5. Objektorientierte und agentenorientierte Modellierung von Fahrerlosen Transportsystemen

Ein in dieser Arbeit ausführlich behandeltes Beispiel für Simulation sind *Fahrerlose Transportsysteme* (FTS). Ein FTS besteht dabei aus *Fahrerlosen Transportfahrzeugen* (FTF), die automatisch gesteuert Transportstücke innerhalb einer Verkehrszone von einem Ort zum anderen transportieren.

FTS werden hier als ein komplexes Beispiel für Agentenbasierte Simulation benutzt. Anhand des Beispiels werden dabei auch die Möglichkeiten objekt- und agentenorientierter Modellierung diskutiert.

Kapitel 5 bietet eine Einführung in Fahrerlose Transportsysteme, die sowohl objekt- als auch agentenorientiert modelliert werden, wobei insbesondere auf den Aspekt der Auftragsvergabe eingegangen wird. In Kapitel 6 werden dann Fahrerlose Transportsysteme benutzt, um ein konkretes Beispiel für die Spezifikation eines Agentenbasierten Simulationssystems darzustellen.

Fahrerlose Transportsysteme (FTS) bestehen aus einem Steuerungssystem und einem oder mehreren Fahrerlosen Transportfahrzeugen[52] (FTF), die Güter auftragsgetrieben innerhalb einer - meist mit einem Fahrkurs durchzogenen - Verkehrszone von einem Ort zu einem anderen transportieren. Dabei ist das Steuerungssystem für die Vergabe von Transportaufträgen, für die Wegplanung und für die Verkehrsregelung zuständig. FTS werden überwiegend für die innerbetriebliche Materialflussabwicklung, zum Beispiel für die Verkettung von Fertigungseinrichtungen und Montageplätzen, eingesetzt.

Heutige Systeme sind zwar im Regelbetrieb hinreichend effizient, im Hinblick auf Veränderungen und Störungen aber meistens zu wenig flexibel und robust. Eine Analyse des *Fraunhofer Instituts Systemtechnik und Innovationsforschung* [ISI01] bestätigt diese Diagnose. Im Fazit der Studie heißt es: „Aktuelle Automatisierungskonzepte können in vielen Fällen nicht die notwendige Flexibilität gewährleisten. Durch kleine Seriengrößen und starke Veränderungen der Umsätze in den vergangenen Jahren wurden die Grenzen der klassischen Automatisierung deutlich.“ Im Falle von FTS liegen die Ursachen für die mangelnde Flexibilität und Robustheit vor allem in der zentralen Steuerung und der damit verbundenen Annahme, dass die im zentralen Steuerungssystem verfügbaren Informationen vollständig und aktuell sind. Weitere Ursachen sind eine leitliniengebundene Fahrzeugführung und die Einschränkung auf unidirektionale Fahrstrecken. Demgegenüber lassen dezentral gesteuerte FTS größere Flexibilität und Robustheit und damit auch ein insgesamt günstigeres Leistungsverhalten erwarten.

Ein FTS mit dezentral gesteuerten Fahrzeugen lässt sich auf natürliche Weise als Multiagentensystem betrachten. Die in der FTS-Domäne vorkommenden Agententypen sind: FTF, Auftragsmanager, (Fertigungs-)Maschinen, Verkehrsregler und menschliche Operateure. Als autonomer Agent kann ein FTF

- seinen Weg selbst planen
- Transportaufträge annehmen und ablehnen
- über die Auftragsvergabe mit einem Auftragsmanager verhandeln
- mit anderen FTF ‚Geschäfte machen’ (z.B. Transportstücke austauschen)

Ein FTS kann mit Hilfe der objektorientierten Modellierungssprache *UML* [OMG04a] modelliert werden. Dabei können jedoch die agentenspezifischen Konzepte, wie Wahrnehmung, Kommunikation, Überzeugungen und Verpflichtungen, nur unzureichend erfasst werden. Um ein agentenbasiertes FTS *agentenorientiert* zu modellieren, wird eine Modellierungssprache benötigt, die zwischen Agenten und Objekten unterscheidet und die Modellierung der Interaktion zwischen Agenten erlaubt. *AORML* [Wag03] ist eine auf UML basierende agentenorientierte Modellierungssprache, die diese Anforderungen erfüllt und neben der herkömmlichen objektorientierten Modellierung von Fahrkurs und Transportaufträgen die agentenorientierte

[52] Strenggenommen muss es sich nicht um *Fahr*zeuge handeln, sondern es könnten auch (U-)Boote oder Flugzeuge zum Einsatz kommen. Der englische Begriff *Automatically Guided Vehicle (AGV)* drückt dies besser aus.

Modellierung der Interaktion zwischen FTF, Auftragsmanagern, (Fertigungs-)Maschinen und Verkehrsreglern erlaubt.

In Abschnitt 5.1 werden FTS, FTF, Fahrkurs, Übergabestationen, Transportaufträge und Orte mit Hilfe von UML-Klassendiagrammen und die Transportauftragsvergabe mit Hilfe von UML-Sequenzdiagrammen objektorientiert modelliert. In den Abschnitten 5.2, 5.3 und 5.4 werden die Modelle aus Abschnitt 5.1 auf unterschiedliche Weise agentenorientiert erweitert. In Abschnitt 5.2 wird mit Hilfe von *AUML*-Sequenzdiagrammen [OPB00], die eine Erweiterung von UML-Sequenzdiagrammen darstellen, die Interaktion zwischen den FTS-Agenten, anstatt wie zuvor der -Objekte, modelliert. In Abschnitt 5.3 wird mit Hilfe von *AORML*-Diagrammen, die eine Erweiterung von UML-Klassendiagrammen darstellen, zwischen den Objekttypen Ort, Fahrkurs, Fahrkurselement, Transportauftrag und Transportstück auf der einen und den Agententypen Transportauftragsmanager, FTF und Fertigungsmaschine unterschieden. Schließlich wird in Abschnitt 5.4 die agentenorientierte Analyse- und Entwurfsmethode *Gaia* [WJK00] dazu verwendet, ein FTS mit *Rollen*- und *Protokollschemata* zu beschreiben. Abschnitt 5.5 gibt Referenzen zu verwandten Arbeiten. Abschnitt 5.6 enthält eine kurze Zusammenfassung, in der auch die verwendeten Modellierungsmethoden miteinander verglichen werden. Im Anhang A werden Begriffe aus dem Bereich FTS in einem Glossar erläutert. Im Glossar können auch die in diesem Kapitel verwendeten Abkürzungen nachgeschlagen werden.

5.1 Modellierung in UML

UML [OMG04a] ist die Standardsprache für objektorientierte Analyse und objektorientierten Entwurf. Vor allem aufgrund ihrer weiten Verbreitung in Forschung und Industrie und der damit verbundenen Möglichkeit der Kommunikation eignet sie sich sehr gut zur Modellierung. Im Folgenden wird die Domäne Fahrerloser Transportsysteme (FTS) in UML modelliert. Dabei wird zunächst bewusst auf agentenorientierte Modellierung verzichtet.

5.1.1 Fahrerlose Transportsysteme (FTS)

Ein FTS besteht aus beliebig vielen Fahrerlosen Transportfahrzeugen (FTF), einem Fahrkurs und beliebig vielen Übergabestationen. Der Fahrkurs ist die Strecke, die von FTF abgefahren werden kann. Übergabestationen sind Bereiche, in denen Transportstücke abgelegt und aufgenommen werden können. Abbildung 12 zeigt das UML-Diagramm für ein FTS.

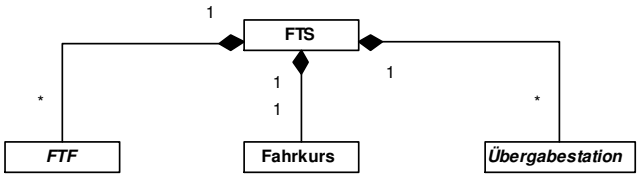

Abbildung 12: UML-Klassendiagramm für Fahrerlose Transportsysteme (FTS)

5.1.2 Fahrerlose Transportfahrzeuge (FTF)

FTF unterscheiden sich durch die Art ihrer Führungstechnologie. Leitliniengebundene FTF fahren eine Strecke an Leitlinien entlang, die sie technisch nicht verlassen können. Orientierungsmarkengebundene FTF können sich frei im Raum bewegen und bestimmen ihre Position anhand von Orientierungsmarken. Oftmals haben aber auch orientierungsmarkengebundene FTF eine festgelegte Fahrstrecke, die sie aber in Ausnahmefällen (etwa bei der Umfahrung eines Hindernisses) verlassen können und dürfen.

Eine andere Art der Unterscheidung von FTF bildet ihre Kapazität, das ist die Anzahl der Transportstücke (TS), die ein FTF gleichzeitig befördern kann. Kann nur ein TS befördert werden, spricht man von einem *Single-Load-Carrier*, während ein FTF, das mehrere TS befördern kann, *Multiple-Load-Carrier* genannt wird. Bei Ebben [Ebb01] gibt es diese Unterscheidung nicht. Dort wird zwischen den englischen Begriffen *cargo* und *load* unterschieden, wobei jedes FTF nur ein *load* transportieren kann, dieses aber vor dem Beladen aus mehreren *cargos* zusammengesetzt werden kann, die dann nach dem Abladen wieder gesplittet werden.

Bei Ebben [Ebb01] gibt es außerdem noch spezielle Abschleppfahrzeuge, die nicht die Aufgabe haben, TS zu transportieren, sondern liegengebliebene FTF abzuschleppen. Da sich die Kapazität auf die Anzahl der TS bezieht, die ein FTF transportieren kann, haben Abschleppfahrzeuge die Kapazität 0.

Abbildung 13 zeigt das UML-Diagramm für FTF.

