Adaption von Auftragsvergabestrategien in der Referenzfallstudie Produktionstechnik

Sören Balko

Otto-von-Guericke-Universität Magdeburg
Institut für Technische und Betriebliche Informationssysteme
Postfach 4120, D–39016 Magdeburg, Germany
balko@iti.cs.uni-magdeburg.de

1. Mai 2000

Zusammenfassung

Gestiegene Anforderungen an Sicherheit und Zuverlässigkeit haben auch in ingenieurwissenschaftlichen Anwendungsbereichen zu einem verstärkten Einsatz von formalen Spezifikationstechniken geführt. Die Spezifikation dieser technischen Informationssysteme schließt sich entsprechend dem klassischen Softwarelebenszyklus einer Anforderungsanalyse an und bildet die Grundlage für die spätere Implementierung. Diese klassische Vorgehensweise weist insbesondere in bezug auf langlebige Informationssysteme einige Nachteile auf. Typischerweise treten mit steigender Lebensdauer von technischen Systemen in zunehmendem Maße veränderte Randbedingungen auf, die eine entsprechende Anpassung des Systems erfordern. Diese Anpassungen betreffen aber häufig nur stark eingegrenzte Bereiche des Systems und verändern nicht dessen Gesamtcharakteristik.

Treten solche Änderungen im spezifizierten System auf, so erforderte dies bislang eine erneute Spezifikation des gesamten Systems. Daraus ergibt sich ein hoher Aufwand zur Wartung dieser Systeme. Alternativ könnten Techniken zur *Adaption* formaler Spezifikationen eingesetzt werden. Darunter verstehen wir eine evolutionäre Anpassung formaler Spezifikationen an geänderte Rahmenbedingungen. Klassische Spezifikationstechniken weisen bislang keine Merkmale für Spezifikationsevolutionen auf, die es gestatten würden, festgelegte Teile der Spezifikation zu verändern ohne dabei Anforderungen wie Terminierbarkeit und Beweisbarkeit zu verletzen. Dieser Nachteil bildet die Motivation für das Projekt *Semantik adaptiver Workflows* (SAW). Im

Rahmen dieses Projekts entwickeln wir Vorschläge für die Realisierung adaptiver Spezifikationen. Als Teil des DFG Schwerpunktprogramms "Spezifikationstechnik" bearbeiten wir eine Referenzfallstudie, die eine Umsetzung dezentraliert gesteuerten Materialflusses zum Gegenstand hat.

In Kapitel 1 stellen wir die Referenzfallstudie "Holonischer Materialfluß" vor. Daran anschließend präsentieren wir in Kapitel 2 eine (statische) Spezifikation in der Spezifikationssprache OmTroll. In Kapitel 3 diskutieren wir einige ausgewählte Adaptionsansätze. Kapitel 4 beschäftigt sich mit der Umsetzung dieser Vorschläge in eine adaptive Spezifikation. In Kapitel 5 legen wir den aktuellen Projektstand dar und geben einen Ausblick auf ausstehende Arbeiten.

1 Holonischer Materialfluß

Im weiteren Verlauf greifen wir zur Illustration unseres Ansatzes auf eine durchgehende Referenzfallstudie zurck. In einem stark vereinfachten Szenario sollen Werkstücke sequentiell in drei Arbeitsschritten bearbeitet werden. In der Grundstufe soll jeder dieser Arbeitsschritte jeweils von genau einer Maschine durchgeführt werden. Wir bezeichnen diese Maschinen im folgenden vereinfachend mit M1, M2 und M3. Jede dieser Maschinen verfügt neben dem Bearbeitungsplatz weiterhin über lokale Vor- und Nachpuffer, die eine begrenzte Anzahl von Werkstücken aufnehmen können. Die vollständig unbearbeiteten Werkstücke werden in einem Eingangslager (EIN) zur Verfügung gestellt. Zunächst können wir von einer unbegrenzten Kapazität des Eingangslagers ausgehen. Die komplett bearbeiteten Werkstücke werden in einem Ausgangslager (AUS) abgelegt. Wir gehen zunächst davon aus, daß in EIN jederzeit eine unbegrenzte Anzahl neuer Werkstücke vorhanden ist und AUS beliebig viele bearbeitete Werkstücke aufnehmen kann. Die Bestückung von EIN sowie die Entleerung von AUS sind nicht Gegenstand dieser Referenzfallstudie. Da keine feste Verbindung zwischen den Maschinen besteht, wird der Transport der Werkstücke zwischen den Maschinen sowie zwischen dem Eingangs-/Ausgangslager und den Maschinen durch *Holonische Transportsysteme* (HTS) gewährleistet. Es handelt sich dabei um freifahrende Transportroboter. Die HTS bewegen sich frei in dem Szenario und unterliegen keiner zentralen Steuerung. Sie besitzen also eine hohe Autonomität. Die Kommunikation zwischen Maschinen und HTS erfolgt über Funk. Störungen in der Kommunikation sowie Totalausfälle einzelner HTS sind in der Grundstufe nicht vorgesehen und müssen nicht berücksichtigt werden. In Abbildung 1 ist das Szenario abstahiert dargestellt. Um die Anlieferung (Abholung) eines Werkstücks zu (von) einer Maschine auszulösen, müssen Transportjobs

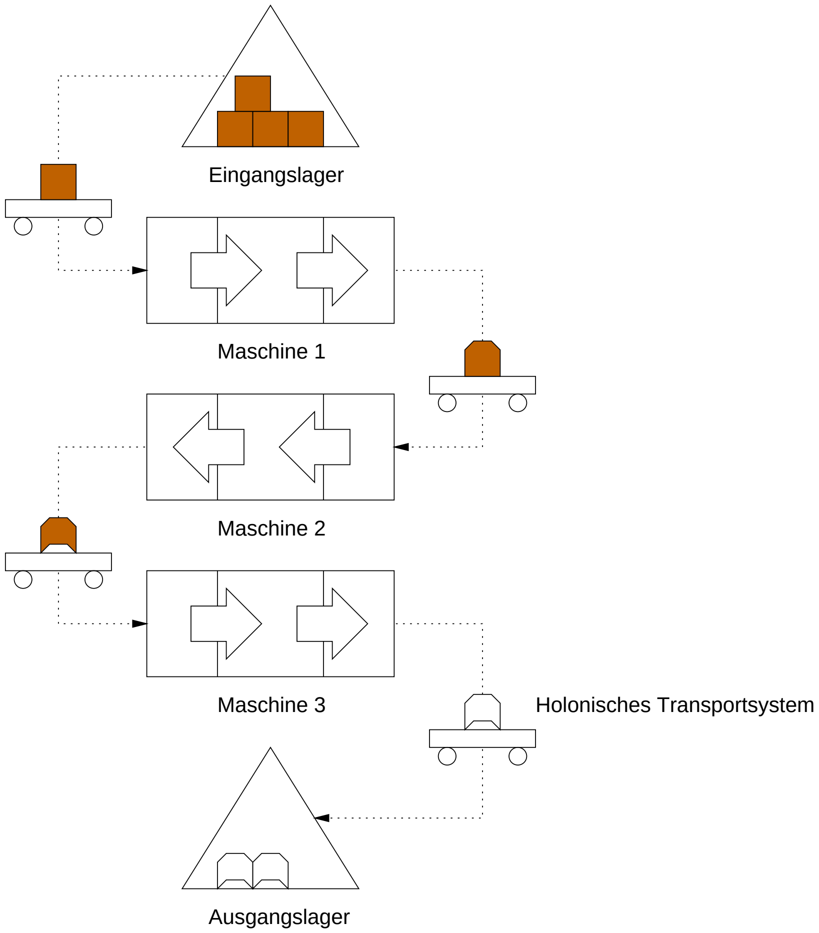

Abbildung 1: Referenzfallstudie "Holonischer Materialfluß"

in Auftrag gegeben werden. Dementsprechend ist zwischen Anforderungs- (DEMAND) und Abholungsjobs (OFFER) zu unterscheiden. Transportjobs werden von HTS ausgeführt. Die Maschinen agieren als die Initiatoren für die Transportaufträge, entscheiden jedoch nicht über die Ausführunsparameter (Reihenfolge, Zeitpunkt, Zuordnung). Ebenso sind in Ausbaustufen dieses Szenarios andere HTS als Auftraggeber für die Transportjobs denkbar, beispielsweise zur Übernahme eines eigenen Werkstücks. Die Steuerung des gesamten Produktionsablaufs muß also in erster Linie von den HTS übernommen werden. Die Zuordnung eines Auftrags zu einem HTS muß dezentral zwischen den HTS ausgehandelt werden. Eine für die statische Spezifikation verwendete einfache Variante besteht aus der Bestimmung von fiktiven Kostenwerten welche durch den Transport des Werkstücks mit jedem einzelnen HTS entstehen würden. Diese Werte können beispielsweise der dazu beanspruchten Zeit, dem benötigten Weg oder dem Energieverbrauch entsprechen. Die HTS tauschen ihre Kostenwerte untereinander aus und erteilen dem HTS den Zuschlag, das den günstigsten Wert anbietet. Gleichzeitig ist zu erwarten, daß dadurch ein guter Durchsatz erreicht wird. In Abbildung 2 wird die Aushandlung des Transportzuschlags illustriert. Dabei sendet zunächst eine

