Naschmarkt - Preisbildung durch bilaterale Verhandlungen lernender Softwareagenten

Kennzahl J151
Matrikel-Nr.: 9250011

Diplomarbeit

Eingereicht von
Bernhard E. Beran

am Institut für
Informationsverarbeitung und Informationswirtschaft,
Abteilung für Angewandte Informatik
insbesondere Betriebsinformatik

an der WIRTSCHAFTSUNIVERSITÄT WIEN
Studienrichtung: Betriebswirtschaft

Begutachter: Univ.Prof.Dkfm.Dr. Wolfgang H. Janko
Betreuender Assistent: Dr. Michael Hahsler

Wien, Freitag, 13. Juli 2001

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, daß ich diese Diplomarbeit selbstständig verfaßt, keine anderen als die angegebenen Hilfsmittel und Quellen verwendet, mich auch sonst keiner unerlaubten Hilfsmittel bedient und diese Diplomarbeit weder im In- noch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Bernhard E. Beran Freitag, 13. Juli 2001

Inhaltsverzeichnis

Zusammenfassung

Diese Arbeit beschäftigt sich mit automatisierten Verhandlungen, also der Übertragung der Aufgabe, sich über die Aufteilung von Ressourcen einig zu werden, an den Computer. Basierend auf einem von Osborne und Rubinstein [Osborne and Rubinstein, 1990] entworfenen Marktmodell wird gezeigt, welche Faktoren auf die Preisbildung Einfluss nehmen. Damit soll gezeigt werden, wie mit automatisierten Verhandlungen Allokationsprobleme auf künstlichen Märkten kosteneffizient gelöst werden können.

Um die Grundlagen der Preisbildung aufzuarbeiten, werden die Marktmodelle der vollkommenen und der monopolistischen Konkurrenz kurz erläutert, um darauf aufbauend die Transaktionskostentheorie darzustellen. Mit ihrer Hilfe kann das Entstehen und die Größe von Verhandlungsspielräumen erklärt werden.

Wie diese Verhandlungsspielräume aufgeteilt werden, erklärt die Spieltheorie, wobei das sequentielle Gleichgewicht als Lösungskonzept verwendet wird.

Das von Osborne und Rubinstein aufgestellte Marktmodell wird konkretisiert, um daraus eine Simulation zu entwickeln, in dem Transaktionskosten, die Risikoeinstellung der Händler und deren Geduld Einfluss auf den Marktpreis nehmen. Dabei verhandeln nutzenorientierte, lernende Softwareagenten bei gegebenen Transaktionskosten um den Preis eines homogenen Gutes.

Kapitel 1

Einleitung

Internet und World Wide Web werden ein immer mehr an Bedeutung gewinnender Kanal für die Abwicklung von wirtschaftlichen Transaktionen. Anderson Consulting (`http://www.ac.com`), ein führendes Beratungsunternehmen, meldet für das Jahr 2000 einen Umsatz von 167,1 Milliarden U.S. Dollar für Business-to-Business eCommerce und 50,7 Milliarden U.S. Dollar für Business-to-Consumer eCommerce. Bereits für 2001 wird im Business-to-Business Bereich ein Umsatz von 320 Mrd. U.S.Dollar prognostiziert, der sich 2002 verdoppeln sollte.

Trotz dieser Zahlen und der großen technischen Möglichkeiten haben sich die Geschäftspraktiken noch kaum den neuen Medien angepasst. Der elektronische Handel ist noch immer weitgehend nicht automatisiert. Obwohl Produkt- und Preisinformationen über das Internet immer leichter zugänglich werden und der Schrift- und Zahlungsverkehr über elektronische Medien abgewickelt wird, sind noch immer Menschen in allen Phasen des Kaufprozess integriert.

Moderne Technologien, wie intelligente Softwareagenten, könnten einen Beitrag zur Umgestaltung von wirtschaftlichen Abläufen darstellen. Softwareagenten sind Programme, die im Auftrag des Anwenders von ihm vorgegebene Aufgaben selbständig ausführen [Maes *et al.*, 1999].

In dieser Arbeit wird das Konzept der intelligenten Softwareagenten auf das Verhandlungsproblem angewandt. Wenn ein Verkäufer eine höhere Preisvorstellung als der Käufer hat, so müssen sich beide im Verhandlungsprozess auf einen von beiden akzeptierten Preis einigen. Wie sich dieser Preis bilden könnte, wenn die Verhandlungspartner keine Menschen sondern Softwareprogramme sind, soll im Folgenden beleuchtet werden.