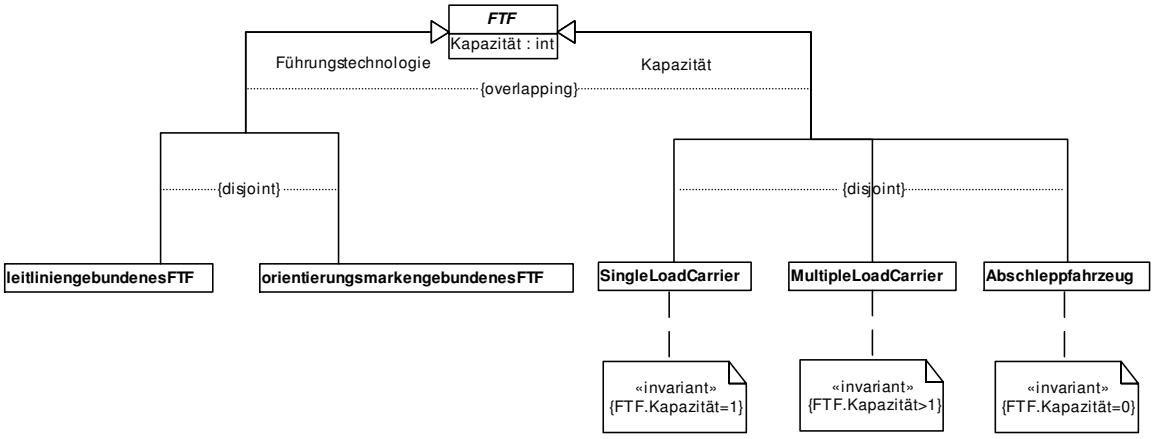

Abbildung 13: UML-Klassendiagramm für Fahrerlose Transportfahrzeuge (FTF)

5.1.3 Fahrkurs

Die Strecke, die von FTF abgefahren werden kann, bezeichnet man als Fahrkurs. Ein Fahrkurs besteht aus Knoten und Segmenten, ist demnach ein Graph mit den Knoten als Ecken und den Segmenten als Kanten.

Segmente sind Fahrkursstrecken, die an zwei Knoten angrenzen, sie können aus mehreren Geraden und Kurven bestehen. Segmente können entweder nur in eine Richtung durchfahren werden (unidirektionale Segmente[53]) oder in beide Richtungen (bidirektionale Segmente). Bei letzteren dürfen sich keine zwei FTF gleichzeitig in entgegengesetzte Richtungen im Segment befinden. Strecken, die breit genug dafür sind, dass entgegenkommende FTF aneinander vorbeifahren können, werden nicht als ein bidirektionales Segment, sondern als zwei entgegengesetzt orientierte unidirektionale Segmente betrachtet.

Knoten, die mehr als zwei Segmente verbinden, werden *Kreuzungen* genannt. Bei Kreuzungen spricht man von unidirektionalen Kreuzungen, wenn alle angrenzenden Segmente unidirektional sind. Spezielle unidirektionale Kreuzungen sind Verzweigungen, wenn nur ein Segment in die Kreuzung einmündet und Zusammenführungen, wenn nur ein Segment aus der Kreuzung herausführt. *Buchten* sind als Ausweichstellen für entgegenkommende FTF oder als Überholmöglichkeit für in die gleiche Richtung fahrende FTF gedacht. Buchten verbinden immer genau zwei Segmente miteinander. Weitere spezielle Knoten sind *Parkplätze* und *Batteriestationen*. Auf Parkplätzen können sich FTF aufhalten, wenn sie gerade keinen Transportauftrag zu erledigen haben. An Batteriestationen können FTF, sofern sie batteriebetrieben sind, ihre Batterie aufladen oder gegen eine aufgeladene austauschen. Es gibt weitere Knotentypen. Das sind Knoten, die nur an ein Segment angrenzen (das Ende einer Sackgasse) und Knoten, die zwar wie Buchten an zwei Segmente angrenzen, aber nicht genügend Platz zum Ausweichen bieten. Bei solchen Knoten ist es nur sinnvoll, sie als Knoten zu modellieren, wenn an ihnen eine Übergabestation liegt.

Betrachtet man wie bei Ebben [Ebb01] nicht nur FTS für den innerbetrieblichen Bereich, sondern sogenannte *externe FTS*, so bietet sich an, hierarchisch strukturierte Fahrkurse zu betrachten. Externe FTS sind solche, die sich über ein großes Gebiet erstrecken oder ein zusammenhängendes Netz von Firmen miteinander verbinden. Ein Fahrkurs kann dann beliebig viele Teilfahrkurse haben, die jeweils Ein- und Ausgänge haben. Auch wenn somit eine beliebige Schachtelungstiefe möglich ist, scheinen Hierarchien mit mehr als 3 oder 4 Ebenen im Allgemeinen nicht viel Sinn zu machen.

Abbildung 14 zeigt das UML-Diagramm für den Fahrkurs.

[53] Unidirektionale Segmente sind vergleichbar mit Einbahnstraßen im Straßenverkehr.

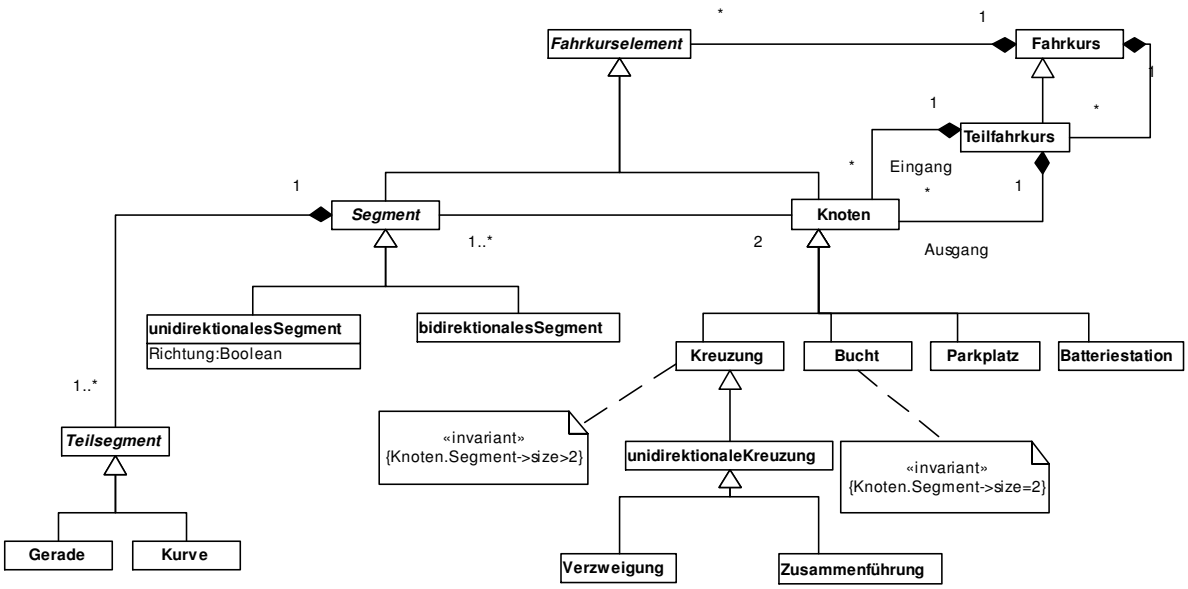

Abbildung 14: UML-Klassendiagramm für den Fahrkurs eines FTS

Man kann hierarchische Fahrkurse auch verwenden, um Kreisverkehre zu modellieren. Würde man einen Kreisverkehr mit z.B. vier einmündenden Straßen als eine Kreuzung modellieren, wäre das eine ungenaue Abbildung des Kreisverkehrs in Hinblick auf Strategien zur Kollisionsvermeidung. Bei Kreuzungen wäre eine geeignete Strategie, dass nur ein FTF sich gleichzeitig in der Kreuzung befinden darf. In einem Kreisverkehr können durchaus zwei FTF gleichzeitig fahren, ohne sich in die Quere zu kommen, etwa, wenn ein FTF aus Richtung Süd kommt und in Richtung Ost fahren will und das andere aus Richtung Nord kommt und in Richtung West fahren will[54]. Verwendet man die hierarchische Strukturierungsmöglichkeit, so bildet man einen Kreisverkehr als einen Teilfahrkurs ab, bei dem jeder Knoten je nach Orientierung der angrenzenden Segmente Eingang, Ausgang oder beides sein kann.

Schrecker [Schr00] unterscheidet zwischen sechs verschiedenen Fahrkurstopologien. *Unidirektionale Fahrkurse* bestehen ausschließlich oder überwiegend aus unidirektionalen Segmenten. Analog bestehen *bidirektionale Fahrkurse* ausschließlich oder überwiegend aus bidirektionalen Segmenten. Fahrkurse mit sowohl unidirektionalen als auch bidirektionalen Segmenten werden als *gemischt orientierte Fahrkurse* bezeichnet. Eine spezielle unidirektionale Fahrkurstopologie ist *Single Loop*, hier besteht der Fahrkurs aus einer einzigen Schleife. Hat man mehrere solcher Schleifen, spricht man von *Multiple Loop*. Dabei ist jedes FTF einer Schleife zugeordnet. Schleifenübergreifende Transporte werden durch Transferstationen realisiert, an denen ein FTF einem anderen ein oder mehrere TS überreichen kann. Ein *segmentierter Fahrkurs* besteht aus mehreren abgetrennten Teilabschnitten, die jeweils nur von einem FTF befahren werden. Auch hier werden Transferstationen eingesetzt. Im Gegensatz zu den oben beschriebenen Teilfahrkursen eines hierarchisch strukturierten Fahrkurses können FTF nicht zwischen verschiedenen Teilabschnitten hin- und herfahren. Denkbar wäre auch, einen Fahrkurs in mehrere abgetrennte Teilabschnitte zu zerlegen, wobei sich innerhalb eines Teilabschnitts auch mehrere FTF befinden können. Auf jeden Fall muss innerhalb eines Teilabschnitts von jedem Knoten zu jedem anderen ein Weg existieren, da es sonst Knoten gäbe, die nicht mehr erreichbar sind.

5.1.4 Übergabestation

Übergabestationen sind Bereiche, in denen TS von FTF abgelegt und aufgenommen werden können. Dabei wird zwischen *Senken*, *Quellen*, *Zwischenlagern* und *Transferstationen* unterschieden.Jede Übergabestation[55] befindet sich an einem Knoten, wobei sich an einem Knoten beliebig viele Übergabestationen befinden können.

---

[54] Rechtsverkehr wird hierbei angenommen.

[55] Der Begriff *Übergabestation* wird von Schrecker [Schr00] verwendet. Balko [Bal00] verwendet den Begriff *Pufferelement* für "das Ein- und Ausgangslager sowie die lokalen Maschinenpuffer". Dieser Begriff wird hier

Senken sind Bereiche, in die FTF die Möglichkeit haben, TS abzulegen, die von außerhalb des Fahrkurses entnommen werden. Im Gegensatz dazu werden in *Quellen* von außerhalb des Fahrkurses TS abgelegt, die durch FTF entnommen werden. Dabei kann mit "außerhalb des Fahrkurses" sowohl an der Grenze des zu modellierenden Systems als auch innerhalb des Systems (z.B. der Fertigungsbereich einer Maschine) gemeint sein. *Zwischenlager* sind Bereiche, in die FTF sowohl TS ablegen können als auch aus ihnen welche entnehmen können. Sie dienen zur temporären Ablage von TS. In *Transferstationen* kann ein FTF einem anderen ein oder mehrere TS überreichen.

