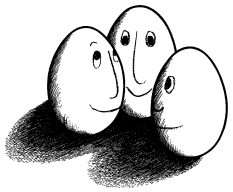
**Diplomarbeit**

# Produktionsplanung mit Hilfe von Multiagentensystemen

Sascha Lüdecke



Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

18. Oktober 2001

**Betreuer:**

Prof. Dr. Katharina Morik
Dipl. Inform. Stefan Haustein

Hiermit bestätige ich, dass ich diese Arbeit selbstständig verfasst und keine anderen, als die angegebenen Quellen und Hilfsmittel verwendet habe. Sämtliche Zitate wurden als solche kenntlich gemacht.

(Sascha Lüdecke)

**Danksagung**
Diese Diplomarbeit ermöglicht haben: meine Frau Hayet durch ihre grenzenlose Geduld, Dirk Burkamp und Oliver Steinbach durch Korrekturlesen und konstruktive Kritik. Desweiteren meine Freunde und Familie durch Aufmunterung und offene Ohren.

Es gibt nur eine Ausflucht vor der Arbeit: Andere für sich arbeiten zu lassen.

– (Immanuel Kant, dt. Phil., 1724-1804) –

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einleitung

Heutige Unternehmen in der produzierenden Industrie verfügen über umfangreiche Produktionsanlagen und stellen eine Vielzahl unterschiedlicher Produkte her. Sie stehen vor der Aufgabe, den Produktionsprozess angemessen zu steuern, um wirtschaftlich arbeiten zu können und treffen dabei auf sehr unterschiedliche Probleme. So muss zum Beispiel die Vorratshaltung und Nachbestellung notwendiger Rohstoffe geregelt werden, um den Produktionsprozess nicht zum Stillstand kommen zu lassen (Lagerhaltung). Hier ist zu planen, welche Mengen vorzuhalten sind und in welchem Abstand bzw. Umfang nachbestellt wird. Gerät ein zum Beispiel Lieferant in Lieferschwierigkeiten, oder wird ein Liefervertrag gekündigt, müssen weitere Lieferanten beauftragt werden. Geschieht dies zu einem sehr späten Zeitpunkt, so kann es zu einem Mangel an Rohstoffen und somit einem Stillstand wenigstens von Teilen der Produktion kommen. Da die Produktionsanlagen oft hohe Kosten in der Anschaffung und/order während des laufenden Betriebs (Betriebskosten) verursachen, ist dies natürlich zu vermeiden. Neben der Lagerhaltung, ist die Kapazitätsplanung der Produktionsanlagen und Arbeitskräfte ein kritisches Element. Beide müssen in ausreichender „Kapazität“ vorhanden sein, um das Produktionsziel zu erreichen (Anzahl der herzustellenden Produkte bzw. zu erfüllenden Aufträge von Kunden). Da aber zum Beispiel die Nachfrage nach Produkten etwa saisonbedingten Schwankungen unterliegt, ist die Kapazitätsplanung von großer Bedeutung. Eventuell müssen weitere Arbeitskräfte eingestellt, die vorhandenen Kurzarbeit leisten, oder auch neue Produktionsanlagen beschafft werden. Das wichtigste Problem aber ist die genaue Planung und Steuerung des Ablaufes des Produktionsprozesses selbst. Um die vorhandenen Produktionsanlagen und Arbeitskräfte optimal einzusetzen, aber auch, um Produktionstermine einzuhalten, ist es notwendig die einzelnen Bearbeitungsschritte eines Produktes auf den unterschiedlichen Produktionsanlagen zeitgenau abzustimmen. Einem Unternehmen entstehen zum einen hohe Kosten durch nicht gut ausgelastete Produktionsanlagen und zum anderen durch halbfertige Produkte, die sich noch in der Produktion befinden und etwa auf eine Produktionsanlage warten müssen (Kapitalbindung). Diese beiden Kostenfaktoren sind widersprüchlich, da im Extremfall durch die Bereitstellung sehr großer bzw. redundanter Kapazitäten zwar alle Produkte ohne Verzögerung den Produktionsprozess durchlaufen, aber auch hohe Leerlaufzeiten entstehen, um die Verfügbarkeit aller Arten von vorhandenen Produktionsanlagen zu gewährleisten. Demgegenüber steht eine maximale Ausnutzung der vorhandenen Kapazitäten, die ihrerseits zu langen Warteschlangen vor den einzelnen Produktionsanlagen führt.

Um diese komplexen Aufgabenstellungen zu handhaben, kommen in den Unternehmen *Produktionsplanungssysteme* zum Einsatz, die den Produktionsprozess planen und

steuern. Die gängigen Systeme verfolgen dabei einen zentralisierten Ansatz, d.h. ein einzelnes System führt die Planung und Steuerung der Produktion durch. Aufgrund der hohen Komplexität der Aufgabe und der Tatsache, das lokale Informationen z.B. über Störungen an das System zur Berücksichtigung gemeldet werden müssen, ist ihre Leistungsfähigkeit aber begrenzt, in der Praxis ermitteln sie höchstens eine gute, fast nie aber eine optimale Lösung. Neuere Systeme verfolgen einen dezentralen Ansatz, indem etwa den Produktionsanlagen eine gewisse Autonomie in der Planung zugestanden wird. Sie erhalten eine Menge von Produkten und sind selbst dafür verantwortlich, die Bearbeitungsreihenfolge zu bestimmen. Mit einem dezentralen Ansatz können Teile der Berechnung also an den Ort der Handlung verlegt werden, um zum einen die Rechenzeit zu verteilen, und zum anderen, um schnell auf lokale Einfüsse zu reagieren.

Aus der Sicht der künstlichen Intelligenz bietet es sich hier an, so genannte *Multiagentensysteme* zur Lösung des Problems einzusetzen. Ein solches System besteht dabei aus vielen, autonom handelnden Softwareeinheiten, die miteinander kommunizieren, um gemeinsam ein Problem zu lösen. Die einzelnen Agenten orientieren ihre Vorgehensweise in der Regel an für sie geltende Ziele und müssen nicht immer kooperativ zusammen arbeiten. Multiagentensysteme versprechen flexiblere Lösungen, als zentralisierte Ansätze und sollen aufgrund ihrer Natur einfacher zu handhaben bzw. anzupassen sein, da sich einzelne Agenten leicht austauschen lassen. Auch, wenn bereits viele Publikationen zum Thema Multiagentensysteme erschienen sind, stehen konkrete Nachweise für ihre Eignung noch aus.

## 1.1 Aufgabenstellung

Diese Arbeit befasst sich mit der Evaluation von Koordinationsmechanismen in Multiagentensystemen. Zentrale Vorraussetzung für den Einsatz von Multiagentensystemen sind Aussagen über ihre Eignung und Verhalten in unterschiedlichen Situationen. Da das Verhalten wesentlich von der Koordinationsform beeinflußt wird, ist ihre Wahl ausschlaggebend für den Erfolg oder Misserfolg eines Systems.

Ziel der Arbeit ist es, ein Multiagentensystem für die Ablaufplanung als Teilproblem der Produktionsplanung, das die Koordinationsformen elektronischer Markt und Kontrakt Netz beherrscht, zu entwerfen und zu implementieren. Anschliessend sollen anhand noch zu bestimmender Kriterien die folgenden Fragestellungen untersucht werden:

1. Wie unterscheiden sich die verschiedenen Koordinationsmechanismen und unter welchen Bedingungen arbeiten sie gut?

2. Sind die jeweiligen Koordinationsmechanismen in der Lage flexibel auf Änderungen der Umgebung zu reagieren?

3. Eignen sich Multiagentensysteme, um Probleme der Produktionsplanung zu lösen?

## 1.2 Überblick

Die Arbeit gliedert sich wie folgt: Kapitel 1 enthält die Einleitung in das Thema. Kapitel 2 wird in den Problembereich der Produktionsplanung einführen und die Aufgabenstellung der Ablaufplanung erläutern. Kapitel 3 beschreibt den Begriff des Multiagentensystems und geht neuere Arbeiten zum Entwurf von Multiagentensystemen ein. Kapitel 4 bereitet knapp auf Experimente mit Multiagentensystemen vor und nennt einige bisherige Arbeiten zu diesem Thema. Kapitel 5 stellt die Implementierung eines Multiagentensystems vor, die im weiteren für Experimente verwendet wird. Die Experimente sind in Kapitel 7 beschrieben. Abgeschlossen wird die Arbeit mit Kapitel 8, das nocheinmal die wichtigsten Ergebnisse zusammenfasst.

# 2 Produktionsplanung

In diesem Kapitel soll in den Problembereich der Produktionsplanung eingeführt werden. Nach einigen Grundbegriffen aus der Produktionswirtschaft, wird das Aufgabenfeld der Produktionsplanung beschrieben. Anschließend werden die Grenzen der verbreiteten Lösungsansätze aus der Industrie. Der Inhalt dieses Kapitels richtet sich nach [Jahnke und Biskup 1999], und deckt sich mit Teil eins: „Grundlagen der Produktion“ aus [Jehle et al. 1990].

## 2.1 Produktion

In den beiden folgenden Abschnitten soll der Begriff der *Produktion* sowie die Struktureigenschaften von Produktionsprozessen erläutert werden.

### 2.1.1 Was ist Produktion?

Das Buch über „Planung und Steuerung der Produktion“ von Jahnke und Biskup [Jahnke und Biskup 1999] beginnt mit folgendem Satz:

> Das Hauptanwendungsgebiet der Produktionsplanung ist die Sachgüterproduktion, die die Haupttätigkeit eines Industriebetriebes darstellt.

Bei der Produktionsplanung geht es also um die Produktion von Gütern, genauer gesagt um Sachgüter. Jahnke und Biskup beschreiben den Begriff „Produktion“ im weitesten Sinne als den „Gewinnungs- oder Transformationsvorgang, bei dem bestimmte Güter in definierten Mengen zielgerichtet eingesetzt werden, um andere Güter herzustellen.“ Die hergestellten Güter nennt man dabei *Produkte*, während man die eingesetzten Güter als *Produktionsfaktoren* bezeichnet.

Als Haupttätigkeit eines Industriebetriebs bedeutet „industrielle Produktion“ oft, dass die einzelnen Arbeitsgänge auf verschiedene Arbeitskräfte/Maschinen aufgeteilt werden und für große, anonyme Märkte produziert wird. Dementsprechend sind die Einsatzkräfte weitestgehend spezialisiert[Jahnke und Biskup 1999, Abschnitt 1.1, Abgrenzung des Gegenstandes]. Die handwerkliche Fertigung grenzt sich davon insofern ab, als eine eher auftragsbezogene und individuelle Fertigung erfolgt. Die Arbeitskräfte haben dementsprechend ein breiteres Tätigkeitsfeld.

Die zur Produktion notwendigen Produktionsfaktoren werden in der Literatur unterschiedlich unterteilt. Die Volkswirtschaft teilt Produktionsfaktoren in Kapital, Arbeit und Boden ein. Kapital bezeichnet die Leistungen und Güter, in deren Bereitstellung

selbst Güter einfließen, also Maschinen, Werkzeuge oder die Leistungen gelernter Arbeiter. Arbeit ist zu verstehen als die Leistung der ungelernten Arbeitskräfte, während Boden Güter in unbearbeitetem Zustand bezeichnet (etwa Rohstoffe). Diese Einteilung ist für die meisten Betriebe aber zu ungenau, weshalb Gutenberg [Gutenberg 1983] (zitiert in [Jahnke und Biskup 1999, Seite 12]) in Arbeit, Betriebsmittel und Werkstoffe unterteilt:

**Arbeit** umfasst hier sowohl objektbezogenes Arbeiten in der Produktion, als auch dispositive Arbeit, etwa die Planung und Steuerung der Produktion;

**Betriebsmittel** umfasst die technische Ausstattung des Betriebes, die zur Produktion verwendet wird und längerfristig vorhanden ist, also Maschinen und Werkzeuge, aber auch auch Grundstücke;

**Werkstoffe** umfassen alle Güter, die im Laufe der Produktion verbraucht werden, nämlich die verbrauchten Rohstoffe (z.B. Holz für die Tischherstellung), Hilfsstoffe, die nicht direkt in das Produkt einfließen (z.B. Leim für die Tischherstellung), und Betriebsstoffe, die zum Betrieb der Maschinen notwendig sind (z.B. Strom oder Schleifmittel), aber nicht in das Produkt miteinfließen.

Wie in Tabelle 2.1 dargestellt, werden weitere Einteilungen vorgenommen, von denen die Einteilung in Potential- und Verbrauchsfaktoren im Abschnitt 2.2 auf Seite 11 zum Tragen kommt.

<table>
<tr>
<td rowspan="3">Arbeit</td>
<td rowspan="2">dispositive<br>Arbeit<br>objektbezogene</td>
<td rowspan="3">dispositiver Faktor</td>
<td rowspan="3">Potential-<br>faktoren</td>
</tr>
<tr>
<td rowspan="2">Boden</td>
</tr>
<tr>
<td rowspan="2">Betriebsmittel</td>
<td rowspan="2">Elementar-<br>faktoren</td>
</tr>
<tr>
<td rowspan="2">Kapital</td>
<td>Rohstoffe<br>Werkstoffe Hilfsstoffe<br>Betriebsstoffe</td>
<td>Repetierfaktoren</td>
</tr>
</table>

Tabelle 2.1: Systematisierung der Produktionsfaktoren

### 2.1.2 Struktur der Produktion in einem Betrieb

Die Produktion in einem Betrieb lässt sich anhand verschiedener Strukturmerkmale klassifizieren. Diese sind auf den Input, den Output und den Produktionsprozess bezogen.

#### Input

Hinsichtlich des Inputs unterscheidet man je nach Anzahl der benötigten Vorprodukte zwischen einer *einteiligen Produktion*, etwa wenn nur ein Werkstück wie z.B. bei der

Produktion von Schrauben bearbeitet wird, und einer *mehrteiligen Produktion*, wenn sich das Endprodukt aus mehreren Teilen zusammensetzt wie z.B. bei der Produktion eines Tisches.

### Output

Hinsichtlich des Outputs unterscheidet man zwischen einem *Ein-Produkt-Unternehmen*, das nur ein Produkt herstellt wie etwa ein Energieversorger, und einem *Mehr-Produkt-Unternehmen*, das mehrere Produkte anbietet. Letztere sind der Regelfall.

Weiter ist der Anstoß der Produktion maßgeblich: gibt es eine hohe Zahl von Varianten, so dass keine Standardprodukte, sondern einzelnen Kundenwünschen entsprechende Produkte gefertigt werden, so spricht man von einer *Auftragsfertigung* (z.B. im Schiffsbau). Kann der Kunde aus einer vorgegebenen Anzahl von Produktvarianten, etwa verschiedenen Ausstattungen bei Automobilen, wählen, so spricht man von einer *bedingten Auftragsfertigung*. Geht es jedoch um standardisierte Produkte, wie z.B. das einfache Konsumgut Bleistifte, so spricht man von einem *anonymen Markt*.

### Produktionsprozess

Bei dem Produktionsprozess werden mehrere Dimensionen betrachtet.

Zunächst die **Anzahl der Produktionsstufen**, nämlich die Anzahl der notwendigen Produktionsschritte, die zur Herstellung des Endproduktes nötig sind. *Einstufige Produktion* liegt vor, wenn während der Produktion nur ein Betriebsmittel belegt wird, etwa bei der Herstellung von Ziegeln. *Mehrstufige Produktion* liegt vor, wenn mehrere Produktionsschritte bis zum Endprodukt nötig sind, etwa bei der Produktion komplexer Chemikalien.

Dann die **Verbundenheit der Produktion**: werden bei der Herstellung mehrere Produkte gefertigt, die in festem Mengenverhältnis stehen, so spricht man von einer *verbundenen Produktion*, etwa bei der Raffinierung von Erdöl, das mehrere Nebenprodukte abwirft. *Unverbundene Produktion* liegt dagegen vor, wenn die unterschiedlichen Endprodukte in keinem technisch-bedingt festen Mengenverhältnis zueinander stehen.

Desweiteren der **Organisationstyp der Fertigung**: von einer *Werkbankfertigung* spricht man, wenn alle Werkzeuge um einen Arbeitsplatz angeordnet sind, etwa bei einem Zahnarzt. Kennzeichnend ist hier eine stark handwerkliche Arbeitsweise, die oft bei wenig mechanisierter Fertigung oder Ländern mit niedrigem Lohnniveau anzutreffen ist. Die *Baustellenfertigung* liegt vor, wenn Produktionsfaktoren zum Standort des Produktes geschafft werden, etwa beim Hoch- oder Tiefbau. Dieser Organisationstyp wird vor allem dann eingesetzt, wenn das Produkt groß oder schwer zu transportieren ist.

Der häufigste Organisationsstyp jedoch ist die *produktionsmittelorientierte Fertigung*, bei der die Herstellung an den Betriebsmitteln stattfindet. Man unterscheidet bei diesem Typ grob zwischen zwei Arten[1], der *Werkstattfertigung* und der *Reihen- bzw. Fließfertigung*.

[1] [Jahnke und Biskup 1999] unterscheiden auch zwischen Fließ- und Reihenfertigung, diese Formen sind sehr ähnlich und wurden daher zusammengefaßt.

Bei der Werkstattfertigung sind funktionsgleiche Betriebsmittel räumlich nah beieinander angeordnet sind. Der Transport zwischen den Betriebsmitteln erfolgt je nach Bedarf und nimmt aufgrund der relativ großen Entfernung zwischen den Betriebsmitteln unterschiedlicher Funktionsgruppen geraume Zeit in Anspruch und verursacht nicht unerhebliche Kosten. Die Durchlaufzeiten für einen Auftrag, also die Zeit zwischen dem Beginn der Produktion und der Fertigstellung sind in der Regel also hoch, wodurch sich höhere Zins- und Lagerkosten ergeben, als in der Reihenfertigung (s.u.). Durch die räumliche Anordnung der Betriebsmittel ist der Fertigungsprozess zudem unübersichtlich und schwer kontrollierbar. Diese Organisationsform trifft man vor allem im Maschinenbau an, bei der Produkte in der Bohrerei, Gießerei usw. bearbeitet werden.

Bei der Reihenfertigung und der Fließfertigung, bei der die Betriebsmittel in der für alle Produkte gleichen Reihenfolge des Materialflusses angeordnet sind. Dadurch lassen sich Transportzeiten und -wege vereinheitlichen und verkürzen. Man spricht hier auch von einer Straßenfertigung, die dann sinnvoll ist, wenn für alle Produkte eine einheitliche Bearbeitungsreihenfolge vorliegt. Bei der Fließfertigung erfolgt der Transport zwischen den Betriebsmitteln dabei typischerweise automatisch und taktgesteuert (z.B. Montageprozesse, etwa Automobil- oder Elektrogeräteherstellung). Durch die Anordnung der Betriebsmittel entsprechend der Bearbeitungsreihenfolge ergeben sich niedrigere Durchlaufzeiten als bei der Werkstattfertigung und eine bessere Übersicht über den Fertigungsprozess. Die Zins- und Lagerkosten für Halbfabrikate sind deutlich niedriger als bei der Werkstattfertigung.

### 2.1.3 Klassifikation eines Betriebs nach Strukturmerkmalen der Produktion

Wird im Rahmen der Produktionsplanung (siehe Abschnitt 2.2 auf Seite 11) von einer zentralen Stelle festgelegt, wann und wo welche Materialien und Teile anzuliefern bzw. anzufertigen sind, so spricht man von einem *Push-Prinzip*, da die Werkstoffe praktisch durch den Produktionsprozess geschoben werden. Falls die Fertigung einer Produktionsstufe hingegen durch Nachfrage einer nachfolgenden Stufe ausgelöst wird, spricht man von einer Produktion nach dem *Pull-Prinzip*, die Produkte werden aus dem Produktionsprozess herausgezogen. Eine Steuerung der Produktion durch das Pull-Prinzip findet gerade in der Just-in-Time Produktion – auch als „Produktion auf Abruf“ bezeichnet – Anwendung[2] In der Regel finden sich beide Prinzipien in Produktionsprozessen: bis zu einem gewissen Punkt, dem *Push-Pull-Punkt* werden Teilprodukte auf Vorrat produziert bzw. bestellt, die später durch Nachfrage vom Markt in die Herstellung der endgültigen

[2]Diese Steuerungsform geht auf den Produktionsingenieur Taiichi Ohno des Unternehmens Toyota zurück: er hat ein Verfahren entwickelt, das die Vorteile der Großserienfertigung hat, nämlich die hohen Stückzahlen, die geringe Bearbeitungsdauer je Auftrag und die geringen Anforderungen an Personal und Qualität, aber nicht ihre Nachteile, nämlich hohe Rüstzeiten, große Lagerbestände und das häufig schlecht motiviertes Personal. Sein Grundgedanke ist es, jegliche Verschwendung zu vermeiden, zum Beispiel Verzögerungen im Produktionsprozess oder schlechte bzw. fehlerhafte Produkte, die ausgetauscht werden müssen. Das Verfahren ist kein Steuerungsprinzip im eigentlichen Sinne, vielmehr handelt es sich um einen kontinuierlichen Optimierungsprozess, der darauf abzielt in jeder Produktionsstufe die notwendigen Materialien genau dann bereitzustellen, wenn sie benötigt werden.

Produkte einfließen. So werden gleiche Teile vorproduziert, die in unterschiedlichen Konfigurationen eines Produktes vorkommen, etwa Tischbeine oder Tischplatten. So kann später relativ zügig das Endprodukt zusammengesetzt werden.

Viele der oben genannten Strukturmerkmale treten in einem Betrieb gemeinsam auf. Dementsprechend lassen sich wie in Abbildung 2.1 dargestellt, drei unterschiedliche Industrietypen ausmachen: auftragsorientierte Einzelfertigung, marktorientierte Großserien- und Massenfertigung und gemischte Serienfertigung.

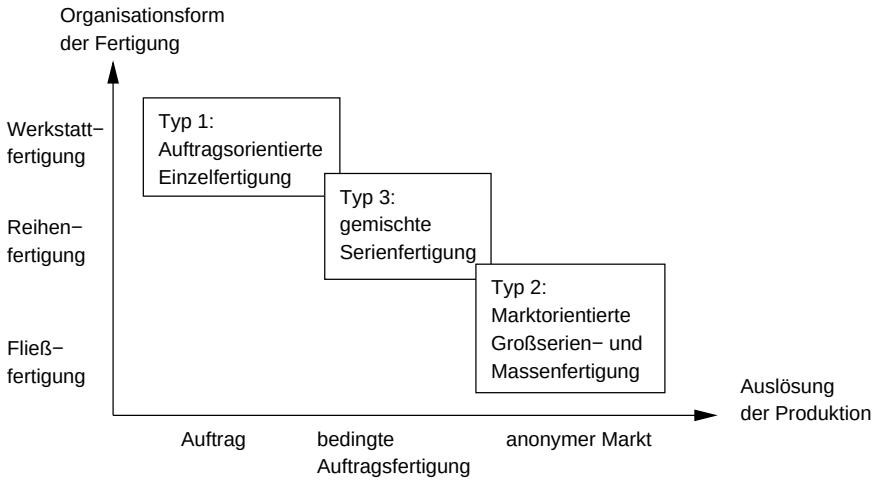



Abbildung 2.1: Klassifikation der Industrietypen nach Strukturmerkmalen

### Typ 1: Auftragsorientierte Einzelfertigung

Kennzeichnend für die *auftragsorientierte Einzelfertigung* ist, dass eine Nachfrageprognose nicht oder nur sehr ungenau erstellbar ist und individuelle Produkte hergestellt werden. Typischerweise sind die Kunden aber bereit, längere Lieferfristen in Kauf zu nehmen. Die Betriebsmittel sind in Form von Werkstätten organisiert, was eine hohe Flexibilität hinsichtlich der unterschiedlichen Produktausstattungen garantiert. Im Vergleich zur Reihenfertigung ist die Ausstattung des Betriebes kostengünstiger, da keine hochautomatisierten Fertigungsstraßen benötigt werden. Die Arbeitskräfte sind dem gegenüber aber besser ausgebildet, um auf die individuellen Kundenwünsche eingehen zu können. Dementsprechend fallen vergleichsweise geringe Gemeinkosten, dafür um so höhere Einzelkosten an, die dem Produkt direkt zurechenbar sind. Wesentlich bei diesem Industrietyp ist die Lagerhaltung, da für die unterschiedlichen Kundenwünsche oft verschiedene Bauteile vorgehalten bzw. vorproduziert werden müssen, insbesondere dann, wenn kurze Reaktionszeiten garantiert werden sollen, was wiederum vergleichsweise hohe Lagerhaltungskosten zur Folge hat. Der Push-Pull-Punkt ist aufgrund der unsicheren Spezifikation der Produkte auf einer frühen Stufe des Produktionsprozesses anzusiedeln.

Neben der Planung der Ausstattung der einzelnen Werkstätten mit Betriebsmitteln, ist hier die Planung der Bearbeitungsreihenfolge (Ablaufplanung) eines Auftrags in den

einzelnen Werkstätten die wichtigste Fragestellung.

### Typ 2: Marktorientierte Großserien- und Massenfertigung

Die *marktorientierte Großserien- und Massenfertigung* zeichnet sich durch stark standardisierte Produkte aus. Da solche Produkte meist von vielen Betrieben angeboten werden, nehmen die Kunden keine Wartezeit in Kauf. Die Nachfrage lässt sich aber relativ genau vorhersagen, weshalb hier angefangen bei den Werkstoffen, bis hin zum Endprodukt auf Vorrat produziert bzw. gelagert wird, um eine direkte Befriedigung der Nachfrage zu gewährleisten. Sofern die Produktion überhaupt durch eine Nachfrage angestoßen wird, liegt der Push-Pull Punkt in der letzten Produktionsstufe. Bedingt durch die Fertigung in großen Mengen, wird die Produktion in Form der Fließfertigung organisiert, da solche Mengen mit einer Werkstattfertigung kaum zu produzieren sind. Da auch die Ausprägung der Produkte bekannt ist, kommen im Vergleich zur auftragsorientierten Einzelfertigung weniger qualifizierte Arbeitskräfte zum Einsatz. Entsprechend sind die Investitionen für die Betriebsmittel wesentlich höher, wodurch sich ein großer Gemeinkostenanteil für die einzelnen Produkte ergibt.

Wichtige Fragestellung bei diesem Typ ist zum einen die Ausgestaltung des Fertigungssystems, vor allem hinsichtlich der Taktung und Produktionsgeschwindigkeit, zum anderen die Nachfrageprognose und entsprechende Vorhaltung von Fertigprodukten, um Nachfrageschwankungen zu begegnen.

### Typ 3: Gemischte Serienfertigung

Die *gemischte Serienfertigung* ist in jeder Hinsicht zwischen der auftragsorientierten Einzelfertigung und der marktorientierten Großserien- und Massenfertigung anzusiedeln. Als Beispiele gelten die Produktion von Computern, Stereoanlagen oder Automobilen. Die Nachfrage kann hier meist nur auf höherer Ebene abgeschätzt werden, etwa 1000 Computer, deren genaue Konfiguration erst bei Anforderung durch den Kunden festliegt. Bedingt durch diese Varianz nehmen die Kunden kurze Lieferzeiten in Kauf. Die Produktion kann sowohl durch einen anonymen Markt, als auch durch bedingte Auftragsfertigung angestoßen werden, was eine Ansiedlung des Push-Pull Punktes innerhalb der Produktionsstufen zur Folge hat. Kommt es zu einer Massenfertigung für einen anonymen Markt, so meist nur saisonbedingt oder im Rahmen von Sonderaktionen. Die Produktion ist bei diesem Typ in Form einer Reihen- oder Fließfertigung organisiert, wobei letztere eine höhere Flexibilität, aber auch eine geringere Produktionsmenge hat. Meistens werden Produkte bei diesem Typ in Serien hergestellt, bei denen Umrüstung der Maschinen zwischen den Serien erforderlich ist.

Wie bei der marktorientierten Groß- und Serienfertigung sind die wesentlichen Fragestellungen hier die Ausgestaltung des Fertigungssystems und die Prognose der Nachfrage. Hinzu kommt eine Planung der Losgrößen der Serien (Mengen der Produkte eines Typs, die ohne Unterbrechung durch Umrüsten hergestellt werden), deren Terminierung und Produktionshäufigkeit.

## 2.2 Das Aufgabenfeld der Produktionsplanung

Die Produktionsplanung umfasst den gesamten Bereich von der prinzipiellen Entscheidung, ein Produkt anzubieten, bis hin zur Herstellung des einzelnen Produktes.

### 2.2.1 Einteilung der Produktionsplanung

Die Produktionsplanung lässt sich in die drei ineinander übergehenden Bereiche strategische, taktische und operative Planung einteilen[Jahnke und Biskup 1999, Seiten 13ff.]:

**strategische Produktionsplanung:** Üblicherweise werden die Entscheidungen in diesem Bereich vom oberen Management getroffen und beziehen sich auf die Fragestellung, in welchen Produktbereichen der Betrieb produzieren soll und wie die Produktpalette aussehen soll. Als grobes Mengengerüst werden meist angestrebte Marktanteile angegeben. An dieser Stelle muss auch über den Organisationstyp der Produktion entschieden werden (Industrietyp), da er entsprechende Investitionen nachsich zieht. In der Regel ist die strategische Planung eine langfristige Planung.

**taktische Produktionsplanung:** Die Entscheidungen in diesem Bereich fällt oft das mittlere Management. Die einzelnen Produkte aus dem zuvor erstellten Portfolio werden hinsichtlich technischer Details präzisiert und entsprechende Stücklisten benötigter Teile sowie Arbeitspläne erstellt. Die Arbeitspläne werden später zur Ermittlung der Vorbereitungszeiten, Durchlaufzeiten, Kapazitätsgrenzen und auch der Planung der Löhne verwendet. Zudem werden für jedes Produkt Angaben über die in den nächsten sechs Monaten zu produzierenden Mengen erstellt, z.B. 200 Tische, 800 Stühle sowie 1000 Regale. Weiterhin werden Tiefe und Breite des Produktprogramms festgelegt, z.B. Anzahl der verschiedenen Tische (Programmtiefe) und ob komplette Wohnzimmer oder nur einzelne Möbel als Produkt angeboten werden (Programmbreite). Ebenso wird geplant, welche Teilprodukte selbst hergestellt werden und welche zugekauft werden sollen, demnach dann die Verteilung von innerbetrieblichen Lagern und die Planung des notwendigen Personals vorgenommen wird. Hier werden also die Potentialfaktoren geplant. In der Regel ist die taktische Planung eine mittelfristige Planung.

**operative Produktionsplanung:** Dies ist die feinere Planung des Produktionsprozesses, d.h. wann welche Produkte (Ablaufplanung) in welchem Umfang (Losgrößenplanung) hergestellt, welche Werkstoffe dafür benötigt (Materialplanung), wie die Teilprodukte zwischen den einzelnen Herstellungspunkten transportiert werden (Wegeplanung) und wie die Vorratshaltung aussehen soll (Lagerhaltung im Rahmen der Bestell- und Lagerhaltungspolitik). Hier werden also unter anderem die Repetierfaktoren geplant. Die operative Planung ist in der Regel eine kurzfristige Planung.

### 2.2.2 Steuergrößen der Produktionsplanung

Der Produktionsprozess lässt sich durch verschiedene Faktoren, sogenannten *Steuergrößen*, beeinflussen, die je nach Industrietyp verschieden stark ins Gewicht fallen (siehe Tabelle 2.2). Sie lassen sich aus den Besonderheiten der unterschiedlichen Industrietypen (siehe Abschnitt 2.1.3 auf Seite 8) ableiten[3]: blendet man die Personalplanung und Bestell- und Lagerhaltungspolitik (im Bereich Beschaffung der Verbrauchsfaktoren) aus, so ergeben sich als Steuergrößen[Jahnke und Biskup 1999, Seite 26]:

- die *Intensität* und *Einsatzzeit* der Betriebsmittel;
- die Anzahl der zu einem *Los* oder *Fertigungsauftrag* zusammengefassten Produkteinheiten, die Auflagehäufigkeit dieser Lose und deren zeitliche Abfolge auf den Betriebsmitteln;
- die Höhe der *Lagerbestände* zwischen den einzelnen Produktionsstufen und des Absatzlagers;
- die Festlegung der *Bearbeitungsreihenfolge* von Aufträgen auf Maschinen und deren *Terminierung* (Ablaufplanung)

| | Bedeutung[4] je Typ | | |
|---|---|---|---|
| Steuergröße | Typ 1 | Typ 2 | Typ 3 |
| Intensität und Einsatzzeit | ∘ bis + | + | + |
| Losgrößenplanung | - | - | + |
| Lagerhaltungsplanung | + | - | ∘ bis + |
| Ablaufplanung | + | - | ∘ |

Tabelle 2.2: Wichtigkeit der Steuergrößen für die verschiedenen Industrietypen

Zwischen den Steuergrößen bestehen Abhängigkeiten, so resultieren z.B. die Lagerbestände aus der Intensität und Einsatzzeit der Betriebsmittel, während die Losgrößenplanung direkt mit der Ablauf- und Lagerhaltungsplanung zusammenhängt.