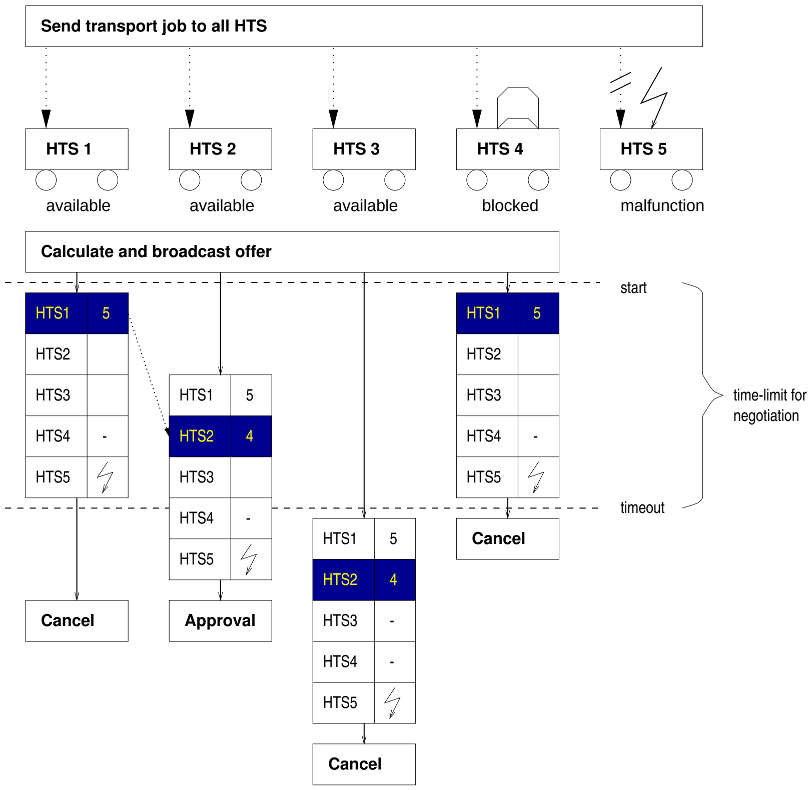

Abbildung 2: Aushandlung von Transportzuschlägen

Maschine einen Transportauftrag an alle HTS. In diesem Beispiel gibt es fünf HTS, wobei $HTS_1 \ldots HTS_3$ für neue Aufträge zur Verfügung stehen, HTS_4 derzeit einen anderen Transport durchführt und HTS_5 dem System aus anderen Gründen nicht zur Verfügung steht. Wir gehen davon aus, daß sich HTS_5 beim Eintritt in diesen Fehlerzustand bei den anderen HTS abgemeldet hat und damit für die Auftragsverhandlung ausfällt.

Unmittelbar nach Eingang des Transportauftrags gibt HTS_4 durch eine entsprechende Botschaft den übrigen HTS bekannt, daß es wegen Blockierung nicht am weiteren Verhandlungsprozeß teilnimmt. Alle verbleibenden HTS berechnen nun einen fiktiven Kostenwert, der bei Abarbeitung dieses Auftrags entstehen würde. Als erster gibt HTS_1 seinen Kostenwert ($w_1 = 5$) bekannt. Damit erhält HTS_1 vorerst den Zuschlag. Etwas später hat HTS_2

seinen Kostenwert ($w_2 = 4$) berechnet. Wegen $w_1 > w_2$ übernimmt HTS_2 damit den Zuschlag. HTS_1 kann ab diesem Moment die Auftragsverhandlung abbrechen und auf neue Aufträge warten.

Um eine Terminierung des Auftragsverhandlungsprozesses sicherzustellen, geben wir einen festen Zeitrahmen für diesen Vorgang vor. Im Beispiel "schafft" es HTS_3 nicht, innerhalb dieser Zeitspanne sein Angebot abzugeben. Damit scheidet HTS_3 ebenfalls aus. Da HTS_2 bei Ablauf des Verhandlungszeitrahmens den vorläufigen Zuschlag hatte, übernimmt es die Auftragsabarbeitung und sendet eine entsprechende Bestätigung an die auslösende Maschine sowie die anderen HTS. Sofern sich zu diesem Zeitpunkt noch andere HTS im Auftragsverhandlungsprozeß befinden (HTS_3), brechen diese den Vorgang nun auch ab und warten auf neue Aufträge.

Die Effekte dieser Auftragsvergabestrategie auf die Optimierung des Durchsatzes im Gesamtsystem hängen im Wesentlichen von der Qualität der Kostenwerte ab. Zum einen müssen die Kostenwerte den tatsächlichen Aufwand für die Bewältigung eines Auftrags wiederspiegeln. Zum anderen müssen die für die Berechnung notwendigen Parameter (Position, Gewicht, usw.) möglichst genau erfaßt werden können. Transportaufträge können in unserem Beispiel ausschließlich von Maschinen initiiert werden. Da bei einem DEMAND-Auftrag ein Werkstück angeliefert und bei einem OFFER-Auftrag ein Werkstück von der Maschine abgeholt werden sollen, benötigen Aufträge in jedem Fall einen *Partner*. Der Partner für einen Auftrag ist das Objekt, das bei einem DEMAND-Auftrag das unbearbeitete Werkstück zur Verfügung stellt und bei einem OFFER-Auftrag das bearbeitete Werkstück übernimmt. Als Partner kommen daher andere Maschinen, HTS sowie das Ein- und Ausgangslager in Frage. In Abbildung 3 sind die beiden Auftragsarten illustriert. Die ursprüngliche (vor Aufnahme der Auftragsbearbeitung)

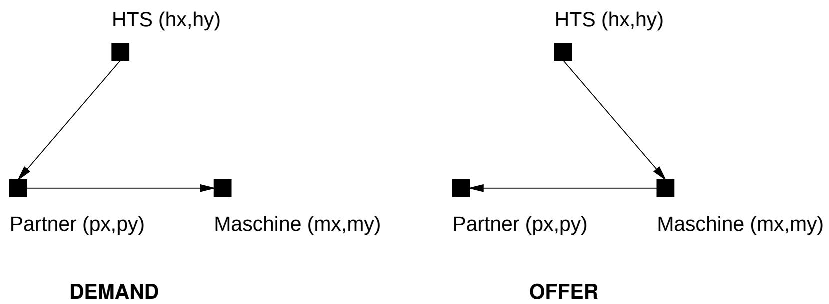

Abbildung 3: Auftragsvarianten

Position des HTS befindet sich an den Koordinaten (hx, hy). Wir gehen davon aus, daß sich alle Teile des Szenarios auf einer Höhenebene befinden.

Daher ist die Erfassung einer zusätzlichen Höhenkoordinate nicht erforderlich. Auftragsauslösende Maschine und Partnerobjekt befinden sich an den Koordinaten (mx, my) respektive (px, py). Im Fall des DEMAND-Auftrags fährt das HTS unbeladen zum Partnerobjekt und von dort beladen zur Maschine. Beim OFFER-Auftrag fährt das HTS unbeladen zur Maschine und von dort beladen zum Partner. Neben den Positionen sind folgende Kenngrößen bekannt und können in die Kostenwertberechnung einfließen:

- g_h: Leergewicht des HTS
- g_w: Gewicht des Werkstücks
- v^0_{max}: Höchstgeschwindigkeit des unbeladenen HTS
- v^b_{max}: Höchstgeschwindigkeit des beladenen HTS
- a^0_+: Beschleunigung des unbeladenen HTS
- a^0_-: Bremsbeschleunigung des unbeladenen HTS
- a^b_+: Beschleunigung des beladenen HTS
- a^b_-: Bremsbeschleunigung des beladenen HTS

Zum Zeitpunkt der Auftragsannahme soll sich das HTS nicht bewegen und unbeladen sein. Als Kostenwert nehmen wir die benötigte Zeit für die Abarbeitung des Auftrags an:

$$t = t^0 + t^b$$

Dabei ist t^0 die Zeit für die Leerfahrt und t^b die Zeit für die beladene Fahrt. Weiterhin sind folgende Wege berechnungsrelevant:

- HTS $\leftrightarrow$ Maschine: $s_{hm} = \sqrt{(x_h - x_m)^2 + (y_h - y_m)^2}$
- HTS $\leftrightarrow$ Partner: $s_{hp} = \sqrt{(x_h - x_p)^2 + (y_h - y_p)^2}$
- Maschine $\leftrightarrow$ Partner: $s_{mp} = \sqrt{(x_m - x_p)^2 + (y_m - y_p)^2}$

Dabei gelten für den DEMAND-Auftrag beispielsweise folgende Gleichungen:

$$\begin{aligned}
s_{hp} &= \frac{a^0_+}{2}(t^0_a)^2 + v^0_{max} t^0_b - \frac{a^0_-}{2}(t^0_c)^2 \\
t^0_a &= \frac{v^0_{max}}{a^0_+} \\
t^0_c &= \frac{-v^0_{max}}{a^0_-}
\end{aligned}$$