1.1 Motivation und Beitrag

Verhandlungen, wie sie auf Märkten statt finden, werden schon lange von der Mikroökonomie und der Spieltheorie untersucht. Mikroökonomie und Spieltheorie stellen also ein wichtiges Werkzeug für das Design von Multi-Agenten-Systemen dar. Diese Disziplinen ermöglichen eine gezieltere Herausarbeitung der Probleme, die in Zusammenhang mit der Auswahl von Verhandlungsmechanismen und Strategien verbunden sind. Speziell die Spieltheorie bietet Methoden, mit denen Verhandlungsprobleme klassifiziert und Anleitungen für optimale Handlungen gegeben werden können [Binmore and Vulkan, 1997].

Obwohl das wirtschaftliche Potential des Einsatzes von intelligenten Softwareagenten noch nicht vollkommen abgeschätzt werden kann, kristallisieren sich die Vorteile dieser Technologie heraus. Für die Anwender ist eine Verbesserung der Effizienz und Effektivität zu erwarten. Agenten sind in der Lage, Aufgaben rascher durchzuführen. Auch im Vergleich mit erfahrenen Usern erzielen Agenten oft bessere Ergebnisse. Menschen begnügen sich oft mit der erstbesten Lösung. Speziell bei der Informationsflut im Internet werden bessere Alternativen oft nicht betrachtet, weil der Aufwand für die Suche zu hoch ist. Agenten können diese Probleme lösen, sie erhöhen die Transparenz von Märkten, weil sie Informationen aus mehr Quellen vergleichen können [Brenner *et al.*, 1998, S. 15f].

BargainFinder, um ein Beispiel zu nennen, ermöglicht es, zehn Händlerseiten nach dem günstigsten Angebot für eine CD zu durchsuchen. Der Anwender spezifiziert Interpret und Titel eines Albums. Als Antwort erhält der Anwender eine Liste mit Händlern, die das gewünschte Album führen. So können Preise rasch verglichen werden, für detaillierte Informationen über Lieferkonditionen bietet BargainFinder Verweise auf die Händler-Homepages an, von denen das Produkt bezogen werden kann [Brenner *et al.*, 1998, S. 270f].

Bei der Betrachtung bereits existierender Marktplätze und experimenteller Systeme (Kapitel 2) zeigt sich, dass automatisierte Verhandlungen in der Praxis noch nicht eingesetzt werden. Arbeiten über automatisierte Verhandlungen berücksichtigen noch kaum volkswirtschaftliche und spieltheoretische Erkenntnisse und bauen oft auf intuitiven Annahmen über menschliches Verhalten auf.

Hier wird der Problembereich sowohl aus volkswirtschaftlicher als auch aus spieltheoretischer Sicht betrachtet. Zum einen wird geklärt, warum ein Verhandlungsspielraum entsteht und welche Faktoren auf dessen Größe Einfluss nehmen. Hier steht der Beitrag der Volkswirtschaftslehre im Vordergrund. Andererseits wird gezeigt, welche Variablen für die Aufteilung dieses Spielraums relevant sind, dabei werden spieltheoretische Erkenntnisse verarbeitet.

Durch die Kombination von Volkswirtschaftslehre und Spieltheorie soll in dieser Arbeit versucht werden, ein geschlossenes Modell zu entwickeln.

Aus einer praxisorientierten Sichtweise betrachtet geht diese Arbeit von zwei

Fragestellungen aus:

- Welche einfache und für den Anwender noch nachvollziehbare Methode kann entwickelt werden, um einen Softwareagenten so zu parametrisieren, dass er noch im Sinne seines Auftraggebers handelt?
- Kann ein Softwareagent so programmiert werden, dass seine Handlungen noch als plausibel beurteilt werden können, wenn man sie mit menschlichem Handeln vergleicht?

1.2 Aufbau der Arbeit

Diese Arbeit gibt in Kapitel 2 einen Überblick über technische Möglichkeiten des Einsatzes von Softwareagenten in einem typischen Beschaffungsprozess und führt in die Problemstellung der automatisierten Verhandlungen ein. Dabei werden bestehende Lösungsansätze in systematischer Gliederung beschrieben und Probleme aufgezeigt. Somit können die Ansätze, an denen sich diese Arbeit orientiert, hervorgehoben werden und mit anderen Lösungswegen verglichen werden.