Ausgangslager, aus denen von außerhalb des Systems TS entnommen werden und *Eingangspuffer* einer (Fertigungs-)Maschine sind spezielle Senken. Analog sind *Eingangslager* und *Ausgangspuffer* spezielle Quellen. Die Entnahme der TS aus Senken wird durch *Transportstück-Konsumenten* (TS-Konsumenten) vorgenommen. Entsprechend legen *Transportstück-Produzenten* (TS-Produzenten) TS in Quellen. Maschinen übernehmen die Rolle des TS-Konsumenten und des TS-Produzenten. Eine Fertigungsmaschine ist somit sowohl ein TS-Konsument als auch ein TS-Produzent. Werden FTS in Umschlagplätzen wie etwa Bahnhöfen oder Häfen modelliert, so können *externe Güterverkehrsmittel* (z.B. LKWs oder Schiffe) TS-Konsument oder TS-Produzent sein. Abbildung 15 zeigt das UML-Diagramm für die Übergabestation.

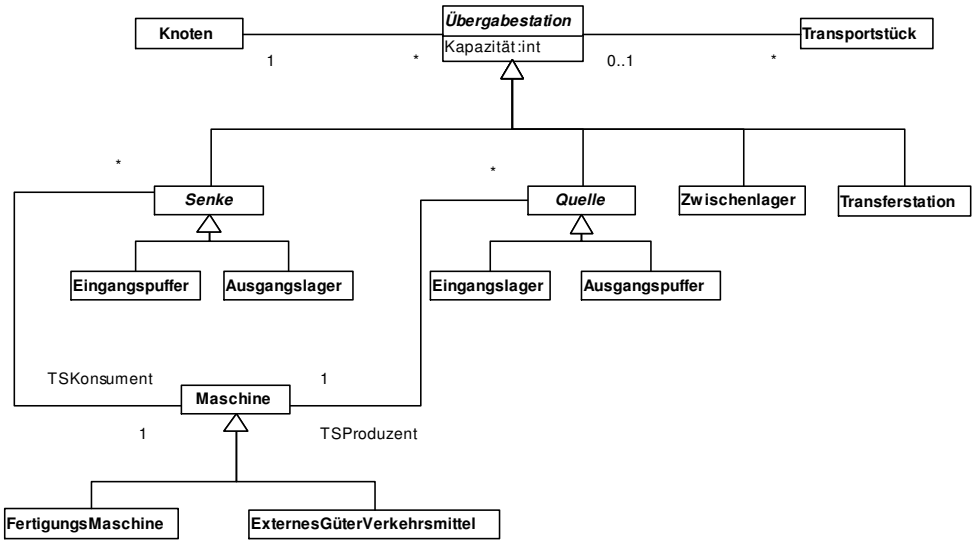

Abbildung 15: UML-Diagramm für die Übergabestationen eines FTS

5.1.5 Transportauftrag

Ein *Transportauftrag* (TA) ist die Aufgabe, ein TS von einem Ort (dem Startort des TA) zu einem anderen (dem Zielort) zu transportieren. TA werden an FTF vergeben, die sie dann ausführen. Man unterscheidet zwischen *angebotsinduzierten TA*, die dadurch entstehen, dass sich TS an einem Ort befinden, die woanders hin transportiert werden sollen und *nachfrageinduzierten TA*, die dadurch entstehen, dass an einem Ort ein Mangel an TS herrscht. Angebotsinduzierten TA geht ein *Abholauftrag* (AA) eines TS-Produzenten voraus, während nachfrageorientierte TA einem *Beschaffungsauftrag* (BA) eines TS-Konsumenten folgen. Ein TA hat eine Startzeit und eine Endzeit. Die Startzeit ist der Zeitpunkt, zu dem das TS frühestens am Startort abgeholt werden kann. Die Endzeit ist der späteste Zeitpunkt, zu dem das TS am Zielort sein muss. Abbildung 16 zeigt das UML-Diagramm für den Transportauftrag.

---

nicht verwendet, da ja ein Oberbegriff für Lager und Puffer gesucht wird. Bei Ebben [Ebb01] wird als englischer Begriff für *Übergabestation* das Wort *dock* verwendet.

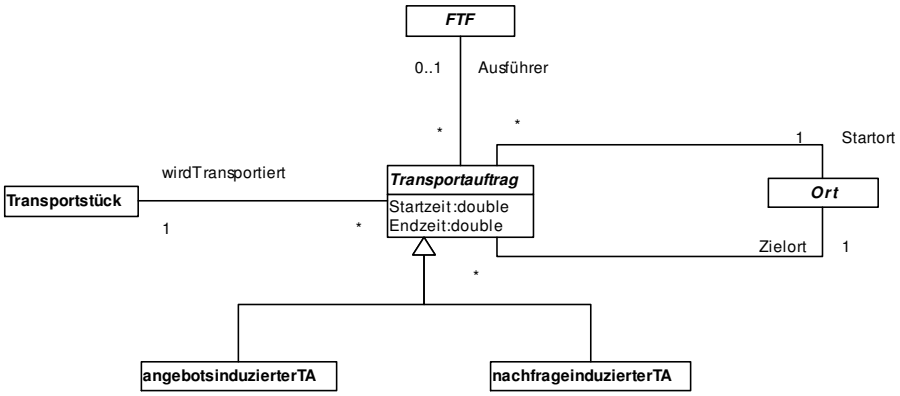

Abbildung 16: UML-Klassendiagramm für Transportaufträge

5.1.6 Ort

Orte sind anfahrbare Positionen im Fahrkurs, die Startort oder Zielort eines TA sein können. Zu diesen gehören Übergabestationen und Knoten. Jeder Ort hat Koordinaten. Weitere Punkte mit Koordinaten sind Orientierungsmarken, an denen sich orientierungsmarkengebundene FTF orientieren. Jeder Ort kann eine Orientierungsmarke haben, es gibt aber auch Orte ohne Orientierungsmarke. Umgekehrt kann eine Orientierungsmarke zu einem Ort gehören, muss es aber nicht. Abbildung 17 zeigt das UML-Diagramm für den Ort.

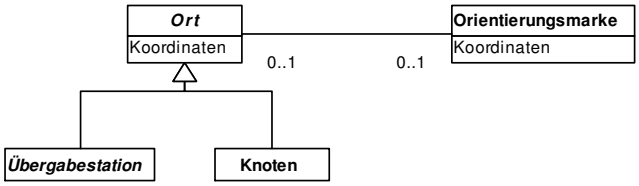

Abbildung 17: UML-Klassendiagramm für Orte

5.1.7 Auftragsvergabe

Unter Auftragsvergabe versteht man die Zuordnung von TA zu FTF. Prinzipiell lassen sich dabei drei Szenarios unterscheiden:

1) Es gibt einen zentralen TA-Manager, der vollständige Information hat und über die Vergabe der TA entscheidet.

2) Es gibt einen zentralen TA-Manager, der unvollständige Information hat, aber über die Vergabe der TA entscheidet. Er holt sich die fehlenden Informationen durch Nachfrage bei den FTF.

3) Mehrere (dezentrale) TA-Manager haben unvollständige Information.

In jedem der Szenarios kann es noch

a) nachfrageinduzierte TA und

b) angebotsinduzierte TA

geben. Dargestellt werden im Folgenden Diagramme zu den Fällen 1a), 1b), sowie für 2a) und 3a). Diagramme zu 2b) und 3b) unterscheiden sich nur in der Auftragsvergabe, so dass sie weggelassen werden.

Der Fall 1a) stellt die Auftragsvergabe eines nachfrageinduzierten TA durch einen zentralen TA-Manager mit vollständiger Information dar, d.h. der TA-Manager weiß insbesondere zu jedem Zeitpunkt, wo sich welches FTF gerade befindet. Abbildung 18 zeigt den Fall 1a) als Sequenzdiagramm in UML. Ein TS-Konsument stellt fest, dass er ein TS benötigt (z.B. kann der Eingangspuffer einer Maschine leer sein) und erteilt deshalb dem TA-Manager einen BA. Dieser generiert einen TA, ermittelt den Startort für den TA und dann das beste FTF für diesen Startort. Diesem FTF erteilt er dann den TA, woraufhin dieses ihn ausführt und seine Erledigung an den TA-Manager meldet.

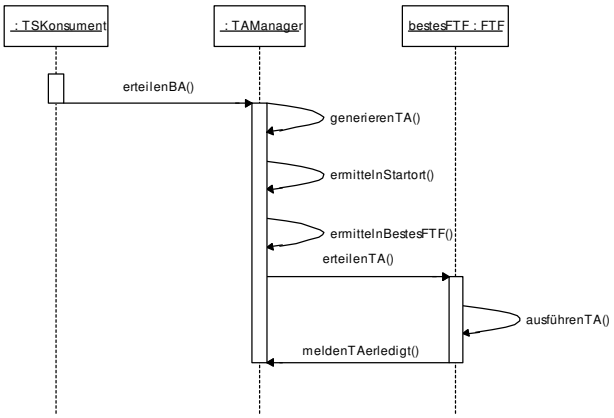

Abbildung 18: UML-Sequenzdiagramm für den Fall 1a: Vergabe eines nachfrageinduzierten Transportauftrages durch einen zentralen TA-Manager mit vollständiger Information

Der Fall 1b) zeigt die Auftragsvergabe eines angebotsinduzierten TA durch einen zentralen TA-Manager mit vollständiger Information. Abbildung 19 zeigt den Fall 1b) als Sequenzdiagramm in UML. Ein TS-Produzent stellt fest, dass er ein TS abzugeben hat (z.B. kann der Ausgangspuffer einer Maschine voll sein) und erteilt deshalb dem TA-Manager einen AA. Dieser generiert einen TA, ermittelt den Zielort für den TA und dann das beste FTF.[56] Dem ausgewählten FTF ordnet er dann auch hier den TA zu, worauf dieses ihn ausführt und seine Erledigung an den TA-Manager meldet.