Die Zeit t^0, die das unbeladene HTS von (x_h, y_h) bis (x_p, y_p) benötigt, ergibt sich aus

$$t^0 = t_a^0 + t_b^0 + t_c^0$$

Dabei repräsentieren t_a^0 und t_c^0 die Zeiten, die das HTS zum Beschleunigen auf die Maximalgeschwindigkeit v_{max}^0 bzw. zum Abbremsen benötigt. Während der Zeit t_b^0 befindet sich das HTS in der Maximalgeschwindigkeit. Zunächst sind t_a^0 und t_b^0 mit Hilfe des Parameters v_{max}^0 zu berechnen. Daraus können anschließend die Wege

$$\begin{aligned} s_a &= \frac{a_+^0}{2}(t_a^0)^2 \\ s_c &= \frac{-a_-^0}{2}(t_c^0)^2 \end{aligned}$$

berechnet werden. Übersteigt die Summe aus s_a und s_c den Gesamtweg s_{hp}, so erreicht das HTS seine Maximalgeschwindigkeit nicht. Dieser Zusammenhang ist in Abbildung 4 dargestellt. In diesem Fall gilt für die Maximalgeschwin-

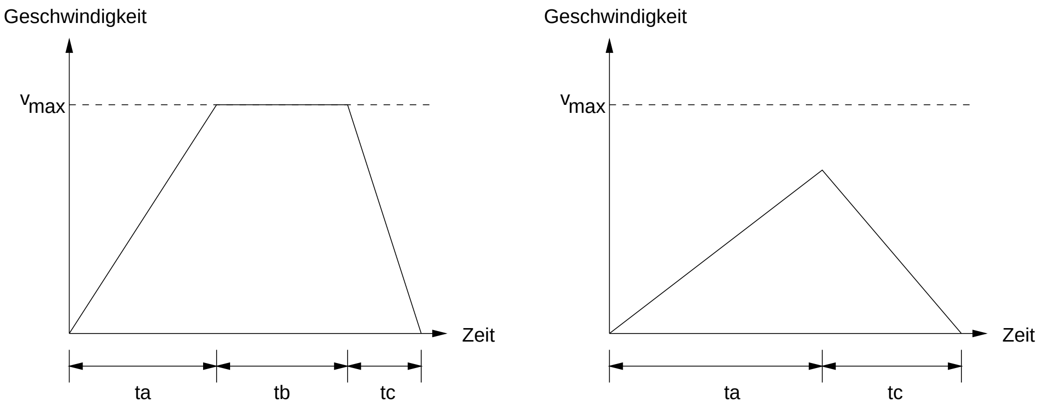

Abbildung 4: Bewegungskurve von HTS

digkeit $v_{max}^?$:

$$v_?^0 \leq v_{max}^0$$

Das HTS muß unmittelbar nach dem Ende der Beschleunigungsphase und dem Erreichen von $v_?^0$ den Bremsvorgang einleiten, so daß es zu keinem Zeitpunkt eine gleichmäßige Geschwindigkeit einnimmt. Daher gelten für diesen Fall folgende Gleichungen:

$$\begin{aligned} s_{hp} &= \frac{a_+^0}{2}(t_a^0)^2 - \frac{a_-^0}{2}(t_c^0)^2 \\ t_a^0 &= \frac{v_?^0}{a_+^0} \end{aligned}$$

$$t_c^0 = \frac{-v_?^0}{a_-^0}$$

Beispiel 1.1 Für die Bewegung des unbeladenen HTS zum Partnerobjekt sind folgende Parameter bekannt. Das HTS kann unbeladen mit $a_+^0 = 5\frac{m}{s^2}$ beschleunigen und bremst mit $a_-^0 = -4\frac{m}{s^2}$ ab. Dabei kann es eine Maximalgeschwindigkeit von $v_{max}^0 = 5\frac{m}{s}$ erreichen.

Variante 1 Die Entfernung zwischen HTS und Partnerobjekt beträgt $s_{hp} = 10m$. Zunächst gehen wir davon aus, daß das HTS seine mögliche Maximalgeschwindigkeit erreichen kann und berechnen so $t_a^0 = 1s$ und $t_c^0 = 1,25s$. Damit ergeben sich für den Beschleunigungsweg $s_a = 2,5m$ und $s_c = 3,125m$. Die Summe aus s_a und s_c liegt unter dem Wert von s_{hp}. Somit erreicht das HTS offensichtlich seine Maximalgeschwindigkeit v_{max}^0 und bewegt sich in der Zeit t_b^0 mit gleichmäßiger Geschwindigkeit. Für t_b^0 ergibt sich somit:

$$t_b^0 = \frac{s_{hp} - s_a - s_c}{v_{max}^0}$$

In unserem Beispiel bewegt sich das HTS für $t_b^0 = 0,875s$ gleichmäßig mit der Geschwindigkeit v_{max}^0. Insgesamt benötigt das unbeladene HTS für den Weg s_{hp} eine Zeit von $t^0 = t_a^0 + t_b^0 + t_c^0 = 6,5s$

Variante 2 Die Entfernung zwischen HTS und Partnerobjekt soll in dieser Variante nur $s_{hp} = 5m$ betragen. Offensichtlich kann das HTS seine maximal mögliche Geschwindigkeit v_{max}^0 nicht erreichen, da der Abbremsvorgang nicht in der verbleibenden Strecke möglich wäre. Daher beschleunigt das HTS bis zu der unbekannten Geschwindigkeit $v_?^0$ und bremst unmittelbar im Anschluß an die Beschleunigungsphase wieder ab. Damit bleibt zunächst die Geschwindigkeit $v_?^0$ zu ermitteln:

$$\begin{aligned}
s_{hp} &= \frac{a_+^0 (v_?^0)^2}{2(a_+^0)^2} - \frac{a_-^0 (-v_?^0)^2}{2(a_-^0)^2} \\
s_{hp} &= \frac{(v_?^0)^2}{2a_+^0} - \frac{(v_?^0)^2}{2a_-^0} \\
s_{hp} &= \frac{(v_?^0)^2 (a_-^0 - a_+^0)}{2a_+^0 a_-^0} \\
v_?^0 &= \sqrt{\frac{2s_{hp} a_+^0 a_-^0}{a_-^0 - a_+^0}}
\end{aligned}$$

Für unser Beispiel beträgt der Wert für die unbekannte maximal erreichte Geschwindigkeit damit $v_?^0 = 4,714\frac{m}{s}$. Entsprechend ergeben sich für $t_a^0 = 0,942s$

und $t_c^0 = 1,179s$. Da das HTS keine gleichmäßige Maximalgeschwindigkeit einnimmt, beträgt der Wert $t_b^0 = 0s$. Insgesamt benötigt das unbeladene HTS für den Weg s_{hp} also eine Zeit von $t^0 = t_a^0 + t_b^0 + t_c^0 = 2,121s$ □

Die Berechnung des Zeitbedarfs für die beladene Fahrt kann dementsprechend mit Hilfe der passenden Beschleunigungs- und Geschwindigkeitsparameter durchgeführt werden.

2 Statische Spezifikation

In diesem Abschnitt wollen wir nun eine statische Spezifikation der vorgestellten Fallstudie präsentieren. Mit Hilfe der Spezifikationssprache TROLL werden wir die Komponenten des Szenarios sowie die grundlegenden Abläufe in den HTS und Maschinen beschreiben. Grundsätzlich handelt es sich bei TROLL [DH97] um eine objektorientierte textuelle Spezifikationssprache zur formalen Beschreibung von Informationssystemen. In Ergänzung zur textuellen Notation existiert weiterhin die grafische Variante OMTROLL. OMTROLL-Diagramme basieren auf OMT und erlauben eine grafische Darstellung vieler TROLL-Konzepte. Darüber hinaus können TROLL-Spezifikationen aus OMTROLL-Diagrammen abgeleitet und anschließend verfeinert werden. In Abbildung 5 werden die spezifizierten Komponenten der Fallstudienbe-

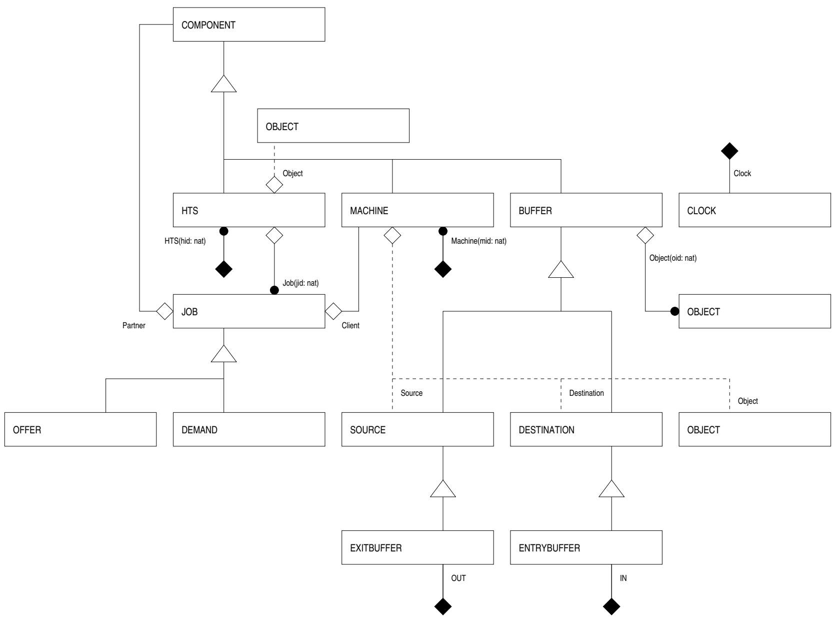

Abbildung 5: *Object Community* Diagramm

schreibung mit ihren Spezialisierungs- und Aggregationsbeziehungen dargestellt. Folgende Objekte wurden identifiziert:

- COMPONENT
  Bei allen aktiven Komponenten des Szenarios handelt es sich um Spezialisierungen dieser abstrakten Klasse. Alle Merkmale bezüglich Kommunikation und Fehlerbehandlung werden in dieser Klasse spezifiziert.