Es besteht ein großer Bedarf, die verschiedenen Steuergrößen über alle Produktionsstufen hinweg zu koordinieren, da z.B. leere Lager oder fehlende Materialien zu Ausfallzeiten führen können, die unter Umständen den gesamten Produktionsprozess beeinflussen können (eine Übersicht über Störfaktoren geben [Corsten und Gössinger 1997]). Diese Aufgabe übernehmen Produktionsplanungs und -steuerungssysteme.

### 2.2.3 Das Teilproblem der Ablaufplanung

In der Produktionsplanung müssen eine Vielzahl von Planungsproblemen gelöst werden. Neben der Losgrösen-, Lagerhaltungs-, Bestell- und Transportplanung, ist die Ablaufplanung, also die Planung wann auf welcher Maschine welches Los bzw. ein Teilschritt einer Produktion ausgeführt werden soll, wichtig (siehe dazu auch Kapitel 3.2

[3] ebenfalls im Bereich operative Produktionsplanung angedeutet

[4] Legende: - = geringe Bedeutung, ∘ = mittlere Bedeutung, + = große Bedeutung

aus [Jahnke und Biskup 1999]). Sie ist insbesondere bei der Werkstattfertigung wichtig, da bei der Reihen- und Fließfertigung die Ausprägung der Produkte bekannt und die Betriebsmittel bereits entsprechend der Bearbeitungsreihenfolge angeordnet sind.

Unter Ablaufplanung als Bestandteil der kurzfristigen Produktionsplanung wird meist die Bestimmung der Bearbeitungsreihenfolgen und deren Terminierung auf den einzelnen Betriebsmitteln verstanden. Im allgemeinen geht man davon aus, dass notwendige Betriebsmittel, Arbeitskräfte und Rohstoffe in ausreichender Menge als Ergebnis der lang- und mittelfristigen Planung vorhanden sind. Die Ablaufplanung ist vor allem bei der auftragsorientierten Fertigung wichtig, da bei der Serienfertigung die Betriebsmittel bereits hinsichtlich der Bearbeitungsreihenfolgen angeordnet sind. Gerade bei der Massenproduktion verändern sich diese nur selten.

Die Ablaufplanung kann hinsichtlich drei unterschiedlicher Zielsetzungen erfolgen:

**auftragsbezogen:** Hier steht die Durchlaufzeit eines Auftrages im Vordergrund. Um Lagerhaltungs- und Zinskosten gering zu halten, die durch das Verweilen eines Auftrages in der Produktion entstehen, sollen die Durchlaufzeiten oder auch deren Summe minimiert werden.

**kundenbezogen:** Hier steht die Termintreue des Produktionsprozesses im Vordergrund. Ziel ist es, die Fertigstellungstermine möglichst einzuhalten und diese werder zu unter- noch zu überschreiten. In beiden Fällen entstehen für das Unternehmen durch die notwendige Lagerhaltung oder den Vertrauensverlust bei dem Kunden vermeidbare Kosten. Auch innerbetrieblich kann die Termintreue relevant sein, etwa wenn eine hintere Produktionsstufe auf Teilprodukte warten muss.

**maschinenbezogen:** Hier steht die Auslastung der Maschinen im Vordergrund. Ziel ist es, Leerlaufzeiten zu vermeiden und eine möglichst hohe Auslastung zu erreichen. Somit verringert sich der Anteil der Fixkosten (Betriebs- und Wartungskosten) an einem Auftrag und Betriebsmittel werden optimal ausgenutzt.

Die in dieser Arbeit behandelte Ablaufplanung soll maschinenbezogen erfolgen. Ziel des später implementierten Systems wird es also sein, eine möglichst hohe Auslastung der Maschinen zu erreichen.

## 2.3 Grenzen heutiger Produktionsplanungsysteme

Nahezu alle Planungsprobleme innerhalb der Produktionsplanung sind sehr komplex (NP-Vollständig[Jahnke und Biskup 1999, Abschnitt 1.4, Anmerkungen zur Komplexitätstheorie], entsprechende Beweise siehe [Jahnke und Biskup 1999, Abschnitt 3.2.2]). Als einfaches Problem der Ablaufplanung werden Konfigurationen mit bis zu drei Maschinen betrachtet [Jahnke und Biskup 1999, Abschnitt 3.2.3, Leichte Ablaufplanungsprobleme]. Entsprechend machen die vorhandenen exakten Verfahren Einschränkungen um eine annähernd gute Lösung zu ermitteln. So geht der Johnson-Algorithmus zur Ablaufplanung [Jahnke und Biskup 1999, Seiten 257ff.] davon aus, dass nur ein zweistufiger

Fertigungsprozess vorliegt. Der Johnson Algorithmus zielt auf eine durchlaufminimale Bearbeitungsreihenfolge ab, indem er Produktarten, die auf der ersten Stufe eine kurze Bearbeitungszeit haben, vorzieht, um für die zweite Stufe ein Auftragspolster zu schaffen. Diese Polster wird dann genutzt, um Produktarten mit längerer Bearbeitungszeit auf Stufe I zu bearbeiten.

Um überhaupt zu einer Lösung zu kommen, werden heuristische Ansätze genutzt, etwa mit Hilfe von Prioritätsregeln (siehe [Jahnke und Biskup 1999, Abschnitt 3.2.4, Schwere Ablaufplanungsprobleme] oder [Jehle et al. 1990, Abschnitt 5.2.2.4.3, Reihenfolgeplanung mit Hilfe von Prioritätsregeln]) die Ablaufplanung in den Griff zu bekommen. Einige Regeln und ihre Kritieren für die Wahl des als nächstes zu bearbeitenden Auftrages:

- SPT (shortest processing time): Auftrag mit der kürzesten Bearbeitungsdauer.
- LWKR (least work remaining): Auftrag mit der kürzesten Restbearbeitungszeit.
- SL (slack): Auftrag mit dem geringsten Schlupf, wobei Schlupf der Abstand zwischen dem aktuellen Zeitpunkt und der Differenz aus Fertigstellungstermin und Restbearbeitungszeit ist.

Diese Heuristiken eignen sich nicht für alle Zielsetzungen der Ablaufplanung gut. So bringt SPT zwar eine maximale Kapazitätsauslastung und sehr geringe Durchlaufzeiten, aber auch große Terminabweichungen mit sich, während SL eine hohe Termintreue ermöglicht, dagegen aber nur mäsige Durchlaufzeiten und eine nur gute Kapazitätsauslastung zur Folge hat (siehe dazu [Jehle et al. 1990, Abbildung 28, Seite 72]).

Produktionsplanungssysteme liefern nicht nur „schlechte“ Planungsergebnissen, sondern reagieren sehr empfindlich auf Störungen oder Änderungen in der Umgebung. Sie erstellen gestützt auf eine Nachfrageprognose zunächst einen globalen Plan der herzustellenden Produkte, um anschließend den Materialbedarf zu ermitteln und danach die Kapazitäts- und Terminplanung durchzuführen[Jahnke und Biskup 1999, Abschnitt 2.1, Die Planungslogik typischer PPS-Systeme]. Kommt es im Verlauf des späteren Produktionsprozesses aber zu Störungen, etwa auf Auftragsebene durch inhaltliche oder terminliche Änderungen, oder auf der Resourcenebene durch Maschinenausfälle[Corsten und Gössinger 1997] muss im schlimmsten Fall das System angehalten und neu gestartet werden[Shen et al. 2001, Seite 234, 1. Absatz].

Ein Ausweg aus diesem Dilemma ist der Ansatz der Dekomposition
[Jahnke und Biskup 1999, Abschnitt 1.5, Dekompositionstrategien in der Produktionsplanung], der das Gesamtproblem der Produktionsplanung in kleinere Teilprobleme unterteilt, die einfacher zu lösen sind. Prinzipiell gibt es bei der Produktionsplanung keinen Planungshorizont, da der Betrieb auf unbestimmte Zeit arbeitet. Die rollierende Planung als Dekompositionsstrategie setzt jedoch einen Planungshorizont fest, bis zu dem der Plan ausgearbeitet wird und plant in gewissem Rythmus neu. So kann sie auf Änderungen der Auftragslage oder Störungen reagieren. Unter Umständen kann die rollierende Planung aber zu sehr schlechten Plänen führen: ist der erste Schritt eines Planes sehr kostenintensiv und wird nur dieser bis zur nächsten Neuplanung ausgeführt, kann

sich im schlimmsten Fall ein kostenmaximaler Plan ergeben. Desweiteren kann sich eine Systemnervosität ergeben, gerade wenn sich die Pläne von Schritt zu Schritt stark unterscheiden. Die ständige Umstellung der Fertigung verursacht aber nicht unerhebliche Kosten.

## 2.4 Ein reduziertes Problem der Produktionsplanung

Wie bereits in Abschnitt 2.2.3 auf Seite 12 erwähnt, soll das Problem der Ablaufplanung mit maschinenbezogener Zielsetzung zugrundeliegende Aufgabenstellung für die weiteren Untersuchung. Dabei werden folgende Annahmen getroffen:

1. Es liegt eine Werktstattfertigung vor[Jahnke und Biskup 1999, Abschnitt 3.2, Ablaufplanung].
2. Die Materialbedarfs- und Lagerhaltungsplanung ist gegeben [Jahnke und Biskup 1999, Abschnitt 3.2, Ablaufplanung].
3. Die Rüstzeiten und -kosten für die Betriebsmittel sind fest.
4. Als Beispielbetrieb wird eine kleine Schreinerei mit etwa acht Maschinen und drei Produktarten angenommen (siehe dazu Abschnitt 6.2 auf Seite 56 bzw. Anhang 7.2 auf Seite 75)
5. Der Transport der Teilprodukte zwischen den Betriebsmitteln wird als gegeben vorrausgesetzt. Dies ist nicht realistisch, da gerade bei der Werkstattfertigung der Transport eine große Rolle spielt. Es würde jedoch den zeitlichen Rahmen dieser Arbeit sprengen, den Transport mit zu implementieren.
6. Es gibt Abschätzungen über die typischen Produktionszeiten der einzelnen Produkte auf den Maschinen um einfach zeitliche Abschätzungen durchführen zu können, die sonst einen erheblichen Kommunikationsaufwand bedeutet hätten.

# 3 Multiagentensysteme

Dieses Kapitel soll in den Themenbereich der Multiagentensysteme (MAS) einführen. Abschnitt 3.1 klärt als den Begriff des "Agenten" und anschließend den eines Multiagentensystems. Abschnitt 3.2 auf der nächsten Seite beschreibt die Methode zur Analyse und Design von Multiagentensystemen, die im Rahmen der Diplomarbeit angewandt werden sollen. Die darin gefundenen Basiselemente liefern Anhaltspunkte für die Auswahl einer geeigneten Agentenplattform in Abschnitt 3.4 auf Seite 25.

## 3.1 Was ist ein Multiagentensystem?

Zunächst soll geklärt werden, was unter dem Begriff eines "Agenten" verstanden wird. Spätestens seit dem Lehrbuch von Russell und Norvig betrachtet man diesen als einen zentralen Begriff der künstlichen Intelligenz [Russell und Norvig 1995]. Er wurde bisher meist in Hinblick auf einen einzelnen Agenten und dessen mentales Modell untersucht. Dabei hat es eine Weile gedauert, bis sich in der Forschergemeide akzeptierte Kategorien [Wooldridge und Jennings 1995] ausgebildet haben und Agenten in Taxonomien geordnet wurden [Franklin und Graesser 1997, Nwana 1996]. Dennoch sind auch diese Ergebnisse nicht unumstritten, was [Castelfranchi 1997] und [Wooldridge 1997] darlegen. Als Begriff eines Agenten scheint sich jedoch folgende Definition zu etablieren:

> "An agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its design objectives." [Jennings und Wooldridge 2000][1]

Also ein System, das die Eigenschaften *Autonomie* und *Flexibilität* hat. Das bedeutet, dass es sich proaktiv verhält und selbst entscheiden kann wann es was tut. Es bedeutet weiter, dass die Aktionen des Systems von der Situation abhängig sind und nicht bloß nach einem Reiz-Reaktionsschema verlaufen. Demzufolge muss das System natürlich auch eine Vorstellung von seinen Handlungsmöglichkeiten bzw. deren Konsequenzen haben. Diesen Umständen trägt das "BDI"-Modell Rechnung, das einem Agenten Überzeugungen (Beliefs), Wünsche bzw. Motivationen (Desires) und konkrete Vorhaben (Intentions) zuschreibt [Wooldridge und Jennings 1995]
[Rao und Georgeff 1995]. Es ist das wohl wichtigste und zur Zeit am weitesten verbreitete mentale Modell für Agenten.

[1] Ein Agent ist ein abgeschlossenes Computersystem, das in einer bestimmten Umgebung arbeitet. Es ist in der Lage, darin flexibel und autonom zu agieren, um seine vorgegebenen Ziele zu erreichen.

Multiagentensysteme sind Populationen von theoretisch beliebig vielen Agenten. Diese arbeiten mal mit- und mal gegeneinander und verfolgen entweder Global- oder Einzelziele. Auch Mischformen sind möglich. Bei ihnen steht dementsprechend die *Interaktion*, also Koordinations- und Planungsmechanismen, die die unterschiedlichen Agenten ein Problem verteilt lösen lassen (z.B. [Bürckert et al. 1998]), im Vordergrund. Hierfür werden zum Beispiel marktliche Mechanismen [Zelewski 1997], unterschiedliche Auktionsformen [McAfee und McMillan 1987] oder Verhandlungsmechanismen, wie z.B. Kontrakt-Netze [Smith und Davis 1981] als mögliche Ansätze herangezogen. Während sich für typische, in Multiagentensystemen anfallende Aufgaben, wie die Kommunikationsprache zwischen Agenten [Finin et al. 1994, FIPA/CA 2000, FIPA/ACL 2000, FIPA/CL 2000], das (hardwaregrenzenübergreifende) Management von Agenten oder Such- und Verzeichnisdienste [FIPA/AMS 2000], bereits Standards herausbilden – wichtige Elemente, die Interoperabilität zwischen verschiedenen Agentensystemen erst ermöglichen – und verschiedene Agentenshells verfügbar sind[ABUILDER 2000], die zwischen Agenten mit unterschiedlichen mentalen Modellen wählen lassen, ist der Entwurf eines Multiagentensystems zu einem bestehenden Problem noch recht wenig erforscht.

Zusammenfassend lässt sich sagen, das *Autonomie*, *Flexibilität* und *Interaktion* wesentliche Merkmale eines Multiagentensystems sind.

## 3.2 Konstruktion von Multiagentensystemen

Ziel dieses Abschnittes ist es, eine Methode zur Analyse und Design von Multiagentensystemen zu finden, die im Rahmen dieser Diplomarbeit eingesetzt wird.

In letzter Zeit erfreuen sich Multiagentensystemen zwar immer größerer Beliebtheit, der Analyse und dem Design wurden bisher noch recht wenig Aufmerksamkeit gewidmet. Auch wenn es bereits Vorschläge für Musterarchitekturen [Horn und Reinke 2000] gibt, sind diese jedoch kaum in den Entwicklungsprozess von Multiagentensystemen vorgedrungen. Selbst die seit geraumer Zeit entwickelten Standards in Bezug auf Multiagentensysteme [FIPA/AMS 2000, MASIF 1997], beschäftigen sich eher mit grundlegenden Aspekten, wie die Verwaltung von Agenten, dem Nachrichtenaustausch zwischen Agenten, Such- bzw. Verzeichnisdienste oder Sprachdefinitionen. Die eigentliche Struktur eines Multiagentensystems, d.h. was ein Agent ist und wie z.B. die Interaktion aussieht, liegt in der Hand des jeweiligen Entwicklers. Im Gegensatz zur Softwaretechnologie (z.B. [Meyer 1988]) haben sich für den Entwurf von Multiagentensystemen noch keine allgemein anerkannten Methoden etabliert und Publikationen in diesem Bereich unterstreichen den jungen Charakter dieses Forschungsfeldes [Jennings und Wooldridge 2000]. Eine Methodenbildung wird unter anderem von [Iglesias et al. 1999] aber als notwendig erachtet, damit sich Multiagentensysteme auch in der Industrie als anerkanntes Paradigma durchsetzen.

Die bisher noch wenigen Ansätze zu dem Thema haben Iglesias etal.
[Iglesias et al. 1999] aufgeführt und darin gemeinsame Grundbausteine identifiziert. Daran angelehnt haben [Wooldridge et al. 2000] eine Methode zur Analyse und Design entwickelt, an der sich die Entwicklung des Multiagentensystems im Rahmen dieser

Diplomarbeit orientieren soll.

Die weiteren Abschnitte gliedern sich wie folgt: Abschnitt 3.2.1 fasst zunächst die Ergebnisse von Iglesias etal. zusammen, Abschnitt 3.2.2 auf der nächsten Seite stellt die Methode von Wooldridge etal. vor und Abschnitt 3.2.3 auf Seite 21 geht abschließend auf noch offene Punkte ein.

### 3.2.1 Bisherige Arbeiten zu Analyse und Design von Multiagentensystemen

Iglesias etal. beschreiben unter anderem Ansätze, die Techniken aus dem Softwaretechnologie erweitern und Ansätze basierend auf dem Knowledge Engineering. Beispielhaft für erstere sei der Artikel von [Burmeister 1996] genannt, der agenten-orientierte Techniken damit motiviert, dass diese eine natürliche Fortsetzung objektorientierter Techniken sind. Objektorientierte Techniken reichen aber an verschiedenen Punkten nicht aus:

- Die innere Struktur und das Verhalten von Agenten ist wesentlich komplexer als die einzelner Objekte und Agenten sind eher mit Teilsystemen vergleichbar. Es gibt keine Techniken, mit denen man die Planungs- und Inferenzprozesse modellieren kann, die z.B. im BDI-Modell [Wooldridge und Jennings 1995] [Rao und Georgeff 1995] auftreten.

- In Multiagentensystemen spielt die soziale Struktur der Population eine große Rolle, sie ist zudem auch sehr dynamisch und nicht mit einer Vererbungshierarchie oder Assoziationen vergleichbar.

- Die Interaktion zwischen Agenten beschränkt sich nicht auf einfache Nachrichten (etwa Methodenaufrufe), sondern besteht aus komplexen Nachrichten unterschiedlichen Typs [FIPA/ACL 2000, FIPA/CA 2000]. Die Interaktion bei Agenten ist weiter auch kontextabhängig und erstreckt sich meist über mehrere Nachrichten hinweg.

Über Methoden basierend auf Knowledge Engineering sagen Iglesias etal., dass in Agentensystemen zwar die entsprechenden Probleme der Wissensverarbeitung vorhanden sind, die Ansätze aber dem verteilten Charakter von Multiagentensystemen nicht Rechnung tragen. Weiter beziehen sich diese Ansätze "nur" auf die Definition des Wissens der Agenten – leisten da auch wertvolle Dienste – aber nicht auf den gesamten Prozess bis zum Entwurf des Multiagentensystems. Daher sollen diese Ansätze hier nicht näher betrachtet werden, auch wenn sie mit Erfolg in einigen Projekten zum Einsatz kamen.

Iglesias etal. finden in den von ihnen verglichenen Ansätzen gemeinsame Aspekte, die in der Analyse und dem Design von Multiagentensystemen ausgearbeitet werden müssen:

**Agenten Modell:** Fähigkeiten, Wissen und Aufgaben der Agenten, sowie innerer Aufbau (z.B. ob rein reaktiv oder BDI-Modell).

**Soziale Struktur:** Bekanntschaften zwischen den Agenten, Rollen, die die Agenten einnehmen können, sowie mögliche Interaktionen.

### 3.2.2 Analyse und Design von Multiagentensystemen – Die Methode von Wooldridge etal.

[Wooldridge et al. 2000] nehmen die von Iglesias etal. identifizierten Elemente auf und entwickeln eine Methodologie zur agenten-orientierten Analyse und Design. Sie bezeichnen ihren Prozess als *organisational design* und teilen ihn in eine abstrakt gehaltene Analyse- und eine konkretere Designphase ein.

#### Analysephase

Ziel der Analysephase ist es, das System und seine Struktur zu verstehen, also die Organisation des Systems zu erfassen. Die Ergebnisse dieser Phase müssen nicht notwendigerweise ein Gegenstück im späteren System haben.

In dieser Phase soll zunächst ein **Rollenmodell** gebildet werden, das die verschiedenen benötigten Rollen zusammen mit Rechten und Verantwortlichkeiten identifiziert. Rechte (permissions) beziehen sich hier auf die Art und Menge der Ressourcen, die einer Rolle zur Verfügung stehen, und darauf wo und wie der Ressourcenverbrauch begrenzt ist. Verantwortlichkeiten definieren im wesentlichen die Funktionalität einer Rolle, und umfassen neben den auszuführenden Tätigkeiten (*liveness expressions*) auch einzuhaltende Bedingungen (*safety expressions*). Erstere sind als über Operatoren verknüpfte Tätigkeiten formuliert, letztere als sicherzustellende Bedingungen (vergleichbar mit Invarianten).

Neben dem Rollenmodell soll in der Analysephase auch das **Interaktionsmodell** gebildet werden, das aus einer Menge von Protokolldefinitionen für Interaktionen zwischen Rollen besteht. Eine solche Definition wird durch folgende Felder beschrieben: Zweck der Interaktion, Initiator, Partner, Eingaben (von Seiten des Initiators), Ausgaben (von Seiten des Partners) und Arbeitsschritten (die der Initiator während der Interaktion durchführt). Auf eine genaue Spezifikation der einzelnen Nachrichten die ausgetauscht werden, wird in der Analysephase bewusst verzichtet.

Nach Entwicklung des Interaktionsmodells kann das Rollenmodell meist schon verfeinert werden. Wooldridge etal. schlagen daher vor, diese beiden Schritte zu iterieren. Tabelle 3.1 auf der nächsten Seite enthält eine Vorlage für Rollenspezifikationen.

#### Designphase

Die Ergebnisse der Analysephase sind Basis für die Designphase. Deren Ziel ist es im Gegensatz zu Designphasen aus der Softwaretechnologie nicht, das System soweit zu spezifizieren, dass man es danach direkt implementieren kann, sondern vielmehr die Ergebnisse der Analysephase soweit umzusetzen, dass man anschließend mit traditionellen Methoden aus der Softwaretechnologie fortfahren kann. Es sollen dabei drei Modelle gebildet werden. Erstens ein **Agentenmodell**, das Agententypen anhand der von ihnen realisierten Rollen beschreibt, eine Vererbungshierarchie wird aber noch ausgelassen. Auch

| Rollen Schema | *Name der Rolle* |
|---|---|
| Beschreibung | *kurze Beschreibung der Rolle* |
| Protokolle | *Interaktionen an denen die Rolle teilnimmt* |
| Rechte | *Rechte der Rolle* |
| Verantwortlichkeiten | |
| Handlungen | *Aktivitäten* |
| Zusicherungen | *Invarianten* |

Tabelle 3.1: Vorlage für Rollenschemata

die ungefähre Anzahl der Instanzen je Typ soll angegeben werden, d.h. ob nur ein Agent oder mehrere oder gar beliebig viele. Zweitens ein **Servicemodell**, das die Funktionen bzw. Hauptaufgaben der einzelnen Agententypen mit Vor- und Nachbedingungen sowie Ein- und Ausgaben beschreibt (dies kann man aus den Protokollen und den Rollenverantwortlichkeiten ableiten). Drittens ein **Bekanntschaftsmodell**, das zusammenfasst, welche Agententypen miteinander kommunizieren und somit die Kommunikationspfade im System beschreibt.

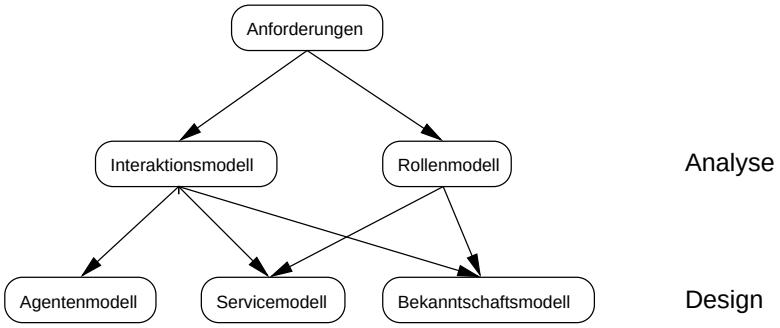



Abbildung 3.1: Zusammenhang zwischen Elementen aus Analyse und Design

#### Nach den beiden Phasen

Die so gewonnenen Modelle sollen dann als Basis für einen normalen Softwareentwicklungsprozess dienen, um letztendlich zu einem implementierten Multiagentensystem zu kommen. Für eine beispielhafte Anwendung der Methode siehe [Wooldridge et al. 2000] und [Bussmann et al. 2001].

### 3.2.3 Diskussion des Ansatzes von Wooldridge etal.

Die von Wooldridge etal. vorgeschlagene Methode ist durchaus vielversprechend, da sie die wesentlichen Merkmale, die Iglesias etal. identifiziert haben, enthält. Auch wenn sie bereits mit Erfolg zum Einsatz kam [Jennings et al. 1996], ist zum Bau eines größeren

Systems – wie aus der Softwaretechnologie bekannt – Erfahrung nötig. Unter anderem sind folgende Punkte anzumerken:

- Bei großen Systemen müssen im schlimmsten Fall bei $n$ Agenten $\sum_{i=1}^{n-1} i = \frac{n^2-n}{2}$ Interaktionen (jeder mit jedem) modelliert werden. Eine angemessene soziale Struktur ist notwendig, um die Kommunkationspfade zu beschränken.
- Es gibt zwar inzwischen eine Anzahl von Interaktionsprotokollen (z.B. Kontrakt-Netze, Auktionen oder Verhandlungsmechanismen), die sich etabliert haben, Wooldridge etal. berücksichtigen diese allerdings nicht. Dies mag daran liegen, das noch keine Angaben dazu existieren, für welche Aufgabenstellung welche Protokolle am besten geeignet sind. Andere Ansätze stellen die Interaktion bewusst in den Vordergrund, beschränken sich dennoch auf prinzipielle Betrachtungen, indem sie z.B. Wissen über die Interaktionsprotokolle in das Kommunikationsmedium hinein verlagern [Omicini und Zambonelli 2000] und ebenfalls keine Anregungen zur Wahl der Protokolle geben.
- Multiagentensysteme sind komplexe Systeme, die verteiltes, paralleles Problemlösen umsetzen. Problemstellen sind daher auch die richtige Anzahl von Agenten und richtige Ausgestaltung der Eigenintelligenz der einzelnen Agenten (siehe auch [Wooldridge und Jennings 1998]).

Die Methode von Wooldridge etal. ist also nicht ultima ratio und sollte eher als Leitfaden bei der Entwicklung des Multiagentensystems dienen.

## 3.3 Interaktionsprotokolle/Koordinationsmechanismen

Die Interaktion ist ein wesentlicher Bestandteil eines Multiagentensystems (siehe auch Abschnitt 3.1 auf Seite 17). Zur Koordination der unterschiedlichen, sich mitunter widersprechenden Ziele der einzelnen Agenten bzw. Agentengruppen, haben sich einige Interaktionsprotokolle – Koordinationsmechanismen – etabliert, die in einem Agentensystem aber nicht immer exklusiv verwendet werden. Vielmehr kommen innerhalb eines Systems bzw. Agentens oft mehrere Koordinationsmechanismen parallel zum Einsatz. Sie kommen aus so unterschiedlichen Bereichen, wie der verteilten künstlichen Intelligenz, Soziologie oder Betriebswirtschaftslehre. Im folgenden sollen die bekanntesten beschrieben werden: Auktionen, marktlichen Mechanismen und Kontraktnetze.

### 3.3.1 Auktionen

Eine Art der Interaktionsprotokolle sind Auktionen. Dabei wird immer ein Objekt (oder Auftrag) zur Versteigerung ausgeschrieben und Bewerber können Gebote dafür abgeben. Die vier grundlegenden Auktionsformen unterscheiden sich danach, ob offen oder verdeckt geboten wird und ob der zu zahlende Preis vom ersten oder zweiten Bieter abhängt. Die Wahl einer Auktion als Interaktionsprotokoll impliziert, dass wesentliche Eigenschaften sowohl der Agenten als auch der Versteigerungsobjekte als Zahl modelliert

werden müssen, was nicht immer gelingen wird. Die Auktion als Interaktionsprotokoll hat jedoch ihren Vorteil in ihrer Einfachheit: nicht nur der Implementierungsaufwand, sondern auch die Anzahl der ausgetauschten Nachrichten ist relativ gering, was die Auktion auch für größere Agentenpopulationen interessant macht.

Folgende Auktionsformen sind gebräuchlich (siehe auch [Klemperer 2000]):

**Englische Auktion:** Hier wird offen und mehrmals mit aufsteigendem Preis geboten. Die Auktion endet, wenn es keine Mitbieter mehr gibt, d.h. in einer gewissen Zeitspanne kein höheres Gebot mehr beim Auktionator eingeht. Interessanterweise hängt, der zu zahlende Preis lediglich von der Zahlungsbereitschaft des letzten Mitbieters (zweiter Bieter) ab. Nachteilig an dieser Auktionsform ist, das das Ende zeitlich nicht abzuschätzen ist. Dies ist die Auktionsform, die in englischen Auktionshäusern praktiziert wird.

**Holländische Auktion:** Auch hier wird offen geboten, jedoch mit absteigendem Betrag. Der Auktionator beginnt mit einem Höchstbetrag und reduziert diesen solange, bis sich ein Bieter meldet (erster Bieter). Sinkt der Preis unter ein bestimmtes Niveau oder ist eine gewisse Zeitspanne überschritten, so verfällt das Angebot. Der Vorteil dieser Auktionsform liegt in der zeitlichen Limitierung. Sie wird vorwiegend im holländischen Blumengrosshandel aber auch im israelischen Fischgrosshandel eingesetzt.

**Vickery Auktion:** Unter diesem Begriff fallen die beiden verdeckten Auktionsformen, die sich darin unterscheiden, nach wem sich der Preis richtet, also dem ersten oder dem zweiten Bieter. In beiden Fällen wird von jedem Bieter genau ein verdecktes Gebot abgegeben und der meistbietende erhält den Zuschlag. Im ersten Fall muss er sein Gebot (erster Bieter), im zweiten Fall das zweithöchste Gebot (zweiter Bieter) bezahlen [Vickery 1961]. Diese Auktionsform wird oft bei der Versteigerung von Briefmarken auf dem Postweg verwendet.

### 3.3.2 Elektronische Märkte

Bei marktlichen Mechanismen werden zwei Typen von Agenten unterschieden: Güter austauschende *Konsumenten* und Güter verwertende *Produzenten*, wobei ein Produzent natürlich gleichzeitig Konsument seiner Basisgüter sein kann. Zwischen diesen Parteien kommt es zu Verhandlungen über den zu zahlenden Preis für ein Gut, die sich an der Maximierung der Nützlichkeit – gemessen als Skalar – orientieren. Idealerweise erreicht der Markt ein Gleichgewicht (Equilibrium), in dem maximale Nützlichkeit und maximaler Profit bei allen Beteiligten erlangt werden. Für einen Markt ist es notwendig, die Güter, die Konsumenten, die Produzenten mit ihren "technischen" Eigenschaften und vor allen das Biete- und Handelverhalten der Agenten zu modellieren. Leider müssen aber viele Aspekte durch einen Preis beschrieben werden, weshalb sich wie bei den Auktionen das Problem stellt, wichtige Eigenschaften auf eine Zahl zu reduzieren. Es gibt bei diesem Verfahren keinen Mechanismus, der das Geschehen explizit kontrolliert. Ist eine große Anzahl von Agenten im System vorhanden, eigenen sich Marktmechanismen gut, da der

Kommunikationsaufwand vergleichsweise gering ist. Damit geht eine gute Erweiterbarkeit des Systems einher, da neue Agenten nur eine kleine Schnittstelle implementieren müssen.