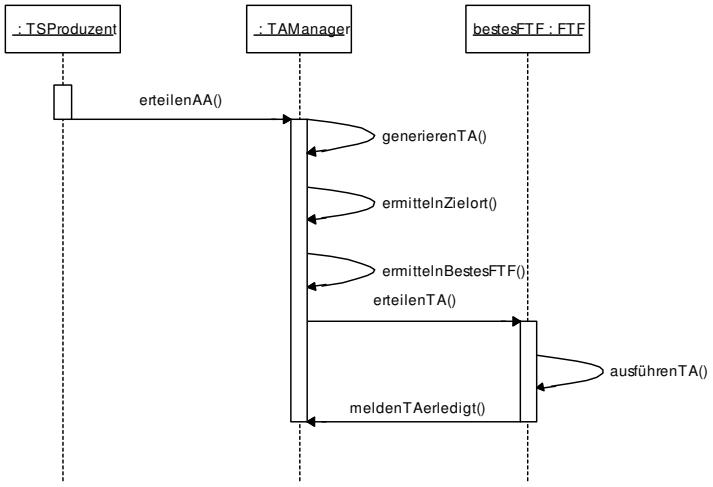

Abbildung 19: UML-Sequenzdiagramm Fall 1b: Vergabe eines angebotsinduzierten Transportauftrages durch einen zentralen TA-Manager mit vollständiger Information

Der Fall 2) beschreibt die Auftragsvergabe eines TA durch einen zentralen TA-Manager mit unvollständiger Information. Abbildung 20 zeigt den Fall 2a) als Sequenzdiagramm in UML. Fall 2b) unterscheidet sich nur in der Auftragsentstehung und ist deshalb nicht dargestellt. Nach dem Generieren des TA fügt der TA-Manager den TA in einen zentralen Pool von TA[57]. Die FTF erfahren

[56] Anders als im Fall 1a) ist die Wahl des FTF i.a. unabhängig von der Wahl des Zielorts, weil das FTF ja zunächst zum Startort fahren muss und somit eher räumliche Nähe zum Startort für die Wahl des FTF von Bedeutung ist.

[57] Die Trennung von TA-Manager und Pool erfolgt unter objektorientierten Gesichtspunkten, wobei der Pool ein Objekt ist, das TA verwaltet. Agentenorientiert wäre es sinnvoller, TA-Manager und Pool zu verschmelzen.

vom neuen TA, indem sie vom Pool informiert werden[58]. FTF1 und FTF2 beschließen daraufhin, sich für den neuen TA zu bewerben, während FTF3 sich nicht bewirbt (weil es z.B. gerade belegt oder aufgrund eines Fehlers kurzfristig außer Betrieb ist). Nach Ablauf der Bewerbungsfrist geht der TA-Pool davon aus, dass keine weiteren Bewerbungen eingehen. Er vergleicht die beiden Angebote, ermittelt FTF2 als bestes FTF und erteilt ihm dann den TA, worauf dieses ihn ausführt und seine Erledigung an den TA-Manager meldet.

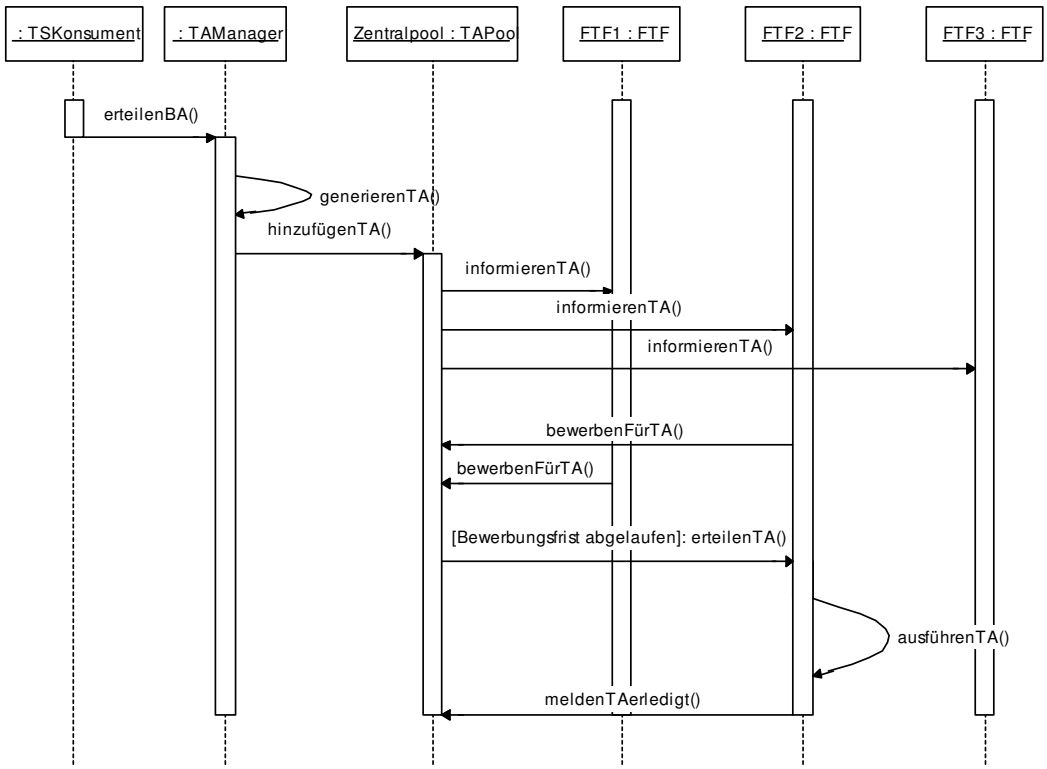

Abbildung 20: UML-Sequenzdiagramm für den Fall 2: Vergabe eines Transportauftrages durch einen zentralen TA-Manager mit unvollständiger Information

Der Fall 3) stellt die Auftragsvergabe durch mehrere dezentrale TA-Manager mit unvollständiger Information dar. Abbildung 21 zeigt den Fall 3a) als Sequenzdiagramm in UML. Fall 3b) unterscheidet sich nur in der Auftragsentstehung und ist deshalb nicht dargestellt. Die TS-Konsumenten K1 und K2 erteilen den TA-Managern M1 bzw. M2 einen BA. Diese generieren jeweils einen TA und fügen ihn den Pools P1 bzw. P2 hinzu. Denkbar ist hier, dass jeder TS-Konsument bzw. TS-Produzent seinen eigenen TA-Manager hat oder, dass sich mehrere einen TA-Manager teilen[59]. Auch hier erfahren die FTF von den neuen TA, indem sie von den Pools informiert werden. FTF1 entschließt sich, sich sowohl für den Auftrag in P1 als auch für den in P2 zu bewerben, während FTF2 sich nur für den Auftrag in P2 und FTF3 nur für den Auftrag in P1 bewirbt. P1 vergleicht die Angebote von FTF1 und FTF3 und ordnet FTF1 dann den TA zu, worauf dieses ihn auszuführen beginnt. P2 vergleicht die Angebote von FTF1 und FTF2 und ordnet FTF1 dann den TA zu. Da FTF1 aber nun schon einen anderen TA übernommen hat, lehnt es den TA aus P2 ab. Daraufhin ordnet P2 dann FTF2 den Auftrag zuordnet, worauf dieses ihn auszuführen beginnt. Am Ende melden FTF1 und FTF2 die Erledigung ihrer jeweiligen TA.

---

[58] Eine weitere Möglichkeit wäre, dass die FTF periodisch beim Pool nachfragen (im Diagramm nicht dargestellt).

[59] Auf jeden Fall sollte es sich bei der Zuordnung von TA-Managern zu TS-Konsumenten bzw. TS-Produzenten nicht um feste, sondern lediglich um bevorzugte TA-Manager handeln. Sollte ein TA-Manager gerade überlastet sein oder ausfallen, so können die TS-Konsumenten und TS-Produzenten auch einem anderen TA-Manager ihren BA bzw. AA erteilen. Dadurch wird die Robustheit des Systems erhöht.

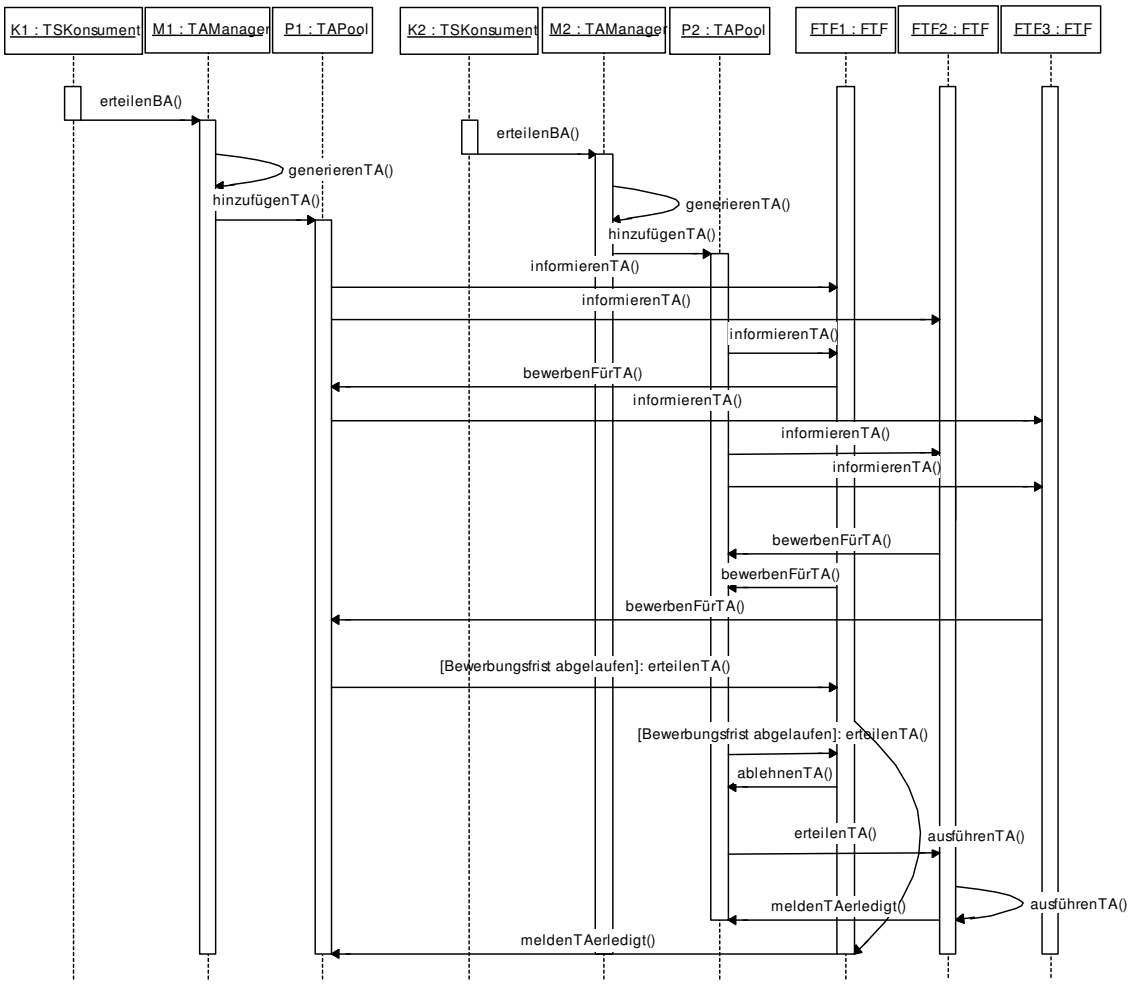

Abbildung 21: UML-Sequenzdiagramm für den Fall 3: Vergabe eines Transportauftrages durch mehrere dezentrale TA-Manager mit unvollständiger Information