- OBJECT
  In dieser Klasse werden die Werkstücke beschrieben. Während in der Grundstufe des Systems ausschließlich eine Sorte von Werkstücken bearbeitet wird, sollen in der Ausbaustufe auch verschiedene Werkstücke mit unterschiedlichen Arbeitsschritten berücksichtigt werden.

- JOB
  In der Klasse JOB werden alle Aufträge zusammengefaßt, die durch ein HTS bearbeitet werden. Prinzipiell ist so zwischen Anlieferungs- und Abholungsjobs zu unterscheiden.

- OFFER
  Diese Spezialisierung von JOB beschreibt Abholungsjobs, also Aufträge, die durch ein Angebot (an bearbeiteten Werkstücken) einer Maschine zustande kommen.

- DEMAND
  In dieser Spezialisierung von JOB werden Anlieferungsjobs spezifiziert. Es handelt sich dabei um Aufträge, die durch den Bedarf (an unbearbeiteten Werkstücken) einer Maschine entstehen.

- HTS
  Die Spezifikation eines HTS umfaßt die Auftragsverhandlung, die Auftragsbearbeitung sowie die Behandlung weiterer Fälle (z. B. *Batterie leer*).

- MACHINE
  Eine Maschine besteht aus lokalen vor- und nachgeschalteten Puffern sowie dem Bearbeitungsplatz selbst. Abhänging von der spezifizierten Auftragsinitialisierungsstrategie müssen die Maschinen Aufträge bei den HTS auslösen. Die eigentliche Werkstückbearbeitung ist nicht Gegenstand dieser Spezifikation.

- BUFFER
  Diese abstrakte Klasse bildet die Generalisierung aller Pufferelemen-

te des Szenarios. Dies umfaßt das Ein- und Ausgangslager sowie die lokalen Maschinenpuffer.

- CLOCK
  Um eine komponentenübergreifende systemweit einheitliche Zeitvorgabe zu gewährleisten, wird die (nicht physisch vorhandene) Klasse CLOCK spezifiziert. Wir verwenden Zeitangaben beispielsweise bei der Festlegung des Zeitlimits für die Auftragsverhandlung.

- SOURCE, DESTINATION
  Diese beiden Klassen bilden die maschinenlokalen Puffer zur Zwischenlagerung von Werkstücken vor und nach Bearbeitung durch die Maschine.

- ENTRYBUFFER, EXITBUFFER
  Diese Klassen beschreiben das Eingangslager EIN und das Ausgangslager AUS. Im Gegensatz zur Grundstufe haben diese Lager in der Ausbaustufe eine begrenzte Kapazität können also komplett entleert oder überfüllt sein.

Nachdem wir in Abbildung 5 mit dem Object-Community Diagramm die Objekte im Szenario identifiziert und kurz erläutert haben, wollen wir nun ausgewählte Abläufe betrachten. Zur Illustration einer adaptiven Spezifikation eignen sich insbesondere die Abläufe in den Maschinen und den HTS, die wir unter den Begriffen *Auftragsverhandlung* und *Auftragsvergabe* betrachten werden. Wie wir im weiteren Verlauf erläutern werden, ergeben sich hier vielfältige sinnvolle Adaptionsmöglichkeiten. OmTroll bietet zur Spezifikation von Abläufen State Diagrams. In Abbildung 6 ist das State Diagram für eine Maschine dargestellt. Mittels des *birth*-Ereignisses start geht die Maschine in den Zustand ready über. Für den Fall, daß sich im lokalen Eingangspuffer (SOURCE) noch Werkstücke befinden (Source.available $>$ 0) kann das nächste Werkstück aus dem Puffer entnommen und bearbeitet werden (process_Object). Dieser Vorgang wird hier nicht detailliert beschrieben, da er für die Auftragsvergabestrategie ohne Bedeutung ist. Damit geht die Maschine in den Zustand processed über. Insofern der lokale Ausgangspuffer wenigstens einen freien Platz hat (Dest.free $>$ 0) kann des nunmehr bearbeitete Werkstück abgelegt werden (next_Object) und erneut der Zustand ready eingenommen werden. Für den Fall, daß im Eingangspuffer keine weiteren Werkstücke bereitstehen (Source.available $=$ 0), wird ein Auftrag ausgelöst (send_JobD). Dabei wird mittels Funkkommunikation eine Anfrage an alle HTS im System gesendet, welche die Art des benötigten

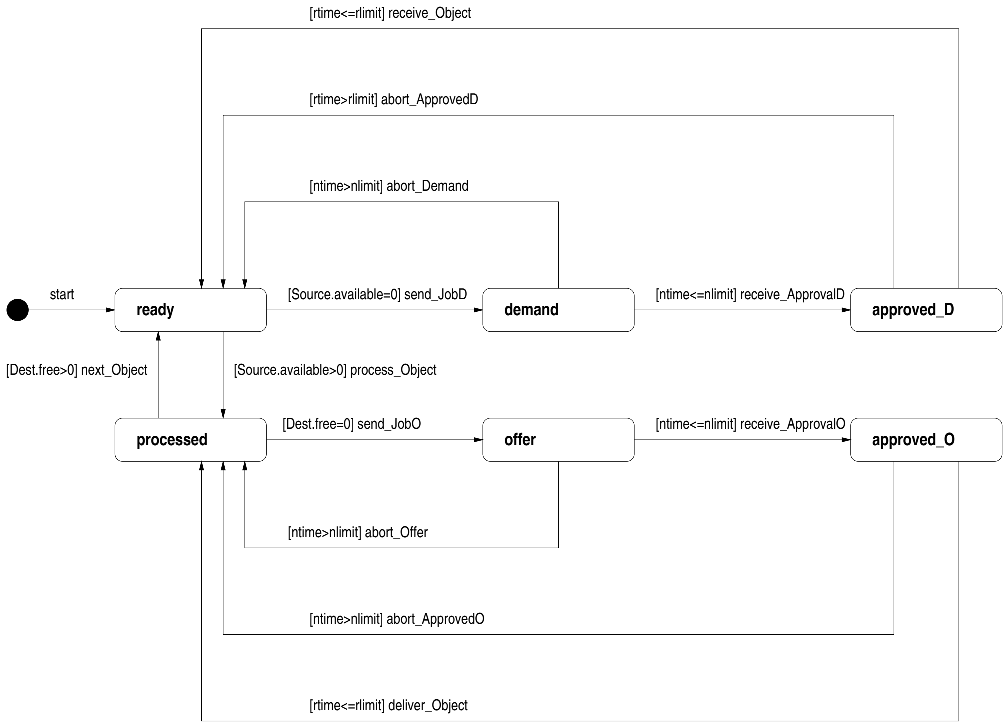

Abbildung 6: State Diagram für MASCHINE

Werkstücks, die aktuelle Zeit, das Zeitlimit für die Auftragsverhandlung sowie die Auftragsart (DEMAND) enthält. Gleichzeitig geht die Maschine in den Zustand demand über. Sie verharrt in diesem Zustand, bis das vorgegebene Zeitlimit überschritten wurde (ntime $>$ nlimit). Dann geht sie wieder in den Zustand ready über, um den Auftrag erneut zu senden. Trifft jedoch innerhalb des Zeitlimits (ntime $\leq$ nlimit) die Bestätigung eines HTS ein (receive_ApprovalD), so geht die Maschine in den approved_D-Zustand über. Auch jetzt gibt es für die Abarbeitung des Auftrags ein vorgegebenes Zeitlimit. Wird das benötigte Werkstück innerhalb dieses Limits (rtime $\leq$ rlimit) angeliefert (receive_Object), so wird der Eingangspuffer aufgefüllt und die Maschine kann nach Übergang in den ready-Zustand mit einem neuen Bearbeitungszyklus beginnen. Wird auch hier das Zeitlimit für die Werkstückanlieferung überschritten (rtime $>$ rlimit), so muß der bereits initiierte Auftrag nachträglich angebrochen werden (abort_ApprovedD). Nur so kann ein blockierungsfreier Betrieb der Maschine sichergestellt werden. Danach geht sie erneut in den ready-Zustand über. Die Vorgehensweise bei fehlender Lagerkapazität in Ausgangspuffer (Dest.free $=0$) ist äquivalent.

Das State Diagram beschreibt einen einfachen Ablauf der Auftragsvergabe. Immer, wenn der Weiterbetrieb der Maschine stockt, wird ein Transportauftrag ausgelöst. Offensichtlich ist diese Vorgehensweise praktikabel, stellt jedoch keinen optimierten Systemdurchsatz sicher. Wie wir später sehen werden, eröffnet sich durch die Adaption der Auftragsvergabestrategie die Möglichkeit zur Verbesserung der Performanz.