In Kapitel 3 werden Märkte aus mikroökonomischer Sicht betrachtet, um die Grundlagen für ein Marktmodell zu entwickeln und zu zeigen, warum Verhandlungen zur Preisbestimmung notwendig sind. Dabei wird vor allem auf die Transaktionskostentheorie zurückgegriffen. Sie versucht zu klären, warum Verhandlungsspielräume bei der Preisbildung bestehen und welche Faktoren auf diese Spielräume Einfluss nehmen. Das Modell der vollkommenen Konkurrenz liefert dabei die Grundlage für einen vorherrschenden Marktpreis, der durch Transaktionskosten verzerrt wird.

Um zu erklären, wie sich Verhandlungspartner auf einen Preis einigen, wird in Kapitel 4 die Spieltheorie herangezogen. Spieltheoretische Lösungskonzepte zeigen einen geeigneten Weg auf, Verhandlungsstrategien für Softwareagenten zu entwickeln. Zunächst werden das Verhandlungsspiel erklärt und die Spielregeln dargestellt um im Anschluss daran ein Lösungskonzept für dieses Spiel zu präsentieren. Mit diesem Lösungskonzept, dem sequentiellen Gleichgewicht, wird gezeigt, welche Strategien Verhandlungspartner anwenden, um sich auf einen Preis zu einigen.

Da für die Beurteilung des Verhandlungsausgangs eine Nutzenfunktion modelliert werden muss, werden die Prinzipien der Nutzentheorie nach von Neumann - Morgenstern ebenfalls in Kapitel 4 dargestellt.

Nachdem die theoretischen Grundlagen sowohl aus volkswirtschaftlicher als auch spieltheoretischer Sicht aufgearbeitet wurden, wird in Kapitel 5 daraus ein geschlossenes Simulationsmodell entwickelt und dokumentiert. Dabei wird vom Marktmodell mit asymmetrischer Informationsverteilung

und dem Risiko des Verhandlungsabbruchs nach Osborne und Rubinstein [Osborne and Rubinstein, 1990] ausgegangen. Die Größe des Verhandlungsspielraums ist von als gegeben angenommenen Transaktionskosten und der Risikoeinstellung der Händler abhängig. Die Risikoeinstellung der Marktteilnehmer und ihre Zeitpräferenzen werden anhand eines Preislimits und eines Zeitlimits bestimmt.

Mit der Simulation soll gezeigt werden, dass das spieltheoretische Modell nach Osborne und Rubinstein herangezogen werden kann, um einen Mechanismus zu entwickeln, der die entgegengesetzten Interessen von Käufern und Verkäufern effizient ausgleicht. Die Effizienz des Mechanismus kann anhand der Kosten, die er verursacht, beurteilt werden.

Im Vordergrund steht die Entwicklung eines lernenden Softwareagenten, der Verhandlungsprobleme selbständig lösen kann. Aus mehreren dieser Agenten wird dann ein künstlicher Markt für die Simulation aufgebaut.

Die Ergebnisse der Simulation werden in Kapitel 6 dargestellt. Die Resultate werden dabei mit den Schlussfolgerungen verglichen, die aus den Kapiteln 3 und 4 hervorgegangen sind. Es wird gezeigt, wie sich unterschiedliche Transaktionskosten auf die Verhandlungsstrategie und die erreichten Verhandlungslösungen auswirken. Weiters zeigt sich, dass der Interessensausgleich zwischen Käufern und Verkäufern nach einer "Lernphase"geringere zeitabhängige Verhandlungskosten verursacht, weil rascher eine Einigung erzielt werden kann. Trotzdem können die Agenten Vorteile, die in ihren Präferenzen hinsichtlich Verhandlungszeit und Preis begründet liegen, immer noch ausnutzen.

In der abschließenden Zusammenfassung im Kapitel 7 wird auf die Probleme des verwendeten Lösungsansatzes eingegangen. Die restriktiven Annahmen der Spieltheorie und deren mathematische Komplexität stehen dabei im Mittelpunkt der Überlegungen.

In Hinblick auf die Einsatzmöglichkeiten zeigt sich derzeit, dass es bis zur Praxisreife automatisierter Verhandlungen noch ein langer Weg ist. Vorallem die Komplexität der Spieltheorie und deren restriktive Modellannahmen machen den praktischen Einsatz zur Zeit noch nicht möglich.