5.2 Agentenorientierte Erweiterungen in AUML

AUML [OPB00] ist eine agentenorientierte Erweiterung von UML. Bei Sequenzdiagrammen werden in AUML keine Methodenaufrufe sondern Kommunikationsakte modelliert, die man grafisch daran erkennt, dass die Pfeilspitzen offen sind. Sind mehrere Kommunikationsakte durch einen senkrechten dicken Balken verbunden, so heißt dies, dass sie gleichzeitig stattfinden (AND). Sind sie durch einen schmalen Balken verbunden, der in der Mitte eine Raute enthält, so kann eine beliebige Teilmenge der Kommunikationsakte nebenläufig abgeschickt werden (OR). Enthält die Raute zusätzlich ein Kreuz, so muss genau einer der Kommunikationsakte abgeschickt werden (XOR). Abbildung 22 zeigt die drei Darstellungsmöglichkeiten nebenläufiger Kommunikation grafisch.

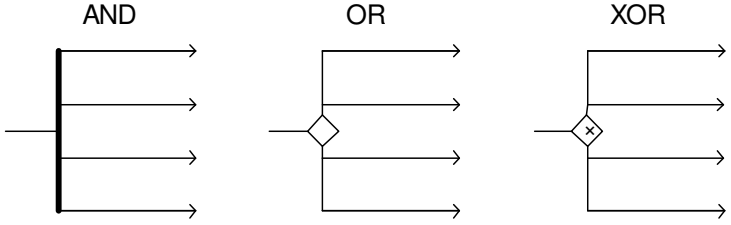

Abbildung 22: Nebenläufige Kommunikation in AUML-Sequenzdiagrammen

Eine Variante des Ablaufs in Fall 2) aus Abschnitt 5.1.7 könnte wie folgt aussehen. Der Zentralmanager schickt an die vier FTF gleichzeitig ein *cfp* (call for proposal, Bitte um Bewerbungen). FTF1 und FTF2 senden beide ein *propose* (Bewerbung), während FTF3 ein *refuse* (Ablehnung)

sendet. FTF4 ist entweder defekt oder schafft es nicht rechtzeitig, eine Bewerbung abzusenden, so dass es keine Nachricht sendet. Nach dem Ablauf der Bewerbungsfrist vergleicht der Zentralmanager die beiden Angebote von FTF1 und FTF2, sendet FTF1 ein *reject-proposal* (Ablehnung der Bewerbung) und FTF2 ein *accept-proposal* (Annahme der Bewerbung). FTF2 führt den TA aus und sendet dem Zentralmanager ein *inform-done* (Erledigungsmeldung). Die Namen der Nachrichtentypen entsprechen dem FIPA Contract-Net-Protokoll [FIPA00]. Abbildung 23 zeigt den Fall 2) als Sequenzdiagramm in AUML. Anders als im entsprechenden UML-Sequenzdiagramm (Abbildung 20) hat man in AUML den Vorteil, dass man darstellen kann, dass die Bitte um Bewerbungen gleichzeitig an alle FTF gesendet wird und dass die Bewerbungsannahme und die Bewerbungsablehnung ebenfalls gleichzeitig an die sich bewerbenden FTF versendet wird. In UML hat man nur die Möglichkeit, Nachrichten darzustellen, die nacheinander verschickt werden.

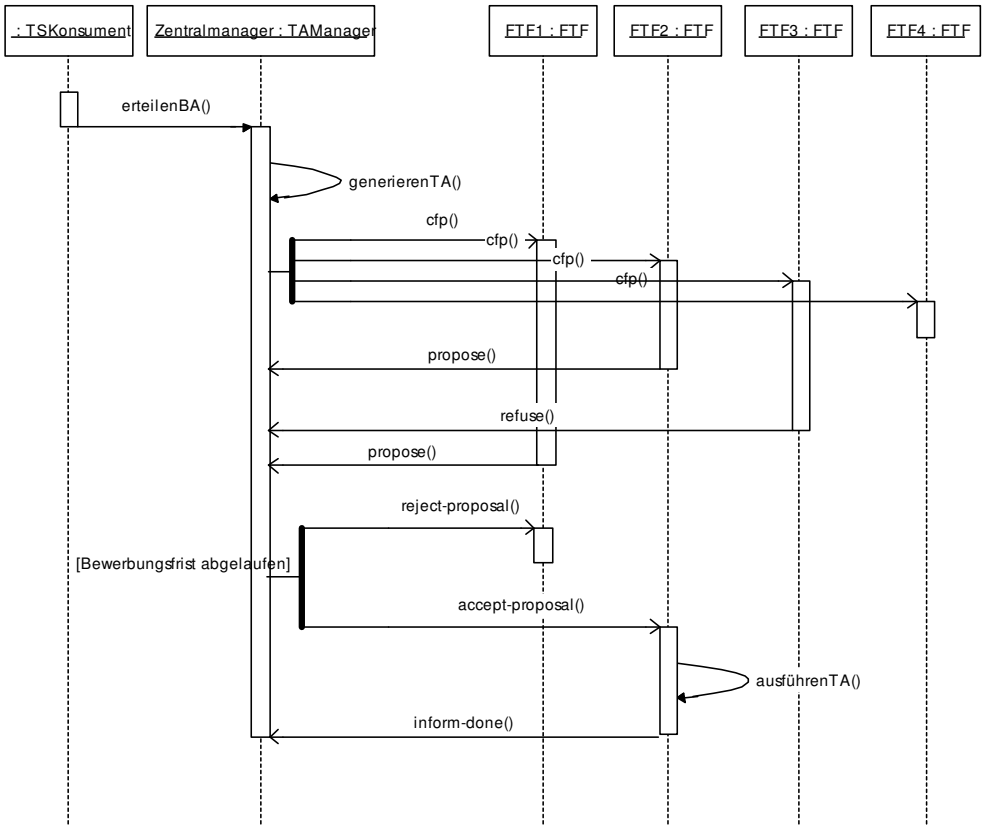

Abbildung 23: AUML-Sequenzdiagramm für den Fall 2: Vergabe eines Transportauftrages durch einen zentralen TA-Manager mit unvollständiger Information

Anders als im FIPA Contract-Net-Protokoll ist in Fall 3) aus Abschnitt 5.1.7 die Einreichung einer Bewerbung für den Bewerber nicht bindend. Deshalb wird eine Erweiterung vorgeschlagen, die dem Bewerber die Möglichkeit gibt, auf den Zuschlag zu verzichten. Dies ermöglicht Bewerbern, sich bei mehreren Anbietern, deren Zuschlag sich gegenseitig ausschließt, zu bewerben.

Abbildung 24 zeigt das Original FIPA Contract-Net-Protokoll und die vorgeschlagene Erweiterung. Neu sind die Kommunikationsakte *accept-order* (Annahme der Bewerbungsannahme) und *reject-order* (Ablehnung der Bewerbungsannahme).

