIS, SRINANDAN DASMAHAPATRA, DAVID DUPPLAW, ANDREW PEET, CARLES ARÚS, BERNARDO CELDA, SABINE VAN HUFFEL und MAGÍ LLUCH I ARIET: *HealthAgents: distributed multi-agent brain tumor diagnosis and prognosis.* Appl. Intell., 30(3):191–202, 2009.

[22] HAYES-ROTH, BARBARA, A. COLLINOT, V. DABIJA, J. DRAKOPOULOS, D. GABA, G. GOLD, M. HEWETT, R. HEWETT, A. MACALALAD, A. SEIVER, S. UCKUN, A. VINA und R. WASHINGTON: *An Architecture for Adaptive Intelligent Systems*, 1995.

[23] HIRCHE, RÜDIGER: *AIS in Theorie und Praxis.* Delius Klasing, 2009.

[24] IEC: *IEC 61993-2. Maritime navigation and radiocommunication equipment and systems - Automatic identification systems (AIS). Part 2*, 12 2001.

[25] IMO: *Resolution A.917(22). GUIDELINES FOR THE ONBOARD OPERATIONAL USE OF SHIPBORNE AUTOMATIC IDENTIFICATION SYSTEMS (AIS)*, 01 2002.

[26] IMO: *SOLAS : consolidated text of the International Convention for the Safety of Life at Sea, 1974, and its Protocol of 1988 : articles, annexes and certificates*. International Maritime Organization, 4. Auflage, 2004. Incorporating all amendments in effect from 1 July 2004.

[27] IMO: *RECOMMENDATION ITU-R M.1371-3. Technical characteristics for an automatic identification system using time division multiple access in the VHF maritime mobile band*, 2007.

[28] MAES, PATTIE: *Artificial Life Meets Entertainment: Lifelike Autonomous Agents*. Commun. ACM, 38(11):108–114, 1995.

[29] MORANDINI, M., L. PENSERINI und A. PERINI: *Automated Mapping from Goal Models to Self-Adaptive Systems*. In: *Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on*, Seiten 485–486, 2008.

[30] MURCH, RICHARD und TONY JOHNSON: *Agententechnologie: die Einführung*. Addison-Wesley, München [u.a.], 2000.

[31] PERINI, ANNA und ANGELO SUSI: *Developing Tools for Agent-Oriented Visual Modeling*. In: *MATES*, Seiten 169–182, 2004.

[32] POKAHR, ALEXANDER und LARS BRAUBACH: *JADEX Tool Guide*. Universität Hamburg, Release 0.96 Auflage, Juni 2007.

[33] POKAHR, ALEXANDER und LARS BRAUBACH: *JADEX. User Guide*. Universität Hamburg, Release 0.96 Auflage, Juni 2007.

[34] POKAHR, ALEXANDER, LARS BRAUBACH und WINFRIED LAMERSDORF: *Jadex: Implementing a BDI-Infrastructure for JADE Agents*. EXP - in search of innovation (Special Issue on JADE), 3(3):76–85, 9 2003.

[35] POKAHR, ALEXANDER, LARS BRAUBACH, ANDRZEJ WALCZAK und WINFRIED LAMERSDORF: *Jadex - Engineering Goal-Oriented Agents*. In: BELLIFEMINE, F., G. CAIRE und D. GREENWOOD (Herausgeber): *Developing Multi-Agent Systems with JADE*, Seiten 254–258. Wiley & Sons, 1 2007.

[36] RAO, A. S. und M. P. GEORGEFF: *BDI-agents: from theory to practice*, 1995.

[37] RAO, ANAND S. und MICHAEL P. GEORGEFF: *Modeling Rational Agents within a BDI-Architecture*, 1991.

[38] RUSSELL, STUART J. und PETER NORVIG: *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1st Auflage, January 1995.

[39] SMITH, R. G.: *The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver*. Computers, IEEE Transactions on, C-29(12):1104–1113, August 2006.

[40] SUSI, ANGELO, ANNA PERINI, JOHN MYLOPOULOS und PAOLO GIORGINI: *The Tropos Metamodel and its Use*. Informatica, 29:401–408, 2005.

[41] TAOM4E-INTERNETSEITE. http://sra.itc.it/tools/taom4e/.

[42] WOOLDRIDGE, MICHAEL: *Introduction to MultiAgent Systems*. John Wiley & Sons, June 2002.

[43] WOOLDRIDGE, MICHAEL und NICHOLAS R. JENNINGS: *Intelligent Agents: Theory and Practice*. Knowledge Engineering Review, 10:115–152, 1995.

[44] YU, ERIC SIU-KWONG: *Modelling strategic relationships for process reengineering*. Doktorarbeit, University of Toronto, Toronto, Ont., Canada, Canada, 1995. Adviser-Mylopoulos, John.