Kapitel 2

Automatisierte Verhandlungen

Eine Verhandlung ist ein Prozess, bei dem zwei oder mehrere Parteien multilateral über die Aufteilung von Ressourcen handeln, in der Absicht gegenseitig Vorteile zu erzielen. Eine Verhandlung ist dann automatisiert, wenn die Verhandlungsfunktion selbst von Computern ausgeführt wird [Beam and Segev, 1997, S. 3].

Zwischen Menschen stellen Verhandlungen extrem komplexe Prozesse dar. Die Modellierung solcher Abläufe mit dem Ziel, diese Aufgabe zu automatisieren, bringt durch die im Modell notwendige Abstraktion und Vereinfachung viele Probleme mit sich.

Der Abschnitt 2.1 dieses Kapitels gliedert die Einsatzmöglichkeiten von Softwareagenten im Beschaffungsprozess. Abschnitt 2.2 systematisiert die verschiedenen Lösungsansätze, die für Verhandlungsprobleme verwendet werden können. Danach stellt Abschnitt 2.3 das Konzept der Softwareagenten vor.

Welche Systeme zur Lösung und Unterstützung von Verhandlungen heute existieren, beschreibt Abschnitt 2.4.

Das Kapitel schließt mit einem Überblick über die Problemfelder, die automatisierte Verhandlungen mit sich bringen.

2.1 Agenteneinsatz im Electronic Commerce

In einem Beschaffungsprozess kann die Agenten-Technologie in allen Phasen angewandt werden. So könnte der Papierverbrauch eines Unternehmens von einem Softwareagenten überwacht werden, der anhand der Verbrauchsmuster rechtzeitig Kauf-Agenten beauftragt, potentielle Lieferanten zu kontaktieren. Die Agenten sammeln nun Informationen über Produkte, Preise, Liefer- und Zahlungskonditionen und wählen einen Lieferanten aus. Sie führen selbständig Verhandlungen mit Anbietern durch, schließen letztendlich einen Vertrag ab und sorgen für die Abwicklung [Maes *et al.*, 1999].

Systematisch lassen sich sechs Phasen im Beschaffungsprozess herausarbeiten [Maes *et al.*, 1999]:

1. Identifikation des Bedarfs
   Am Beginn des Beschaffungsvorgangs muss der Käufer sich dem Bedürfnis nach einem bestimmten Produkt bewusst werden. So bietet das Internet-Auktionshaus atrada.de (`http://www.atrada.de`) einen Angebotsagenten, der User über ihren Interessen entsprechende Angebote informiert.

2. Produktsuche
   Nachdem der Bedarf erkannt wurde, muss ein Produkt gefunden werden, dass diesen Bedarf decken kann.

   PersonaLogic (`http://www.personalogic.com`) hilft Konsumenten, ihre Wünsche hinsichtlich verschiedener Güter zu spezifizieren. Dabei wird der Anwender durch einen Fragebogen geleitet, der durch Beurteilung von Produkteigenschaften durch den User geeignete Güter auswählt.

   Diese System verwenden meist kollaborative Filtertechniken [Goldberg *et al.*, 1992]. Dabei werden Produktbeurteilungen mit denen von anderen Anwendern verglichen. Konsumenten mit ähnlichen Beurteilungsprofilen erhalten Produktempfehlungen voneinander.

3. Händlersuche
   Bei der Händlersuche unterstützen Systeme wie BargainFinder von Anderson Consulting oder Jango (`http://jango.excite.com`) den Konsumenten. BargainFinder ersucht um Preisauskünfte bei verschiedenen Händlersites. Ähnlich arbeitet Jango, allerdings werden hier die Anfragen vom Webbrowser des Users aus gestartet und nicht von einer zentralen Seite. Damit kann das Problem, dass viele Händler solche Anfragen von Dienstleistern abblocken, umgangen werden.

4. Verhandlungsführung
   In dieser Phase werden die Vertragskonditionen bestimmt. Während im Konsumentengeschäft bisher Preise durch die Anbieter festgesetzt werden sind in Business-to-Business Geschäften Preisfestsetzungen durch Verhandlungen in der Überzahl. Durch das Internet zeigt sich hier mit dem Erfolg der Online-Auktion jedoch auch ein Trend hin zur Preisfestsetzung durch Verhandlungen bei Konsumentengeschäften.

   Einen Überblick über den Stand der Verhandlungsunterstützung durch verschiedene Systeme gibt Abschnitt 2.4.