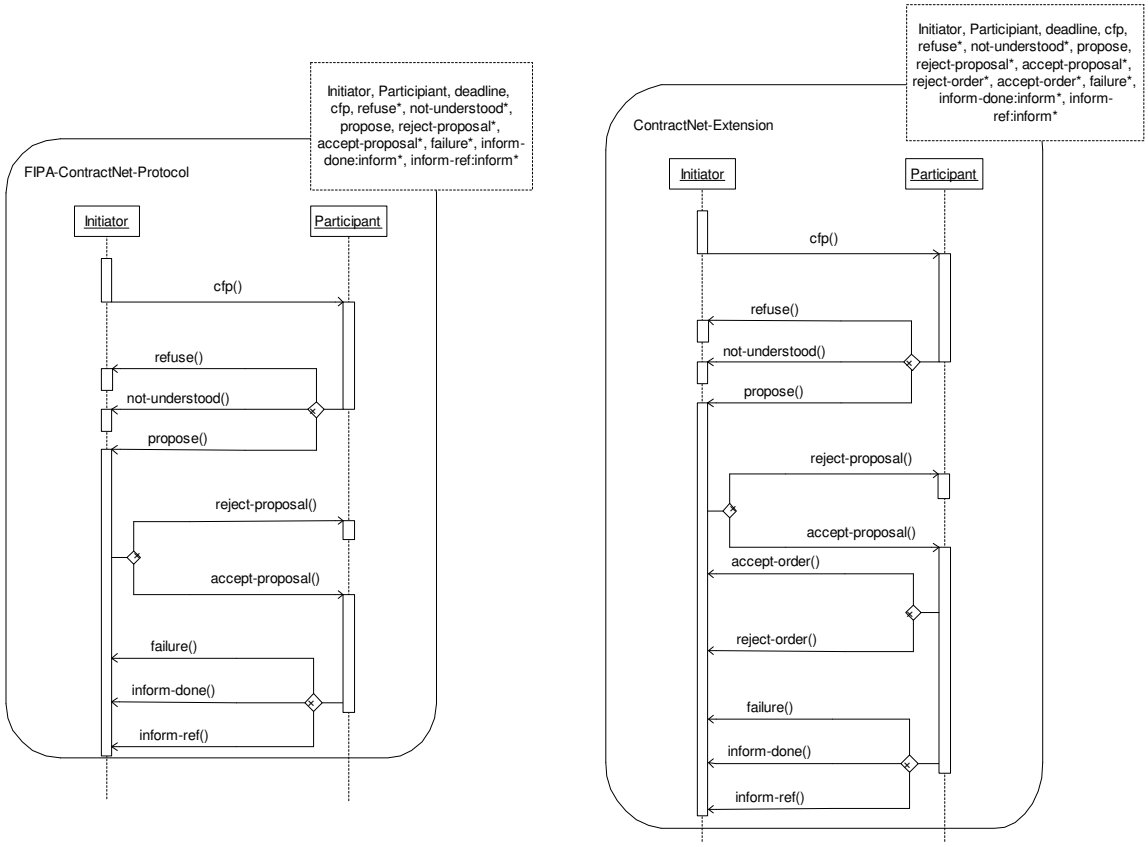

Abbildung 24: FIPA Contract-Net-Protokoll und vorgeschlagene Erweiterung

Abbildung 25 zeigt eine Variante des Ablaufs in Fall 3 aus Abschnitt 5.1.7 als Sequenzdiagramm in AUML. M1 schickt an FTF1, FTF2 und FTF3 gleichzeitig ein *cfp*, woraufhin sich FTF1 und FTF3 mittels *propose* bewerben. M2 sendet an FTF1 und FTF2 ebenfalls gleichzeitig ein *cfp*. FTF3 erhält von M2 kein *cfp*, da es dort nicht an Aufträgen interessiert ist und sich deswegen nicht angemeldet hat.[60] FTF2 entschließt sich nun, sich bei M2 zu bewerben und bei M1 sich nicht zu bewerben, sondern ihm ein *reject* zu schicken. FTF1, das sich schon bei M1 beworben hat, bewirbt sich nun auch bei M2. M1 vergleicht die beiden Angebote von FTF1 und FTF3 und sendet FTF1 ein *accept-proposal*. FTF3 erhält noch kein *reject-proposal*, da ja noch nicht klar ist, ob FTF1 den Auftrag annimmt. Nachdem M1 von FTF1 das *accept-order* (Annahme der Bewerbungsannahme) erhalten hat, sendet er ein *reject-proposal* an FTF3. M1 vergleicht die beiden Angebote von FTF1 und FTF2 und sendet FTF1 ein *accept-proposal*. Da dieses aber nun schon einen anderen Auftrag angenommen hat, sendet es ein *reject-order* (Ablehnung der Bewerbungsannahme), woraufhin M2 nun FTF2 den Zuschlag mittels *accept-proposal* erteilt. FTF2 nimmt mittels *accept-order* an. FTF1 und FTF2 führen den jeweiligen TA aus und senden dem entsprechenden TA-Manager ein *inform-done*. Auch hier hat man bei AUML gegenüber dem entsprechenden UML-Diagramm (Abbildung 21) den Vorteil, dass man die Gleichzeitigkeit der Bitte um Bewerbungen ausdrücken kann.

[60] FTF können sich dynamisch bei TA-Managern zur Auftragsvergabe an- und abmelden (im Diagramm nicht dargestellt).

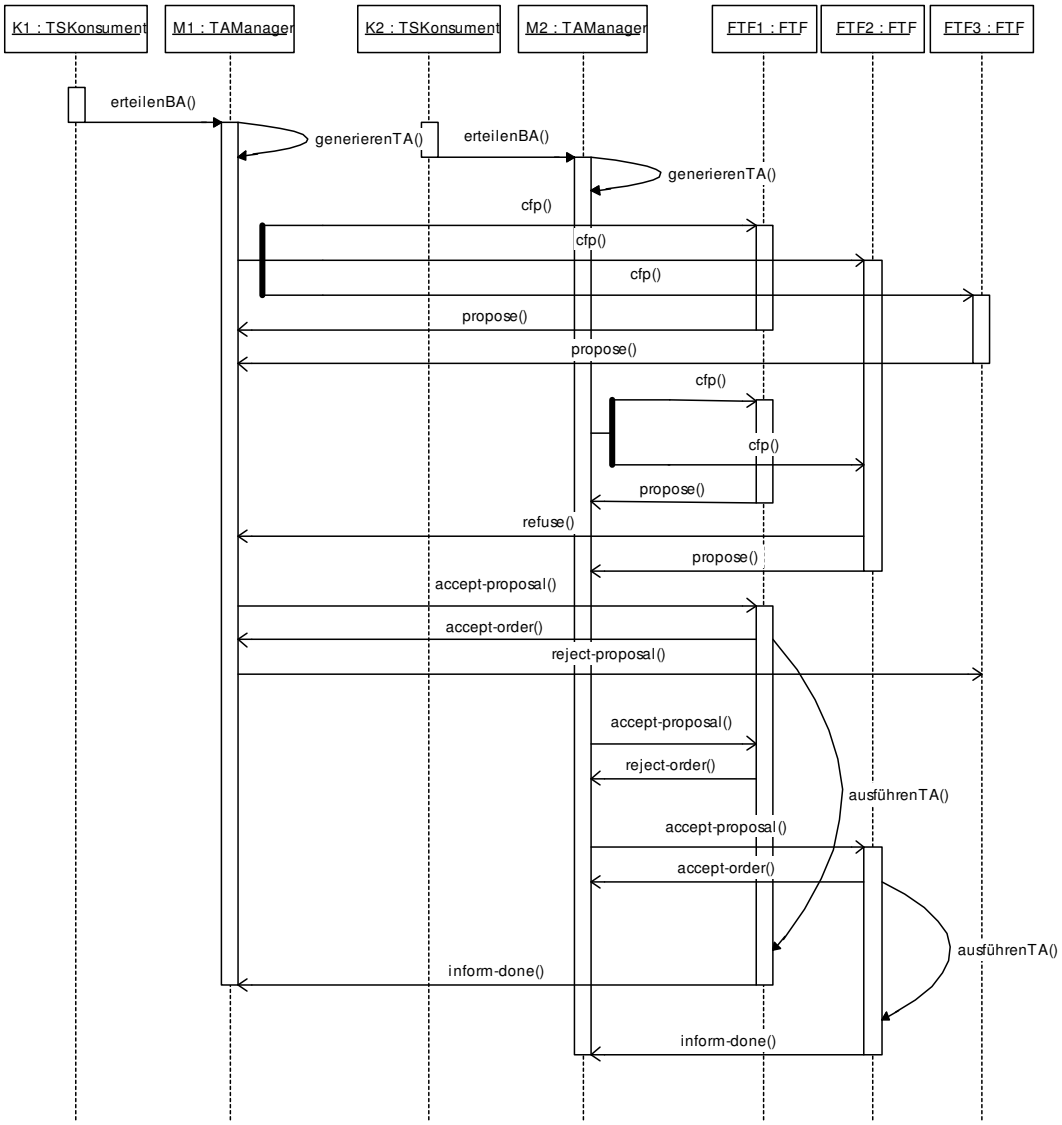

Abbildung 25: AUML-Sequenzdiagramm für den Fall 3: Vergabe eines Transportauftrages durch mehrere dezentrale TA-Manager mit unvollständiger Information

5.3 Agentenorientierte Erweiterungen in AORML

In der von Wagner [Wag03] vorgeschlagenen, auf UML basierenden Agenten-Objekte-Relationen-Modellierungssprache *AORML* ist eine Entität ein Agent, ein Ereignis, eine Verpflichtung / ein Anspruch oder ein gewöhnliches Objekt. Spezielle Relationen zwischen Entitätstypen unterschiedlicher Kategorien, wie z.B. die Relationen *send* und *receive* zwischen Agententypen und Kommunikationsereignis- oder Nachrichtentypen, ergänzen die aus der ER-Modellierung stammenden fundamentalen Relationen *Assoziation*, *Aggregation/Komposition* und *Generalisierung*. In AORML wird das Verhalten eines Agenten bzw. Systems, regelbasiert (mit Hilfe von *Reaktionsregeln*) spezifiziert.

Da in AORML zwischen Agenten und Objekten unterschieden wird, ist es zunächst notwendig zu untersuchen, bei welchen Klassen aus Abschnitt 5.1 es sich um Agententypen und bei welchen um Objekttypen handelt. In der grafischen Notation sind Agenten- und Objekttypen Rechtecke, wobei bei den Agententypen die Ecken abgerundet sind.

Agententypen sind *FTF* und *Maschine* jeweils mit allen Unterklassen. Die Objektklassen *TAManager* und *TAPool* aus dem objektorientierten Modell fallen zu einem weiteren Agententypen, ebenfalls *TAManager* genannt, zusammen.

Objekttypen sind *Ort* (einschließlich aller Unterklassen, also auch *Übergabestation* und *Knoten*), *Fahrkurs*, *Fahrkurselement* (einschließlich Unterklassen), *Transportstück* und *Transportauftrag*.

Ereignistypen sind *erteilenAA*, *erteilenBA*, *ausführenTA*, sowie die in dem erweiterten Contract-Net-Protokoll aus Abbildung 24 aufgeführten Nachrichtentypen. Wenn ein FTF eine Nachricht vom Typ *accept-order* verschickt, geht es eine Verpflichtung vom Typ *ausführenTA* ein, d.h. eine Verpflichtung, das Handlungsereignis *ausführenTA* zu kreieren.

Man unterscheidet zwischen externen AOR-Modellen, die eine Sicht auf mehrere Agenten von außen zeigen und internen AOR-Modellen, die die Sicht eines einzelnen Agenten darstellen. Im Folgenden werden nur externe AOR-Modelle vorgestellt.

Agententypen, Objekttypen und die Relationen, in denen sie zueinander stehen, kann man in einem *Agentendiagramm* darstellen. Objekttypen-Rechtecke werden innerhalb eines Agententypen-Rechtecks gezeigt, wenn es sich um interne Informationen der Agenten dieses Typs handelt. Handelt es sich jedoch um Informationen, die auch für andere Agenten zugänglich ist, werden die entsprechenden Objekttypen-Rechtecke als eigenständige Elemente außerhalb eines Agententypen-Rechtecks gezeigt.

Gemäss dem in Abbildung 26 gezeigten Agentendiagramm verwaltet der TAManager beliebig viele Transportaufträge. Das FTF hat zu jedem Zeitpunkt keinen oder einen Transportauftrag und transportiert unbestimmt viele Transportstücke (im Falle des Single-Load-Carriers höchstens ein Transportstück). Eine Fertigungsmaschine hat Eingangs- und Ausgangspuffer. In den Puffern können sich mehrere Transportstücke befinden.