### 3.3.3 Kontraktnetze

Dieser bereits 1980 in die verteilte KI eingeführte Koordinationsmechanismus löst das Problem der Zuordnung von Aufgaben an Agenten [Smith 1980]. Die beiden hier vorkommenden Agententypen, *Manager* und *Contractor* finden über eine feste Vorgehensweise zu einer Verteilung der Aufgaben. Wie bei den Marktmechanismen können Agenten beide Rollen übernehmen, etwa wenn eine Aufgabe aus verschiedenen Teilaufgaben besteht. Jeder Manager hat dabei die Kontrolle über seinen Bereich, so dass im System letztendlich eine verteilte Kontrolle über den Gesamtprozess ausgeübt wird. Dabei geht ein Manager folgende Punkte ab:

1. Ausschreibung einer Aufgabe
2. Entgegennahme von Angeboten
3. Erteilung eines Zuschlages
4. Entgegennahme des Ergebnisses

Ein Contractor vollzieht folgende Schritte:

1. Ausschreibung erhalten
2. Bewertung der eigenen Fähigkeiten bezüglich der Aufgabe
3. Abgabe eines Angebots
4. Ausführung und Abgabe des Ergebnisses, sofern der Auftrag erteilt wird.

Ein mögliches Problem kann die Tatsache werden, dass der am besten für eine Aufgabe geeignete Agent gerade beschäftigt ist, und ein Auftrag daher an einen weniger geeigneten Agenten geht.

### 3.3.4 Andere Interaktionsprotokolle

Ebenfalls in den Bereich der Interaktionsprotokolle lassen sich die Verhaltensweisen *Broker* und *Mediator* einordnen. Beide sind darauf gerichtet, dass eine Gruppe von Agenten zusammenarbeitet bzw. zusammenfindet. Ein Broker lässt sich mit einem Suchdienst vergleichen, der auf Anfragen hin Adressen passender Agenten zurückliefert. Ein Mediator dagegen arbeitet als Mittelsmann und Koordinator einer Gruppe von Agenten und leistet eventuell notwendige Übersetzungsarbeit zwischen den Agenten.

## 3.4 Der FIPA Standard

Bei der Implementierung komplexer Systeme, ist es sinnvoll auf Standards zurückzugreifen. Bezüglich der Multiagentensysteme sind dies die Spezifikationen des FIPA Standards [FIPA], die bereits in verschiedenen Implementierungen vorliegen. Von den momentan[2] verfügbaren, soll später eine anhand einfacher Kriterien für diese Arbeit ausgewählt werden.

### 3.4.1 Der FIPA-Standard

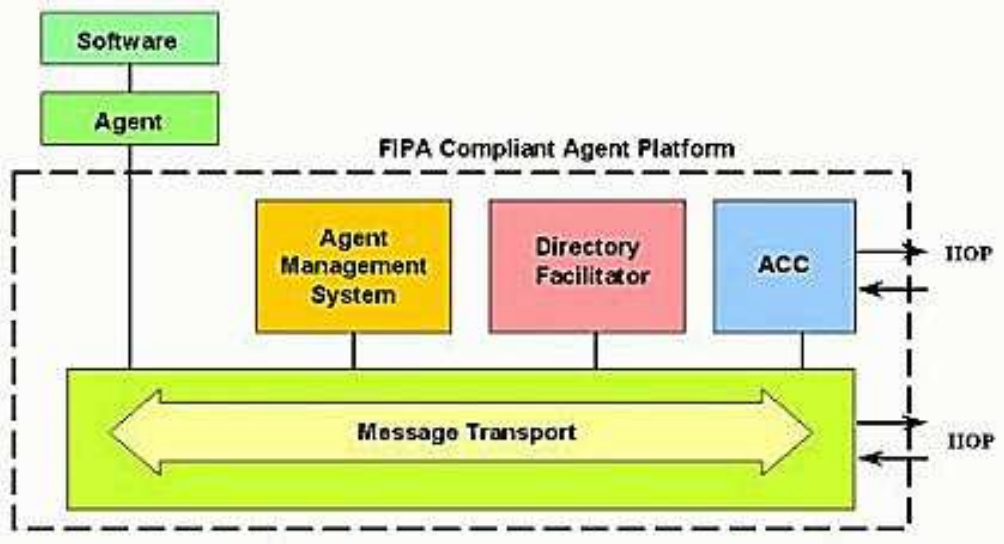



Abbildung 3.2: FIPA Agentenplattform

Der Grundgedanke des FIPA-Standards ist es, das mentale Modell der einzelnen Agenten so wenig wie möglich festzulegen, um eine große Zahl verschiedenster Agenten und Agententypen miteinander arbeiten lassen zu können. Der Schwerpunkt der FIPA-Spezifikationen liegt in der Definition grundlegender Mechanismen zur Verwaltung vieler, verschiedener Agenten über Hardwaregrenzen hinweg. Die Spezifikationen werden im Internet publiziert [FIPA].

#### Agentenplattform

Basis des Standards ist eine Agentenplattform, die jeweils auf einem Rechner läuft – die Zusammenarbeit mehrerer Plattformen über Hardware- und Harwareplattformgrenzen hinweg ist vorgesehen. Eine Plattform besteht dabei neben den eigentlichen Agenten aus den folgenden Elementen (siehe auch Abbildung 3.2) [FIPA/AMS 2000]:

[2] April 2001

**Message Transport:** Eine Transportebene, die den Nachrichtenaustausch innerhalb einer Instanz einer Plattform organisiert.

**Agent Communication Channel:** Dieser Baustein übernimmt den plattformübergreifenden Nachrichtenaustausch. Dazu wird das IIOP[3] verwendet.

**Directory Facilitator:** Ein Verzeichnisdienst, in dem die Agenten Dienste (*Services*) anbieten bzw. danach suchen können. Dieser Dienst erlaubt prinzipiell eine verteilte Suche über mehrere physikalisch getrennte Plattformen hinweg.

**Agent Management System:** Dieser Teil kontrolliert den Zugriff auf die Agentenplattform und vergibt eindeutige ID's an Agenten. Alle Agenten aus der Plattform werden hier eingetragen.

### Agentenkommunikation

Auf den Nachrichtenaustausch zwischen den Agenten beziehen gleich mehrere FIPA-Spezifikationen. [FIPA/ACL 2000] legt zunächst die Struktur einer Nachricht fest. Deren wesentliche Elemente sind Absender und Empfänger, verwendete Sprache, Inhalt, Interaktionskontext und Art des kommunikativen Aktes. Letzerer basiert auf der Theorie der Sprechakte, die besagt, dass kommunikative Äußerungen auch immer Handlungen sind (die sich wiederum in verschiedene Klassen einordnen lassen) [Austin 1979]. Verschiedene kommunikative Akte spezifiziert [FIPA/CA 2000], z.B. "Bestätigung", "Ablehnung", "Einverständnis", "Abbruch", "Fehler" oder "Information". Wie erwähnt, hat jede ausgetauschte Nachricht einen Inhalt. [FIPA/CL 2000] ist ein Rahmen für die momentan vorgesehenen Sprachen SL[FIPA/SL 2000], KIF[FIPA/KIF 2000], RDF[FIPA/RDF 2000] und CCL[FIPA/CCL 2000]. Es ist nicht zwingend vorgegeben, dass ein Agent eine der Sprachen verstehen muss, einzig die im Austausch mit dem Agenten Management System verwendete Sprache muss er beherrschen. Mit Hilfe der Nachrichten können Agenten nun miteinander kommunizieren, um ihre Handlungen zu koordinieren. Hierfür sind bereits verschiedene Interaktionsprotokolle, wie Auktionen oder Kontraktnetze (siehe auch Abschnitt 3.3 auf Seite 22) spezifiziert [FIPA/IP 2000].

[3] Internet Inter Orb Protocol – Protkollspezifikation aus CORBA, zum Nachrichtenaustausch in einer verteilten Middleware gedacht

# 4 Vorbereitung für Experimente mit einem MAS

Die bisher in der Produktionsplanung und in anderen Bereichen eingesetzten Systeme verfolgen einen zentralisierten Ansatz, d.h. ein einzelnes Programm errechnet den vollständigen Produktionsplan und steuert die Produktion. Aufgrund der hohen Komplexität der Aufgabe und der Tatsache, das lokale Informationen z.B. über Störungen an das System zur Berücksichtigung gemeldet werden müssen, bieten sich Multiagentensysteme als Möglichkeit an, hier bessere Ergebnisse zu erzielen. So können Teile der Berechnung an den Ort der Handlung verlegt werden, um zum einen die Rechenzeit zu verteilen, und zum anderen, um schnell auf lokale Einflüsse zu reagieren. Teilt man nämlich die Aufgabe der Produktionsplanung in eine Grobplanung und eine Feinplanung (vgl. taktische und operative Produktionsplanung, Abschnitt 2.2.1 auf Seite 11), lassen sich die erwähnten Eigenschaften von Multiagentensystemen, also Autonomie, Flexibilität und Interaktion (siehe Abschnitt 3.1 auf Seite 17) ausnutzen. Es ist aber zu prüfen, ob der Einsatz eines Multiagentensystems wirklich eine Verbesserung bringt.

## 4.1 Experimente mit Multiagentensystemen

Multiagentensystemen sind schon seit längerem Gegenstand experimenteller Vergleiche mit klassischen Verfahren der Produktionsplanung. Meist wurde dabei ein bestimmter Koordinationsmechanismus bzw. Varianten desselben untersucht (dazu Abschnitt 2.3 auf Seite 13). Einige Vergleiche sollen im folgenden vorgestellt werden.

So zum Beispiel Krothaphalli und Deshmukh[Krothapalli und Deshmukh 1999]. Ihr Agentensystem besteht aus Maschinen- und Produktagenten, die untereinander die Produktion aushandeln. Krothapalli und Deshmukh implementierten dazu drei unterschiedliche Koordinationsmechanismen, die auf einem Markt bzw. einer Auktion basieren. Die unterscheiden sich in der Art, in der ein Produktagent eine Maschine auswählt: über die kürzeste Bearbeitungsdauer, über den Preis, den ein Kunde bezahlt (die Wartezeit eines Produktes wird dazu in eine Geldsumme umgerechnet) und über die noch verbleibende Zeit bis zum Fertigstellungstermin. Sie vergleichen die drei Koordinationsformen anhand der Summe Wartezeiten eines Auftrags im System, der Durchlaufzeit, der Pünktlichkeit und der Auslastung der Maschinen mit einer zentralen Produktionssteuerung.

Im Gegensatz dazu vergleichen [Saad et al. 1995] einen Kontrakt Netz basierten Ansatz mit klassischen Methoden und ziehen dabei verschiedene Zeitliche Kriterien zu

Rate. Wie auch Krothapalli und Deshmukh kommen sie zu dem Ergebnis, das sich der Einsatz von Multiagentensystemen lohnt. Beide geben aber an, das ihre Experimente erst erste Anfänge waren und das Gebiet noch weiter untersucht werden muss.

### 4.1.1 Kriterien für die spätere Bewertung

Die bisherigen Vergleiche von Multiagentensystemen mit klassischen Produktionsplanungverfahren wurden in Hinblick auf Pünktlichkeit der Produktion, Maximierung der Auslastung oder die Minimierung der Kosten durchgeführt. Für diese Arbeit wurden die Auslastung der Maschinen, sowie die Produktionserfolge als maßgebliches Kriterium in Bezug auf die Anwendung gewählt. Da von einer Werkstattproduktion ausgegangen wird, in der der Kunde typischerweise bereit ist, längere Lieferzeiten in Kauf zu nehmen, wird davon ausgegangen, das genug Planungspielraum für die pünktliche Fertigstellung bleibt. Da die beiden Koordinationsmechanismen miteinander verglichen werden sollen, werden auch die Mengen der ausgetauschten Nachrichten und der Ressourcenverbrauch einander gegenübergestellt.

### 4.1.2 Wahl der Koordinationsmechanismen

Aus den gängigen Koordinationsmechanismen (siehe Abschnitt 3.3 auf Seite 22) wurden das *Kontrakt Netz* und der *Markt* ausgewählt. Das Kontrakt Netz, da dies einer der komplexeren Koordinationsmechanismen ist. Hier wird mit die größte Anzahl von Nachrichten ausgetauscht, dafür ist jedoch eine verteilte Kontrolle des Gesamtprozesses durch die Verträge zwischen den Agenten gegeben. Der elektronische Markt, da hier vergleichsweise wenige Nachrichten ausgetauscht werden, dafür aber keine bzw. nur kaum Kontrolle über den Gesamtprozess möglich ist. Da diese beiden Mechanismen sehr gegensätzlich sind, wurden sie für diese Arbeit ausgewählt.

Auf die Implementierung weiterer Koordinationsmechanismen wurde aus zeitlichen Gründen verzichtet, da es den Rahmen der Arbeit gesprengt hätte.

## 4.2 Anforderungen an ein MAS

Um die Koordinationsmechanismen miteinander vergleichen zu können, musste ein geeignetes Multiagentensystem zum Einsatz kommen. Geeignet bedeutet dabei folgendes:

- Das System sollte konform zum FIPA Standard implementiert sein.
- Das System sollte möglichst gut steuer- und verstehbar sein.
- Das System sollte keine Simulation, sondern ein Einsatz von vollständig programmierten Agenten sein.
- Das System sollte das Problem der Ablaufplanung lösen.
- Das System soll überschaubar sein und möglichst auf zusätzliche Dinge verzichten.

- Es sollte möglich sein, verschiedene Koordinationsmechanismen, sofern nicht schon vorhanden, nachzupflegen.

Da in der Literatur immer Simulationen von Agenten oder sehr komplexe Systeme oder Systeme, die nicht dem FIPA Standard entsprachen, zum Einsatz kamen und keine Problemlösung der Ablaufplanung mit Hilfe von Multiagentensystemen zu finden war, fiel die Entscheidung, ein eigenes Multiagentensystem zu implementieren. Der damit verbundene Aufwand hat Vorteile:

- Durch eine eigene Implementierung ist die Einsicht in die Funktionsweise der Mechanismen größer.
- Die Besonderheiten einer eigenen Implementierungen sind bekannt, und müssen nicht erst, wie bei Implementierungen anderer, „gefunden“ werden.
- Durch eine eigene Implementierung bleibt das System notwendigerweise klein und Nebeneffekte durch unbekannte Optimierungen treten nicht auf.

### 4.2.1 Implementierungen des FIPA Standards

Um ein eigenes Agentensystem auf Basis des FIPA Standards zu implementieren, musste eine geeignete Plattform ausgewählt werden. Zwischen gibt es mehrere ausgereifte Implementierungen (siehe Tabelle 4.1 auf der nächsten Seite) , die zum Teil mit grafischen Oberflächen ausgeliefert werden. Trotz des hohen Reifegrades dieser Produkte, fiel die Wahl auf die am Lehrstuhl für künstliche Intelligenz entwickelte Plattform XFAP. Dies geschah aus folgenden Gründen:

- Es wurden bereits frühere Version von XFAP eingesetzt, so dass vor Ort Erfahrungen mit diesem Agentensystem bestanden
- Der Einsatz der Agentenplattform trägt zur Weiterentwicklung derselben bei.
- Da der Entwickler Stefan Haustein am Lehrstuhl arbeitet, war ein direkter und schneller Kontakt möglich. Bei anderen Systemen hätte es unter Umständen lange dauern können, bis Fehler behoben werden.
- Die Agentenplattform XFAP ist recht „klein“ und damit gut überschaubar. Der Einarbeitungsaufwand ist entsprechend geringer als bei umfangreicheren Umgebungen.

| System | Hersteller | Lizenz | Sprache |
|---|---|---|---|
| FIPA-OS | Freie Entwickler | Freie Software (GPL) | Java |
| JADE | CSELT (inzw. Telekom Italia) | Freie Software (LGPL) | Java |
| XFAP | Stefan Haustein | Freie Software (GPL) | Java |
| Zeus | British Telecom | Freie Software | Java |
| April | Freie Entwickler | Freie Software (LGPL) | Java |

Tabelle 4.1: Implementierungen des FIPA Standards

# 5 Ein Multiagentensystem für die Produktionsplanung

In diesem Kapitel soll dass für die Experimente implementierte Multiagentensystem vorgestellt werden. Bei seinem Design und Entwurf habe ich mich grob an die von Wooldridge etal. beschriebene Methode (siehe Abschnitt 3.2.2 auf Seite 20) gehalten. Viele Aspekte ergaben sich jedoch bereits aus der Aufgabenstellung, so dass z.B Analyse- und Design-Phase zusammenfallen. Es folgt die Analyse der notwendigen Bestandteile für das MAS, der Entwurf des Datenmodells, die konkrete Beschreibung interner Mechanismen der Agenten, sowie ein kurzer Abriss der übrigen verwendeten Hilfsmittel. Das Kapitel wird mit einer Beschreibung der Besonderheiten abgeschlossen, die bei der Implementierung auffielen.

## 5.1 Analyse und Design des MAS

Die Anforderungen an das System sind klar abgegrenzt: Aus technischer Sicht sollte ein Multiagentensystem implementiert werden, dass die beiden Koordinationsmechanismen Kontrakt Netz und elektronischer Markt beherrscht (siehe Abschnitt 4.2 auf Seite 28). Aus Sicht der Anwendung war das Problem der Ablaufplanung (siehe Abschnitt 2.4 auf Seite 15) zu lösen, d.h. die Planung der zeitliche Abfolge auf den einzelnen Maschinen.

Daraus ergaben sich die allgemeinen Rollen *Maschine*, *Buchhaltung* und Lagerhaus. Hinzu kommen für das Kontrakt Netz Maschinen, die Auftragnehmer und Auftraggeber sind, sowie Kunden die als Auftraggeber auftreten; für den Markt eine *Marktplattform*, die als Broker Produzenten und Maschinen zusammenbringt, ein *Produzent* als Mediator, der eingehende Kundenaufträge annimmt und deren Teilaufträge über den Markt weiter gibt, sowie Maschinen, die über den Markt Aufträge erhalten. Die Schemata in den Tabellen 5.1 auf der nächsten Seite bis 5.6 auf Seite 34 präzisieren die Aufgabenbereiche.

Wie aus den Rollenschemata ersichtlich, sind neben den Koordinationsmechanismen Kontrakt Netz und Markt kaum weitere Interaktionsprotokolle notwendig, die über eine einfach Abfrage und Übermittlung von Daten hinausgehen. Sie wurden daher nicht explizit modelliert.

In der Designphase werden aus den Rollenschemata die verschiedenen Agententypen gewonnen, die jeweils eine oder auch mehrere Rollen umsetzen können. In dem vorliegenden Anwendungsfall entsprechen diese aber bereits den Rollenschemata, weshalb an dieser Stelle nur die Anzahl der Vorkommen angegeben wird. So gibt es in einem laufenden System genau einen Buchhaltungsagenten, beliebig viele Lageragen-

| Rollen Schema | **Maschine** |
|---|---|
| Beschreibung | Diese Rolle repräsentiert eine Maschine und übernimmt die lokale Ablaufplanung, sowie Steuerung der Produktion. Die Maschine muss die notwendigen Rohstoffe aus einem Lagerhaus anfordern und kann im allgemeinen davon ausgehen, dass alle Teilprodukte bereits vorhanden sind. |
| Protokolle | Mark, Kontrakt Netz, Abfragen an Buchhaltung und Lagerhäuser |
| Rechte | Kann Rohstoffe aus Lagerhäusern anfordern, Transaktionen bei der Buchhaltung initiieren und eine physikalische Maschine steuern |
| Verantwortlichkeiten | |
| Handlungen | Produziert eingehende Teilaufträge, im Falle des Kontrakt Netzes Weitergabe von Teilaufträgen an andere Maschinen |
| Zusicherungen | Abrechnung der verbrauchten Rohstoffe und hergestellten Produkte (auch Abrechnung im Fehlerfall), evtl. Einlagerung von Teilprodukten im Fehlerfall |

Tabelle 5.1: Rollenschema Maschine

| Rollen Schema | **Buchhaltung** |
|---|---|
| Beschreibung | Diese Rolle verwaltet für jeden vorkommenden Agenten ein Konto. Jeder Agent kann eine Transaktion von oder auf sein Konto veranlassen und Kontostände bzw. Transaktionen abfragen. Es wird davon ausgegangen, dass alle Agenten kooperativ sind, weshalb keine Sicherheitsmaßnahmen getroffen wurden. |
| Protokolle | Abfragen an die Buchhaltung |
| Rechte | Kontoverwaltung |
| Verantwortlichkeiten | |
| Handlungen | Verbucht Transaktionen |
| Zusicherungen | Korrektheit der Konten bzw. Transaktionsabfragen |

Tabelle 5.2: Rollenschema Kontoverwalter

| Rollen Schema | **Lagerhaus** |
|---|---|
| Beschreibung | Diese Rolle verwaltet den Inhalt eines Lagers. In der Regel werden hier Rohstoffe vorgehalten, Produkte können aber auch eingelagert werden (vor allem Teilprodukte unvollständiger Fertigung) |
| Protokolle | Abfragen zur Einlagerung und Bereitstellung |
| Rechte | - |
| Verantwortlichkeiten | |
| Handlungen | Ein- und Auslagerung von Rohstoffen und Produkten |
| Zusicherungen | - (möglich auch: Warnungen bei geringen Lagerbeständen, evtl. Anstoß von Bestellvorgängen) |

Tabelle 5.3: Rollenschema Lagerhaus

| Rollen Schema | **Kunde/Auftraggeber** |
|---|---|
| Beschreibung | Diese Rolle stellt einen Kunden bzw. Auftraggeber dar und gibt Aufträge an das System. Die hergestellten Produkte werden danach in Empfang genommen. |
| Protokolle | Auftragsweitergabe an Produzenten (Markt) oder eine Maschine (Kontrakt Netz) |
| Rechte | Darf Aufträge erteilen |
| Verantwortlichkeiten | |
| Handlungen | Autragserteilung und Statusbericht |
| Zusicherungen | Abrechnung mit dem Kunden |

Tabelle 5.4: Rollenschema Kunde/Auftraggeber

| Rollen Schema | **Markt** |
|---|---|
| Beschreibung | Diese Rolle ist die Plattform für einen Markt, indem eingehende Aufträge an Interessenten gemeldet werden, die wiederrum Angebote an den Auftraggeber geben. Dieser wählt daraufhin eine geeignete Maschine aus. Die Maschinen müssen sich dafür bei dem Markt als Interessenten für gewisse Produktarten eintragen. |
| Protokolle | Weitergabe von Aufträgen an Maschinen |
| Rechte | - |
| Verantwortlichkeiten | |
| Handlungen | Markt |
| Zusicherungen | Weitergabe der Aufträge an potentielle Käufer (Maschinen) |

Tabelle 5.5: Rollenschema Markt

| Rollen Schema | **Produzent** |
|---|---|
| Beschreibung | Diese Rolle nimmt Kundenaufträge entgegen und gibt die jeweiligen Teilaufträge an den Markt weiter. Aus den daraufhin eingehenden Angeboten von Maschinen wird dann ausgewählt. |
| Protokolle | Auftragsannahme von Kunden, Verkäufer bei Markt |
| Rechte | Zerlegung von Aufträgen, Management der Produktion der Teilaufträge |
| Verantwortlichkeiten | |
| Handlungen | s.o. |
| Zusicherungen | s.o. |

Tabelle 5.6: Rollenschema Produzent

ten, Maschinenagenten entsprechend der Situation in dem Betrieb und beliebig viele Kundenagenten (die Anzahl muss nicht der Anzahl der Kunden entsprechen, vorstellbar ist auch ein Kundenagent je Benutzerschnittstelle bzw. Sachbearbeiter). Im Falle des Marktes kommt wenigstens ein Markt und wenigstens ein Produzent hinzu; im Falle des Kontrakt Netzes keine weiteren Agenten.

Das Servicemodell ist ebenso einfach abzuhandeln. Da die Koordinationsmechanismen bereits besprochen sind (siehe Abschnitt 3.3 auf Seite 22) müssten nur noch die fehlenden Interaktionen spezifiziert werden. Diese sind jedoch so einfach strukturiert, dass darauf verzichtet werden kann.

Aus den Rollenschemata ergibt sich, welcher Agententyp mit welchem kommunizieren wird. Die Abbildungen 5.1 und 5.2 auf der nächsten Seite zeigen die Kommunikationspfade für die beiden Koordinationsmechanismen.

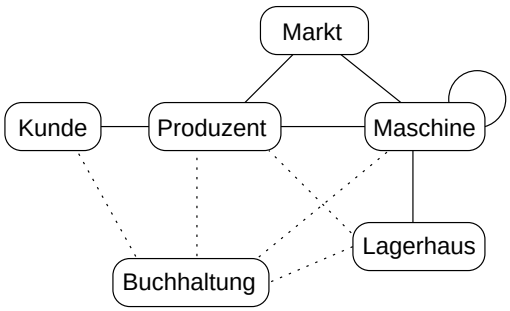



Abbildung 5.1: Kommunikationspfade im MAS mit Markt

Damit kann bereits eine Vererbungshierarchie der Agenten angegeben werden. Die gemeinsamen Eigenschaften aller Agenten (Nachrichten senden, Protokollierung von Nachrichten, etc.) wurden in der Klasse `DAAgent` zusammengefasst. Alle weiteren Agenten erben von dieser Klasse, bis auf `ContractMachine` und `MarketMachine`, die von der Klasse `Machine` abgeleitet sind. Eleganter wäre an dieser Stelle sicher gewesen, der ContractMa-

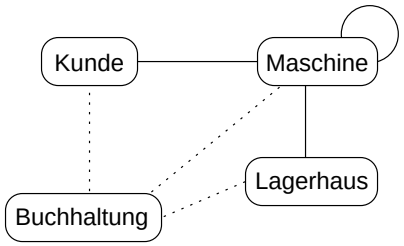



Abbildung 5.2: Kommunikationspfade im MAS mit Kontrakt Netz

schine und AuctionMachine einen Maschinenagenten zur Seite zu stellen. So hätte eine Maschine an zwei verschiedenen Mechanismen teilnehmen oder ihn zur Laufzeit wechseln können. Dies hätte aber einen weiteren, nicht unerheblichen Synchronisationsaufwand nach sich gezogen, weshalb im Rahmen dieser Arbeit darauf verzichtet wurde. Zudem sollten die beiden Koordinationsmechanismen als solche miteinander verglichen werden. Es ist aber ein interessanter Gedanke, etwa bei besonderer Systemlast oder veränderten Auftragsaufkommen durch einen Wechsel des Koordinationsmechanismus darauf reagieren zu können. Die Vererbungshierarchie ist in Abbildung 5.3 dargestellt.

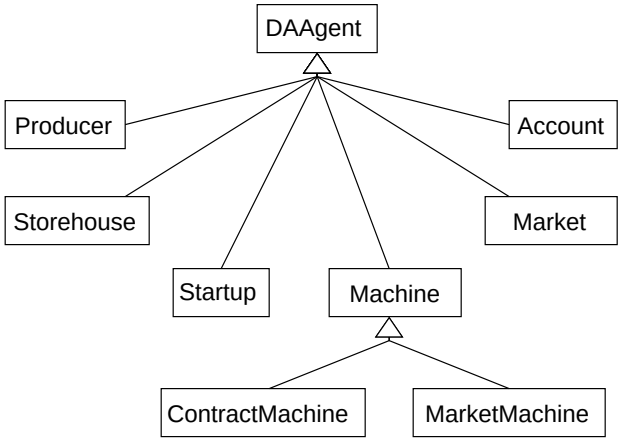



Abbildung 5.3: Vererbungshierarchie der Agenten

## 5.2 Modellierung des Anwendungsbereichs

Im folgenden wird das Datenmodell des implementierten MAS beschrieben. Seine wesentlichen Bestandteile sind Rohstoffe, Produkte und Aufträge, die zwischen den Agenten ausgetauscht werden. Produkte sind dabei nicht direkt vertreten, sondern durch Arbeitspläne und Datenfelder von Aufträgen repräsentiert.

Alle abgebildeten Klassendiagramme sind um die Methoden zur Serialisierung und eventuelle Hilfsvariablen bzw. interne Methoden gekürzt, um den Blick nicht zu verstel-

len. Sie wurden mit dem freien Programm ArgoUML Version 0.9.3[1] erzeugt, das gut dafür geeignet ist Software zu entwerfen und auch umfangreichere Projekte zu handhaben. In den Diagrammen taucht desöfteren der Rückgabewert `void??` auf. Dabei handelt es sich um die Konstruktoren der Klasse, deren korrekte Darstellung in der verwendeten Version noch fehlerhat ist.

### 5.2.1 Rohstoffe und Arbeitspläne

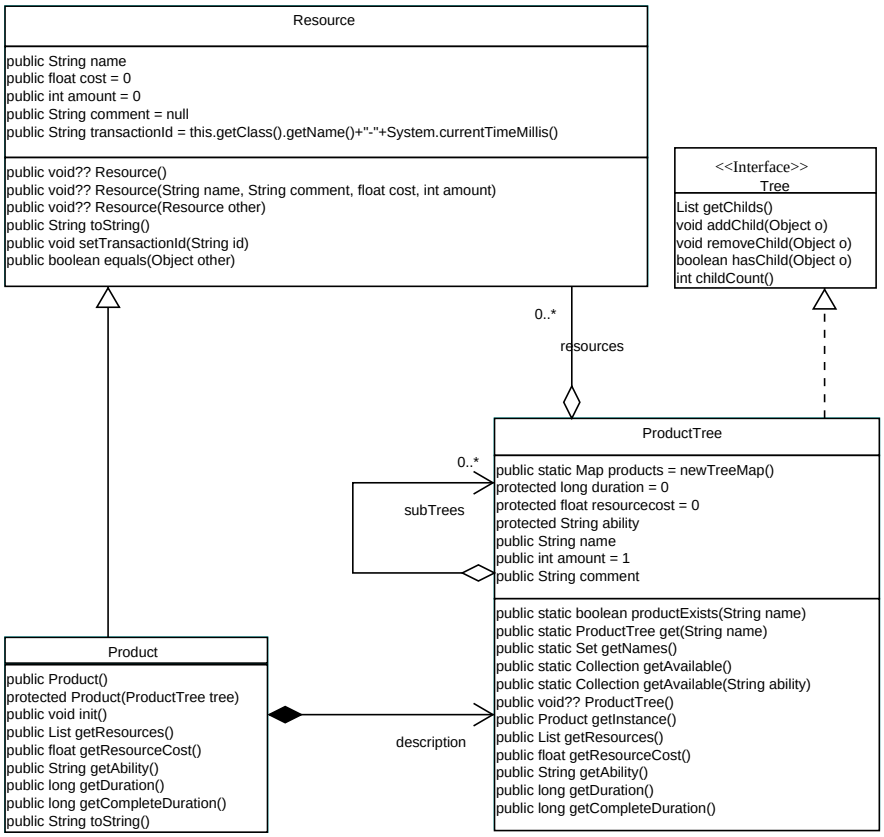



Abbildung 5.4: Klassendiagramm von Resource, Product und ProductTree

**Rohstoffe:** Die zur Produktion notwendigen *Rohstoffe* werden von Instanzen der Klasse `Ressource` repräsentiert. Jedem Rohstoff sind dabei neben einem Namen Kosten je Einheit und die Menge benötigter Einheiten zugeordnet.

[1] Homepage: `http://argouml.tigris.org/`

**Arbeitspläne** Die *Arbeitspläne* für die Produkte werden von Instanzen der Klasse `ProductTree` repräsentiert. Jeder Arbeitsplan ist ein Baum[2], dessen Knoten diejenigen Teilprodukte darstellen, die für den übergeordneten Knoten benötigt werden. Der oberste Knoten ist das Produkt selbst (Beispiel siehe Abbildung 5.5). Zu jedem Knoten gehört eine Liste der benötigten Rohstoffe (`resources`), eine Zeitspanne (`duration`), die die Produktionsdauer repräsentiert und zur Berechnung von Produktionsbeginn und -dauer im Rahmen der zeitlichen Vorgaben eines Auftrages herangezogen wird, sowie eine Tätigkeit (`ability`), die von der jeweiligen Maschine bei der Produktion ausgeführt werden muss. Über die Tätigkeit wird später die Bestimmung geeigneter Maschinen für einen Auftrag gehandhabt. Desweiteren ist im Arbeitsplan die Menge der benötigten Teilprodukte für ein Produkt gespeichert.

Abbildung 5.4 auf der vorherigen Seite stellt die besprochenen Klassen dar.

Die einzelnen Zweige eines Baumes können parallel bearbeitet werden, wobei zu Beginn der Bearbeitung eines Knotens alle Teilprodukte hergestellt worden sein müssen. Im Beispiel Abbildung 5.5 kann der Stuhl also erst produziert werden, wenn sowohl das Skelett als auch der Sitz fertiggestellt wurden.