Nachdem wir den Vorgang der Auftragsvergabe in den Maschinen betrachtet haben, wollen wir nun die Abläufe bei der Auftragsverhandlung in den HTS spezifizieren. In Abbildung 7 ist das State Diagram für die HTS dargestellt. Nach Eintritt des *birth*-Ereignisses start geht das HTS in den Zustand

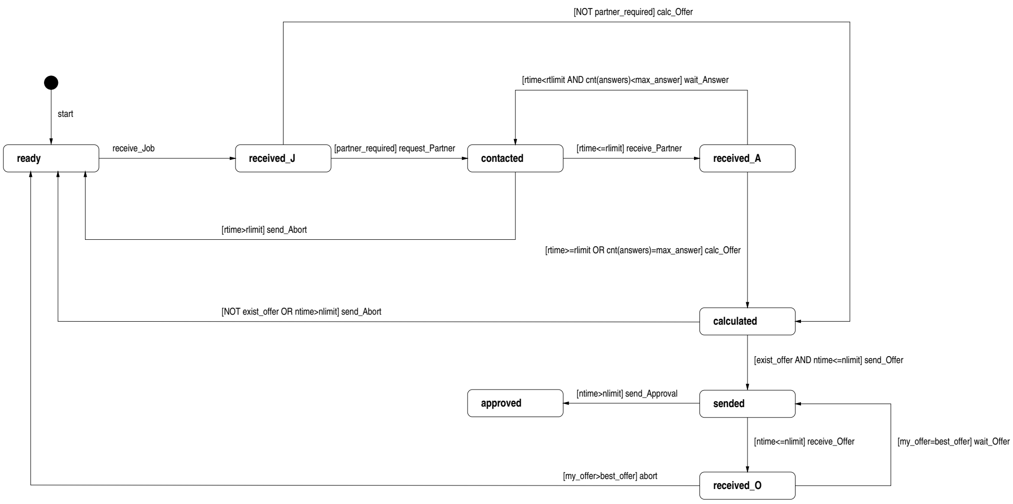

Abbildung 7: State Diagram für HTS

ready über. Trifft über Funk eine Auftragsanfrage von einer Maschine ein (receive_Job), so geht das HTS in den Zustand received_J über. Dieser Zustand markiert den Beginn der nun folgenden Auftragsverhandlung. Über die Auftragsübernahme kann nicht autonom, sondern nur in Abstimmung mit den anderen HTS entschieden werden. Abhängig davon, ob ein Partnerobjekt für die Ausführung dieses Auftrags notwendig ist (partner_required), muß durch eine Funkanfrage ein potentieller Partner gefunden werden (request_Partner). Danach geht das HTS in den Zustand contacted über. Alternativ kann das HTS in einigen Fällen auch ohne Partner auskommen. Das kann beispielsweise dann der Fall sein, wenn ein zuvor übernommener Auftrag wegen Zeitüberschreitung abgebrochen wurde (NOT partner_required). Dann kann unverzüglich ein Angebotswert berechnet werden (calc_Offer) und in den Zustand calculated übergegangen werden. Anders verhält es sich, wenn das HTS auf die Bestätigung eines potentiellen Partners warten muß. Wird ein dafür vorgesehenes Zeitlimit überschritten (rtime $>$ rlimit), bricht das HTS die Angebotsverhandlung ab und gibt sein Ausscheiden aus dem Vorgang den anderen HTS bekannt (send_Abort), um danach wieder in den ready-Zustand zurückzukehren. Innerhalb des Zeitlimits ($\text{rtime} \leq \text{rlimit}$), sammelt das HTS die Parameter aller eingehenden Partnerobjekte (receive_Partner). Bis zum Ende des Zeitlimits oder dem Erreichen einer vordefinierten maximalen Anzahl von Partnerobjekten ($\text{cnt(answers)} < \text{max\_answer}$) wartet des HTS auf weitere Angebote von den Partnerobjekten (wait_Answer). Ist das Zeitlimit erreicht ($\text{rtime} \geq \text{rlimit}$) oder ist eine bestimmte Anzahl von Partnerangeboten eingetroffen ($\text{cnt(answers)} = \text{max\_answer}$), wird aus den Auftrags- und Partnerparametern ein Angebot berechnet (calc_Offer). Danach geht das HTS in den Zustand calculated über. Möglicherweise konnte aus den eingegangenen Partnerangeboten kein Angebot berechnet werden oder die Verhandlungszeit wurde bereits überschritten ($\text{NOT exist\_offer OR rtime} > \text{rlimit}$). Auch in diesem Fall muß nun die Verhandlung abgebrochen werden (send_Abort) und der Zustand ready eingenommen werden.

Anderenfalls wird nun der Angebotswert an die anderen teilnehmenden HTS versendet (send_Offer). Dabei geht das HTS in den Zustand sended über. In diesem Zustand wartet es auf weitere Angebote kokurrierender HTS. Innerhalb der Verhandlungszeitspanne ($\text{ntime} \leq \text{nlimit}$) können so weitere Angebote eintreffen (receive_Offer). Bei jedem dieser Angebote wird das eigene Angebot verglichen. Fällt dieser Vergleich negativ aus ($\text{my\_offer} > \text{best\_offer}$), bricht das HTS die Verhandlung unverzüglich ab (abort), ohne weitere Benachrichtigungen an andere HTS zu senden. Ist das eigene Angebot besser als das eingegangene Angebot ($\text{my\_offer}=\text{best\_offer}$), wartet das HTS auf weitere Konkurrenzangebote (wait_Offer) und geht dafür wieder in den Zustand sended über. Ist die Verhandlungszeit schließlich verstrichen ($\text{ntime} > \text{nlimit}$), übernimmt das HTS die Abarbeitung dieses Auftrags und schickt eine Bestätigung an die auslösende Maschine (send_Approval) und geht in den Zustand approved über.

Die in Abbildung 7 beschriebene Auftragsverhandlung der HTS ist mit der Auftragsvergabe aus Abbildung 6 *kompatibel*. Allerdings ist das Verhalten der Maschinen ebenso wie das der HTS nicht komplett spezifiziert. Obwohl Maschinen auch potentielle Partnerobjekte für Transportaufträge darstellen, ist die Partnerfunktionalität hier nicht spezifiziert. Ebenso fehlt die formale Beschreibung der eigentlichen Auftragsausführung durch die HTS im State Diagram. Ein weiteres Problem ist durch die fehlende Nebenläufigkeit bei der Behandlungs eingehender Funksignale gegeben. Zur Vermeidung des Verlustes eingehender Nachrichten sollten diese Signale in einem separaten Nachrichtenpuffer gespeichert und auch so spezifiziert werden. Alle diese Mängel sind jedoch für die Erläuterung unseres Adaptionsansatzes irrelevant und müssen daher in vertiefenden Papieren beschrieben werden.

Um die objektübergreifende Kommunikation zwischen Maschinen und HTS zu beschreiben, ist in Abbildung 8 mit dem Communication Diagram ein weiteres OmTroll-Diagramm dargestellt. Neben der bereits diskutier-

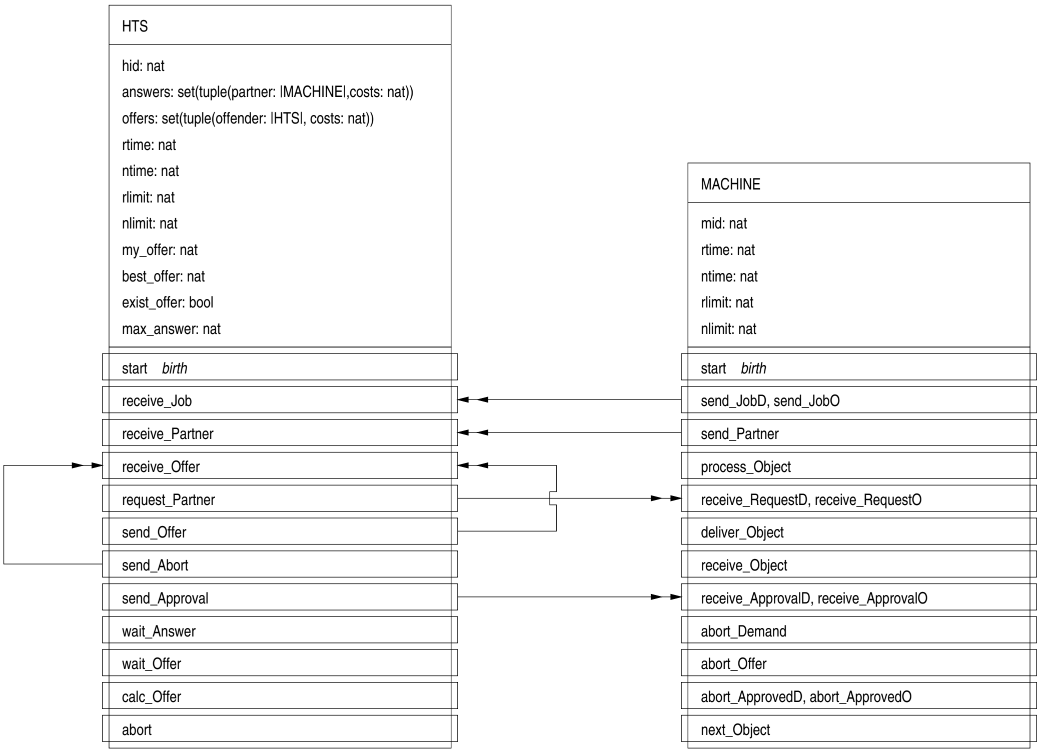

Abbildung 8: Communication Diagram für HTS und Maschinen

ten Auftragsvergabe- und -verhandlungskommunikation sind weiterhin mit send_Partner und receive_Partner die Kommunikationselemente für die Suche von Partnerobjekten angegeben.