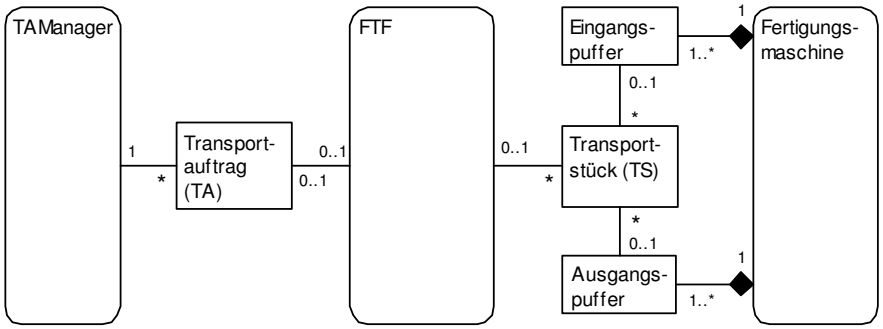

Abbildung 26: AOR-Agentendiagramm

Ein *Interaktionsrahmendiagramm* zeigt die möglichen Aktionsereignistypen und Anspruchs-/ Verpflichtungstypen, die zwischen zwei Agententypen entstehen können.

Zwischen einem FTF und dem zentralen TAManager sieht der Interaktionsrahmen wie folgt aus. Der TAManager kann dem FTF die Nachrichten *informierenTA* und *erteilenTA* senden, während das FTF dem TAManager die Nachrichten *bewerbenFürTA* und *meldenTAerledigt* senden kann. Das FTF kann die Verpflichtung *ausführenTA* gegenüber dem TAManager haben, die es durch die ausgeführte Aktion *ausführenTA* erfüllen kann. Im Falle mehrerer dezentraler TAManager wären noch weitere Nachrichten denkbar. In Abbildung 27 sind mehrere Interaktionsrahmendiagramme zu einem zusammengefasst.

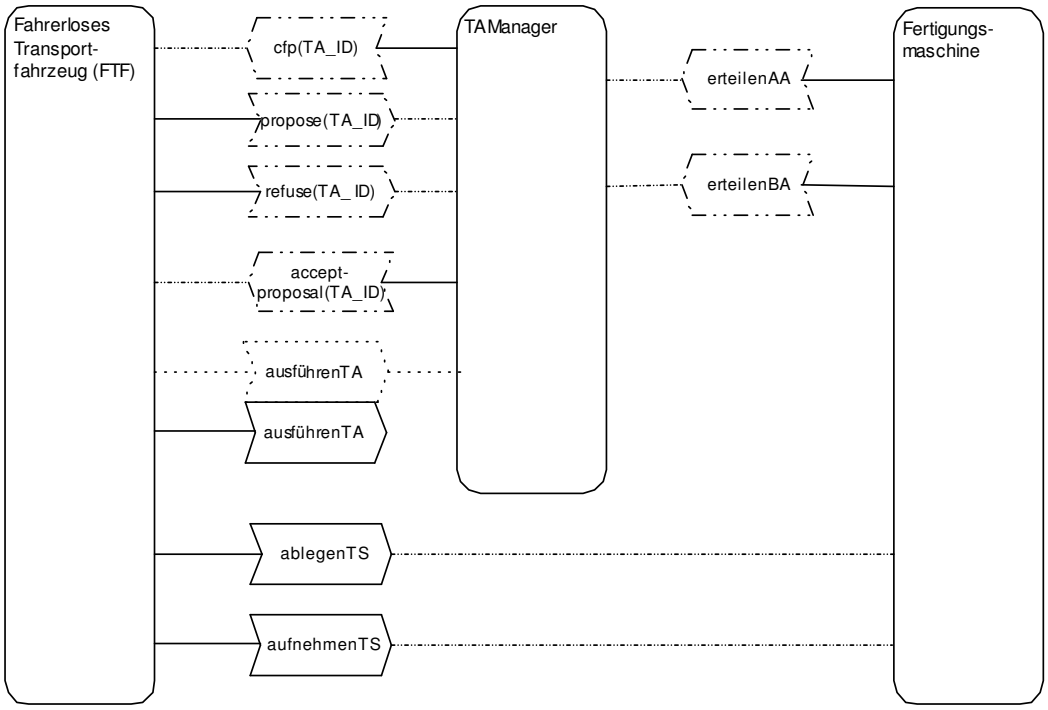

Abbildung 27: AOR-Interaktionsrahmendiagramm

Ein *Interaktionsmusterdiagramm* zeigt allgemeine Interaktionsmuster in Form einer Menge von Reaktionsregeln. Im Falle mehrerer dezentraler Transportauftragsmanager (im Diagramm nur einer beteiligt) könnte ein Interaktionsmusterdiagramm wie in Abbildung 28 aussehen.

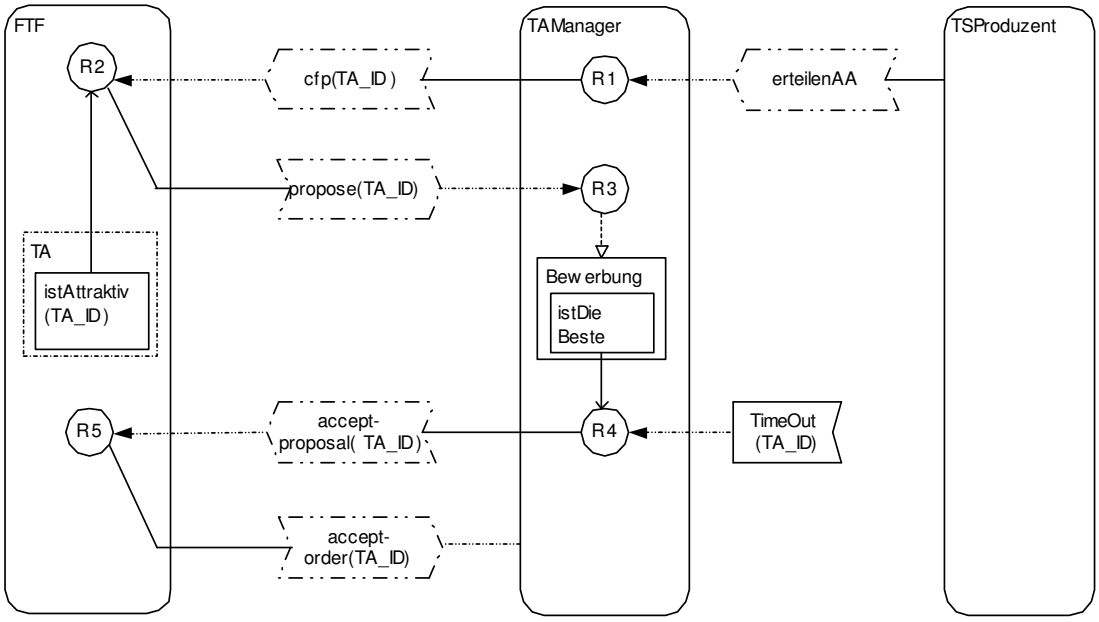

Abbildung 28: AOR-Interaktionsmusterdiagramm

Die Bezeichnungen der Nachrichtentypen sind hier wie in Abschnitt 5.2 an das FIPA Contract-Net-Protokoll sowie die vorgeschlagene Erweiterung angelehnt.

Reaktionsregel R1 ist die Regel, die angestoßen wird, wenn ein *TAManager* ein *erteilenAA* erhält. Er generiert einen TA und sendet an alle bei ihm angemeldeten FTF ein *cfp*.

Wenn ein *FTF* ein *cfp* erhält (Reaktionsregel R2), so wertet es aus, ob der TA attraktiv ist[61], indem es die Funktion/Methode TA.istAttraktiv(TA_ID) aufruft. Ist die Antwort ja, so antwortet es mit einem *propose*. Anderenfalls sendet es ein *refuse* (im Diagramm nicht dargestellt).

Erhält der *TAManager* ein *propose* (Reaktionsregel R3), so speichert er die Bewerbung in seiner Bewerbungstabelle.

Wenn der *TAManager* eine Nachricht *TimeOut* erhält (Reaktionsregel R4), so vergleicht er alle in seiner Tabelle gespeicherten Bewerbungen und sendet an das *FTF*, das die beste Bewerbung abgegeben hat, ein *accept-proposal*.

Der Empfang von *accept-proposal* löst bei einem FTF Reaktionsregel R5 aus. Wenn das FTF momentan keinen zugewiesenen TA hat, sendet es *accept-order* an den TAManager. Hat es einen TA, etwa weil es zwischen dem Senden von *propose* und dem Empfang von *accept-proposal* von einem anderen TAManager ein *accept-proposal* erhalten hat, so sendet es ein *reject-order* an den TAManager (im Diagramm nicht dargestellt).

Im Diagramm nicht dargestellt ist Reaktionsregel R6. Empfängt der TAManager ein *accept-order*, so sendet er an alle FTF, die sich ebenfalls für den TA beworben haben, ein *reject-proposal*.

Ebenfalls im Diagramm nicht dargestellt ist Reaktionsregel R7. Empfängt der TAManager ein *reject-order*, so sendet er an das FTF mit der nächstbesten Bewerbung ein *accept-proposal*.

Man kann Reaktionsregeln auch in einer formalen Regelspezifikationssprache darstellen. Im folgenden Beispiel wird R2 dargestellt, wobei Variablen als Präfix ein Fragezeichen vorangestellt wird.

ON RECEIVE cfp(?TA_ID) FROM ?TAManager

IF TA.istAttraktiv(?TA_ID)

THEN SEND propose(?TA_ID) TO ?TAManager

5.4 Agentenorientierte Erweiterungen in Gaia

Gaia [WJK00] ist eine Methode für agentenorientierte Analyse und agentenorientierten Entwurf. Im Rahmen der Modellierung wird hier nur auf die Analyse eingegangen. In der Analyse werden die am System beteiligten Rollen in sogenannten Rollenschemata und deren Interaktionen in Form von Protokollen modelliert.