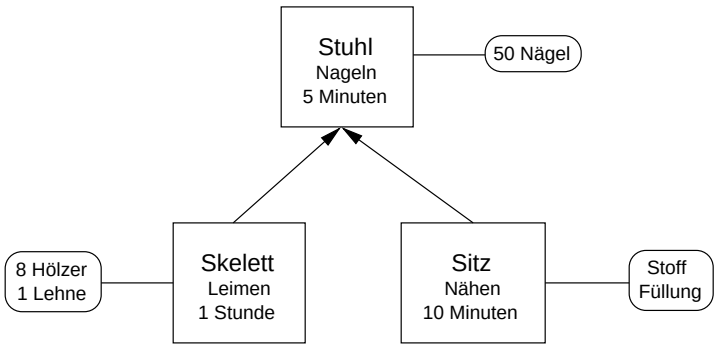



Abbildung 5.5: Beispiel für einen Arbeitsplan zur Produktion eines Stuhls

Jeder Knoten eines Arbeitsplans kann in mehreren anderen Arbeitsplänen verwendet werden. Somit lassen sich Produktvarianten modellieren, die oft bis auf wenige Änderungen ähnliche Arbeitspläne haben.

Sofern das Multiagentensystem nicht auf mehreren Computer verteilt läuft, gibt es von jedem Arbeitsplan genau eine Instanz. Bei einer räumlichen Verteilung gäbe es eine je verwendetem Computer, was bei Änderungen der Arbeitspläne einen Abgleich erforderlich macht. Da Arbeitspläne in der vorliegenden Implementierung an ein entferntes System verschickt werden können, stellt dies kein Problem dar. Dieses Konzept hat seine Grenzen, wenn das Multiagentensystem auf sehr vielen Computern verteilt ist und häufige neue Arbeitspläne eingehen – die Größe des Beispielbetriebes (siehe Abschnitt 6.2 auf Seite 56) ist aber noch auf einem Computer handhabbar, zumal auch die Menge der verschiedenen Arbeitspläne in den Experimenten begrenzt und von vornherein bekannt

[2]Ein Baum ist eine Datenstruktur, die eine Wurzel mit Nachfolgern hat. Jedem Nachfolger können selbst wiederum Nachfolger zugeordnet sein, so dass sich eine hierarchische Struktur ergibt.

ist. Das Konzept kann aber leicht durch ein anderes Verfahren, etwa dem Information Layer [Haustein 1999], ersetzt werden.

### 5.2.2 Aufträge bzw. Produkte

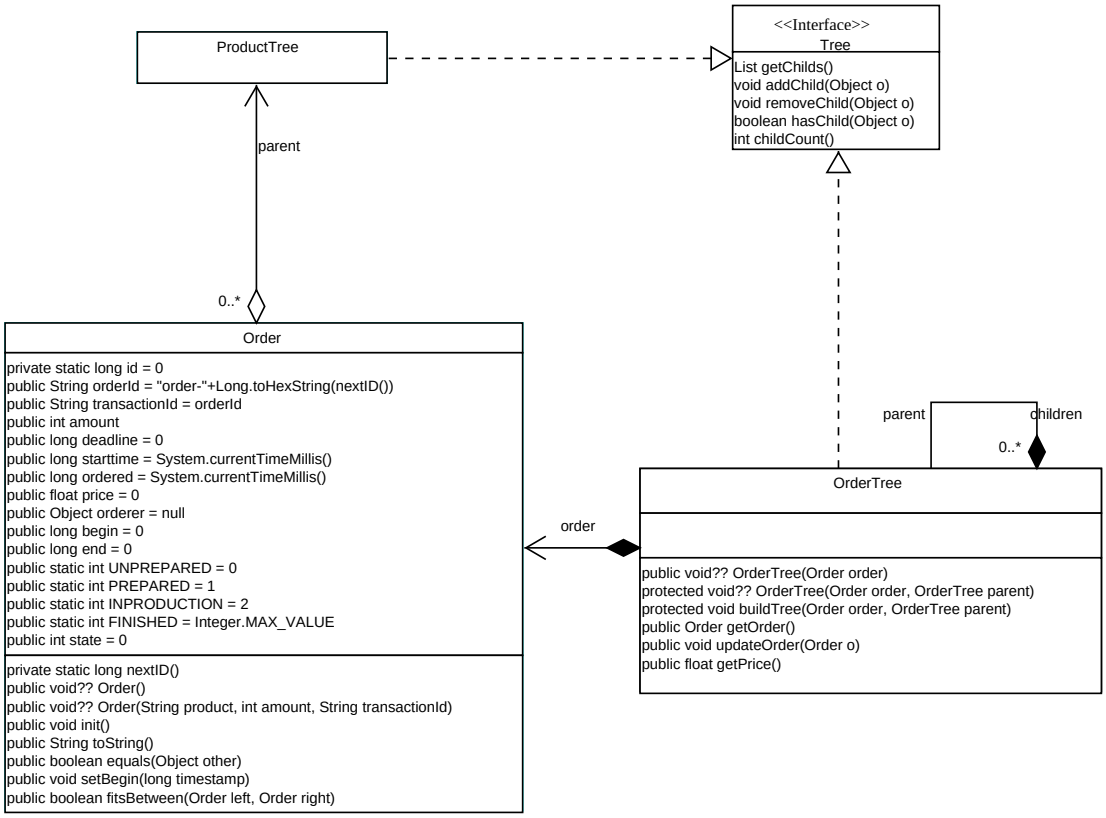



Abbildung 5.6: Klassendiagramm von Order und OrderTree

Der *Auftrag* ist neben den Arbeitsplänen das zentrale Element des Datenmodells. Alle Koordinationsmechanismen laufen darauf hinaus, dass ein Agent einem anderen einen Auftrag erteilt und letzterer entsprechende Rückmeldungen gibt. Es wurde auf eine explizite Modellierung der hergestellen Produkte verzichtet, da im Rahmen dieser Arbeit lediglich die notwendigen Daten über den Bauplan (`product`), die Menge (`amount`) und den Preis (`price`) für die Agenten wichtig sind. Diese sind je einem Auftrag zugeordnet.

Neben den Daten über das Produkt selbst, hat ein Auftrag einen Startzeitpunkt (`starttime`), vor dem mit der Produktion nicht begonnen werden kann, z.B. notwendige Teilprodukte erst dann verfügbar sein werden, sowie einen Endzeitpunkt (`deadline`), zu dem der Auftrag spätestens abgearbeitet worden sein muss. Für die lokale zeitliche Planung einer Maschine (lokale Ablaufplanung) gibt es zudem temporäre Variablen, die den konkreten Produktionsbeginn und das Produktionsende aufnehmen. Die Methode `fitsBetween()` testet, ob der Auftrag zeitlich zwischen zwei anderen Aufträge einge-

ordnet werden kann. Ist dies der Fall, werden diese Variablen auf den frühestmöglichen Zeitpunkt zwischen den beiden Aufträgen gesetzt.

Um die Verwaltung der Aufträge innerhalb der Agenten zu erleichtern, kann sich ein Auftrag zudem in einem der vier folgenden Zustände befinden:

**unvorbereitet** sind Aufträge, deren notwendige Rohstoffe oder Teilprodukte erst noch bei einer Maschine eingehen müssen;

**vorbereitet** sind Aufträge, deren Bearbeitung nichts mehr im Wege steht;

**in Produktion** sind Aufträge, die gerade produziert werden;

**fertiggestellt** sind Aufträge, die vollständig und fehlerfrei abgearbeitet wurden.

Jeder Auftrag hat einen automatisch vergebenen, eindeutigen Bezeichner `orderId` anhand dessen sich sein Weg durch das System nachvollziehen lässt und trägt eine Transaktionsnummer für die spätere Abrechnung. Eine Zerlegung eines Auftrags in seine Teilaufträge entsprechend der Arbeitspläne und des zeitlichen Rahmens wird mit der Klasse `OrderTree` vorgenommen, die ebenfalls die notwendigen Mengen der Teilprodukte berechnet.

Die beschriebenen Klassen sind in Abbildung 5.6 auf der vorherigen Seite zu sehen.

Aus Zeitgründen wurde auf eine explizite Transportkomponente verzichtet. Zwar sind gerade bei der Werkstattfertigung die Transportkosten ein nicht unerheblicher Faktor, diese Komponente hätte jedoch mehr Zeit in Anspruch genommen, als für diese Arbeit zur Verfügung stand. Abgesehen von dem Aufwand, diese Komponente zu erstellen, wäre es nicht schwer das bestehende System entsprechend zu erweitern. Anstelle des momentanen impliziten Materialflusses durch die Erfolgsmeldungen, müssten die Agenten lediglich eine weitere Subkomponente erhalten, die sich mit den Transportagenten auseinandersetzt.

## 5.3 Interne Strukturen der einzelnen Agenten

Nachdem die verschiedenen Agenten allgemein vorgestellt und das Datenmodell besprochen wurde, sollen die einzelnen Agenten nach einem Blick auf die von allen verwendeten Hilfsklassen genauer betrachtet werden.

### 5.3.1 Gemeinsame Hilfsklassen

Einige Klassen werden von allen Agenten genutzt: die Klasse `Conversation`, die eine Grundstruktur für alle Koordinationsmechanismen bereitstellt, weiter Klassen zum Nachrichtenaustausch zwischen den Agenten und Klassen, um aus einer Liste von eingehenden Angeboten auszuwählen.

**Conversation, MessageDispatcher und MessageReceiver:** Um im Rahmen einer Konversation zu einem Ergebnis zu kommen, müssen die Agenten eine Vielzahl von Nachrichten austauschen. Es bietet sich an, die Konversationen als endliche Automaten zu beschreiben: sie befinden sich in einem Zustand, den sie als Reaktion auf eingehende Nachrichten oder interne Ereignisse wechseln. Angefangen von einem Startzustand führen sie dabei Handlungen aus und/oder stoßen selbst wieder Ereignisse an, bis ein Endzustand erreicht ist. Die Koordinationsmechanismen werden jeweils durch eine oder mehrere unterschiedliche Konversationsarten (z.B. Kontrakt Netz: Auftraggeber und Auftragnehmer) modelliert.

Dementsprechend besitzt die Klasse `Conversation` eine `run()` Methode, die solange die Methode `tick()` aufruft, bis ein Fehler auftritt oder einer der Endzustände erreicht ist. Innerhalb von tick() werden dem aktuellen Zustand entsprechende unterschiedliche Handlungen ausgeführt und/oder auf externe Ereignisse gewartet (`Thread.wait()`). Eingehende Nachrichten, der Ablauf der Wartezeit oder andere Ereignisse veranlassen tick() entweder die Ausführung normal fortzusetzen, oder mit einer Fehlermeldung zu reagieren. Die Implementierung der Methode tick() und die Handhabung der Nachrichten ist den Unterklassen von `Conversation` überlassen.

Weiter muss ein Agent in der Lage sein, mehrere Konversationen gleichzeitig zu führen. Dazu implementiert er das Interface `MessageDispatcher`, bei dem sich alle `MessageReceiver` (zu denen auch Conversation gehört) mit einem Bezeichner registrieren, so dass alle Nachrichten mit diesem Bezeichner an sie weitergeleitet werden. Beginnt ein Agent eine Konversation, so erzeugt er entsprechend ein neues Objekt der Klasse Conversation, an das die zugehörigen Nachrichten weitergegeben werden. Da jede Konversation als eigener Thread läuft, kann ein Agent viele Konversationen führen.

Abbildung 5.7 auf der nächsten Seite stellt die erwähnten Klassen dar.

**Nachrichtenklassen:** Die ausgetauschten Nachrichten zwischen den Agenten sind eigenständige Objekte und werden als solche an die Agentenplattform zur Versendung weitergegeben. Gemeinsame Oberklasse ist `Message` aus XFAP, die bereits alle Methoden und Datenfelder enthält, um eine vollständige FIPA SL0-Nachricht zu repräsentieren (zu FIPA ACL siehe [FIPA/ACL 2000]). Der notwendige Inhalt der Nachrichten wird von den Klassen modelliert, die in Tabelle 5.7 auf Seite 42 samt Zweck aufgelistet sind. Die oberen sechs Klassen kommen in den Koordinationsmechanismen zum Einsatz, die anderen sind für die sonstigen Interaktionen vorgesehen.

**Selektoren bzw. Auswahlverfahren:** In beiden Koordinationsmechanismen müssen Agenten aufgrund der eingehenden Angebote eine Auswahl treffen. Für verschiedene Auswahlverfahren ist die abstrakte Klasse `MachineSelector` gedacht, die eine Liste von Estimated Objekten aufnimmt (`addList()` und diese sukzessive bei Aufrufen von `next()` zurückliefert. Mit der Methode `create()` kann eine Instanz eines Auswahlverfahrens geklont werden, so dass die Klone mit verschiedenen Angebotslisten aber einem gemeinsamen Datenbereich arbeiten können, etwa um über die Zeit bestimmte Agenten bei der Auswahl zu bevorzugen. Bisher ist nur ein Verfahren implementiert, das

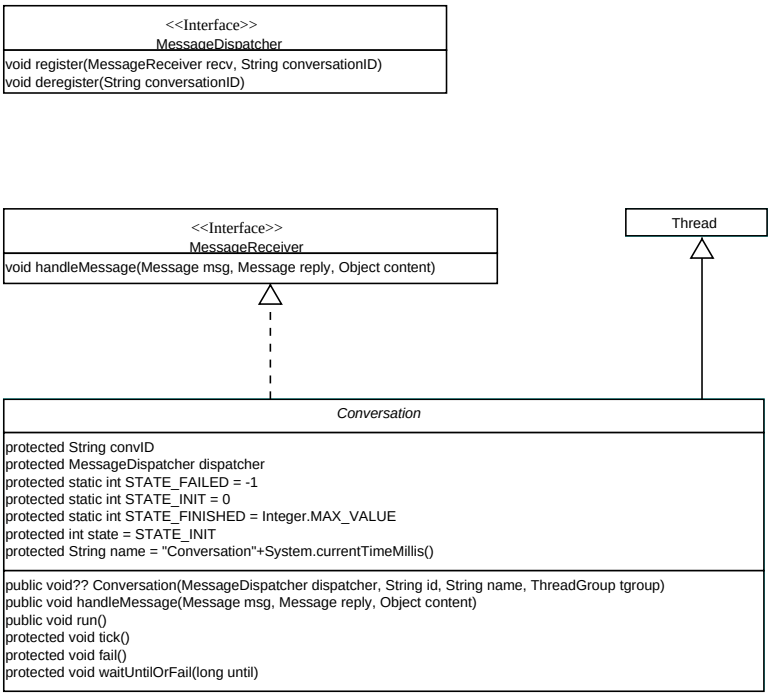



Abbildung 5.7: Klassendiagramm von Conversation, MessageDispatcher und MessageReceiver

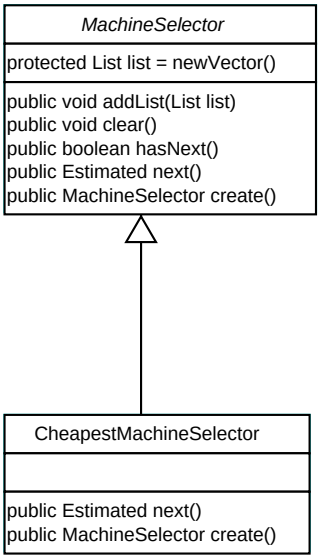



Abbildung 5.8: Klassendiagramm MachineSelector und CheapestMachineSelector

| **Canceled** | |
|---|---|
| Zweck: | Ein Auftrag wurde abgebrochen |
| Inhalt: | Auftrag |
| Sender/Empfänger | Maschinen, Produzenten ⇒ Maschinen, Kunden, Produzenten |
| **Estimated** | |
| Zweck: | Abschätzung eines Auftrages hinsichtlich Preis und Fertigstellung |
| Inhalt: | Geschätzter Auftrag |
| Sender/Empfänger | Maschine, Produzent ⇒ Maschine, Kunde, Produzent |
| **Failed** | |
| Zweck: | Ein Auftrag ist fehlgeschlagen |
| Inhalt: | Auftrag |
| Sender/Empfänger | Maschinen, Produzenten ⇒ Maschinen, Kunde, Produzenten |
| **Finished** | |
| Zweck: | Ein Auftrag ist fertiggestellt |
| Inhalt: | Auftrag |
| Sender/Empfänger | Maschinen, Produzenten ⇒ Maschinen, Kunde, Produzenten |
| **Produce** | |
| Zweck: | Aufforderung, einen Auftrag herzustellen |
| Inhalt: | Auftrag |
| Sender/Empfänger | Maschinen, Kunden, Produzenten ⇒ Maschinen, Produzenten |
| **Scheduled** | |
| Zweck: | Information, dass ein Auftrag angenommen und lokal eingeplant wurde |
| Inhalt: | Geschätzter Auftrag |
| Sender/Empfänger | Maschinen, Produzenten ⇒ Kunden, Produzenten |
| **Deploy** | |
| Zweck: | Objekt nachladen, z.B. Arbeitspläne, aber auch der Start weiterer Agenten |
| Inhalt: | Serialisiertes Objekt |
| Sender/Empfänger | alle ⇒ Startup |
| **Balance** | |
| Zweck: | Kontostand |
| Inhalt: | Alle Kontobewegungen passend auf eine CheckBalance Abfrage |
| Sender/Empfänger | Account ⇒ Alle |
| **CheckBalance** | |
| Zweck: | Kontostandabfrage |
| Inhalt: | Suchmuster für Abfrage |
| Sender/Empfänger | Alle ⇒ Account |
| **Transaction** | |
| Zweck: | Geldtransfer |
| Inhalt: | Betrag, Ziel der Quelle der Überweisung |
| Sender/Empfänger | Alle ⇒ Account |



Tabelle 5.7: Nachrichtentypen für das MAS

immer den günstigsten Agenten auswählt. Andere Verfahren, die nicht nur den Preis, sondern auch den Zeitpunkt der Fertigstellung des Auftrages oder andere Eigenschaften des Auftrages bzw. der angebotsstellenden Agenten mit in Betracht ziehen, sind aber leicht hinzuzufügen. Sie müssten lediglich das obige Interface umsetzen.

### 5.3.2 Der Produzent

Der erste Agent, der genauer beschrieben werden soll, ist der Produzent. Er nimmt an dem Markt teil und ist Schnittstelle zu den Kunden: er nimmt Aufträge an, handhabt deren Zerlegung in Teilaufträge entsprechend der Arbeitspläne und sorgt dafür, dass diese von Maschinen produziert werden (er arbeitet also als Mediator). Dazu bietet er die Aufträge über den Markt an und wählt aus den daraufhin eingehenden Angeboten eines aus (siehe auch Sequenzdiagramm Abbildung 5.9). Maschinen, deren Angebot nicht angenommen wurde, werden nicht benachrichtigt.

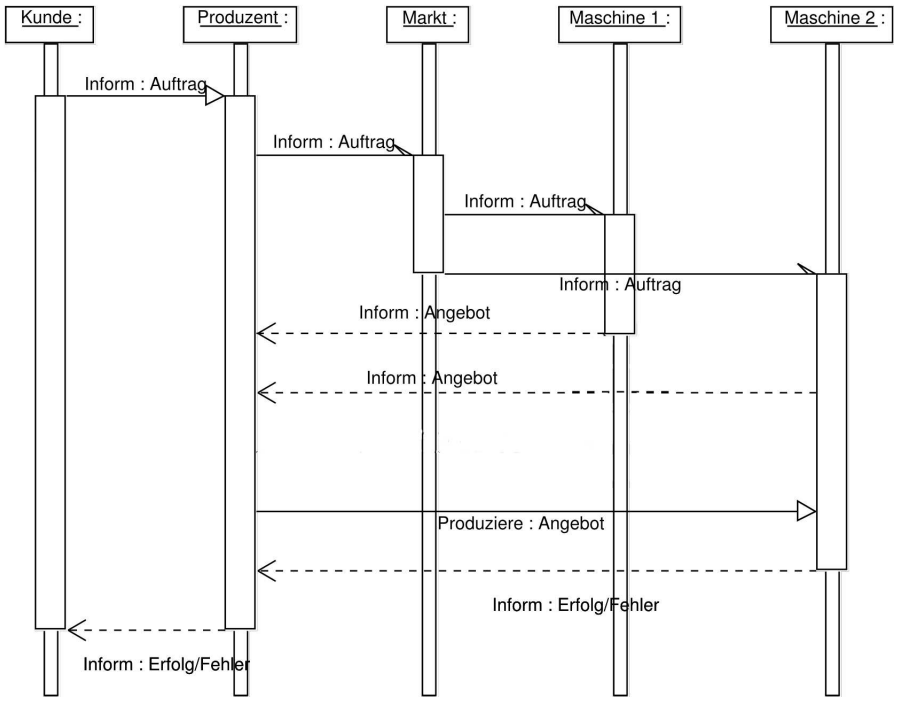



Abbildung 5.9: Sequenzdiagramm Produzent

Intern wird zu einem eingehenden Auftrag zunächst ein dem Arbeitsplan entsprechender Baum aus Aufträgen erzeugt (ein OrderTree), der entsprechende Zeit- und Mengenvorgaben für die Teilaufträge berechnet. Jedem Knoten dieses Baumes wird ein NodeThread zugeordnet, der zunächst wartet bis seine Nachfolger erfolgreich die Aufträge vergeben haben und die Teilprodukte eingetroffen sind. Daraufhin vergibt er seinen Auftrag über den Markt an eine entsprechende Maschine, die sogleich mit der Produktion beginnt. Ein Auftrag wird also erst dann an den Markt gegeben, wenn alle Teilaufträge

bearbeitet sind. NodeThreads, deren Auftrag fertiggestellt wurde, beenden sich selbst. Die anfallenden Kosten werden im Laufe der Bearbeitung aufsummiert und zum Ende dem Kunden bei der Buchhaltung in Rechnung gestellt. Kosten für den Produzenten selbst werden dabei nicht erhoben, da davon ausgegangen wird, dass das System an anderer Stelle abgeschrieben wird.

Andere Ansätze, wie zum Beispiel Mojonation [McCoy], ein marktbasiertes Konzept zum Austausch von Dateien, berechnen zum auch für Suchanfragen und Vermittlungen von Tauschpartnern Kosten (Micropayment). An dieser Stelle ist das sinnvoll, da jede Anfrage Rechenzeit und Bandbreite anderer kostet (Kosten werden in Mojos berechnet, die man durch das Anbieten von Daten oder Dienstleistungen erwirbt und mit denen man Daten oder Dienstleistungen bei anderen „kaufen“ kann).

Tritt bei der Abarbeitung ein Fehler auf, oder kann ein Produkt nicht hergestellt werden, so wird der Fehler zuerst bis zum obersten NodeThread weitergegeben, welcher daraufhin die noch laufenden NodeThreads beendet. Fertiggestellte Teilprodukte werden an das Lagerhaus weitergereicht.

### 5.3.3 Der Markt

Der Markt agiert als Broker, indem er eingehende Meldungen von Produzenten an in Frage kommende Maschinen weitergibt (siehe auch das Sequenzdiagramm in Abbildung 5.9). Die Maschinen teilen dazu zu Beginn dem Markt mit, welche Tätigkeiten sie beherrschen und können sich bei Bedarf wieder abmelden.

### 5.3.4 Die Maschine

Ein Maschinenagent repräsentiert eine physikalische Maschine und ist dafür verantwortlich, dass eingehende Aufträge in einem lokalen Plan angeordnet (lokale Ablaufplanung) und dementsprechend abgearbeitet werden, unterbrochen durch regelmäßige Wartungsphasen. Eine Maschine beherrscht zur Vereinfachung nur eine Verarbeitungsgeschwindigkeit, unabhängig von den auf ihr produzierbaren Fabrikaten (in Anlehnung an [Krothapalli und Deshmukh 1999]). Andere Modellen stellen dies variabel dar (z.B. [Váncza und Márkus 2000]). Bei der Werkstattfertigung kann davon aber ausgegangen werden, dass die einzelnen Maschinen feste Geschwindigkeiten haben, da es sich um eine fertigungstechnische Produktion von Fabrikaten und keine verfahrenstechnische Produktion von Fließgütern handelt. Dennoch ist der Maschine ein Zeitfaktor zugeordnet, durch den die in den Arbeitsplänen abgelegten geschätzten Produktionszeiten geteilt werden. So ist es möglich Maschinen mit unterschiedlicher, aber fester Geschwindigkeiten zu modellieren. Eine Maschine besteht aus mehreren Teilkomponenten:

**Produktion:** Diese Komponente ist für die Abarbeitung des lokalen Plans verantwortlich. Dazu werden Aufträge sukzessive aus dem Plan entnommen, die Maschine mit einer festen Rüstzeit umgerüstet und die Produktion angestoßen. Nach Ende der Produktion eines Auftrages wird die Callback-Methode `productionReport()` mit einem entsprechenden Canceled-, Finished- oder Failed-Objekt (siehe Abschnitt 5.3.1) aufgerufen. Sie stellt die Kosten, die sich aus den Rüstkosten, den laufenden

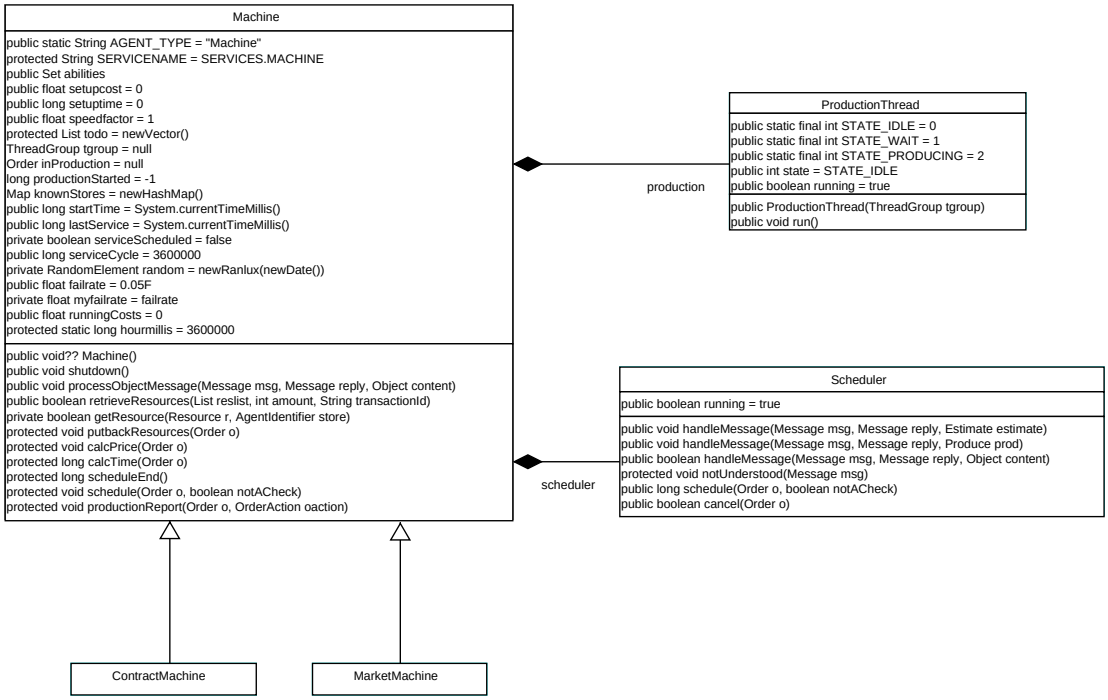



Abbildung 5.10: Klassendiagramm der Maschine(n)

Kosten für die Produktion, sowie den Kosten für Teilprodukte im Falle des Kontraktnetzes zusammensetzen, dem Auftraggeber in Rechnung. Kam es zu einem Fehler oder zum Abbruch der Produktion, so werden die bis dahin angefallenen Kosten einem dafür vorgesehenen Konto bei der Buchhaltung in Rechnung gestellt.

**Scheduler:** Diese Komponente ist für die lokale Planung verantwortlich. Geht ein Auftrag an den Scheduler, so wird zunächst geprüft, ob die notwendigen Rohstoffe in einem Lagerhaus vorhanden sind und ob sich der Auftrag in den lokalen Arbeitsplan einfügen lässt. Ist dies der Fall, so wird er an der frühestmöglichen Stelle eingeplant. Mit der Methode `cancel()` kann ein bereits geplanter Auftrag wieder abgebrochen werden. Ist er bereits in Produktion, wird die Produktionskomponente benachrichtigt, ansonsten wird der Auftrag einfach aus dem Plan entfernt. Die Methode productionReport() handhabt die Abrechnung und Benachrichtigung des Auftraggebers.

**Konversation:** Die Maschine nimmt an den beiden Koordinationsmechanismen Markt und Kontrakt Netz teil. Dazu gibt es zwei Unterklassen `ContractMaschine` und `MarketMachine`, die die jeweiligen Mechanismen umsetzen. Die nächsten beiden Absätze beschreiben diese Komponenten.

**Maschine im Markt** Der Markmechanismus innerhalb der Maschine war einfach umzusetzen. Sobald vom Markt ein Auftrag eingeht, wird mit Hilfe des Schedulers geprüft, ob

er erfüllbar ist. Dabei werden gleichzeitig die anfallenden Kosten berechnet und ein entsprechendes Angebot an den Auftraggeber abgegeben. Ist der Auftrag nicht erfüllbar, so wird die Anfrage ignoriert. Der Auftraggeber benachrichtigt nach Eingang der Angebote die Maschine, die den Zuschlag erhält. Sie gibt diesen dann an ihren Scheduler weiter, der den Auftrag in den lokalen Plan einfügt. Die Produktion kann beginnen, sobald der Zuschlag erteilt wurde.

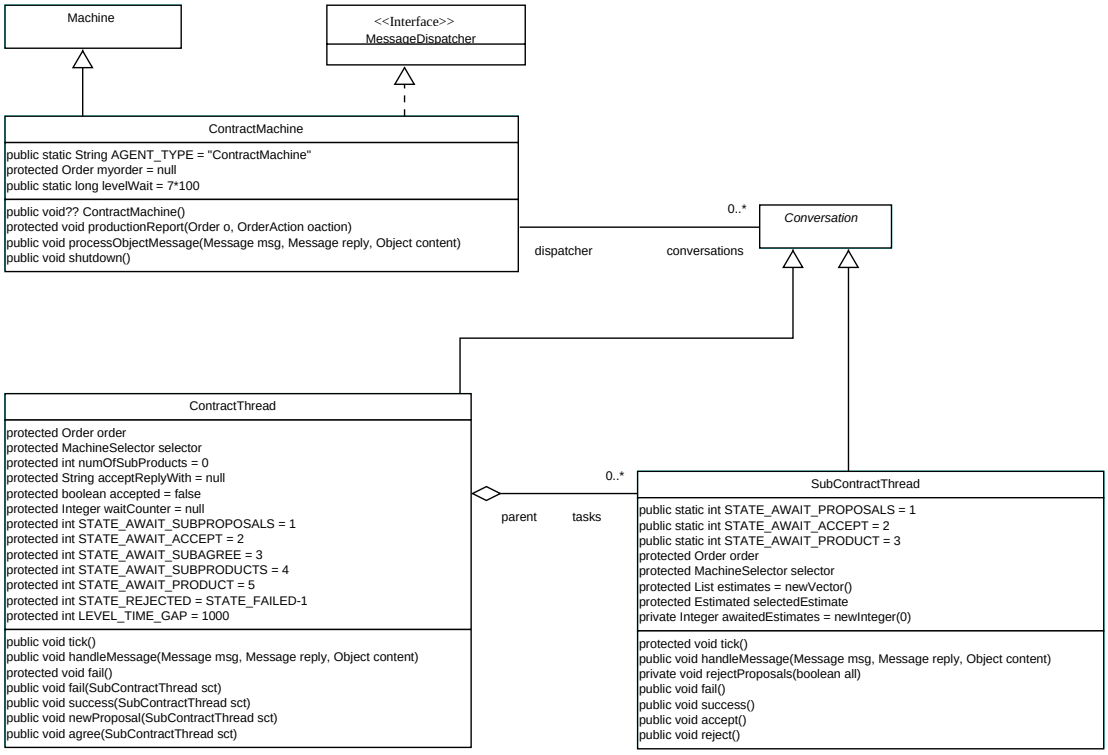



Abbildung 5.11: Klassendiagramm Maschine mit Kontrakt Netz

**Maschine im Kontrakt Netz** Der Kontrakt Netz Mechanismus ist komplexer als der Marktmechanismus, da hier die Maschine anstelle des Produzenten für die Teilaufträge verantwortlich ist und entsprechende Partner auszuwählen hat. Diese Aufgabe übernehmen die Klassen `ContractThread` und `SubContractThread`. Geht ein Auftrag ein, bestimmt ein neu erzeugter ContractThread zunächst die notwendigen Teilaufträge mit den zeitlichen Rahmen und gibt diese an von ihm koordinierte SubContractThreads weiter. Diese suchen über die Suchmechanismen der Agentenplattform geeignete Maschinen und stellen an diese Angebotsnachfragen. Mit Hilfe eines Auswahlverfahrens wird dann aus den eingehenden Angeboten eines ausgewählt (jede Maschine erstellt entweder ein Angebot oder lehnt den Auftrag ab), welches an den ContractThread weitergegeben wird. Sobald Angebote für alle Teilaufträge vorliegen, erstellt der ContractThread darauf basierend selbst ein Angebot und leitet es an den ursprünglichen Auftraggeber weiter.