3 Adaptionsansätze

Im vorangegangenen Abschnitt haben wir die statische Spezifikation von Auftragsvergabe- und -verhandlungsstrategie der HTS und Maschinen in der Referenzfallstudie *Holonischer Materialfluß* präsentiert. Dabei wurden bereits einige Nachteile der gewählten Strategie kurz aufgeführt. Darüber hinaus müssen jedoch auch einige strukturelle Probleme benannt werden, die sich aus der Gesamtcharakteristik der Kombination "Technisches Szenario und statische Spezifikation" ergeben.

So ist beispielsweise zu erwarten, daß die beschriebenen technischen Einrichtungen über einen verhältnismäßig langen Zeitraum genutzt werden. Aus dieser langen Lebensdauer läßt sich wiederum nicht zwingend schließen, daß das Szenario ansonsten unverändert bestehen bleibt. Die Produktionsabläufe können sich ebenso ändern, wie die konkreten Produkte. Diese Änderungen können möglicherweise durch gewollte Umstrukturierungsmaßnahmen, veränderte gesetzlicher Vorgaben, den Austausch einzelner Systemkomponenten oder menschlicher Eingriffe in das System bedingt sein. Häufig beziehen sich diese Änderungen jedoch auf begrenzte Teile der formalen Spezifikation und betreffen zudem häufig die gleiche Thematik, wie beispielsweise die Auftragsvergabestrategie. In unserem Beispiel würden Teile wie die Wertstückbearbeitung, die Funktionsweise von Puffern und Lagern, die Übernahme/Übergabe von Werkstücken zwischen Maschinen und HTS, die Fahrfunktionalität der HTS und die Kollisionsvermeidung zwischen den HTS von einer Anpassung der Auftragsvergabestrategie nicht tangiert. Dennoch machen diese Elemente einen Großteil der Spezifikation aus.

Etablierte Spezifikationstechniken setzen auf die vollständige Beschreibung des Systems in der formalen Spezifikation. Die Umsetzung in ein Softwareprodukt, in unserem Beispiel in eine komplexe Steuerungs- und Kontrollsoftware, schließt sich der Spezifikation an. Spätere Veränderungen an der Spezifikation, die in der entstehenden Software automatisch reflektiert würden, sind nicht vorgesehen. Jede Veränderung im Szenario, die nicht von der formalen Beschreibung vorgesehen war, erfordert so konsequenterweise eine komplette Neuspezifikation und – darauf basierend – Neuimplementierung der Software. Bevor wir adaptive Spezifikationen als mögliche Lösung für diese Problematik einführen, werden wir im folgenden verschiedene Möglichkeiten zur Realisierung von Auftragsvergabe- und verhandlungsstrategien vorstellen. Wir werden diese Möglichkeiten zunächst informal beschreiben und dann mit Hilfe unseres Adaptionsansatzes in eine adaptive Spezifikation umsetzen.

3.1 Auftragsvergabestrategien

Die Auftragsvergabestrategie wird in erster Linie von den Maschinen bestimmt. Sie legt fest, wann und wie Transportaufträge an HTS vergeben werden. Durch die Auswahl einer "guten" Strategie kann der Durchsatz an Werkstücken erheblich gesteigert werden:

- urgent
  Hierbei handelt es sich um die bereits spezifizierte Strategie. Sie legt fest, daß Transportaufträge nur in akuten Fällen initiiert werden. Das

betrifft den Fall, wenn der lokale Eingangspuffer geleert oder der lokale Ausgangspuffer voll ist. Erst dann wird ein Auftrag ausgelöst. Offensichtlich muß die Maschine in diesem Fall dann immer ihren Abarbeitungszyklus unterbrechen, und auf die Vollendung des Auftrags durch ein HTS warten. Trotz dieser durchsatzmindernden Charakteristik gibt es einige sinnvolle Anwendungsfelder. Zum einen kann der Fall eintreten, daß wenige HTS viele Maschinen bedienen müssen. Dadurch wird die Auftragslast für die HTS verringert. Zum anderen wäre diese Strategie praktikabel, wenn die lokalen Eingangs- und Ausgangspuffer jeweils nur einen Pufferplatz besitzen.

- always
  Diese Strategie stellt von ihrer Charakteristik den gegensätzlichen Fall zu urgent dar. Hier werden Transportaufträge immer dann ausgelöst, wenn ein Werkstück aus dem Eingangspuffer entnommen oder im Ausgangspuffer abgelegt wird. Offenbar kann bei dieser Strategie der Systemdurchsatz bei einer genügend großen Anzahl von verfügbaren HTS gegenüber urgent verbessert werden.

- limit
  Die Strategie limit stellt einen Kompromiß zwischen urgent und always dar. Mit Hilfe zweier Parameter wird der Füllstand bzw. Entleerungsgrad von Ein- und Ausgangspuffer überwacht und bei Erreichen dieses festgelegten Limits ein Transportauftrag ausgelöst. Der Vorteil dieser Strategie liegt in einer besseren Anpaßbarkeit an die Anzahl der verfügbaren HTS bzw. deren Auftragslast.

- never
  Die Strategie never nimmt gegenüber den zuvor vorgestellten Strategien eine Sonderrolle ein. Hier wird die Auftragsauslösung nicht von den Maschinen übernommen. Stattdessen müssen sie warten, bis ein HTS anstehende Transportaufgaben “freiwillig” übernimmt. Die Spezifikation der Maschinen wird dadurch erheblich vereinfacht.

In unserer statischen Spezifikation verfolgen alle Maschinen die gleiche Auftragsvergabestrategie. Mögliche Verbesserungen des Durchsatzes könnten ebenfalls dadurch erreicht werden, daß die Maschinen mit unterschiedlichen Strategien arbeiten. So würde eine schnell arbeitende Maschine mit kurzen Bearbeitungszeiten vermutlich mit der Strategie always den Gesamtdurchsatz positiv beeinflussen. Ebenso macht bei einer langsam arbeitenden Maschine die Wartezeit auf angelieferte Werkstücke nur einen geringen Teil der Bearbeitungszeit aus. Um die Auftragslast der HTS zu verringern, wäre die Strategie urgent hier möglicherweise angebrachter.

3.2 Auftragsverhandlungsstrategien

Mit der Auftragsverhandlungsstrategie entscheiden die HTS untereinander über die Übernahme von Transportaufträgen. Auch damit läßt sich der Systemdurchsatz beeinflussen:

- offer
  Die Strategie offer wurde bereits in der statischen Spezifikation verwendet. Grundidee dieser Strategie ist die Berechnung konkurrierender Angebote durch die beteiligten HTS. Das beste Angebot "gewinnt" und übernimmt die Ausführung des Auftrags. Die Praktikabilität dieser Strategie hängt von Art und Qualität der Parameter für die Angebotsberechnung ab. Nur wenn diese Parameter die Berechnung eines aussagekräftigen und vergleichbaren Angebotswertes gestatten, ist dieser Ansatz sinnvoll. Weiterhin entsteht durch den Verhandlungsprozeß ein hohes Kommunikationsaufkommen (Partnersuche, Angebotsvergleich), das ebenfalls Probleme bereiten kann.

- idle
  Mit der Strategie idle soll eine hohe Auslastung der HTS erreicht werden. Dabei sollen aktuell unbeschäftigte HTS freiwillig Aufträge übernehmen. Diese Strategie ist das Gegenstück zur Auftragsvergabestrategie never und kann außerdem mit anderen Auftragsverhandlungsstrategien kombiniert werden.

- assign
  Anders als bisher steht ein HTS hier nicht allen, sondern nur einer bestimmten Auswahl von Maschinen zur Verfügung. Dies ist insbesondere dann sinnvoll, wenn sich die Maschinen in großen räumlichen Entfernungen zueinander befinden. Dabei würden lange Fahrwege anfallen, die mit dieser Strategie teilweise vermieden werden sollen. Eventuell kann so auch die Auftragsverhandlung entfallen.

- sequential
  Diese einfache Strategie legt fest, daß die HTS nacheinander die Maschinen "besuchen". Daher wird keine Auftragsverhandlung benötigt. Allerdings ist diese Strategie nur bei einer geringen Anzahl von HTS sinnvoll.

Auch bei den Auftragsvergabestrategien müssen nicht alle HTS zwingend die gleiche Strategie verfolgen. Ebenso lassen sich einzelne Strategien kombinieren und sind so der aktuellen Systemsituation besser anpaßbar.

Die Liste der hier genannten Strategien erhebt keinen Anspruch auf Vollständigkeit. Vielmehr handelt es sich um eine kleine Auswahl sinnvoller Möglichkeiten zur Umsetzung der Strategien. Viele Strategien sind nur in bestimmten Konstellationen bzw. Systemkonfigurationen nützlich. Damit wird erneut deutlich, daß eine optimale Gestaltung der formalen Spezifikation unmöglich in Unkenntnis der möglichen zukünftigen Entwicklungen erfolgen kann.