Rollenschemata beinhalten eine kurze Beschreibung in natürlicher Sprache, eine Angabe der Protokolle und Aktivitäten, Angabe der Rechte sowie der Verantwortlichkeiten in Form eines Lebenszyklus oder in Form von Sicherheiten[62]. Aktivitäten (unterstrichen dargestellt) benötigen im Gegensatz zu Protokollen keine Interaktion mit anderen Rollen. Rechte beziehen sich auf den Zugriff auf sich im System befindliche Ressourcen. Leserechte werden mit *reads*, Änderungsrechte (diese beinhalten immer auch Leserechte) mit *changes* und Erzeugungsrechte mit *generates* gekennzeichnet. Der Lebenszyklus besteht aus Kombination von Aktivitäten und Protokollen. Dabei bedeutet *x.y*, dass *x* vor *y* ausgeführt wird. Die Bedeutung von *x||y* ist, dass *x* und *y* überlappend ausgeführt werden. Schließlich bedeutet x^{ω}, dass *x* unendlich oft ausgeführt wird.

Als Rollen kommen die bereits in Abschnitt 5.3 erkannten Agententypen in Frage. Eine Fertigungsmaschine erfüllt zwei Rollen: die Rolle *TSProduzent* und die Rolle *TSKonsument*. Die Rollenschemata sehen wie folgt aus.

[61] Im einfachsten Fall ist ein TA genau dann attraktiv, wenn das FTF gerade keinen anderen TA zugewiesen bekommen hat.

[62] Sicherheiten werden hier aus Gründen der Einfachheit nicht modelliert. Damit das Schema von Gaia beibehalten wird, erscheinen sie trotzdem in den Tabellen.

Rollenschema	FTF
Beschreibung	Ein FTF führt TA aus, indem es TS vom Startort zum Zielort bringt. Um neue TA zu erhalten, bewirbt es sich bei einem TAManager.
Protokolle und Aktivitäten	zumStartortFahren, aufnehmenTS, zumZielortFahren, ablegenTS, fürTAbewerben
Rechte	reads TA changes TS
Lebenszyklus	((zumStartortFahren . aufnehmenTS . zumZielortFahren . ablegenTS) || fürTAbewerben) $^{\omega}$
Sicherheiten	

Rollenschema	TAManager
Beschreibung	Ein TAManager nimmt Anforderungen und Angebote für TS entgegen, generiert neue TA, schreibt sie aus und erteilt den Zuschlag.
Protokolle und Aktivitäten	TAgenerieren, TAausschreiben, BewerbungenBeurteilen, ZuschlagErteilen
Rechte	reads TS generates TA
Lebenszyklus	(TAgenerieren . TAausschreiben . BewerbungenBeurteilen . ZuschlagErteilen) $^{\omega}$
Sicherheiten	

Rollenschema	TSProduzent
Beschreibung	Ein TSProduzent stellt TS her und erteilt einem TAManager einen AA.
Protokolle und Aktivitäten	TSerzeugen, AAerteilen
Rechte	generates TS
Lebenszyklus	(TSerzeugen . AAerteilen) $^{\omega}$
Sicherheiten	

Rollenschema	TSKonsument
Beschreibung	Ein TSKonsument erteilt einem TAManager einen BA und konsumiert dann die erhaltenen TS.
Protokolle und Aktivitäten	BAerteilen, TSverbrauchen
Rechte	changes TS
Lebenszyklus	(BAerteilen . TSverbrauchen) $^{\omega}$
Sicherheiten	

Die Protokolle werden ebenfalls durch Schemata dargestellt. Dabei enthalten sie den Zweck des Protokolls in einer kurzen natürlich-sprachlichen Beschreibung, den(die) Initiator(en), den(die)

Antwortenden, die Eingaben, die Ausgaben und die Abarbeitung in einer kurzen natürlich-sprachlichen Beschreibung. Für die in den Rollenschemata genannten Protokolle sehen die Schemata wie folgt aus.

Protokollschema	fürTAbewerben
Zweck	ermöglicht eine Bewerbung für einen TA
Initiator	FTF
Antwortender	TAManager
Eingaben	TA, Bewerbung
Ausgaben	Zuschlag(=Bewerbungsannahme), Bewerbungsablehnung
Abarbeitung	Das FTF bewirbt sich bei einem TAManager um die Ausführung eines TA.

Protokollschema	TAausschreiben
Zweck	Ausschreibung für einen TA
Initiator	TAManager
Antwortender	FTF
Eingaben	TA
Ausgaben	Bewerbung
Abarbeitung	Der TAManager schreibt einen neuen TA aus. Dabei informiert er alle interessierten FTF über den neuen TA und erwartet ihre Bewerbungen.

Protokollschema	ZuschlagErteilen
Zweck	Zuschlag für einen TA
Initiator	TAManager
Antwortender	FTF
Eingaben	TA
Ausgaben	Zuschlagsannahme, Zuschlagsablehnung
Abarbeitung	Der TAManager erteilt den Zuschlag für einen TA, indem er das FTF, das den Zuschlag erhalten hat, benachrichtigt. Erhält er eine Zuschlagsannahme, so schickt er an alle anderen FTF eine Ablehnung, ansonsten erhält ein anderes FTF den Zuschlag.

Protokollschema	AAerteilen
Zweck	Erteilung eines Abholauftrags
Initiator	TSProduzent
Antwortender	TAManager
Eingaben	AA (damit Startort für TA), TS
Ausgaben	TA
Abarbeitung	Der TSProduzent erteilt dem TAManager einen AA und informiert ihn dadurch darüber, dass er TS produziert hat, die abgeholt und zu einem anderen Ort transportiert werden sollen. Der TAManager generiert daraufhin einen TA.

Protokollschema	BAerteilen
Zweck	Erteilung eines Beschaffungsauftrags
Initiator	TSKonsument
Antwortender	TAManager
Eingaben	BA (damit Zielort für TA)
Ausgaben	TA, TS
Abarbeitung	Der TSKonsument erteilt dem TAManager einen BA und informiert ihn dadurch darüber, dass er TS benötigt. Der TAManager generiert daraufhin einen TA.

5.5 Verwandte Arbeiten

Ein Überblick über FTS kann man sich bei Hollier [Hol87] verschaffen. Eine detaillierte Beschreibung von FTS liefert Schrecker [Schr00]. Bei Ebben [Ebb01] werden FTS sehr ausführlich anhand eines Fallbeispiels eines externen FTS diskutiert.

Verschiedene Optionen beim Design von Fahrkursen werden von Majety und Wang [MW95] diskutiert.

Bei Egbelu und Tanchoco [ET84] werden Auftragsvergabestrategien in die beiden Kategorien fahrzeuginitiiert und arbeitsplatzorientiert eingeteilt. Hwang und Kim [HK98] verwenden ein Auktionskonzept zur Auftragsvergabe. Bei van der Heijden et al. [vdH02] wird die Zuordnung von FTF zu Transportaufträgen für ein externes FTS untersucht, während Akturk und Yilmaz [AY96] dies für eine Produktions- oder Warenhausumgebung untersuchen. Balko [Bal00] untersucht ebenfalls verschiedene Auftragsvergabestrategien.

Das Problem der Positionierung leerer FTF ohne auszuführenden Transportauftrag wird bei Kim und Kim [KK97] behandelt.

Bei Seifert et al. [Sei98] werden mehrere dynamische Routing-Strategien mit Hilfe einer Simulationsstudie verglichen.

Heidemann [Hei94] untersucht die Steuerung von bidirektionalen Segmenten mit Hilfe von Ampelanlagen genauer. Die Steuerung von periodisch wechselnden Einbahnstraßen (das entspricht dem hier vorgestellten Konzept von bidirektionalen Segmenten) wird bei van der Heijden et al. [vdH01] untersucht.

McHaney [McH95] untersucht die Beschränkungen von FTF bei Verwendung von Batterien näher.

5.6 Abschließende Bemerkungen

FTS können als verteilte Objekt-Systeme oder als Multiagentensysteme modelliert werden. In jedem Fall ist die objektorientierte Modellierung der passiven Objekte eines FTS (Fahrkurs, Übergabestation, Transportstück, etc.) mit Hilfe von UML grundlegend. In diesem Kapitel wurden als mögliche Alternativen zu UML die agentenorientierten Methoden *AUML*, *AORML* und *Gaia* verwendet.

AUML stellt eine Erweiterung von UML dar, da Elemente hinzukommen, die die Ausdruckskraft der Modelle erhöhen, wie etwa der AND-, OR- und XOR-Split für ausgehende Nachrichten. AUML ist besonders zur Spezifikation von Interaktionsprotokollen geeignet. Die vorgestellten Erweiterungen sind nicht nur für Agentensysteme, sondern allgemein für Modellierung verteilter Systeme geeignet. Allerdings wird grafisch in AUML zwischen Agenten und Objekten nicht unterschieden, so dass sich die Frage stellt, ob man bei AUML wirklich von einem agentenorientierten Ansatz sprechen kann.

AORML erweitert die ontologischen Prinzipien von UML und unterscheidet im Gegensatz zu AUML zwischen den Klassen-Stereotypen <<Agent>> und <<Objekt>>. Es eignet sich besonders gut zur regelbasierten Verhaltensmodellierung und bietet Konstrukte zur Berücksichtigung von Verpflichtungen und Ansprüchen als soziale Kontrollelemente in der Verhaltensmodellierung. Durch die flächenhafte Darstellung von Nachrichten im Stil von Klassen-Rechtecken kann es vorkommen, dass AOR-Sequenzdiagramme mehr Platz beanspruchen als die entsprechenden AUML-Sequenzdiagramme, was in ungünstigen Fällen zu unübersichtlichen Diagrammen führen kann.

Gaia erscheint als nicht-grafische Notation eher ungeeignet für die Modellierung. Außerdem lassen sich viele Sachverhalte nicht in die streng vorgegebenen Schemata pressen, so dass Genauigkeitsverluste auftreten. So lässt sich beispielsweise nur verbal beschreiben, welche weiteren Protokolle durch ein Protokoll angestoßen werden. Auch Verhalten in Abhängigkeit von Bedingungen lässt sich nicht formal, sondern nur natürlich-sprachlich beschreiben. Des Weiteren ist nachteilig, dass Gaia nicht auf UML basiert und deshalb keine guten Aussichten hat, sich in der Praxis zu etablieren. Vorteil von Gaia ist, dass man ohne Werkzeugunterstützung auskommt.

Alles in allem empfiehlt sich deshalb auf Grundlage der Erfahrung bei der Modellierung Fahrerloser Transportsysteme eine kombinierte Modellierung in UML (einschließlich der mit AUML bezeichneten Erweiterungen) und in AORML.