Sobald der ContractThread den Zuschlag oder die Ablehnung erhält, erhalten alle angebotsstellenden Maschinen ihrerseits einen Zuschlag, während die anderen eine Ablehnung erhalten. Abbildung 5.12 zeigt ein Beispiel für ein erfolgreiches Zustandekommen eines Vertrages.

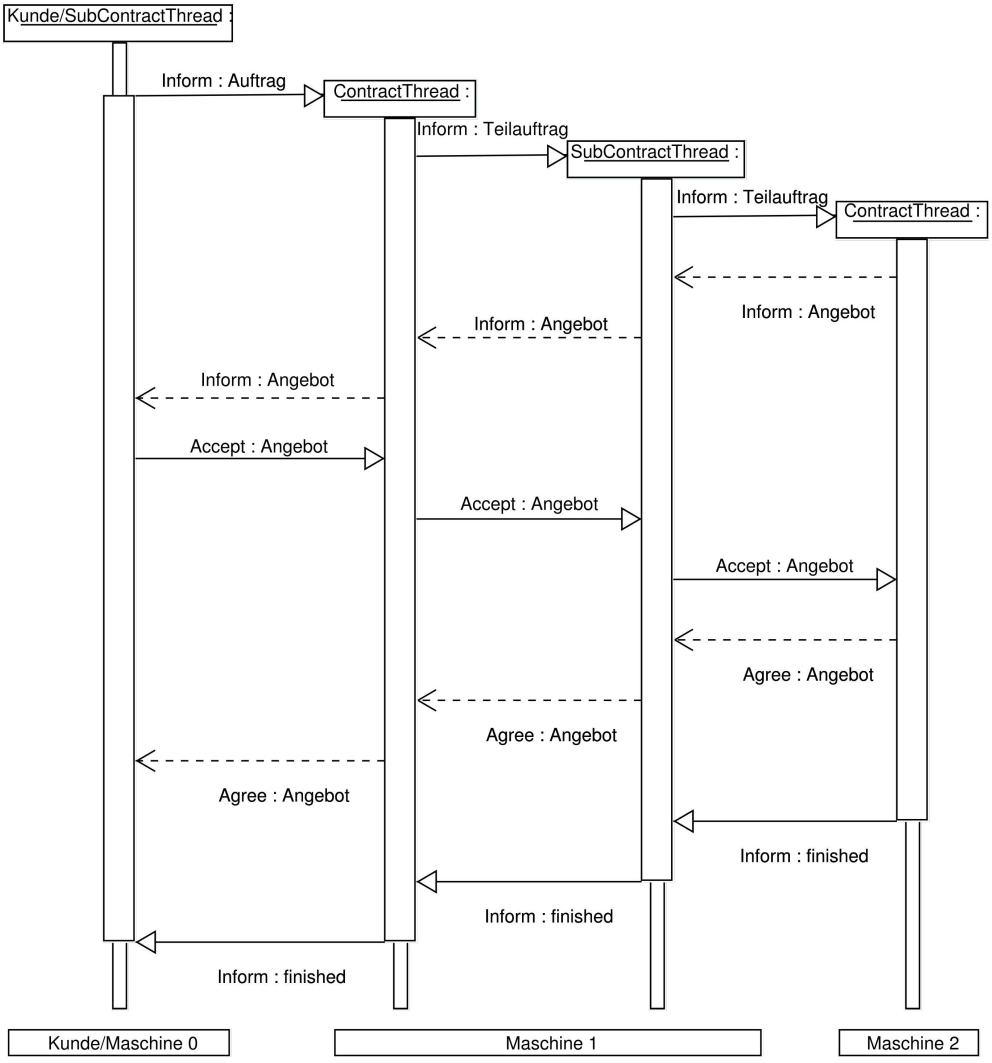



Abbildung 5.12: Beispielhaftes Sequenzdiagramm für ContractThread und SubContractThread

Im Gegensatz zum Marktmechanismus, bei dem ein Auftrag erst dann über den Markt vermittelt wird, wenn seine Teilaufträge erfolgreich produziert wurden, werden hier alle Teilaufträge vor Beginn der Produktion vermittelt. Durch das Paar ContractThread und SubContractThread wird der Arbeitsplan zudem faktisch in einen UND-ODER-Baum aufgespalten: ContractThread und SubContractThread einer Maschine bilden dabei die UND-Kanten, während SubContractThread der einen und Contract-

Thread der nächsten Maschine die ODER-Kanten des Baumes bilden. Nachdem durch Angebotsnachfrage der Baum ausgebildet wurde, wird aus den ODER-Kanten durch das Auswahlverfahren je eine ausgewählt. Daher ist das Beispiel 5.12 auf der vorherigen Seite stark vereinfacht. In Wirklichkeit kommuniziert jeder ContractThread und SubContractThread mit mehreren Partnern (nach rechts hin gesehen), was einen nicht zu vernachlässigenden Synchronisationsaufwand nach sich zieht. Angenommen, es gäbe z.B. zwei Maschinen, die Leimen können und drei die Nähen, so ergibt der einfache Arbeitsplan für einen Stuhl aus Abbildung 5.5 auf Seite 37 die in Bild 5.13 zu sehenden Kommunikationspfade.

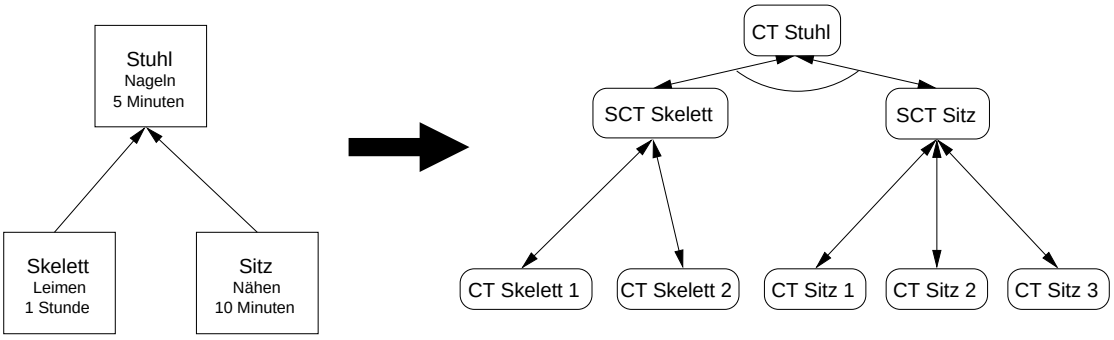



Abbildung 5.13: Kommunikationspfade im Beispiel Kontrakt Netz und Stuhl

Wie bereits durch den Namen angedeutet, sind ContractThreads und SubContractThreads Threads, d.h. parallel laufende Prozesse. Bei der Vergabe der Teilprodukte eines Auftrages (Teilauftrag) muss ein Agent jeweils mit mehreren anderen Agenten kommunizieren. Diese Konversationen durchlaufen dabei unabhängig voneinander verschiedene Zustände (Teilauftrag verschicken, Angebote und Ablehnungen entgegennehmen, ein Angebot auswählen und weitermelden und bei Akzeptanz annehmen bzw. die anderen ablehnen), bis entweder alle Teilaufträge vergeben wurden oder der Vorgang abgebrochen wurde. Die Unabhängigkeit der Konversationen legt eine nebenläufige Implementierung durch Threads nahe: der Programmcode kann sich auf die Behandlung einer einzigen Konversation beschränken und wird dadurch einfacher. Durch genau bestimmte Wartepunkte innerhalb einer Konversation sind die (dazwischen) durchgeführten Aktionen atomar und der Zustand der Konversation ist bei externen Ereignissen (eingehende Nachrichten, Aufforderung zum Abbruch) genau bekannt. Es genügt, wenn alle Konversationen von einer Stelle aus koordiniert werden. Die Umsetzung der Koordination der Threads ist aber eine komplexe Aufgabe, die nicht immer auf Anhieb gelingt. Zudem kostet die Fehlersuche in nebenläufigen Systemen viel Zeit und Nerven, da man die Zustände der einzelnen Teile und ihr Zusammenspiel genau nachverfolgen muss, um den Fehler und letztendlich die Fehlerquelle bestimmen zu können.

Eine alternative Implementierung, die die Konversationen ohne Threads gemeinsam behandelt, ist wesentlich aufwändiger, kann aber zu einem schnelleren System führen. Der Aufwand liegt hier dann nicht nur in der Koordination der Threads, sondern auch in der komplexen zentralen Verwaltung der einzelnen Konversationen und auftretenden

Ereignisse. Die Fehlersuche würde sich an dieser Stelle ebenfalls nicht vereinfachen, da noch immer der Zustand der einzelnen Konversationen nachvollzogen werden muss.

### 5.3.5 Buchhaltung

Die Abrechnung der produzierten Aufträge und sonstiger anfallender Kosten übernimmt ein Agent der Klasse `Account`. Er empfängt eingehende Nachrichten vom Typ Transaction, die die Geldmenge, den Zahlenden, den Empfänger, die zwei Textfelder `transactionID` und `reference` und einen Transaktionstyp enthalten. TransactionID ist ein eindeutiger Bezeichner für diese Transaktion, währen reference normalerweise auf den zugehörigen Auftrag des Kunden zeigt, bei der Abrechnung von Rohstoffenentnahmen aus einem Lagerhaus steht hier der Name des Rohstoffs. Der Transaktionstyp bezieht sich auf die Gründe für die Transaktion: Rohstoffentnahmen, Abrechnung für unfertige, abgebrochene, erfolgreiche oder fehlgeschlagene Aufträge, sowie für an Kunden verkaufte Produkte. Alle Transaktionen werden für eine spätere Abfrage aufgezeichnet und können optional in einer Datenbank abgelegt werden. CheckBalance enthält dieselben Datenfelder wie eine Transaktion. Bei einer Abfrage werden neben dem Kontostand genau diejenigen Transaktionen zurückgeliefert, die in den bei der CheckBalance Instanz gesetzten Feldern übereinstimmen. Produzenten und Maschinenagenten machen davon Gebrauch, um den Betrag zu bestimmen, den sie für ein Produkt in Rechnung stellen müssen (bei Maschinen: eigene Kosten + Kosten für alle Teilprodukte).

### 5.3.6 Lagerhäuser

Die Rohstoffe für die Produktion werden in den Lagerhäusern verwaltet. Alle Lagerhäuser melden sich dynamisch bei dem Agentensystem mit einer Liste der verfügbaren Rohstoffe an, über die die Maschinen später das Lagerhaus finden, dass die gesuchten Rohstoffe enthält. Sobald ein Agent Rohstoffe aus einem Lagerhaus bezieht, berechnet dies bei der Buchhaltung die entsprechenden Kosten. Natürlich können die Lagerbestände zur Laufzeit durch die Zusendung von Rohstoffen oder Produkten (etwa aus Teilproduktionen) jederzeit aufgefüllt werden. Jedes Lagerhaus operiert in einem von drei möglichen Modi:

**realistisch:** Anfragen werden entsprechend der echten Lagerbestände beantwortet. Lagerbestände können auslaufen.

**endlos:** Alle eingelagerten Rohstoffarten sind in beliebiger Menge verfügbar.

**schwarzes Loch:** Alle möglichen Rohstoffarten sind in beliebiger Menge verfügbar.

Die drei Modi wurden implementiert, um das System zügig testen zu können. Zudem sind die benötigten Mengen an Rohstoffen meist durch die Ergebnisse aus der taktischen Produktionsplanung bekannt, so dass sie als ausreichend gegeben vorrausgesetzt werden können.

### 5.3.7 Kunden/Testagenten

Für die Experimente wurden zwei Testagenten implementiert, die die Kunden repräsentieren. Sie setzen deren Anteil an dem jeweiligen Koordinationsmechanismus um und geben eine Abfolge von Aufträgen an das System weiter. Gemeinsame Oberklasse der beiden Testagenten ist `RandomizedTester`, der eine zufällige Abfolge von Aufträgen generieren kann oder einen Auftragstrom aus einer zuvor erzeugten Datei liest (siehe dazu auch Anhang 7.2 auf Seite 73). Dies hat den Vorteil, dass die beiden Koordinationsmechanismen mit identischen Kundenverhalten getestet werden können und so ein Vergleich hinsichtlich der Kriterien möglich ist.

### 5.3.8 Weitere Agenten

Desweiteren gibt es im System neben den Agenten der Agentenplattform und den bereits beschriebenen lediglich den Startup-Agenten. Er ist dafür verantwortlich das System zu initialisieren und liest dazu die Beschreibungen der Arbeitspläne, sowie Parameter für den Start weiterer Agenten aus einem Datenstrom. Dies kann entweder eine Datei oder auch der Inhalt einer ihm zugesendeten Nachricht sein. Damit können zur Laufzeit Agenten gestartet und Arbeitspläne geladen werden.

## 5.4 Weitere Bestandteile des Systems

**Protokollierung:** Neben den beschriebenen Agenten ist das Protokollieren des Systemlaufes ein wichtiges Element im System. Jedes Objekt kann sich dazu einen Ausgabestrom der Klasse `Log` instantiieren und über verschiedene Kanäle (info, log, debug, error, fatal) Nachrichten ausgeben. Die Ausgabeströme werden üblicherweise in eine Datei umgeleitet, womit jeder Agent seine eigene Protokolldatei erhält.

**Serialisierung:** Dazu kommt die Serialisierung, die eine Initialisierung von Objekten aus einer XML[Behme und Mintert 1998] Datei heraus ermöglicht. Über dieses Format ließen sich etwa Arbeitspläne mit anderen bereits in einem Betrieb installieren Systemen tauschen (natürlich wird hier immernoch eine Konvertierung nötig sein, da sich die Strukturen nicht notwendigerweise decken). Die Serialisierung wird dazu verwendet, das System mit unterschiedlicher Konfiguration zu betreiben und evtl. zu einem späteren Zeitpunkt weitere Agenten hinzufügen zu können.

**Datengenerierung und Auswertung:** Die Generierung der Auftragsmengen für die Kundenagenten und die spätere Auswertung der Protkolldaten, wird durch Skripte[3] gehandhabt. Details zu den Skripten sich in Anhang 7.2 auf Seite 73, die Auswertungen sind in Abschnitt 6.3 auf Seite 57 und folgende beschrieben.

[3] . . . in der objektorientierten Programmiersprache Ruby `http://www.ruby-lang.org`

## 5.5 Erfüllt das implementierte Agentensystem die Anforderungen?

In Abschnitt 4.2 auf Seite 28 wurden Anforderungen an das Multiagentensystem genannt, die im folgenden durchgegangen werden:

- *Das System sollte konform zum FIPA Standard implementiert sein:* Die verwendete Agentenbibliothek ist eine Implementierung des FIPA Standards. Auch das Multiagentensystem selbst ist zu diesem Standard konform.
- *Das System sollte möglichst gut steuer- und verstehbar sein:* Das implementierte Agentensystem führ detaillierte Protokolle über den Zystand der einzelnen Teile. So ist ein Programmlauf gut nachvollziehbar. Es lässt sich ebenfalls sehr einfach über den Aufruf eines Skriptes steuern. Da es eine eigene Implementierung ist, ist sie natürlich verstehbar.
- *Das System sollte keine Simulation, sondern ein Einsatz von vollständig programmierten Agenten sein:* Das implementierte System ist in seiner Verwendung in den Experimenten als Modell ausgelegt. Es besteht jedoch aus vollwertigen, einzelnen Agenten und kann bei Bedarf durch den Einsatz echter Zeitangaben und dem Austausch der Produktionskomponente der Maschinenagenten, an reale Maschinen angeschlossen werden.
- *Das System sollte das Problem der Ablaufplanung lösen:* Diese Anforderung wird erfüllt.
- *Das System soll überschaubar sein und möglichst auf zusätzliche Dinge verzichten:* das implementierte System verzichtet auf die Ablaufplanung zunächst nicht betreffende Details und konzentriert sich auf das Problem und die Koordinationsmechanismen.
- *Es sollte möglich sein, verschiedene Koordinationsmechanismen, sofern nicht schon vorhanden, nachzupflegen:* Alle benötigten Koordinationsmechanismen liegen bereits vor, weitere können bei Bedarf nachgepflegt werden.

## 5.6 Probleme in nebenläufigen Systemen

Bei der Implementierung habe ich für den Umgang mit verteilten Systemen und Multiagentensystem im besonderen wichtige Erfahrungen sammeln können. Am meisten hat mich die Arbeit mit den Threads belehrt, da ich hier einige (leider) typische Fehler begangen habe.

**Race Condition in Threads:** Der Kontrakt Netz Mechanismus innerhalb eines Agenten wartet auf Antworten von jedem Agent, dem er einen Auftrag zur Angebotserstellung zugesendet hat. Dazu wird innerhalb eines Threads A in einer while()-Schleife ein Zähler erniedrigt, bis er auf 0 ist:

```
while(counter > 0) {
   this.wait(timout);
   counter--;
}
```

Die anderen beteiligten Threads rufen beim Eingang eines Angebotes nun `notify()` auf, was den aktuellen Thread aufweckt. Dieser erniedrigt dann den Zähler und wartet erneut. Hier liegt eine typische Race-Condition vor: es kann vorkommen, dass zwei Threads B und C gleichzeitig notify() aufrufen. Dann erhält nicht A das Lock um den Zähler zu verändern, sondern der zweite Aufrufer C. Somit führt B's Aufruf von notify() nicht dazu, dass der Zähler verändert wird, denn A wartet ja noch auf den Lock. Nachdem C diesen freigegeben hat, setzt A seine Arbeit fort und hat somit den Aufruf von B verloren.

Die Lösung liegt darin, dass immer das Objekt, das auch den Lock erhält, den Zähler erniedrigt. Also B und C anstelle von A.

**Deadlocks in Threads:** Eine weitere Klippe mit Threads sind Deadlocks, d.h. wenn zwei Threads aufeinander warten. So zum Beispiel bei der Fehlerbehandlung in NodeThreads des Produzenten: Die NodeThreads unterbrechen mit `interrupt()` solange den Vorgänger, bis der oberste NodeThread erreicht ist (interrupt() bewirkt, dass an einem Wartepunkt fail() aufgerufen wird). Dieser teilt dann über kill() allen seinen Nachkommen mit, dass sie sich beenden sollen.

```
public fail() {
  if (parent != null)
    synchronized (parent) { parent.interrupt(); }
  else
    kill();
}

public synchronized kill() {
  if (childs != null) {
      Iterator i = childs.iterator();
      while (i.hasNext()) {
          NodeThread nt = (NodeThread) i.next();
          nt.kill();
      }
  }
}
```

Das Problem ist, dass die run() Methode der Klasse Conversation „synchronisiert“ ist – sie kann also zu einem Zeitpunkt nur einmal betreten werden, im Gegensatz zu nicht-synchronisierten Methoden, die beliebig oft gleichzeitig betreten werden können – damit nur an bestimmten Wartepunkten Unterbrechungen der Handlung möglich sind,

nämlich dann, wenn innerhalb dieser Methode auf externe Ereignisse gewartet wird. Tritt nun aber in zwei Nachfolgern B und C eines Nodethreads A ein Fehler auf, so wird aus der jeweiligen run() Methode heraus fail() aufgerufen, das eigene Lock bleibt bei B bzw. C. Ein klassischer Deadlock: beide versuchen nun den Vorgänger A zu unterbrechen. Dem ersten gelingt das noch, während der zweite auf das Lock von A warten muss. Der Vorgänger A hält weiterhin sein Lock und betritt die kill() Methode. A versucht nun bei B und C kill() aufzurufen, was bei B noch gelingen mag. Da C aber auf das Lock von A wartet und sein eigenes Lock nicht freigegeben hat, muss A auf das Lock von C warten. Aus dieser Situation gibt es dann keinen Ausweg.

Eine scheinbare Lösung besteht darin kill() nicht-synchronisiert zu halten und statt dessen über ein Flag festzuhalten, das sie betreten wurde:

```
private boolean killing = false;

public kill() {
  if (killing)
     return;

  killing = true;

  if (childs != null) {
      Iterator i = childs.iterator();
      while (i.hasNext()) {
          NodeThread nt = (NodeThread) i.next();
          nt.kill();
      }
  }
}
```

Womit man sich wieder eine Race Condition eingehandelt hat. Alternativ könnte der Nachfolger, wie im erstem Problem, das Flag setzten. Es ist aber eleganter zu prüfen, ob ein Thread unterbrochen wurde:

```
public fail() {
  if (parent != null)
    if (!parent.isInterrupted())
      synchronized (parent) { parent.interrupt(); }
  else
    kill();
}
```

**Fehlersuche und Logging mit Agenten:** Leider gibt es noch keine allgemeinen Vorgehensweisen zur Fehlersuche in Multiagentensystemen [Shen et al. 2001]. Im Laufe der Arbeit kristalisierten sich aber folgende Punkte heraus:

- Viele Applikationen verwalten nur wenige Verlaufsprotokolle, etwa eines für Statusinformationen und eines für Fehlermeldungen. Für verteilte Systeme ist das ungeeignet: da viele Komponenten des Systems parallel laufen, ist es schwierig aus einem einzigen Protokoll den Ablauf nachzuvollziehen. Viel einfacher wird es, wenn jeder Komponente ein eigenes Protokoll zugeordnet wird, da man so deren Zustand genau verfolgen kann.

- Es hat sich ebenso als praktisch erwiesen, wenn in diesem Protokoll Meldungen über verschiedene „Kanäle“ (z.B. Info, Debug, Fehler, ...) ausgegeben werden. So lassen sich wichtige Ereignisse, wie der Beginn der Produktion oder Produktionsabbrüche, sehr schnell finden.

- Da die Koordinationsmechanismen als Endliche Automaten modelliert wurden, war die Protokollierung der Zustände wesentlich für eine Fehlersuche, die ansonsten nahezu aussichtslos gewesen wäre. Es hat sich ebenfalls bewährt, die einzelnen Konversationen eines Agenten mit einem eindeutigen Bezeichner zu versehen.

- Die Aufzeichnung der versendeten und empfangenen Nachrichten war ebenfalls von sehr großem Nutzen, da so der Lauf eines Auftrages durch das System relativ gut nachvollzogen werden konnte.

Ein Nachteil solch detaillierter Protokolle ist aber die anfallende Datenmenge: bei einzelnen Experimenten wurden bis zu 300 MB an Protokolldaten erzeugt. Die Fehlersuche gestaltete sich trotz der genannten Punkte als schwer.

**Reaktive und Proaktive Agenten:** Im vorliegenden System gibt es eine Vielzahl von verschiedenen Agenten. Einige davon sind proaktiv (Produzenten, Kunden und Maschinen), d.h. sie ergreifen von sich aus die Initiative und handeln ohne externen Anstoß, während andere rein reaktiv sind (Buchhaltung, in gewissem Rahmen auch Lagerhäuser) und nur auf externe Ereignisse hin tätig werden. Gerade bei letzteren kann man darüber streiten, ob es wirklich Agenten sind oder nur passive Systemkomponenten. Da aber solche Teile des Systems durchaus notwendig sind, wird man kaum ein Agentensystem finden, das nur aus proaktiven besteht.

# 6 Ergebnisse und Auswertung der Experimente

Diese Kapitel bespricht die Experimente, die mit dem Agentensystem aus dem vorhergehenden Kapitel durchgeführt wurden. Zunächst werden die untersuchten Aspekte einzeln mit den zur Bewertung herangezogenen Kriterien vorgestellt. Es folgt eine Beschreibung der Experimentkonfiguration und setzt dann mit der experimentellen Untersuchung der einzelnen Aspekte fort.

## 6.1 Welche Aspekte sollen in den Experimenten untersucht werden?

**Flexibilität:** Ein Aspekt ist die *Flexibilität* des Systems, d.h. wie gut es auf Änderungen der Umgebung reagiert. Darunter fallen Änderungen in der Maschinenkonfiguration, etwa der Weg- oder Ausfall einer Maschine, aber auch das Hinzukommen neuer Maschinen. Eine weitere in der Praxis wohl öfter vorkommende Änderung sind Schwankungen im Auftragseingang, die entweder kurzfristig durch die Arbeitszeiten (etwa, das morgens verstärkt Aufträge eingehen) oder längerfristig durch saisonale Schwankungen beeinflusst sein können. Desweiteren können sich die Baupläne der Produkte ändern oder neue hinzukommen, wenn etwa ein weiteres Produkt in das Angebot mit aufgenommen wird.

Bewertungskriterium sollen an dieser Stelle die Auslastung der der Maschinen und die Erfolgsquoten bei der Produktion sein.

Es wird vermutet, das die Abbruchquote bei dem Markt höher ist als bei dem Kontrakt Netz, denn dieses beginnt erst mit der Produktion, wenn sicher ist, dass ein Produkt vollständig produziert werden kann. Der Markt produziert sobald der erste Termin steht, ohne zu wissen, ob das Produkt überhaupt herstellbar ist. Da er sich auf diese Weise aber nicht festlegt, wird weiter vermutet, das er flexibler auf Änderungen reagieren kann.

**Skalierbarkeit** Bezüglich der Multiagentensysteme ist die *Skalierbarkeit* ein weiterer, wichtiger Aspekt. Es ist eines der Versprechen der Agententechnologie, gut mit umfangreichen Problemstellungen umgehen zu können, besser sogar als monolithische Ansätze. Daher soll untersucht werden, wie sich die beiden Koordinationsmechanismen bei unterschiedlichen Mengen von Maschinen verhalten. Interessant sind hier die Mengen der ausgetauschten Nachrichten und die Gesamtbelastung des Computersystems, etwa durch die Anzahl der parallel laufenden Prozesse.

Es wird vermutet, das der Markt weniger Ressourcen verbraucht als das Kontrakt Netz, da er der einfachere Koordinationsmechanismus ist.

## 6.2 Experimentkonfigurationen

Als Beispielbetrieb wurde eine kleine Schreinerei gewählt, die zwei unterschiedliche Stühle, normale und niedrige Tische sowie Schränke im Angebot hat. Zusätzlich gibt es das Produkt „Esszimmer“, das aus einem Schrank, vier Stühlen und einem normalen Tisch und einem niedrigen Tisch besteht, was nicht jedes Esszimmer widerspiegeln mag, aber einem handelsüblichen Angebot entspricht. Für die Herstellung der Produkte werden Maschinen bereitgestellt, die die folgenden Tätigkeiten ausführen können: Leimen, Nähen, Nageln, Zusammenfügen, Schnitzen, Sägen, Drehen und Bohren. Es gibt also drei unterschiedliche Produktarten und acht verschiedene Maschinentypen. Eine detaillierte Beschreibung ist in Anhang 7.2 zu finden.

Um die vorgestellten Aspekte untersuchen zu können wurden unterschiedliche Datensätze generiert. Reale Daten standen leider nicht zur Verfügung, ein Vergleich der beiden Koordinationsmechanismen ist aber auch mit künstlich generierten Daten möglich. Um die Experimente zu beschleunigen, wurden alle Zeitangaben auf ein Zehntel reduziert. Das bedeutet, das eine Sekunde *Realzeit*, 10 Sekunden *Modellzeit* repräsentiert. Da das System die Zeit im Millisekundenbereich misst, ist somit eine ausreichende Granularität im Bereich einer hundertstel Sekunde gewährleistet.

Die Datensätze wurden in den folgenden Eigenschaften variiert:

- Anzahl der Aufträge (`amount`),
- Zufallszahlengenerator und sein Initialisierungswert (`RNG` und `seed`),
- nur Hauptprodukte oder auch Teilprodukte in den Aufträgen (`maxlevel`, 0 bedeutet dabei nur Hauptprodukte),
- Wartezeit zwischen dem Eingang von zwei Aufträgen (`sleepMin` und `sleepMax`),
- konstante Auftragsrate mit und ohne Auftragsstösen von unterschiedlicher Frequenz (`peakFrequency`), Dauer (`peakDuration`) und Art (`peakModifier` als Wert, durch den die Wartezeit zwischen zwei Aufträgen währen eines Auftragsstoßes geteilt wird und `peakProducts`, womit eine Menge von Produkten angegeben werden kann, die während eines Auftragsstoßes geordert werden).

Die in Bezeichner in Klammern finden sich später in den Graphen zu den Experimenten wieder (siehe auch nächste Abschnitt).

Alle Experimente wurden auf einem handelsüblichen Computer in folgenden Konfiguration durchgeführt:

| | |
|---|---|
| Prozessor | AMD Athlon 1.1 GHz |
| Speicher | 768 MB RAM |
| System | Linux 2.4 |
| Java | JDK 1.3 von IBM |

Die Java Umgebung von IBM wurde gewählt, da sie sich schon im Vorfeld als robuster als die Implementierung von SUN erwiesen hatte. Letztere ergab gerade bei einer großen Anzahl von Threads viele Experimentabbrüche und hatte im Allgemeinen einen höheren Ressourcenbedarf als die Umgebung von IBM (Speicher- und Rechenverbrauch).

Auffallend war, das die Rechenlast bei allen Experimenten, bis auf denen zur Skalierbarkeit, meist niedrig war (unter 10 Prozent) und auch der Speicherbedarf der Experimente meist unter 150 MB RAM lag, gleich welcher Computer verwendet wurde. Das war nicht überraschend, da nur wenige Berechnungen angestellt werden mussten und die Hauptlast auf dem Austausch von Nachrichten und der Verwaltung vieler parallel laufender Prozesse (je Agent und Konversation) lag. Die Experimente liefen ohne Änderung mit derselben Geschwindigkeit auch auf kleineren Computern mit weniger Arbeitsspeicher. Es ist aber anzumerken, das sowohl ältere Linux Versionen als auch Versionen des Betriebssystems SUN Solaris Schwierigkeiten mit der schieren Anzahl der Prozesse hatten, da gerade umfangreichere Experimente oft nach unterschiedlicher Zeit unvermittelt stehen blieben. Leider traten hier keine Fehlermeldungen auf, weshalb der Grund für das Aussetzen nicht ermittelbar war. Dieses Verhalten zeigte sich ebenfalls bei der Java Umgebung von SUN und wird daher entweder in der verwendeten Agentenbibliothek oder in der Implementierung des Multiagentensystems vermutet.

## 6.3 Auswertung der Protokolldaten der Experimente

Aus den Verlaufsprotokollen aller Experimenten wurden anschließend unterschiedliche Werte extrahiert: die ausgetauschten Nachrichten hinsichtlich der verschiedenen Zustände der Koordinationsmechanismen, die Arbeitsbelastung der Maschinen und ihr Produktionsverhalten, die resultierenden Kontostände und Erfolg oder Misserfolg der Aufträge hinsichtlich der einzelnen Produkte. Alle genannten Werte finden sich in den folgenden Abschnitten als Tabellen wieder, sofern sie für das Experiment von Bedeutung waren. Die unterschiedlichen Werte in den Tabellen werden bei ihrer Verwendung im Text erläutert.

Desweiteren wurden Graphen erzeugt, die die Verläufe der Arbeitsbelastung der Maschinen und der Auftragsraten enthalten. Die Werte wurden alle fünf Minuten Modellzeit für den Abschnitt der nächsten 50 Minuten Modellzeit ermittelt und abgetragen (Arbeitsbelastung dabei in Prozent der Aktivität, Auftragsrate in Anzahl der Auftrage innerhalb des Zeitfensters). Die Anzahl der parallel laufenden Prozesse ist in einem eigenen Graph enthalten, der später angeführt wird.

Sowohl die Graphen als auch die Statistiken konnten in den meisten Fällen nur Hinweise geben und haben nicht die manuelle Analyse der Protokolldateien ersetzen können, wenn Gründe für auffällige Graphen oder Werte gesucht wurden.

## 6.4 Vergleich der Flexibilität

Da das implementierte Agentensystem in der Lage ist, jederzeit weitere Agenten zu starten (jedoch nicht bei Bedarf) und der Ausfall von Maschinen auch durch eine geänderte

Auftragsrate an das System dargestellt werden kann, wurden Experimente mit unterschiedlichen Auftragsströmen durchgeführt. Hier gibt es eine Einschränkung: der Ausfall einer Maschine kann dann nicht durch den Auftragsstrom wiedergegeben werden, wenn dieser Maschinentyp nur einmal im System vorkommt. In diesem Fall käme es früher oder später zu einem Stillstand wenigstens eines Teils des Produktionsprozesses. Da das Ende der Ausfallzeit nicht a priori bekannt ist, kann in dieser Situation auch nicht mit zeitlicher Sicherheit geplant werden. Ein großer Teil der dennoch eingeplanten Aufträge würde nicht zeitgerecht ausgeführt werden können.