4 Adaptive Spezifikation

Nachdem wir im letzten Abschnitt mehrere Strategien für die Auftragsvergabe und -verhandlung benannt haben, werden wir nun exemplarisch eine adaptive Spezifikation entwickeln. Dazu werden wir jeweils zwei Varianten herausgreifen. Zunächst wollen wir den Begriff der Adaption von Spezifikation nach unserem Verständnis näher charakterisieren. In bezug auf die bisher verwendeten OmTroll-Diagramme lassen sich Adaptionen grob in zwei Klassen unterscheiden:

- Einfache Adaption: In diesem Fall verändern sich Anzahl, Identität und Reihenfolge der Zustände im State Diagram nicht. Vielmehr können sich Bedingungen (guards) und Ereignisse (events) verändern, die von einem Zustand in einen anderen führen. In Abbildung 9 sind drei Typen einfacher Adaptionen dargestellt.

    1. *Condition Adaption:* Bei diesem Adaptionstyp wird lediglich der guard eines Ereignisses verändert.
    2. *Event Adaption:* Hier wird bei gleichem guard das Ereignis bei einem Zustandsübergang verändert.
    3. *Condition-Event Adaption:* Sowohl guard als auch event eines Zustandsübergangs verändern sich.

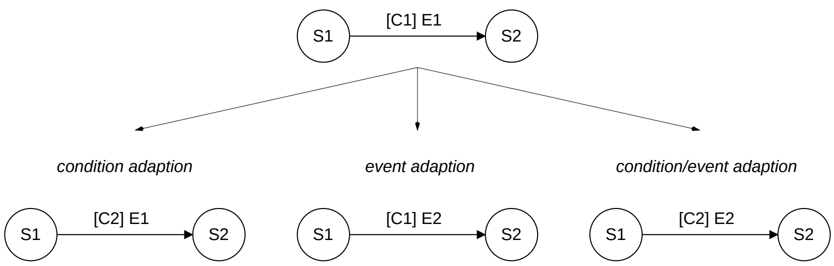

Abbildung 9: Einfache Adaptionen

- Komplexe Adaption: Komplexe Adaptionen erlauben tiefergreifende Veränderungen bei Zustandsübergängen als einfache Adaptionen. Neue Zustände können hinzugefügt oder ersetzt werden. Andere Zustände können ganz aus dem State Diagram entfernt werden. In Abbildung 10 werden komplexe Adaptionstypen dargestellt.

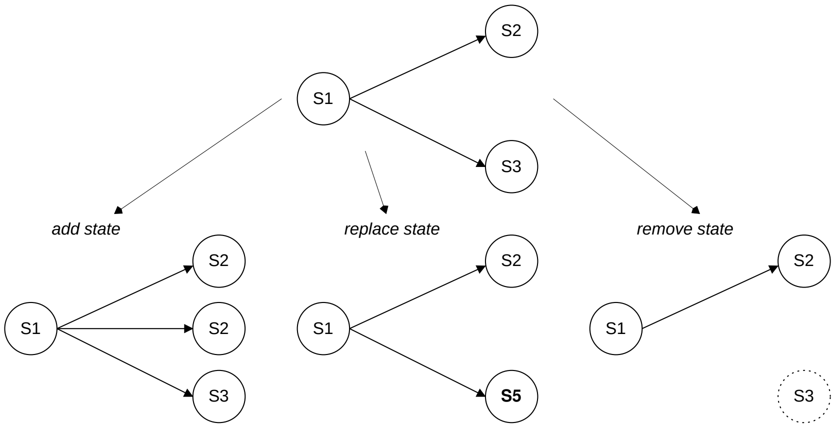

Abbildung 10: Komplexe Adaptionen

Im folgenden werden wir die Adaption der Auftragsvergabestrategien always→urgent und der Auftragsverhandlungsstrategien offer→assign diskutieren. Dazu sollen zunächst die Veränderungen in den State Diagrams veranschaulicht werden. In Abbildung 11 ist eine veränderte Variante des State Diagrams der Klasse MASCHINE dargestellt. Für die guards der Ereignis-

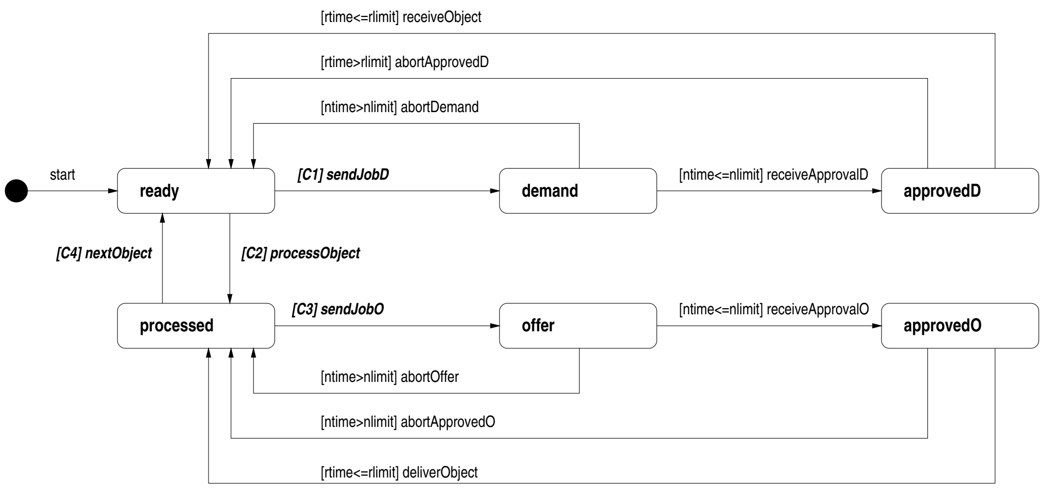

Abbildung 11: Adaptives State Diagram für MASCHINE

se sendJobD, processObject, sendJobO und nextObject wurden die Platzhalter $C_1 \dots C_4$ eingefügt. In Tabelle 1 wird der Adaptionsvorgang anhand der unterschiedlichen Belegungen für $C_1 \dots C_4$ illustriert. Aus den Veränderun-

guard	always	urgent
C_1	Source.available $<$ Source.capacity	Source.available $= 0$
C_2	Source.available $=$ Source.capacity	Source.available > 0
C_3	Dest.free $<$ Dest.capacity	Dest.free $= 0$
C_4	Dest.free $=$ Dest.capacity	Dest.free > 0

Tabelle 1: Adaption: always$\rightarrow$urgent

gen der guards in Tabelle 1 geht hervor, daß es sich bei der Adaption always$\rightarrow$urgent offensichtlich um eine event adaption handelt.

In Abbildung 12 ist eine veränderte Variante des State Diagrams der Klasse HTS dargestellt. Die Unterscheidung der beiden Strategien offer und assign

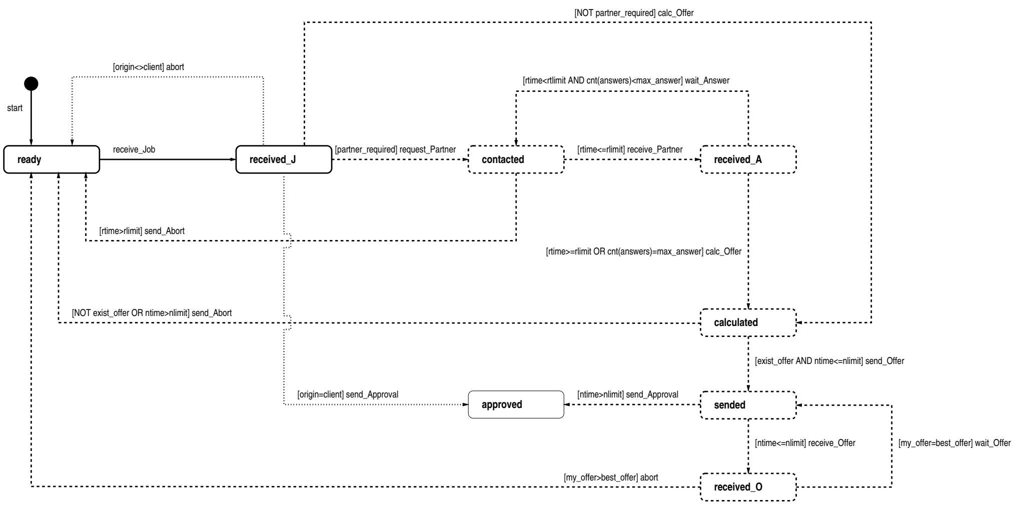

Abbildung 12: Adaptives State Diagram für HTS

erfolgt anhand der Linienarten. Alle mit durchgehenden Linien dargestellten Zustände und Ereignisse sind in beiden Strategien vorhanden. Die gepunkteten Linien sind der Strategie assign zugeordnet, während die getrichelten Linien in der offer-Strategie verwendet werden. In diesem Beispiel ist die assign-Strategie weiter eingeschränkt, indem das HTS genau einer Maschine zugeordnet ist. Alle Aufträge, die nicht von dieser Maschine kommen (origin $\neq$ client) werden zurückgewiesen. Alle anderen Aufträge werden angenommen.