### 6.4.1 Beschreibung der Datensätze

Um die Anpassungsfähigkeit der beiden Koordinationsmechanismen zu untersuchen, wurden Datensätze generiert, die Auftragsströme enthalten, in denen Aufträge in unterschiedlichen Abständen an das System gegeben wurden. Basis dafür war ein zufälliger Abstand zwischen 100 und 200 Sekunden Modellzeit, der im weiteren als „konstante“ Auftragsrate bezeichnet wird. Neben einem solchen Datensatz zu Vergleichszwecken, wurden Datensätze mit Auftragsraten generiert, die Auftragsschübe von langer (alle 10 Stunden einen Schub für die Dauer eine Stunde, jeweils Modellzeit) und von kurzer (alle 150 Minuten einen Schub für die Dauer von 20 Minuten, jeweils Modellzeit). Desweiteren enthalten die Datensätze entweder mit Aufträgen nur aus der Produktpalette (Hauptprodukte, im Beispielbetrieb Stühle, Tische, Schränke und Esszimmer), oder aus dem gesamten Produktionsspektrum, einschließliche der Teilprodukte. Alle Experimente wurden mit einem Agentensystem aus zehn Agenten durchgeführt, wobei für jeden der acht verschiedenen Maschinentypen ein Agent, für das Sägen und das Leimen je zwei Agenten gestartet wurden. In den letzten beiden Datensätzen wurde der Abstand zwischen zwei Aufträgen halbiert, um die Belastung des Systems zu erhöhen.

Es wurden noch weitere Experimente mit anderen Datensätzen durchgeführt, diese sind aber nicht explizit mit angeführt.

Die zeitlichen Verläufe der Auslastung und Auftragsraten der ersten beiden Experimente sind in Abbildung 6.1 (Kontrakt Netz) und 6.2 (Markt) zu sehen, die der übrigen wurden aus Platzgründen in den Anhang 7.2 verschoben[1]. In den Graphen sind jeweils drei Gruppen von Kurven zu sehen: die oberste Gruppe stellt den Verlauf der Auftragsrate und den der Erfolgs- bzw. Fehlerrate. Zusätzlich ist die Summe der Auslastung der Maschinen mit abgetragen. Die übrigen drei Gruppen beinhalten jeweils der Verlauf der Auslastung einer Maschine. Die Tabellen 6.2 und 6.3 beinhalten die Auslastung und das Produktionsverhalten der einzelnen Maschinen in Zahlen. Die Zahlen bezüglich der Aufträge beziehen sich auf alle Aufträge, also einschließlich der Teilaufträge, die an eine Maschine gingen. Bei den Abbruchzahlen handelt es sich um die Aufträge, die vom Auftraggeber abgebrochen wurden, in der Regel, weil ein anderes Teilprodukt nicht hergestellt werden konnte. Der Median in der letzten Spalte bezieht sich auf die in den Graphen dargestellten Auslastungskurven. Er wurde anstelle des Durchschnittes gewählt,

[1] Die Experimente haben in den Abbildungen im Bezeichner zusätzlich die Zeichenfolge `mt19937`, die den verwendeten Zufallszahlengenerator benennt. Da er hier nicht variiert wurde, wurde er im Text aus den Bezeichnern gestrichen

| Bezeichnung | Aufträge | Auftragsrate | Produktarten |
|---|---|---|---|
| lev0-4711/contract | 500 | konstant | nur Hauptprodukte |
| lev0-4711/market | 500 | konstant | nur Hauptprodukte |
| lev0-peaked-long-4711/contract | 500 | lange Spitzen | nur Hauptprodukte |
| lev0-peaked-long-4711/market | 500 | lange Spitzen | nur Hauptprodukte |
| lev0-peaked-short-4711/contract | 500 | kurze Spitzen | nur Hauptprodukte |
| lev0-peaked-short-4711/market | 500 | kurze Spitzen | nur Hauptprodukte |
| 6742-long/contract | 500 | lange Spitzen | alle |
| 6742-long/market | 500 | lange Spitzen | alle |
| 6742-short/contract | 500 | kurze Spitzen | alle |
| 6742-short/market | 500 | kurze Spitzen | alle |
| 9856/contract | 500 | konstant | alle |
| 9856/market | 500 | konstant | alle |
| small-peaked-long-42/contract | 1000 | lange Spitzen | alle |
| small-peaked-long-42/market | 1000 | lange Spitzen | alle |
| small-peaked-long-42-fast/contract | 1000 | 2fach, lange Spitzen | alle |
| small-peaked-long-42-fast/market | 1000 | 2fach, lange Spitzen | alle |

Tabelle 6.1: Datensätze für den Vergleich der Flexibilität

um ein realistischeres Bild der Gesamtauslastung zu geben. Käme es zum Beispiel zu wenigen Lastspitzen und vielen Leerlaufzeiten, wäre der Durchschnitt deutlich höher als der Median und würde eine „gute“ Auslastung der Maschine unterstellen. In der Summenzeile beinhaltet diese Spalte aus demselben Grund den Median über die Mediane. Alle angegebenen Zeiten sind wie auch in den Graphen in Realzeit.

| | | | Aufträge | | | Arbeitszeiten | | | Median |
|---|---|---|---|---|---|---|---|---|---|
| Agent | Anfr. | Angen. | Erfolg | Fehler | Abbr. | Laufz. | Aktiv | Leer | Last % |
| co-carving | 242 | 71 | 65 | 1 | 0 | 7528 | 5296 | 2039 | 48,5 |
| co-drilling | 71 | 19 | 9 | 0 | 0 | 7638 | 712 | 6738 | 0,0 |
| co-glueing-1 | 457 | 133 | 84 | 34 | 0 | 7632 | 4235 | 3397 | 59,5 |
| co-glueing-2 | 343 | 20 | 20 | 0 | 0 | 7600 | 1019 | 6466 | 0,0 |
| co-lathe | 235 | 78 | 75 | 1 | 0 | 7480 | 3499 | 3884 | 59,0 |
| co-nailing | 240 | 50 | 24 | 0 | 0 | 7612 | 3045 | 4401 | 12,5 |
| co-putting | 223 | 71 | 53 | 2 | 0 | 7605 | 2741 | 4710 | 2,0 |
| co-sawing-1 | 423 | 57 | 54 | 3 | 0 | 7548 | 1552 | 5789 | 0,0 |
| co-sawing-2 | 488 | 122 | 104 | 4 | 0 | 7638 | 5107 | 2326 | 50,5 |
| co-sewing | 327 | 79 | 69 | 4 | 0 | 7600 | 3634 | 3702 | 79,5 |
| Summe | 3049 | 700 | **557** | 49 | 0 | 75881 | **30840** | 43452 | 15,75 |

Tabelle 6.2: Maschinenauslastung Experiment Kontrakt Netz lev0-4711

### 6.4.2 Auffälligkeiten an den Graphen

An den Graphen fällt direkt auf, dass der Markt im Vergleich zum Kontrakt Netz eine konstant hohe Fehlerrate aufweist. In Tabelle 6.3 sieht man, das es sich dabei um Abbrüche der Produktion von Seiten des Produzenten handelt. Die Erklärung ist einfach:

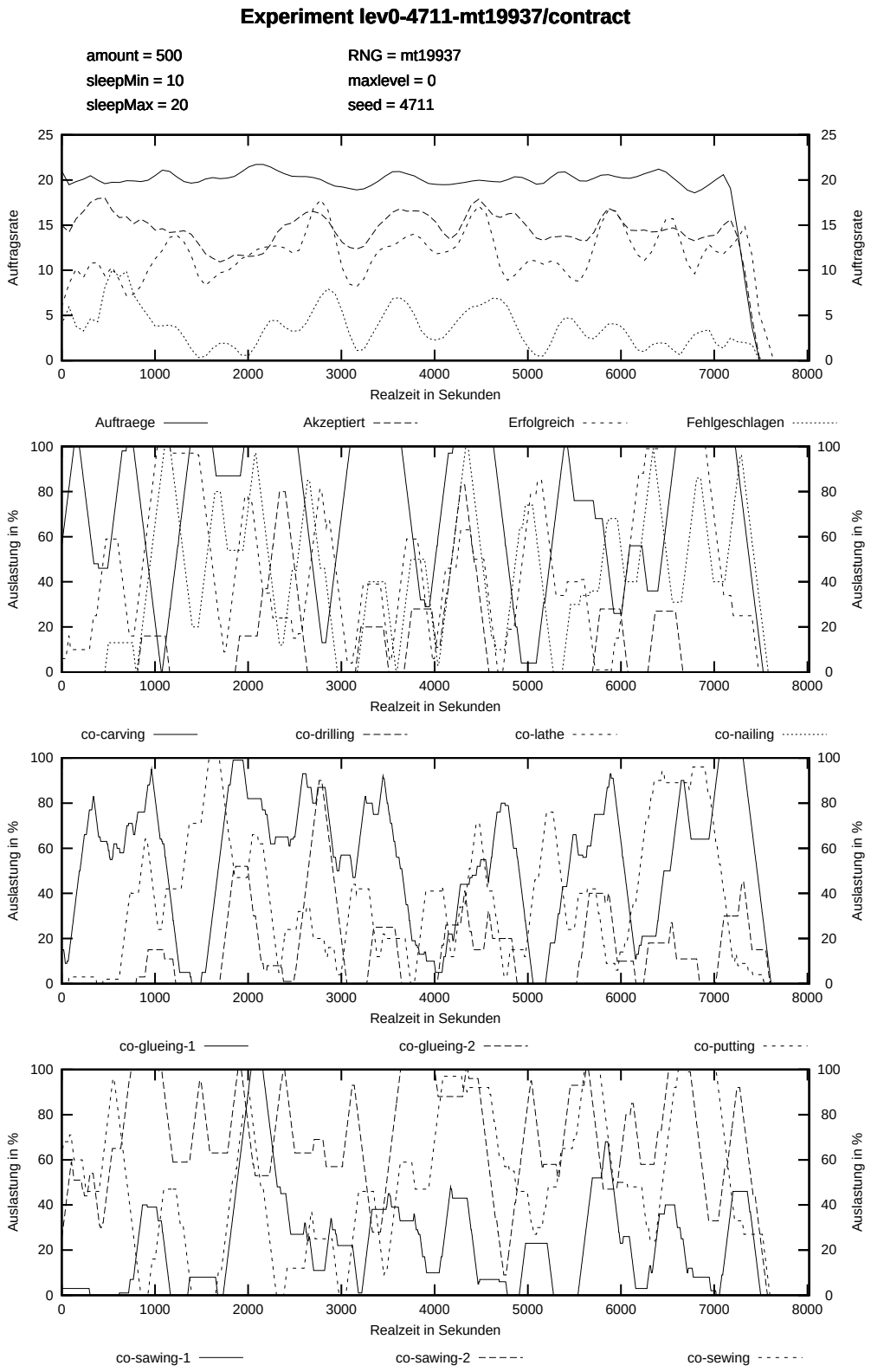



Abbildung 6.1: Experiment Kontrakt Netz lev0-4711

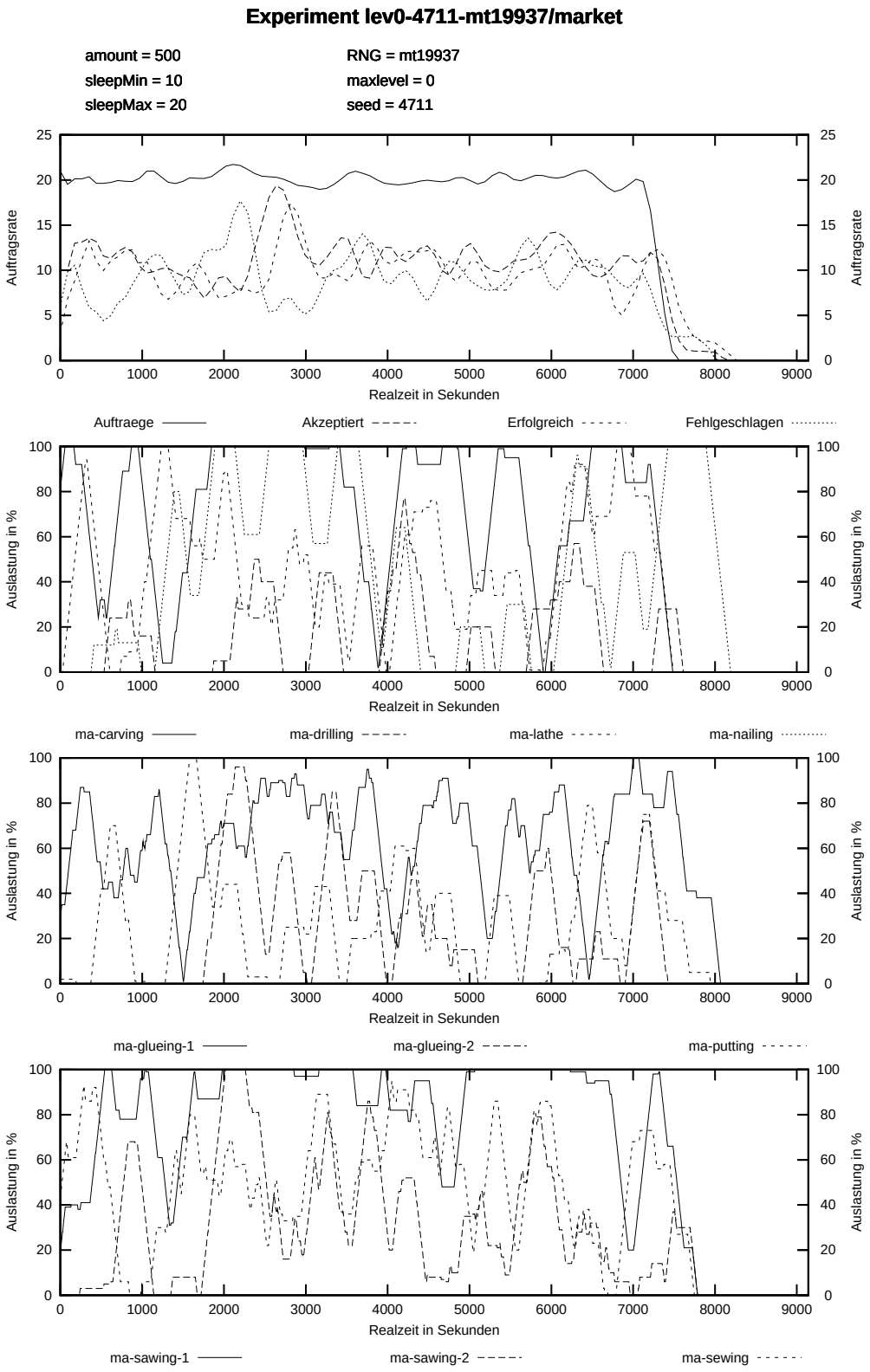



Abbildung 6.2: Experiment Markt lev0-4711

| | | | Aufträge | | | Arbeitszeiten | | | Median |
|---|---|---|---|---|---|---|---|---|---|
| Agent | Anfr. | Angen. | Erfolg | Fehler | Abbr. | Laufz. | Aktiv | Leer | Last % |
| ma-carving | 176 | 76 | 71 | 4 | 1 | 7486 | 5576 | 1806 | 32,0 |
| ma-drilling | 32 | 13 | 12 | 1 | 0 | 7788 | 1043 | 6583 | 0,0 |
| ma-glueing-1 | 554 | 241 | 113 | 44 | 84 | 8070 | 4712 | 2907 | 49,5 |
| ma-glueing-2 | 343 | 30 | 27 | 0 | 3 | 7486 | 1613 | 5720 | 0,0 |
| ma-lathe | 243 | 86 | 67 | 7 | 12 | 7491 | 3362 | 3927 | 35,5 |
| ma-nailing | 75 | 27 | 26 | 1 | 0 | 8194 | 3646 | 4413 | 12,0 |
| ma-putting | 103 | 42 | 41 | 0 | 1 | 7948 | 2055 | 5717 | 42,0 |
| ma-sawing-1 | 561 | 190 | 109 | 8 | 73 | 7788 | 6247 | 1371 | 85,5 |
| ma-sawing-2 | 467 | 101 | 77 | 9 | 15 | 7788 | 2416 | 5261 | 3,0 |
| ma-sewing | 435 | 186 | 95 | 3 | 88 | 7749 | 3906 | 3689 | 70,5 |
| Summe | 2989 | 992 | **638** | 77 | 277 | 77788 | **34576** | 41394 | 15,5 |

Tabelle 6.3: Maschinenauslastung Experiment Markt lev0-4711

während das Kontrakt Netz vor der Produktion alle Teilprodukte terminiert, beginnt der Markt sobald der erste Termin festgelegt ist. Kann der Auftrag aber aufgrund der bestehenden Arbeitslast nicht erfüllt werden, so muss die begonnene Produktion abgebrochen werden.

Im Graphen der Auftragsraten des Marktes fällt eine Spitze in der Fehlerrate um die Zeit 2000 auf. Eine Analyse der Protokolldateien ergab, dass es zu dieser Zeit zu einer verstärkten Nachfrage an Produkten kam, die als Tätigkeit „Sägen“ erforderten. Die entsprechenden Maschinen „ma-sawing-1“ und „ma-sawing-2“ waren dadurch zu 100 % ausgelastet und gaben zu dieser Zeit keine Angebote ab. Der nachfolgende Anstieg der Erfolgsrate ergibt sich aus den von beiden Maschinen produzierten Aufträge.

### 6.4.3 Vergleich der Produktionsraten

Vergleicht man nun die Erfolgsquote bei der Produktion, so ergibt sich folgendes Bild. Die Erfolgszahl liegt bei dem Markt 14,5 % über der des Kontrakt Netzes. Zieht man die Produktionsabbrüche des Marktes von der Anzahl der angenommenen Aufträge ab (diese entstanden bei dem Kontrakt Netz nicht erst), so hat der Markt eine um 9,7 % höhere Erfolgsquote (Differenz der Erfolgsquoten nach Formel (6.1)).

$$Erfolgsquote = \frac{Erfolg}{Annahmen} \tag{6.1}$$

Im Gesamt ist der Markt in der Produktion bei diesem Experiment also erfolgreicher als das Kontrakt Netz. Zieht man die Produktionserfolge bei Kundenaufträgen hinzu, die in Tabelle 6.4 enthalten sind, ergibt sich zunächst ein anderes Bild, das für das Unternehmen nicht weniger wichtig ist. Demnach erledigt das Kontrakt Netz 14,6 % mehr Aufträge als der Markt, was sich aber dadurch relativiert, dass es 25,8 % mehr Aufträge angenommen hat. Vergleicht man die Erfolgsquoten bei Kundenaufträgen, so ist der Markt mit 89,7 % um acht Prozentpunkte besser als das Kontrakt Netz, das nur 81,7 % der angenommenen Aufträge auch produziert, was mit den vorhergehenden Betrachtungen übereinstimmt.

Die höhere relative Erfolgsquote des Marktes wird durch die Ergebnisse der anderen Experimente belegt (siehe Tabellen 6.5 und 6.6): stets liegt die Erfolgsquote des Marktes bei Kundenaufträgen um wenigstens 10 % über der des Kontrakt Netzes, besonders auffallend im letzten Experiment, in dem der zeitliche Abstand zwischen den eingehenden Aufträgen halbiert war.

| | | Kontrakt Netz | | Markt | |
|---|---|---|---|---|---|
| Produkt | Aufträge | Angen. | Erfolgreich | Angen. | Erfolgreich |
| back | 29 | 17 | 16 | 15 | 14 |
| carved-chair | 32 | 19 | 6 | 4 | 4 |
| chair | 25 | 12 | 6 | 8 | 7 |
| chair-seat | 28 | 16 | 16 | 20 | 20 |
| chair-skeleton | 30 | 27 | 21 | 25 | 24 |
| cupboard | 23 | 19 | 9 | 13 | 12 |
| cupboard-back | 26 | 21 | 21 | 19 | 17 |
| cupboard-plate | 24 | 23 | 20 | 22 | 19 |
| cupboard-side | 29 | 27 | 25 | 26 | 23 |
| dining-area | 30 | 0 | 0 | 0 | 0 |
| flat-table | 27 | 20 | 18 | 11 | 11 |
| frame | 26 | 25 | 20 | 26 | 18 |
| seating | 20 | 16 | 16 | 14 | 13 |
| seats | 23 | 16 | 13 | 7 | 7 |
| short-table-feet | 20 | 18 | 18 | 15 | 14 |
| simple-carved-chair | 21 | 19 | 12 | 12 | 12 |
| skeleton | 18 | 16 | 11 | 12 | 8 |
| table | 22 | 16 | 13 | 11 | 11 |
| table-feet | 28 | 24 | 23 | 19 | 16 |
| table-plate | 19 | 15 | 15 | 12 | 11 |
| Total | 500 | 366 | **299** | 291 | **261** |

Tabelle 6.4: Produktionserfolge der Kundenaufträge im Experiment lev0-4711

**Vergleich der Auslastungen der Maschinen** Nachdem gezeigt wurde, wie sich die Produktionsraten verhalten, sollen die Auslastungen der Maschinen in den beiden Koordinationsmechanismen untersucht werden. Als Basis dient auch hier das Experiment `lev0-4711`. Die absolute Auslastung der Maschinen ist bei dem Markt mit einer Aktivitätszeit von 34576 um 13,4 % höher als bei dem Kontrakt Netz ($Aktiv_{Markt}/Aktiv_{KontraktNetz}$). Betrachtete man die Mediane der Auslastung, so gleichen sich beide Werte bis auf 0,25 %. Demnach muss sich die höhere Auslastung des Marktes aus Lastspitzen ergeben. Wie die Graphen in Abbildung 6.3 zeigen, ist die Gesamtauslastung der Systeme bei beiden Koordinationsformen ähnlich, sie bleibt bis auf einige Ausreißer, auf einem Level. Selbst der zeitweilige Wechsel des Levels im Markt Verändert den Median nicht.

Interessanter sind nach Tabelle 6.5 die Experimente `6742-short`, `lev0-peaked-short-4711` sowie `9856`, da sich hier die Mediane deutlich unterscheiden. Die Abbildungen der Grafiken dieser drei Experimente zeigten lediglich, das sich die

| | | | Aufträge | | | | Arbeitszeiten | | |
|---|---|---|---|---|---|---|---|---|---|
| Experiment | Anfr. | Angen. | Erfolg | Fehler | Abbr. | Laufz. | Aktiv | Leer | Median % |
| lev0-peaked-long-4711/contract | 2967 | 628 | 490 | 54 | 0 | 68877 | 26575 | 40523 | 22,5 |
| lev0-peaked-long-4711/market | 2876 | 902 | 544 | 59 | 277 | 62979 | 28884 | 32643 | 23,5 |
| lev0-peaked-short-4711/contract | 2995 | 657 | 486 | 58 | 0 | 67126 | 26484 | 38583 | 35,3 |
| lev0-peaked-short-4711/market | 2910 | 933 | 613 | 67 | 253 | 66576 | 33235 | 31772 | 40,5 |
| 6742-long/contract | 2863 | 672 | 517 | 59 | 0 | 76506 | 26398 | 48436 | 27,5 |
| 6742-long/market | 2741 | 881 | 569 | 74 | 238 | 67680 | 29247 | 35991 | 24,5 |
| 6742-short/contract | 2836 | 666 | 522 | 52 | 0 | 75426 | 26905 | 46188 | 24,0 |
| 6742-short/market | 2761 | 903 | 605 | 61 | 237 | 67185 | 31868 | 33005 | 40,0 |
| 9856/contract | 3047 | 745 | 558 | 58 | 0 | 77721 | 28808 | 47278 | 23,5 |
| 9856/market | 2962 | 995 | 665 | 65 | 260 | 77753 | 34602 | 40823 | 30,8 |
| small-peaked-long-42/contract | 5525 | 1365 | 1081 | 118 | 0 | 136326 | 54649 | 77777 | 21,5 |
| small-peaked-long-42/market | 5265 | 1710 | 1198 | 128 | 385 | 135111 | 62109 | 68951 | 24,0 |
| small-peaked-long-42-fast/contract | 4973 | 930 | 704 | 66 | 0 | 67675 | 37058 | 28883 | 56,0 |
| small-peaked-long-42-fast/market | 4643 | 1222 | 793 | 85 | 344 | 71576 | 46094 | 23597 | 67,0 |

Tabelle 6.5: Summen der Maschinenauslastung in weiteren Experimenten

| | | Kundenaufträge | | | |
|---|---|---|---|---|---|
| Experiment | Erfolgsquote in % | Aufträge | Angen. | Erfolgreich | Anteilig |
| lev0-peaked-long-4711/contract | 78,0 | 500 | 336 | 273 | 81,2 |
| lev0-peaked-long-4711/market | 87,0 | 500 | 241 | 220 | 91,3 |
| lev0-peaked-short-4711/contract | 74,0 | 500 | 346 | 275 | 79,5 |
| lev0-peaked-short-4711/market | 90,1 | 500 | 269 | 248 | 92,2 |
| 6742-long/contract | 76,9 | 500 | 340 | 266 | 78,2 |
| 6742-long/market | 88,5 | 500 | 262 | 237 | 90,5 |
| 6742-short/contract | 78,4 | 500 | 352 | 286 | 81,3 |
| 6742-short/market | 90,8 | 500 | 293 | 265 | 90,4 |
| 9856/contract | 74,9 | 500 | 375 | 299 | 79,7 |
| 9856/market | 90,5 | 500 | 304 | 280 | 92,1 |
| small-peaked-long-42/contract | 79,2 | 1000 | 725 | 594 | 81,9 |
| small-peaked-long-42/market | 90,4 | 1000 | 579 | 527 | 91,0 |
| small-peaked-long-42-fast/contract | 75,7 | 1000 | 495 | 393 | 79,4 |
| small-peaked-long-42-fast/market | 90,3 | 1000 | 312 | 291 | 93,3 |

Tabelle 6.6: Vergleich der Auslastungen aus Tabelle 6.5 auf der vorherigen Seite

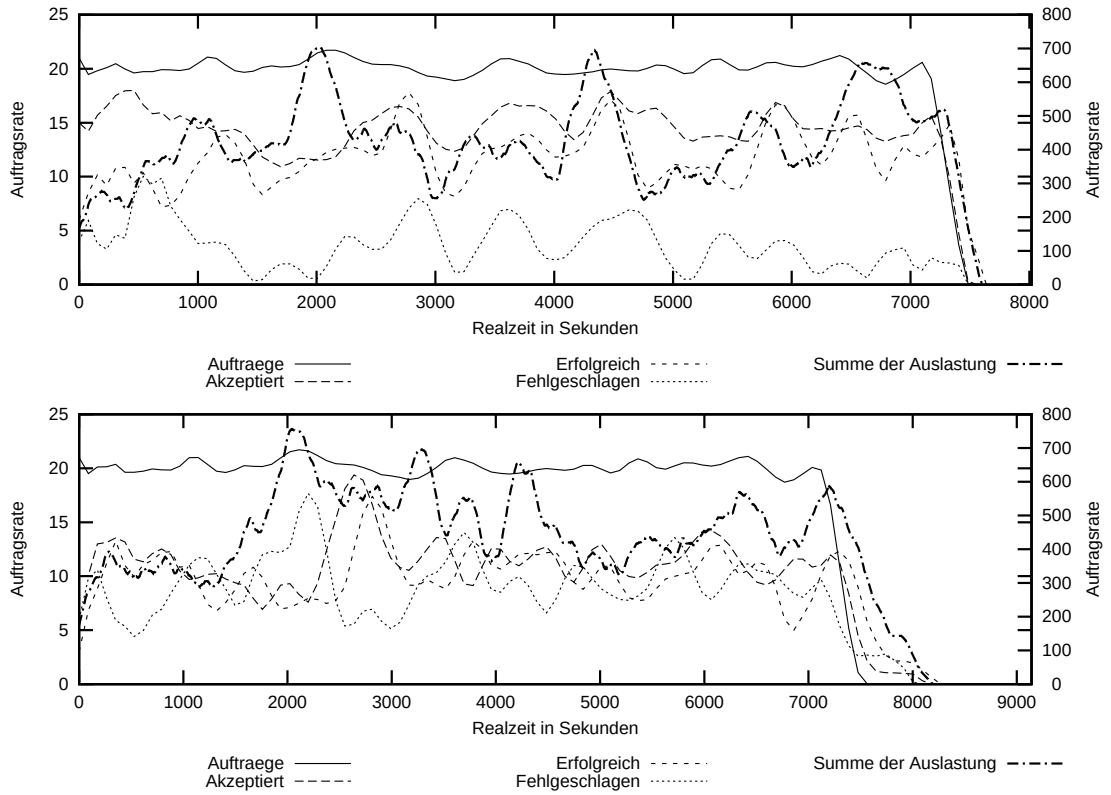



Abbildung 6.3: Abbildung der Gesamtlast im Experiment lev0-4711 Kontrakt Netz (oben) und Markt (unten)

Auslastung sowie alle anderen Werte grob an der Auftragsrate orientieren und deren Änderungen nachvollziehen, zum Beispiel bei `6742-short` in Abbildung 6.4. Ausnahme ist bei dem Markt die Zeit kurz vor dem Punkt 5000: die Akzeptanz- und Erfolgsrate fällt, währen die Fehlerrate steigt. Abbildung 7.5 und die Protokolldateien zeigen, das hier wieder die 100 % Last der beiden Maschinen „ma-sawing-1“ und „ma-sawing-2“ der Grund für den Anstieg der Fehlerrate.

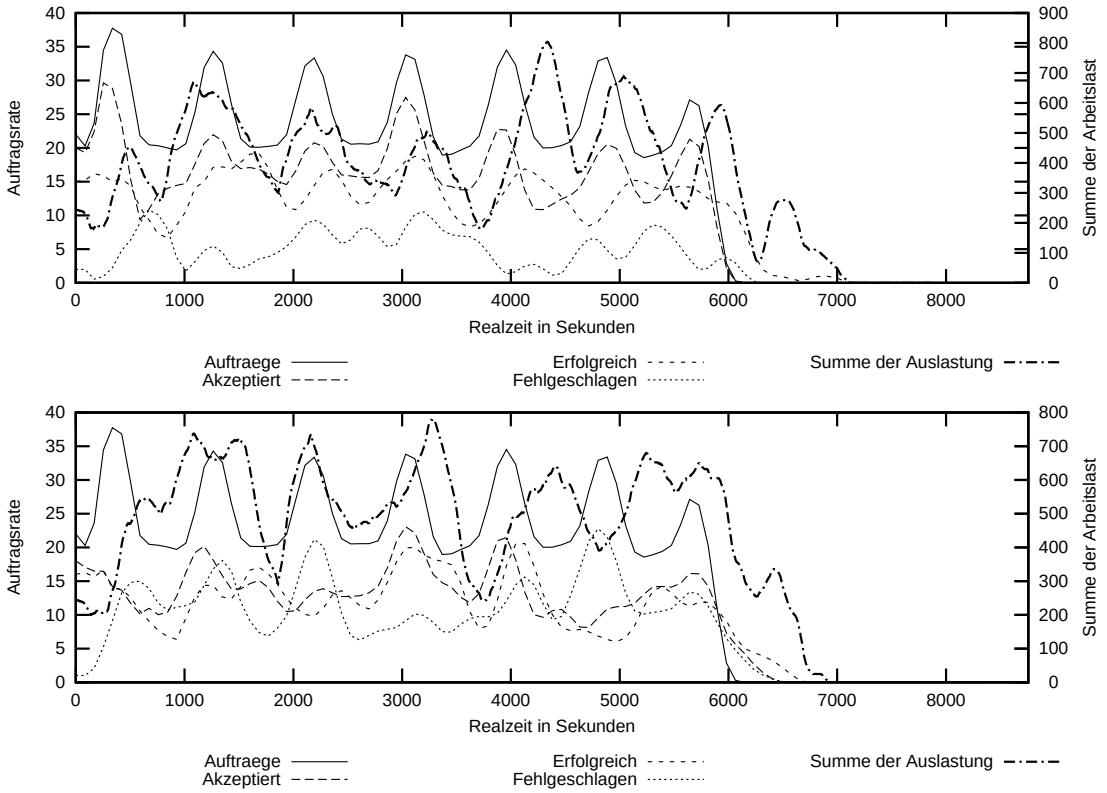



Abbildung 6.4: Abbildung der Gesamtlast im Experiment 6742-short Kontrakt Netz (oben) und Markt (unten)

Aus Tabelle 6.5 ist in der Spalte „Median“ zu sehen, dass die Auslastungen der Maschinen durchaus nicht gleich sind. So erzielt der Markt bei einem Datenstrom mit kurzen Auftragsstösen eine deutlich höhere Auslastung, als das entsprechende Kontrakt Netz. Eine Untersuchung der Protokolldateien ergibt, dass die „mehr“ eingehenden Aufträge bei dem Kontrakt Netz später terminiert werden, als bei dem Markt, da viele Zeiteinheiten bereits durch frühere Verträge belegt sind (siehe auch Abbildung 6.4).

**Diskussion der Ergebnisse** Die Experimente haben gezeigt, das die Koordinationsmechanismen zu unterschiedlichen Ergebnissen führen. Der Markt ist wesentlich agiler als das Kontrakt Netz und reagiert schnell auf Änderungen im System. Dies erreicht er vor allem dadurch, das er sich im Gegensatz zum Kontrakt Netz viele Möglichkeiten offen

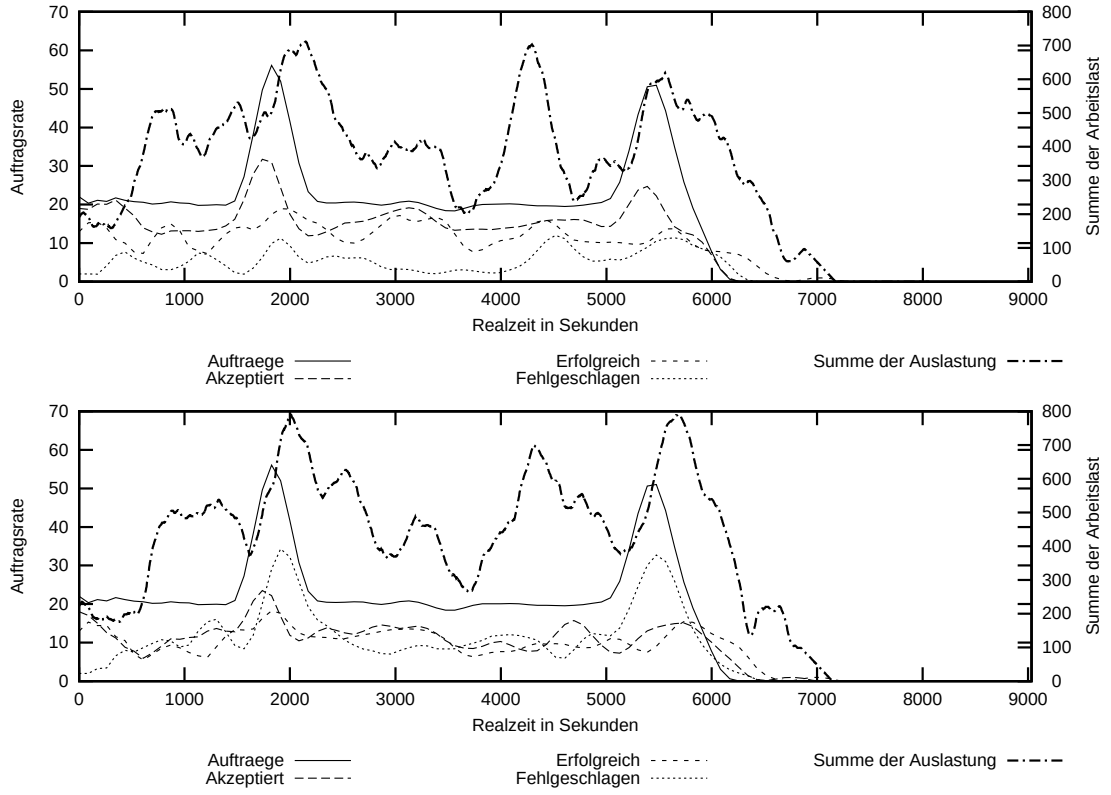



Abbildung 6.5: Abbildung der Gesamtlast im Experiment 6742-long Kontrakt Netz (oben) und Markt (unten)

hält, häufig aber auch auf gut Glück handelt (Beginn der Produktion ohne das Wissen, ob der Auftrag erfüllbar ist), nur um daran zu scheitern (hohe Abbruchquote, die hohe Kosten verursacht). Dadurch, dass der Markt sich nicht durch einen vollständigen Plan festlegt, hält er sich viele Möglichkeiten offen, um später die vorhandenen Kapazitäten besser nutzen zu können.

Das Kontrakt Netz geht mit der kompletten Planung eines Auftrages den sicheren Weg und erzielt dementsprechend eine Abbruchquote von 0, was mit dem Markt kaum vergleichbar ist. So entstehen dem Unternehmen nur wenige Kosten für halbfertige oder „versehentlich“ auf Vorrat produzierte Güter. Das Kontrakt Netz ist durch die komplette Planung allerdings wesentlich schwerfälliger als der Markt und kann nur verhältnismäßig langsam auf Änderungen reagieren (es sei denn, es ist in der Lage die Pläne abzuändern).

## 6.5 Vergleich der Skalierbarkeit

Um zu untersuchen, wie gut sich die beiden Koordinationsmechanismen bei größeren Systemen verhalten, wurden Datensätze mit 1000 Aufträgen und verschiedenen, im Vergleich zum vorhergehenden Abschnitt verkürzten, Auftragsraten generiert. Daraufhin wurden Experimente mit unterschiedliche großen Agentenpopulationen durchgeführt. An dieser Stelle muss leider gesagt werden, das ein Agentensystem mit mehr als 20 Agenten nicht mehr stabil lief. Zwar wurden eine Zeitlang alle Aktionen ausgeführt, doch meist nach kurzer Zeit blieb das System unvermittelt stehen. Die bis dahin generierten Daten waren aufgrund der geringen Menge nicht verwertbar.

Daher sollen an dieser Stelle die Mengen von ausgetauschten Nachrichten, sowie der nebenläufigen Prozesse beschrieben werden.

Aufgrund der einfachen Natur des Marktes ist anzunehmen, das hier wesentlich weniger Prozesse während eines Systemlaufs gestartet werden. Im Experiment `6742-long` wurden insgesamt 30254 Nachrichten bei dem Markt ausgetauscht (siehe Tabelle 6.8). Das Kontrakt Netz war wesentlich sparsamer, es benötigte nur 17858 Nachrichten, um seine Aufgabe zu bewältigen. Diese Zahlenverhältnisse finden sich in allen Experimenten wieder, so dass der Markt wider Erwarten das größere Nachrichtenaufkommen hat.

Die Anzahl der gestarteten, nebenläufigen Prozesse ist bei dem Kontrakt Netz erwartungsgemäß höher als bei dem Markt. Als Beispiel sollen die Graphen in Abbildung 6.6 dienen.

Auch wenn keine größeren Experimente durchgeführt werden konnten, lassen sich anhand der genannten Daten die potentiellen Engpässe bei einem Systemlauf ausmachen. Bei einem Markt müssen wesentlich mehr Nachrichten ausgetauscht werden, was für die verwendete Agentenplattform in der Regel kein Problem darstellte. Bei dem Kontrakt Netz müssen dagegen viele nebenläufige Prozesse erzeugt und gesteuert werden, womit offensichtlich ein Bestandteil des Systems so große Schwierigkeiten hatte, das die Experimente nicht durchzuführen waren.

| Agent | Gesendet | Empfangen | Gesamt |
|---|---|---|---|
| Buchhaltung | 557 | 1290 | 1847 |
| Lagerhaus | 47 | 47 | 94 |
| Kundenagent | 1015 | 1428 | 2443 |
| Maschine co-carving | 442 | 222 | 664 |
| Maschine co-drilling | 629 | 577 | 1206 |
| Maschine co-glueing-1 | 1045 | 492 | 1537 |
| Maschine co-glueing-2 | 628 | 378 | 1006 |
| Maschine co-lathe | 474 | 211 | 685 |
| Maschine co-nailing | 1427 | 1010 | 2437 |
| Maschine co-putting | 1648 | 1205 | 2853 |
| Maschine co-sawing-1 | 661 | 337 | 998 |
| Maschine co-sawing-2 | 880 | 351 | 1231 |
| Maschine co-sewing | 619 | 238 | 857 |
| Summe | 10072 | 7786 | 17858 |

Tabelle 6.7: Ausgetauschte Nachrichten im Experiment Kontrakt Netz 6742-long

| Agent | Gesendet | Empfangen | Gesamt |
|---|---|---|---|
| Buchhaltung | 237 | 2279 | 2516 |
| Lagerhaus | 3112 | 2085 | 5197 |
| Kundenagent | 500 | 997 | 1497 |
| Produzent | 4985 | 4811 | 9796 |
| Maschine ma-carving | 354 | 250 | 604 |
| Maschine ma-drilling | 95 | 71 | 166 |
| Maschine ma-glueing-1 | 957 | 739 | 1696 |
| Maschine ma-glueing-2 | 313 | 308 | 621 |
| Maschine ma-lathe | 384 | 308 | 692 |
| Maschine ma-nailing | 115 | 80 | 195 |
| Maschine ma-putting | 161 | 98 | 259 |
| Maschine ma-sawing-1 | 865 | 817 | 1682 |
| Maschine ma-sawing-2 | 667 | 575 | 1242 |
| Maschine ma-sewing | 883 | 768 | 1651 |
| Markt Agent | 1215 | 1225 | 2440 |
| Summe | 14843 | 15411 | 30254 |

Tabelle 6.8: Ausgetauschte Nachrichten im Experiment Markt 6742-long

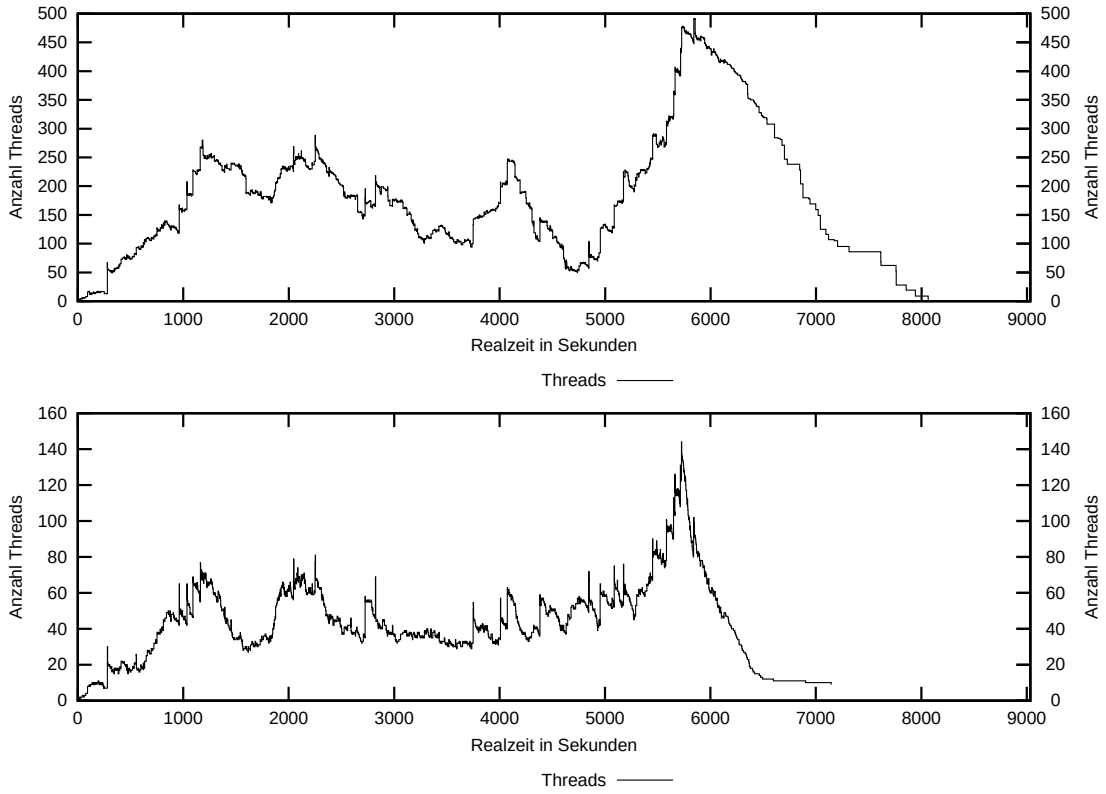



Abbildung 6.6: Anzahl der Threads über die Zeit im Experiment 6742-long

# 7 Zusammenfassung und Ausblick

In diesem Kapitel werden die Arbeit und die gewonnenen Erkenntnisse nocheinmal zusammengefasst. Anschließend wird ein Ausblick darauf gegeben, wie die Arbeit fortgeführt werden könnte.

## 7.1 Zusammenfassung der Arbeit und der Ergebnisse

Die Arbeit gab zunächst eine Übersicht über die Produktionsplanung und deren besonderem Aufgabengebiet, der Ablaufplanung, die vor allem in der auftragsorientierten Fertigung wichtig ist. Dies ist auch der Bereich der Produktionsplanung, in dem sich nicht mit Sicherheit geplant werden lässt, da zum einen die Nachfrage nach den einzelnen Produkten unsicher ist und zum anderen die Ausprägung der Produkte a priori nicht bekannt ist. Die Probleme der zentrale Ansätzen in der Produktionsplanung wurden genannt und der Hinweis auf die vermutete bessere Eignung dezentraler Lösungen gegeben. Als dezentraler Lösungsansatz wurde ein Multiagentensystem implementiert, an dem anschliessend verschiedene Eigenschaften untersucht wurden.

Dabei konnten die folgenden Fragestellungen beantwortet werden:

1. *Wie unterscheiden sich die verschiedenen Koordinationsmechanismen und unter welchen Bedingungen arbeiten sie gut?*

   Das Kontrakt Netz arbeitet zuverlässiger als der Markt, da Handlungen erst dann begonnen werden, wenn sie vollständig geplant sind. Es reagiert langsamer auf Änderungen als der Markt und verbraucht mehr Resourcen.

   Der Markt ist dagegen flexibler als das Kontrakt Netz in der vorliegenden Implementierung, da er nicht durch eine langfristige Planung gebunden ist und so besser auf Ereignisse reagieren kann. Aufgrund der fehlenden Planung arbeitet er aber weniger zuverlässig, was dann eine Rolle spielt, wenn zwischen den Tätigkeiten Abhängigkeiten bestehen, wie zum Beispiel in der Produktion.

2. *Sind die jeweiligen Koordinationsmechanismen in der Lage flexibel auf Änderungen der Umgebung zu reagieren?*

   Beide Koordinationsmechanismen konnten sich an Änderungen in der Umgebung anpassen. Das Kontrakt Netz reagiert empfindlich auf Störungen, die bereits geplante Vorgänge betreffen, dagegen der Markt flexibel darauf reagieren kann.

3. *Eignen sich Multiagentensysteme, um Probleme der Produktionsplanung zu lösen?*

> Diese Frage kann nicht mit Sicherheit mit Ja beantwortet werden. Im Rahmen der Untersuchungen konnte nicht festgestellt werden, ob das Agentensystem auch bei großen Problemstellungen einsetzbar ist. Ein Vergleich mit realen Daten konnte mangels Verfügbarkeit ebenfalls nicht durchgeführt werden, weshalb die damit zusammenhängende Frage, ob Multiagentensysteme die Problem ausreichend gut lösen nicht beantwortet werden.

Es sinnvoll mit dem bestehenden System weiterzuarbeiten. Mit Hilfe weiterer Experimente könnte man die gewonnenen Ergebnisse vertiefen und weitere Aspekte von Koordinationsmechanismen untersuchen. Da das Agentensystem bereits vorhanden ist, könnten auch weitere Koordinationsmechanismen Gegenstand von Untersuchungen sein.

Die Erkenntniss leiden unter der Tatsache, das keine realen Vergleichsdaten zur Verfügung standen. Ein Vergleich der Koordinationsmechanismen etwa mit einer a priori berechneten optimalen Lösung würde weiteren Aufschluss über deren Eignung als Lösungsansatz für die Produktionsplanung geben.

Die vorliegende Implementierung sollte hinsichtlich der Aufgabenstellung erweitert werden, da der gerade bei der auftragsorientierten Werkstattfertigung wichtige Transport nicht umgesetzt wurde.

Dennoch wurden wichtige Erkenntnisse über Eigenschaften von Koordinationsmechanismen und Multiagentensystemen gewonnen, die deren Einsatz vielversprechend erscheinen lassen.

## 7.2 Ausblick

Mit Hilfe des implementierten Multiagentensystems könnten unter Verwendung realer Daten eines Unternehmens Optimierungen des Produktionsprozesses durchgeführt werden. Mit Hilfe verschiedener Testläufe des Systems wäre es mögllich, Engpässe aber auch überflüssige Resourcen zu identifizieren und verschiedene Konfigurationen des Produktionsprozesses durchzuspielen.

Das Implementierte Agentensystem eignet sich aber auch als Platform zur Untersuchung weiterer Koordinationsmechanismen. Für viele Mechanismen ist es noch immer eine offene Frage, unter welchen Umständen sie am besten arbeiten, und wo ihre Grenzen liegen. Aber auch die Erprobung neuer Koordinationsmechanismen, oder von Mischformen könnte untersucht werden. Damit könnte ein wichtiger Beitrag zur Forschung an Multiagentensystemen geleistet werden.

Die zur Bewertung der Koordinationsmechanismen verwendeten Kriterien könnten verfeinert und auf ihre allgemeine Anwendbarkeit hin überprüft werden. Mit Hilfe unterschiedlicher Koordinationsmechanismen wäre es möglich, die Kriterien an Testläufen anzuwenden und weiterzuentwickeln.

# Bedienung der Implementierung

Im folgenden sollen die Bedienung des implementierten Systems kurz beschrieben werden. Alle Aufrufe an das System werden über Skripten in der Sprache Ruby gesteuert. Teilweise werden dabei Bibliotheken für Ruby eingesetzt, die aber alle in dem Quellverzeichnis der Diplomarbeit zu finden sind (`tools/ruby`). Sämtliche Skripte geben bei dem Parameter `--help` eine Übersicht über die akzeptierten Parameter aus.

**Start des Agentensystems:** Das Skript `start.rb` startet das Agentensystem in unterschiedlichen Konfigurationen. Als Parameter werden akzeptiert: `--logdir` um das Zielverzeichnis für die Protokolldateien anzugeben, `--datafile` um die Datei mit dem zu verwendenden Auftragsstrom anzugeben.

Um das System verteilt laufen zu lassen, muss es mit unterschiedlichen Namen und Ports gestartet werden. Dazu dienen `--port` und `--name`. Mit Hilfe von `--remotedf` und `--remoteurl` werden die Instanzen untereinander bekannt gemacht. Es genuegt, eine Instanz zu starten und alle weiteren darauf zu verweisen.

Nach diesen (optionalen) Parametern folgt eine Liste von XML Dateien, aus denen das System initialisiert werden soll.

**Generierung von Experimentdaten:** Das Skript `generateEvents.rb` liest Produktpläne aus XML Dateien und generiert Auftragsströme unterschiedlicher Natur. Die Aufrufparameter mit ihrer Bedeutung sind in Tabelle 7.1 beschrieben.

**Auswertung der Protokolldateien:** Das Skript `makeAllStats.rb` steuer die drei Skripten `makestats.rb` (Auswertung des Produktionsverhaltens), `computeload.rb` (Auswertung der Arbeitslast) und `makeGraphs.rb` (Erzeugung von Graphen aus den Daten der Arbeitslast und des Produktionsverhaltens). Als Parameter werden `--graphs` (Graphen neu erzeugen), `--changed` (nur bei geänderten Protokolldateien Berechnungen anstellen) und `--basedir` (Alle unterhalb dieses Verzeichnisses gefundenen Protokolldateien auswerten) akzeptiert

Sofern der Parameter `--basedir` nicht angegeben wurde, wird eine Liste von Verzeichnissen mit Protkoldateien erwartet.

| Parameter | Wert | Bedeutung |
|---|---|---|
| amount | Integer | Menge der zu generierenden Aufträge |
| seed | Integer | Initialisierungswert für den Zufallszahlengenerator |
| maxlevel | Integer | Maximale Tiefe der gewählten Produkte aus den Arbeitsplänen. Tiefe 0 bedeutet nur Produkte von der obersten Ebene, was dem Produktprogramm des Betriebes entspricht |
| productfile | Dateiname | Datei mit den Definitionen der Arbeitspläne. |
| rng | Algorithmus | ...der für die Generierung von Zufallszahlen verwendet werden soll. Zur Auswahl stehen unter anderem der Standardgenerator von Unix Systemen, der auch von Java verwendet wird, Mersenne Twister (mt19937) [Matsumoto und Nishimura 1998] und Ranlux [Lüscher 1994]. Die beiden letzteren sind in verschiedenen Softwarepaketen für wissenschaftliche Berechnungen implementiert und werden meistens als gut bezeichnet. Leider lies sich kein neuerer Vergleich finden, der diese tatsache objektiv bestätigt. |
| sleepMin, sleepMax | Integer | Minimale und maximale Anzahl der realen Sekunden, die zwischen zwei Aufträgen vergehen sollen. |
| peakFrequency | Sekunden | Anzahl der Sekunden, die zwischen zwei erhöhten Auftragsaufkommen vergehen soll |
| peakDuration | Sekunden | Dauer eines erhöhten Auftragsaufkommens |
| peakModifier | Zahl | Wert, durch den die ursprünglich vorgesehene (sleepMin und sleepMax) Wartezeit zwischen zwei Aufträgen während eines erhöhten Aufkommens geteilt werden soll |

Tabelle 7.1: Parameter für die Generierung von Auftragsströmen

# Beschreibung der Produktpläne des Beispielbetriebs Schreinerei

Der Beispielbetrieb Schreinerei kann drei verschiedene Produktarten herstellen (siehe Abschnitt 6.2), die in Abbildung 7.1 dargestellt sind. Die Tabellen 7.3 und 7.3 listen die weiteren Daten wie Resourcen und Produktionsdauer in Modellzeit auf. Da keine verwendbaren realen Daten zur Verfügung standen, wurden unterschiedliche Produktpläne aufgestellt, die nicht notwendigerweise realistisch sind. Sie sind werden aber als hinreichend sinnvoll angenommen, um Experimente durchführen zu können.

Es gibt im System acht verschiedene Maschinentypen, die je eine der Tätigkeiten Leimen, Nähen, Nageln, Zusammenfügen, Schnitzen, Sägen, Drehen oder Bohren ausführen können. Zusätzliche hat jede Maschine individuelle Rüstzeiten, Rüstkosten, Fehlerraten, Betriebskosten, Servicezyklen, sowie einen Geschwindigkeitsfaktor, durch den die in dem Produktplan angegebene Produktionsdauer geteilt wird.

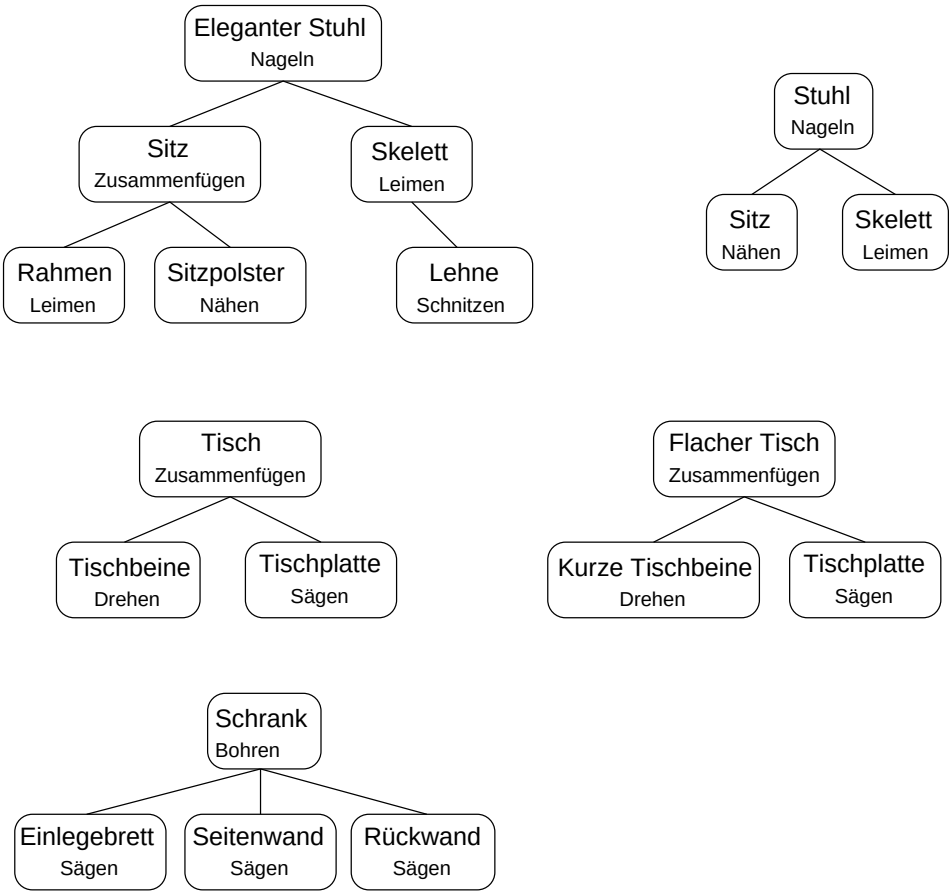



Abbildung 7.1: Produktpläne des Beispielbetriebs

| | |
|---|---|
| **Eleganter Stuhl** | 1 Sitz, 1 Skelett |
| | Nageln, 3 Minuten |
| | 50 Nägel |
| Sitz | 1 Rahmen, 1 Sitzpolster |
| | Zusammenfügen, 10 Sekunden |
| | – |
| Rahmen | – |
| | 4 Holzstücke |
| | Leimen, 150 Sekunden |
| Sitzpolster | – |
| | Nähen, 90 Sekunden |
| | 1x Teurer Stoff, 1x Füllung |
| Skelett | 1 Lehne |
| | Leimen, 4 Minuten |
| | 30 Holzstücke |
| Lehne | – |
| | Schnitzen, 2 Minuten |
| | 1 Holzplatte |
| **Stuhl** | 1 Sitz, 1 Skelett |
| | Nageln, 5 Minuten |
| | 50 Nägel |
| Sitz | – |
| | Nähen, 50 Sekunden |
| | 1x Stoff, 1x Füllung |
| Skelett | – |
| | Leimen, 50 Sekunden |
| | 11 Holzstücke |
| **Tisch** | 4 Tischbeine, 1 Tischplatte |
| | Zusammenfügen, 2 Minuten |
| | – |
| Tischbeine | – |
| | Drehen, 30 Sekunden |
| | je ein Rundholz |
| Tischplatte | – |
| | Sägen, 100 Sekunden |
| | 1 große Holzplatte |

Tabelle 7.2: Detaildaten zu den Produktplänen (I)

| | |
|---|---|
| **Flacher Tisch** | 4 kurze Tischbeine, 1 Tischplatte |
| | Zusammenfügen, 2 Minuten |
| | – |
| Kurze Tischbeine | – |
| | Drehen, 30 Sekunden |
| | je ein Rundholz |
| Tischplatte | – |
| | Sägen, 100 Sekunden |
| | 1 große Holzplatte |
| **Schrank** | 1 Rückwand, 2 Seitenwände, 6 Einlegebretter |
| | Bohren, 2 Minuten |
| | 25 kleine und 8 große Schrauben |
| Einlegebrett | – |
| | Sägen, 30 Sekunden |
| | je eine kurze dickere Holzplatte |
| Seitenwand | – |
| | Sägen, 30 Sekunden |
| | eine lange dickere Holzplatte |
| Rückwand | – |
| | Sägen, 30 Sekunden |
| | eine große, dünne Holzplatte |

Tabelle 7.3: Detaildaten zu den Produktplänen (II)

# Graphen weiterer Experimente

Dieser Anhang enthält die Graphen der im Kapitel 6 erwähnten und aus Gründen der Übersichtlichkeit dort nicht abgedruckten Experimente. Damit die Graphen desKontrakt Netzes und des Marktes zu einem Experiment auf einer Seite abgebildet sind, ist der Rest dieser Seite leer.