Wie Abbildung 12 deutlich macht, sind beim Wechsel zwischen diesen Strategien umfangreichere Veränderungen am State Diagram notwendig. Zustände müssen in das Diagramm eingefügt bzw. aus ihm entfernt werden. Damit handelt es sich bei offer→assign um eine komplexe Adaption.

Im folgenden wollen wir unsere Adaptionsansätze in TROLL-Spezifikationen umsetzen. Wir basieren unseren Ansatz auf der TROLL-Erweiterung *dy*TROLL[TCS96, CRSS98, CST99]. Die Idee dieser Erweiterung besteht in einer Trennung der objektlokalen Spezifikation in einen rigiden und einen dynamischen Anteil. Der rigide Teil enthält die unveränderliche Grundcharakteristik. Der dynamische Teil enthält die adaptiven Elemente, also die Anteile der Spezifikation, die entfernt oder ersetzt werden können. Die Entscheidung welche Spezifikationsanteile rigide oder dynamisch werden, muß zur Entwurfszeit der Spezifikation erfolgen. Die größtmögliche Flexibilität könnte dadurch erreicht werden, daß die komplette Spezifikation dynamisch also austauschbar gestaltet wird. Damit verlieren wir aber alle grundlegenden Zusicherungen an das System. In unserem Beispiel wären dann auch nicht die grundlegenden Eigenschaften von Maschinen und HTS fest in der Spezifikation verankert.

```
object class MASCHINE
  identification
    ByMid:(mid)
  attributes
    mid:nat constant.
    rtime:nat initialized 0.
    ntime:nat initialized 0.
    rlimit:nat initialized 100.
    nlimit:nat initialized 20.
    job:|JOB| initialized nil.
    hts:|HTS| initialized nil.
  components
    Source:SOURCE
    Dest:DESTINATION
    Object:OBJECT
  events
    start birth
      calling
        {Source.available=0} sendJobD,
        {Source.available>0} processObject.
```

```
sendJobD
  changing
    ntime:=0,
    job:=job.create(DEMAND,self)
  calling
    foreach h:HTS do h.receiveJob(job) od,
    abortDemand.

abortDemand
  enabled
    ntime>nlimit
  calling
    sendJobD,
    processObject.

receiveApprovalD(h:|HTS|,j:|JOB|,offer:real)
  enabled
    sometime sendJobD sincelast receiveApprovalD
    and j=job
  changing
    hts:=h,
    rtime:=0,
    rlimit:=offer
  calling
    abortApprovedD.

abortApprovedD
  enabled
    rtime>rlimit
  calling
    sendJobD,
    processObject.

receiveObject(o:|OBJECT|)
  enabled
    sometime receiveApprovalD sincelast sendJobD
  calling
    sendJobD,
    processObject.

processObject
  calling
```

```
    nextObject,
    sendJobO.

sendJobO
  changing
    ntime:=0,
    job:=job.create(OFFER,self)
  calling
    foreach h:HTS do h.receiveJob(job) od,
    abortOffer.

abortOffer
  enabled
    ntime>nlimit
  calling
    sendJobO,
    nextObject.

receiveApprovalO(h:|HTS|,j:|JOB|,offer:real)
  enabled
    sometime sendJobO sincelast receiveApprovalO
    and j=job
  changing
    hts:=h,
    rtime:=0,
    rlimit:=offer
  calling
    abortApprovedO.

abortApprovedO
  enabled
    rtime>rlimit
  calling
    sendJobO,
    nextObject.

deliverObject(!o:|OBJECT|)
  enabled
    sometime receiveApprovalO sincelast sendJobO
  calling
    sendJobO,
    nextObject.
```

```
    nextObject
      calling
        processObject,
        sendJobD.

  axioms
    JobInitiation
      initialized
        sendJobD enabled Source.available = 0.
        processObject enabled Source.available > 0.
        sendJobO enabled Dest.free = 0.
        nextObject enabled Dest.free > 0.

  mutators
    replaceStrategy(Add:set(axiom), Rem:set(axiom))
      enabled
        Rem ⊆ JobInitiation
      changing
        JobInitiation:=JobInitiation-Rem
        JobInitiation:=JobInitiation∪Add.
end object class
```

Um eine adaptive Spezifikation zu erhalten, müssen wir zunächst ein oder mehrere **axioms** festlegen. Axiome sind spezielle Attribute (Meta-Attribute), die TROLL-Formeln als Werte enthalten. Die so festgelegten Axiome enthalten den aktuellen dynamischen Spezifikationsanteil. Um die Werte dieser Axiome zu verändern, benötigen wir des weiteren eine spezielle Art von Ereignissen (**mutators**). In unserem Beispiel ist replaceStrategy solch ein Mutatorereignis. In Äquivalenz zu **events** können wir auch hier guards festlegen. In unserem Beispiel soll das Mutatorereignis replaceStrategy die Auftragsvergabestrategien austauschen. Der guard stellt sicher, daß sich die zu entfernenden Axiome in der Menge der gegenwärtig gültigen Axiome (JobInitiation) befinden.

Das Beispiel zeigt, daß ein Austausch der Auftragsvergabestrategien sehr einfach zu realisieren ist. Im Gegensatz dazu illustriert Abbildung 12 die Komplexität der Adaption von Auftragsverhandlungsstrategien. Nahezu alle Ereignisse der statischen Spezifikation müssen hier entfernt bzw. verändert werden. Aus Platzgründen verzichten wir an dieser Stelle auf eine komplette *dy*TROLL-Spezifikation dieser Adaption.

5 Zusammenfassung

In diesem Papier wurde die Fallstudie "Holonischer Materialfluß" ausführlich vorgestellt. Spezifische Eigenschaften technischer Informationssysteme wurden anhand der Fallstudie illustriert. In besonderer Weise wurden Lösungsansätze für Auftragsvergabe- und -verhandlungsstrategien beschrieben. In der Folge wurde eine OmTroll-basierte Spezifikation entwickelt und erläutert. Anschließend wurden Probleme bei der Verwendung statischer Spezifikationen im technischen Umfeld aufgezeigt. Insbesondere für die erwähnten Strategien wurden mehrere Varianten informell beschrieben. Daraus wurde eine Motivation für adaptive Spezifikationen entwickelt. Eine grobe Klassifikation unterteilte adaptive Spezifikationen in einfache und komplexe Adaptionen. Anschließend wurden OmTroll-basierte adaptive Spezifikationen ausgewählter Auftragsvergabe- und -verhandlungsstrategien präsentiert. Daraus wurde eine *dy*Troll-Spezifikation entwickelt.

Die Troll/*dy*Troll-basierte Spezifikation technischer Szenarien läßt noch einige Wünsche offen. So muß beispielsweise die Möglichkeit zur Darstellung nebenläufiger Abläufe näher betrachtet werden. Weiterhin sind Entwurfsrichtlinien zu erarbeiten, die es dem Spezifikationsentwerfer gestatten, eine sinnvolle Zuordnung rigider und dynamischer Spezifikationsanteile vorzunehmen. Des weiteren sollten Sprachmittel eingeführt werden, die auch eine objektübergreifende Adaption gestatten. Darunter verstehen wir beispielsweise das Hinzufügen oder Entfernen ganzer Klassen.

Literatur

[ABS00] N. Aoumeur, S. Balko und G. Saake. Towards a Three-Level Methodology for Developing Cooperative Information Systems. Preprint, Fakultät für Informatik, Universität Magdeburg, 2000. *To appear.*

[Bal00] Balko, S. Adaptive Specifications of Technical Information Systems. In H. Balsters, B. De Brock und S. Conrad, Herausgeber, *Proc. of the 9th International Workshop on Foundations of Models and Languages for Data and Objects : Database schema evolution and meta-modeling, Dagstuhl, Germany*, 2000. *to appear.*

[CRSS98] S. Conrad, J. Ramos, G. Saake und C. Sernadas. Evolving Logical Specification in Information Systems. In J. Chomicki und G. Saake, Herausgeber, *Logics for Databases and Information Systems*, Kapitel 7, S. 199–228. Kluwer Academic Publishers, Boston, 1998.

[CST99] S. Conrad, G. Saake und C. Türker. Towards an Agent-Oriented Framework for Specification of Information Systems. In J.-J. Ch. Meyer und P.-Y. Schobbens, Herausgeber, *Formal Models of Agents — ESPRIT Project ModelAge Final Report, Selected Papers*, Lecture Notes in Artificial Intelligence, Band 1760, S. 57–73. Springer-Verlag, Berlin, 1999.

[DH97] G. Denker und P. Hartel. Troll – An Object Oriented Formal Method for Distributed Information System Design: Syntax and Pragmatics. Informatik-Berichte, Technische Universität Braunschweig, 1997.

[TCS96] C. Türker, S. Conrad und G. Saake. Dynamically Changing Behavior: An Agent-Oriented View to Modeling Information Systems. In J. Fiadeiro und P.-Y. Schobbens, Herausgeber, *Proc. of the 2nd Workshop of the ModelAge Project (ModelAge'96), Sesimbra, Portugal, January 1996*, S. 257–270, Departamento de Informatica, Universitade de Lisboa, 1996.