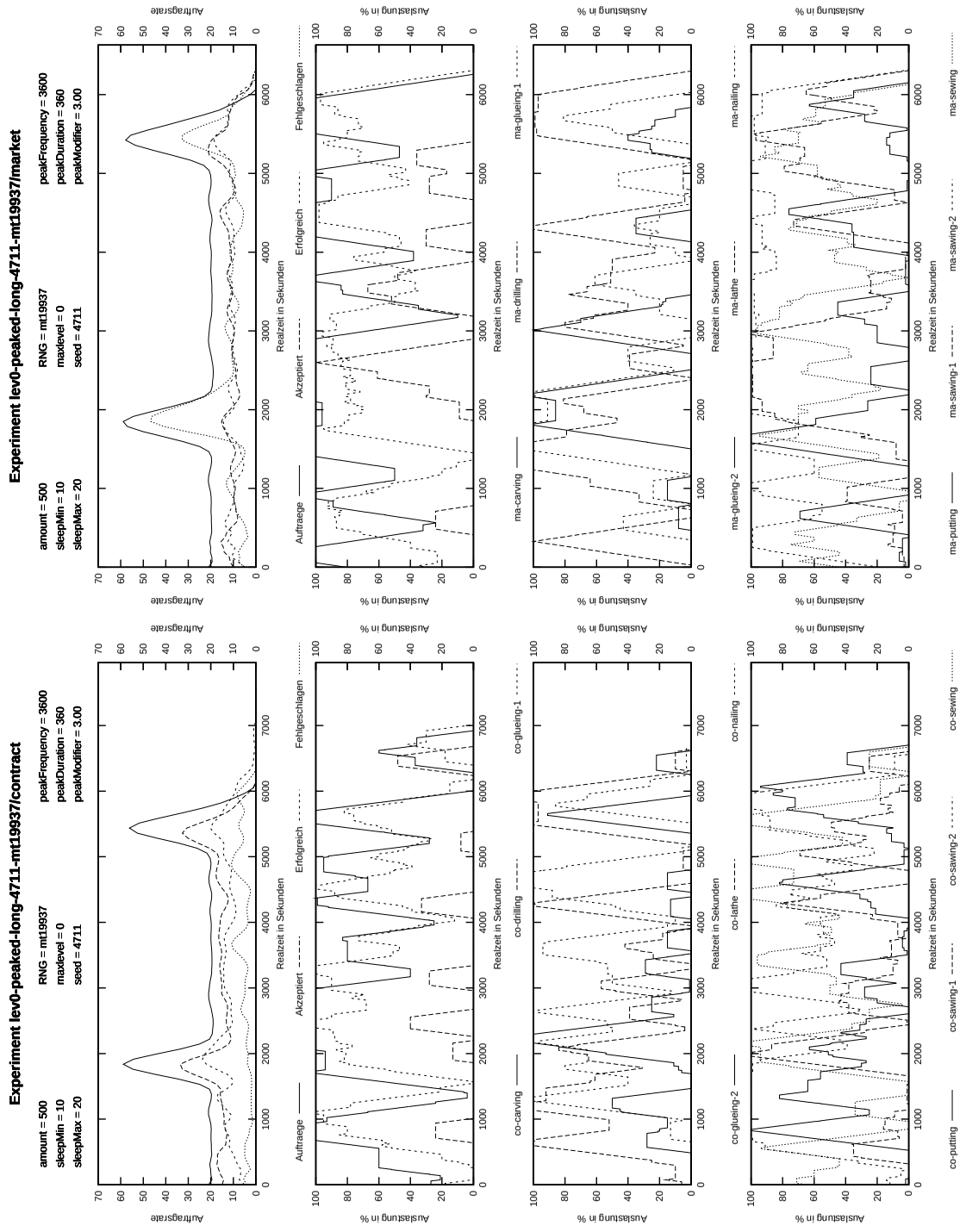



Abbildung 7.2: Graphen für die Experimente lev0-peaked-long-4711

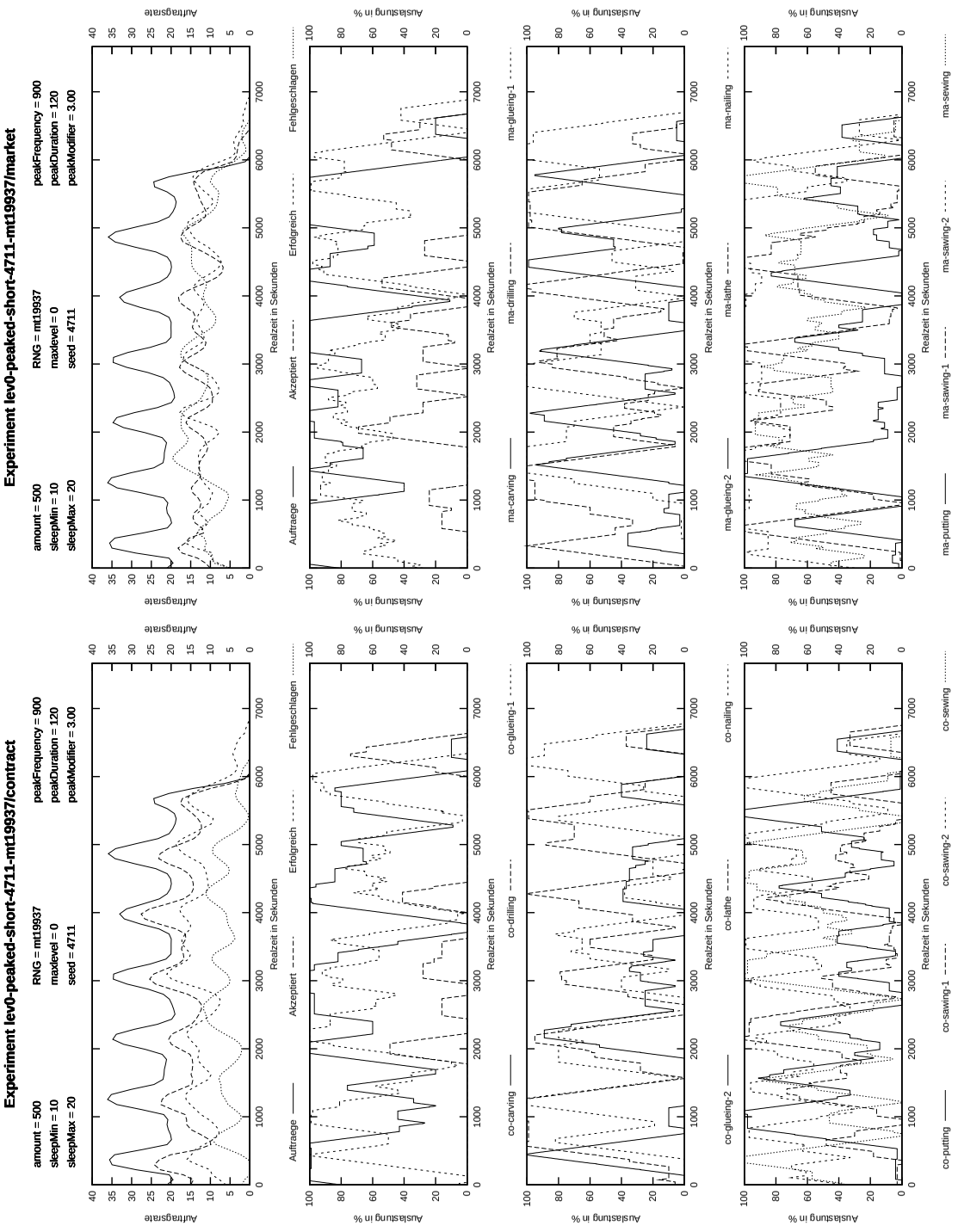



Abbildung 7.3: Graphen für die Experimente lev0-peaked-short-4711

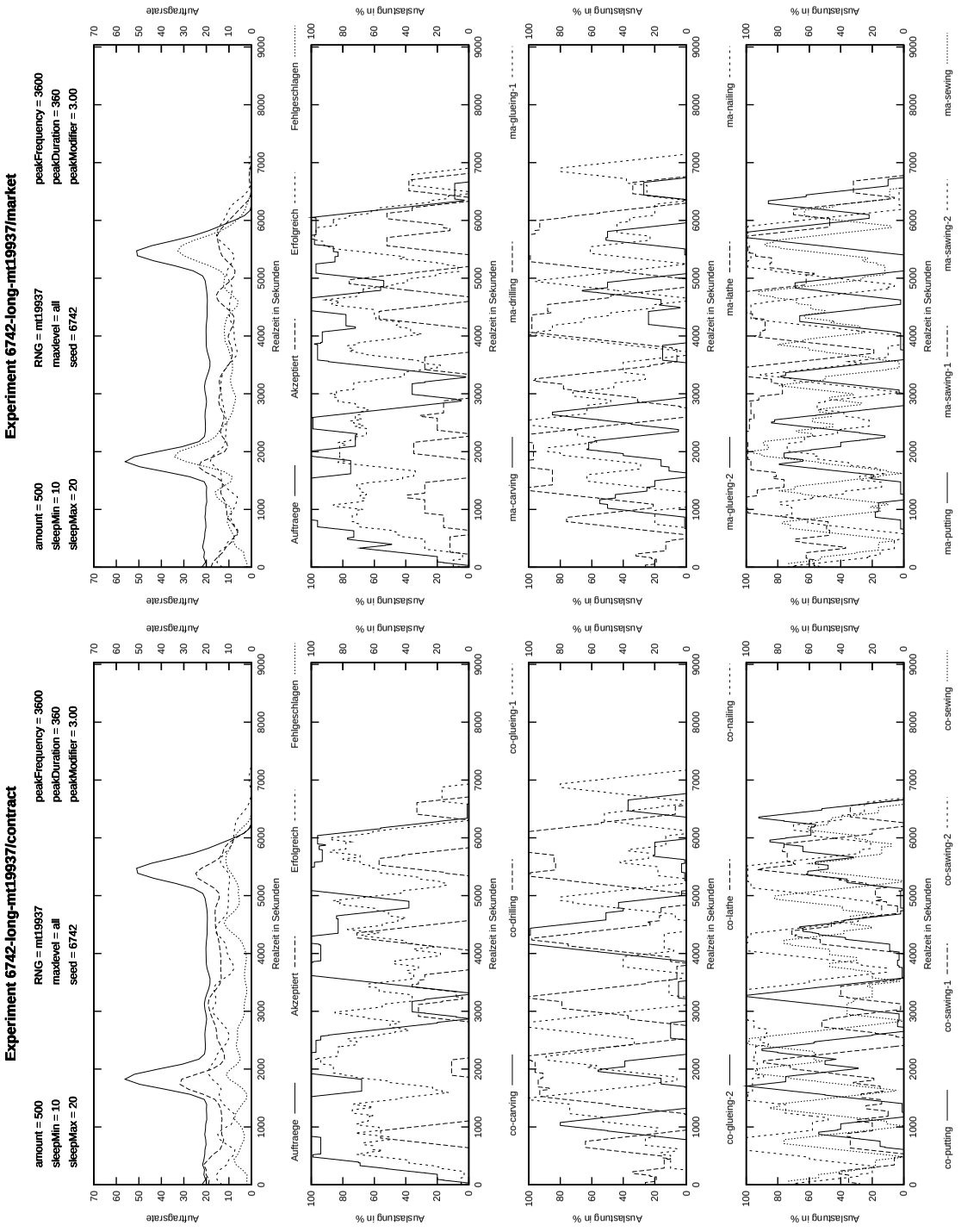



Abbildung 7.4: Graphen für die Experimente 6742-long

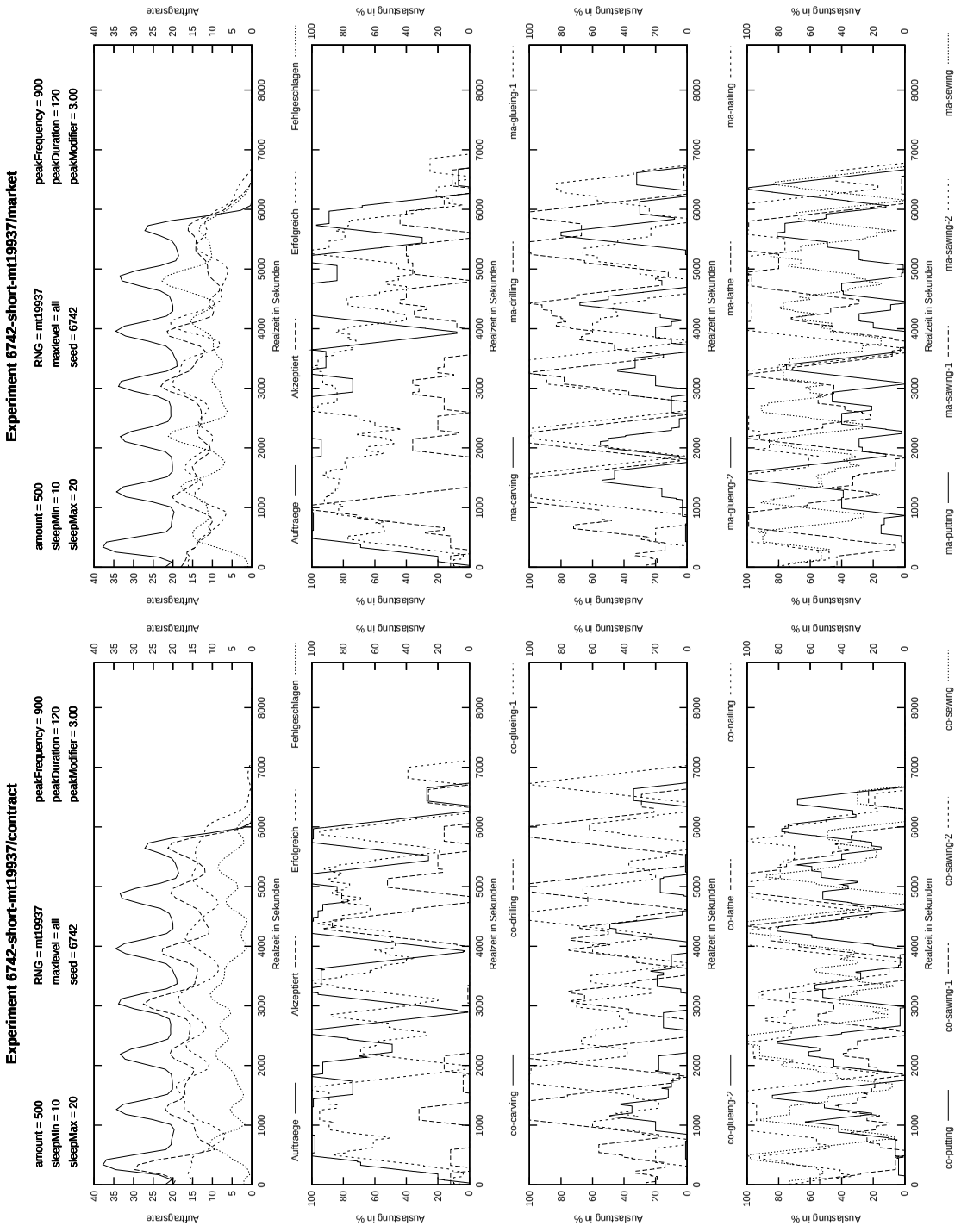



Abbildung 7.5: Graphen für die Experimente 6742-short

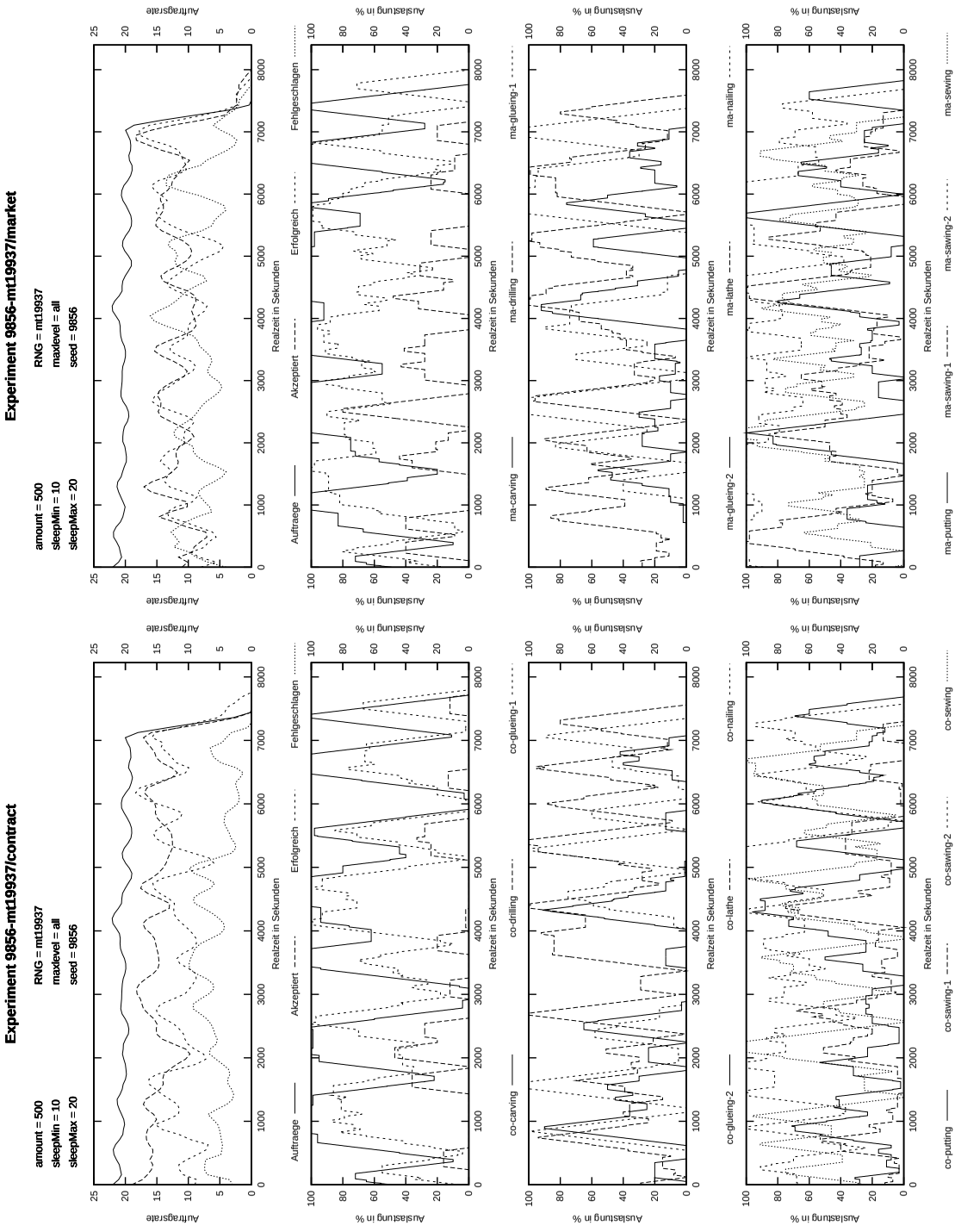



Abbildung 7.6: Graphen für die Experimente 9856

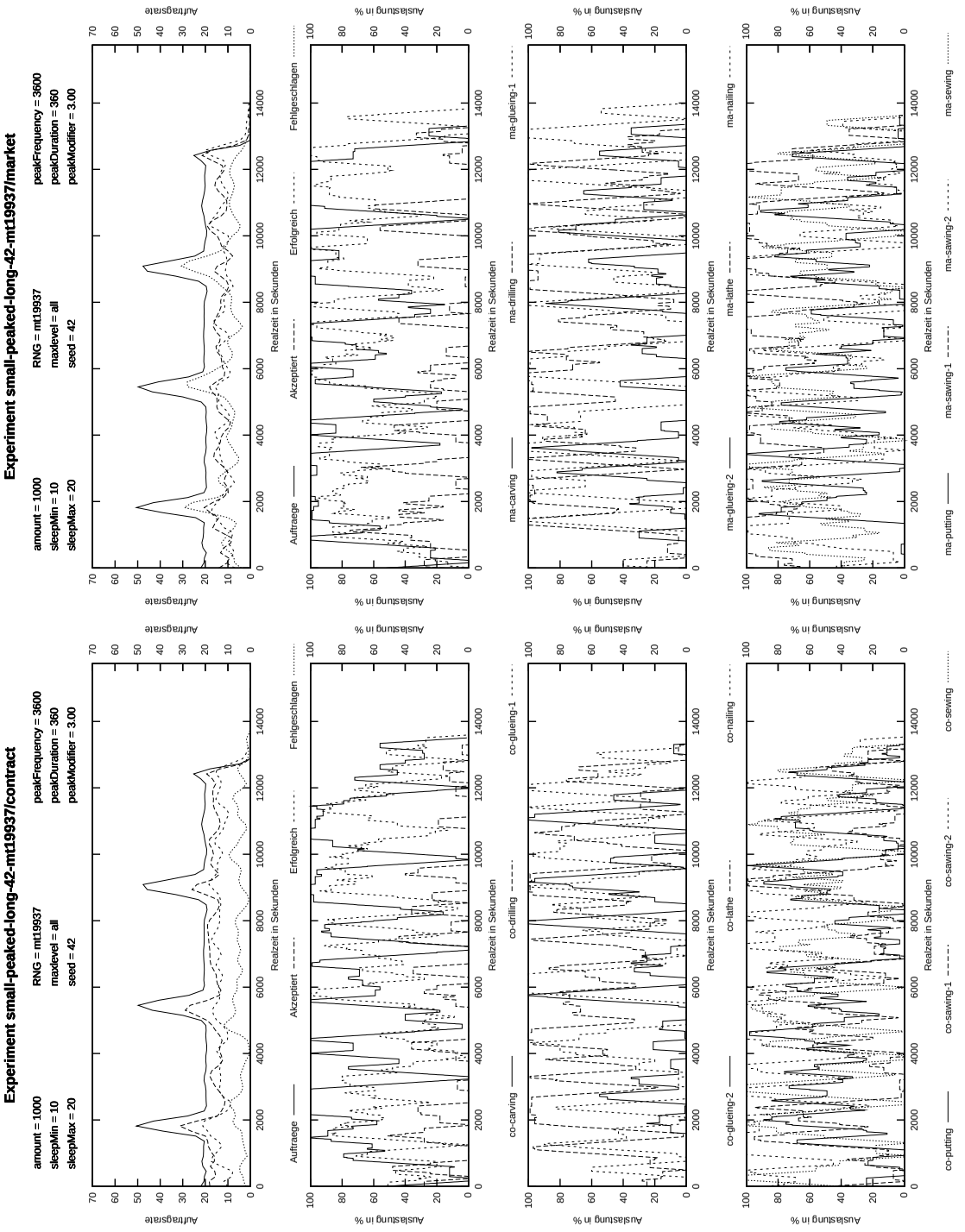



Abbildung 7.7: Graphen für die Experimente peaked-long-42

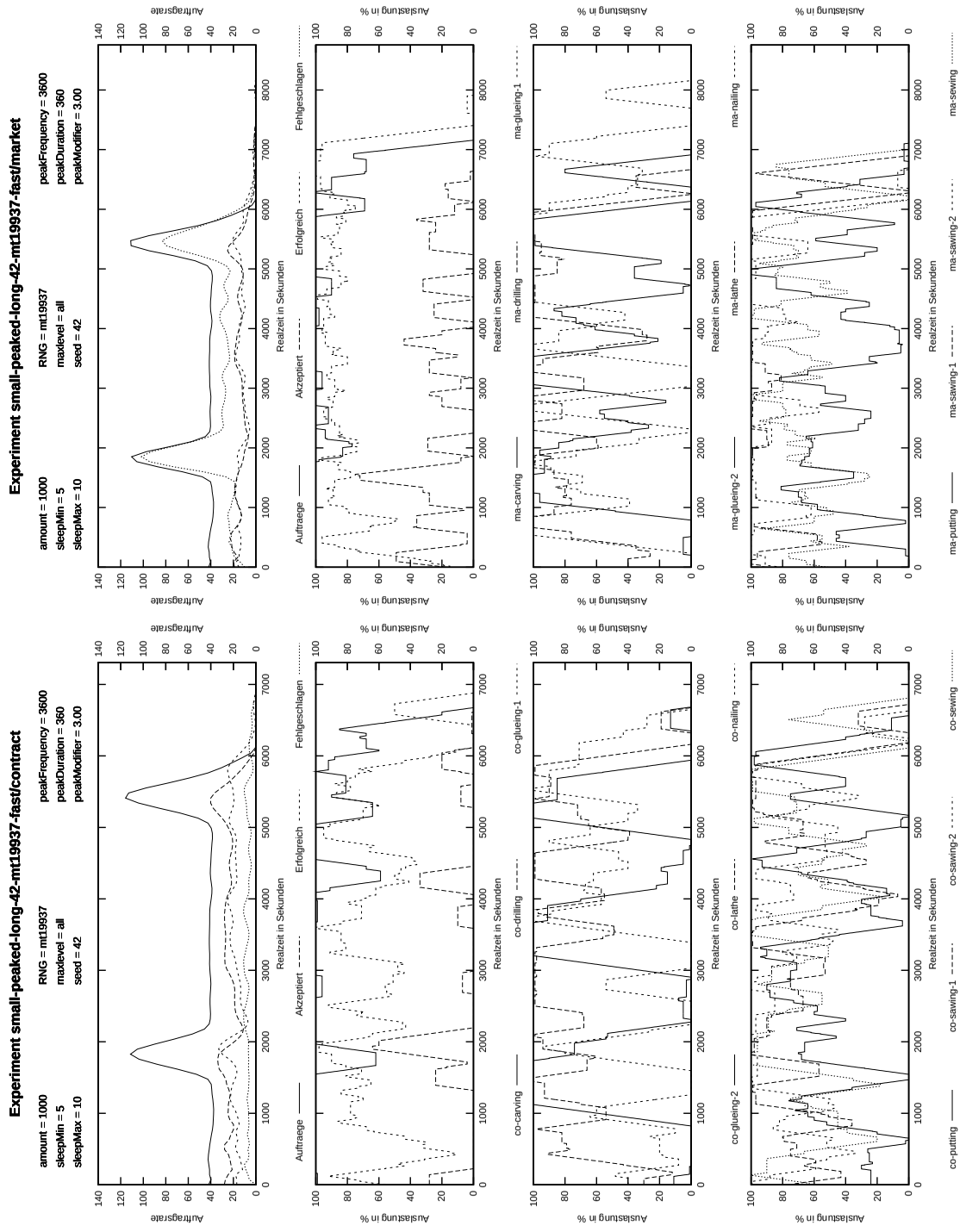



Abbildung 7.8: Graphen für die Experimente peaked-long-42-fast

# Literaturverzeichnis


[ABUILDER 2000] ABUILDER (2000). *Agent Construction Tools*. `http://www.agentbuilder.com/AgentTools/index.html`.

[Austin 1979] Austin, John L. (1979). *Zur Theorie der Sprechakte (how to do things with words)*. Phillip Reclam Jun., Stuttgart. zweite Auflage, zugrundeliegende Originalausgaben von 1962 und 1975.

[Behme und Mintert 1998] Behme, Henning und S. Mintert (1998). *XML in der Praxis – Professionelles Web-Publishing mit Extensible Markup Language*. Addison-Wesley.

[Bürckert et al. 1998] Bürckert, Hans-Jürgen, K. Fischer und G. Vierke (1998). *Teletruck: A holonic fleet management system*. In: Trappl, R., Hrsg.: *Proceedings of the 14th European Meeting on Cybernetics and Systems Research*, Bd. 2, S. 695 – 700.

[Burmeister 1996] Burmeister, Birgit (1996). *Models and methodology for Agent-oriented Analysis and Design*. In: Fischer, Klaus, Hrsg.: *Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems*, Nr. D-96-06 in *DFKI Document*.

[Bussmann et al. 2001] Bussmann, S., N. R. Jennings und M. Wooldridge (2001). *On the Identification of Agents in the Design of Production Control Systems*. In: Ciancarini, P. und M. Wooldridge, Hrsg.: *Agent-Oriented Software Engineering*, Bd. 1975 d. Reihe *LNCS*, S. 141 –162. Springer, Berlin, Germany.

[Castelfranchi 1997] Castelfranchi, Cristiano (1997). *To Be or Not to Be an Agent*. In: Müller, Jörg P, M. J. Wooldridge und N. R. Jennings, Hrsg.: *Intelligent Agents III*, S. 37 – 40, Berlin, Heidelberg, New York. Springer.

[Corsten und Gössinger 1997] Corsten, Hans und R. Gössinger (1997). *Multiagentengestützte Störungsbehandlung auf der Grundlage der opportunistischen Koordinierung*. Schriften zum produktionsmanagement 14, Lehrstuhl für Produktionswirtschaft, Universität Kaiserslautern.

[Finin et al. 1994] Finin, T., R. Fritzson, D. McKay und R. McEntire (1994). *KQML as an Agent Communication Language*. In: *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM94)*, S. 456 – 463. ACM Press.

[FIPA] FIPA. *Foundation of Intelligent Physical Agents*. http://www.fipa.org.

[FIPA/ACL 2000] FIPA/ACL (2000). *FIPA Agent Communication Language Message Structure Specification*. FIPA, Geneva, Switzerland. http://www.fipa.org/specs/fipa00061.

[FIPA/AMS 2000] FIPA/AMS (2000). *Agent Management Specification*. FIPA, Geneva, Switzerland. http://www.fipa.org/specs/fipa00023.

[FIPA/CA 2000] FIPA/CA (2000). *FIPA Communicative Act Library Specification*. FIPA, Geneva, Switzerland. http://www.fipa.org/specs/fipa00037.

[FIPA/CCL 2000] FIPA/CCL (2000). *FIPA Constraint Choice Language – Content Language Specification*. FIPA, Geneva, Switzerland. http://www.fipa.org/specs/fipa00009.

[FIPA/CL 2000] FIPA/CL (2000). *FIPA Content Languages Specification*. FIPA, Geneva, Switzerland. http://www.fipa.org/specs/fipa00007.

[FIPA/IP 2000] FIPA/IP (2000). *FIPA Interaction Protocol Library Specification*. FIPA, Geneva, Switzerland. http://www.fipa.org/specs/fipa00025.

[FIPA/KIF 2000] FIPA/KIF (2000). *FIPA Knowledge Interchange Format – Content Language Specification*. FIPA, Geneva, Switzerland. http://www.fipa.org/specs/fipa00010.

[FIPA/RDF 2000] FIPA/RDF (2000). *FIPA Resource Definition Format – Content Language Specification*. FIPA, Geneva, Switzerland. http://www.fipa.org/specs/fipa00011.

[FIPA/SL 2000] FIPA/SL (2000). *FIPA Semantic Language – Content Language Specification*. FIPA, Geneva, Switzerland. http://www.fipa.org/specs/fipa00008.

[Franklin und Graesser 1997] Franklin, Stan und A. Graesser (1997). *Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents*. In: Müller, Jörg P, M. J. Wooldridge und N. R. Jennings, Hrsg.: *Intelligent Agents III*, S. 21 – 36, Berlin, Heidelberg, New York. Springer.

[Gutenberg 1983] Gutenberg, E. (1983). *Grundlagen der Betriebswirtschaftslehre*, Bd. I. Berlin, 24. Auflage Aufl.

[Haustein 1999] Haustein, Stefan (1999). *Information Environments for Software Agents*. In: Burgard, Wolfram, T. Christaller und A. B. Cremers, Hrsg.: *KI-99: Advances in Artificial Intelligence*, Bd. 1701 d. Reihe *LNAI*, S. 295 – 298, Bonn, Germany. Springer Verlag.

[Horn und Reinke 2000] Horn, Erika und T. Reinke (2000). *Musterarchitekturen und Entwicklungsmethoden für Multiagentensysteme*. Künstliche Intelligenz Schwerpunktheft: Autonome Mobile Systeme, 4/00:48 – 54.

[Iglesias et al. 1999] Iglesias, Carlos A., M. Garijo und J. C. González (1999). *A Survey of Agent-Oriented Methodologies*. In: Müller, Jörg P., M. P. Singh und A. S. Rao, Hrsg.: *Intelligent Agents V – Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, Bd. 1555 d. Reihe *LNCS*, S. 317 – 330. Springer.

[Jahnke und Biskup 1999] Jahnke, Hermann und D. Biskup (1999). *Planung und Steuerung der Produktion*. Verlag Moderne Industrie, Landasberg/Lech, Germany.

[Jehle et al. 1990] Jehle, Egon, K. Müller und H. Michael (1990). *Produktionswirschaft*, Bd. 4 d. Reihe *Grundstudium Betriebswirtschaftslehre*. Verlag Recht und Wirtschaft, 3. Aufl.

[Jennings et al. 1996] Jennings, N. R., P. Faratin, M. J. Johnson, T. J. Norman, P. O'Brien und M. E. Wiegand (1996). *Agent-based business process management*. International Journal of Cooperative Information Systems, 5(2-3):105 – 130.

[Jennings und Wooldridge 2000] Jennings, Nicholas R. und M. Wooldridge (2000). *Agent-Oriented Software Engineering*. In: Bradshaw, J., Hrsg.: *Handbook of Agent Technology*. AAAI/MIT Press. to appear.

[Klemperer 2000] Klemperer, Paul (2000). *The Economic Theory of Auctions*, Kap. Auction Theory: A Guide to the Literature, S. 3 – 62. Edward Elgar.

[Krothapalli und Deshmukh 1999] Krothapalli, N. K. C. und A. V. Deshmukh (1999). *Design of negotiation protocolls for multi-agent manufacturing*. International Journal of Production Research, 37(7):1601 – 1624.

[Lüscher 1994] Lüscher, M. (1994). *A portable high-quality random number generator for lattice field theory simulations*. Computer Phys. Commun., 79:100 – 110.

[MASIF 1997] MASIF (1997). *The Mobile Agent System Interoperability Facilities Specification*. GMD FOKUS and IBM and Crystaliz Inc. and General Magic Inc. and The Open Group.

[Matsumoto und Nishimura 1998] Matsumoto, M. und T. Nishimura (1998). *Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator*. ACM Trans. on Modeling and Computer Simulation, 8(1):3–30.

[McAfee und McMillan 1987] McAfee, Preston R. und J. McMillan (1987). *Auctions and Bidding*. Journal of Economic Literature, XXV(2):699 – 738.

[McCoy] McCoy, Jim. *Mojonation, a peer-to-peer content distribution technology*. AZI Corporate Contacts, 644 Church St., Mountain View, CA 94041. `http://www.mojonation.net`.

[Meyer 1988] Meyer, Bertrand (1988). *Object-oriented software construction*. Prentice Hall.

[Nwana 1996] Nwana, H. (1996). *Software agents: an overview*. Knowledge Engineering Review, 11(3):205 – 244.

[Omicini und Zambonelli 2000] Omicini, Paolo Ciancarini Andrea und F. Zambonelli (2000). *Multiagent System Engineering: the Coordination Viewpoint*. In: *Intelligent Agents VI*, Nr. 1767 in *LNAI*. Springer Verlag.

[Rao und Georgeff 1995] Rao, Anand S. und M. P. Georgeff (1995). *BDI Agents: From Theory to Practice*. In: *Procs. of the First Int. Conf. on Multi-Agent Systems (ICMAS)*, S. 312–319.

[Russell und Norvig 1995] Russell, Stuart J. und P. Norvig (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, USA.

[Saad et al. 1995] Saad, A., G. Biswas und K. Kawamura (1995). *Performance Evaluation of Contract net-Based heterarchical Scheduling for Flexible Manufacturing Systems*. Journal of Intelligen Automation and Soft Computing.

[Shen et al. 2001] Shen, Weiming, D. H. Norrie und J.-P. H. Barthés (2001). *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*. Taylor and Francis, London, New York.

[Smith und Davis 1981] Smith, R. G. und R. Davis (1981). *Frameworks for Cooperation in Distributed Problem Solving*. IEEE Trans. on Systems, Man and Cybernetics, 11(1):61 – 70.

[Smith 1980] Smith, Reid G. (1980). *The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver*. IEEE Transactions on Computers, C-29(12):1104 – 1113.

[Váncza und Márkus 2000] Váncza, J. und A. Márkus (2000). *An Agent Model for Incentive-based Production Scheduling*. Computers in Industry, 43(2):173 – 187.

[Vickery 1961] Vickery, William (1961). *Counterspeculation, auctions, and competitive sealed tenders*. Journal of Finance, 16:8 – 37.

[Wooldridge et al. 2000] Wooldridge, M., N. R. Jennings und D. Kinny (2000). *The Gaia Methodology for Agent-Oriented Analysis and Design*. Journal of Autonomous Agents and Multi-Agent Systems, 3(3):285 – 312.

[Wooldridge 1997] Wooldridge, Michael (1997). *Agents as a Rorschach Test: A Response to Franklin and Graesser*. In: Müller, Jörg P, M. J. Wooldridge und N. R. Jennings, Hrsg.: *Intelligent Agents III*, S. 47 – 48, Berlin, Heidelberg, New York. Springer.

[Wooldridge und Jennings 1995] Wooldridge, Michael J. und N. R. Jennings (1995). *Intelligent Agents: Theory and Practice*. Knowledge Engineering Review, 10(2):115–152.

[Wooldridge und Jennings 1998] Wooldridge, Micheal und N. Jennings (1998). *Pitfalls of Agent-Oriented Development*. In: Sycara, K. P. und M. Wooldridge, Hrsg.: *Agents '98: Proceedings of the Second International Conference on Autonomous Agents*. ACM press.

[Zelewski 1997] Zelewski, S. (1997). *Elektronische Märkte zur Prozeßkoordinierung in Produktionsnetzwerken*. Wirtschaftsinformatik, (39):231 – 243.

# Index