5. Lieferung und Zahlung
   In vielen Fällen erfolgt die Zahlung noch auf herkömmlichen Weg per

Nachnahme. Mit der Verbreitung von sicheren elektronischen Zahlungssystemen ist aber auch hier mit einer vermehrten Abwicklung über das Internet zu rechnen.

SET - Secure Electronic Transfer (`http://www.mastercard.at`) etwa ermöglicht das Versenden von Kreditkarteninformationen ohne das der Händler diese einsehen kann. Für den Händler besteht der Vorteil dieses Systems darin, dass er von der Abwicklungsstelle Auskunft über die Bonität des Kunden erhält. Sofern es sich bei der Transaktion nicht um ein Informationsgut handelt sind bei der Auslieferung herkömmliche Versandvarianten unumgänglich.

6. Service
   Produktservice und Konsumentenbetreuung findet heute schon vermehrt über das Internet statt. So sind für diverse Hardwareprodukte geeignete Treiber kostenlos über das Internet erhältlich, viele Unternehmen bieten auf ihren Homepages Service- und Wartungsanleitungen an.

Ziel aller Bemühungen, Käufern und Verkäufern funktionsfähige Unterstützungssysteme zur Verfügung zu stellen, ist, letztendlich die Vereinfachung von wirtschaftlichen Transaktionen und die damit verbundene Reduktion an Transaktionskosten [Maes *et al.*, 1999].

Die oben dargestellten Phasen des Wirtschaftsprozesses passen sich immer mehr den Möglichkeiten, die durch Internet und WWW entstanden sind, an. Die Unterstützung in der eigentlichen Verhandlungsphase ist heute aber noch unzureichend. Gegenwärtige eCommerce-Werkzeuge und Technologien sind noch nicht in der Lage, die komplexe Aufgabe von Geschäftsverhandlungen zu übernehmen [Beam and Segev, 1996].

Da jedoch Verhandlungen im Wirtschaftsleben von großer Bedeutung sind, sollte die Forschung in dieser Richtung forciert werden.

2.2 Ansätze zur Problemlösung

Das Problemfeld der automatisierten Verhandlungsführung kann anhand zweier Dimensionen analysiert werden. Verhandlungsproblem können nach dem Ziel der Verhandlung in kooperative und nicht kooperative Probleme gegliedert werden. Die zweite Dimension ist durch den Mechanismus, der die Verhandlungsstrategie erzeugt, bestimmt [Beam and Segev, 1996].

2.2.1 Kooperative und nicht kooperative Verhandlungen

Das Ziel, das durch eine Verhandlung erreicht werden soll, kann für die Verhandlungspartner ident sein. In diesem Falle spricht man von kooperativen Verhandlungsproblemen.

Typische kooperative Probleme entstehen bei der Entwicklung einer Softwareapplikation im Team. Jedes Mitglied hat das gleiche Ziel, die Entwicklung einer guten Lösung, jedoch muss über die Aufteilung der einzelnen Aufgaben verhandelt werden. Weil alle Parteien an einem gemeinsamen Ziel arbeiten ist die Preisgabe von Informationen, wie z.B. Kostenangaben oder Fertigstellungstermine, im gemeinsamen Interesse, jede Partei hat dadurch den größtmöglichen Freiheitsgrad in Verhandlungen über einen Zeitplan.

Preisverhandlungen stellen hingegen nicht kooperative Verhandlungsprobleme dar. Während der Käufer einen möglichst niederen Preis erzielen will, ist der Verkäufer bestrebt, genau das Gegenteil zu erreichen. Die Ziele sind verschieden.

Nimmt eine Verhandlungspartei irrtümlich an, in einer kooperativen Verhandlungssituation zu sein, so besteht große Gefahr, dass durch diese Fehleinschätzung kostspielige Fehler entstehen. Der Grund dafür liegt darin, dass die Preisgabe von Informationen an die Verhandlungspartner in kooperativen Verhandlungssituationen von Vorteil ist, sich bei nicht kooperativen Problemen allerdings nachteilig auswirkt. Gibt der Käufer bekannt, welchen Betrag er maximal bereit wäre zu zahlen, so kann der Verkäufer auf diesem Preis beharren und dem Käufer kann kein Vorteil aus einer Verhandlung entstehen.

Preisverhandlungen sollten deswegen immer als nicht kooperative Verhandlungen modelliert werden [Beam and Segev, 1996].

Die Einteilung in kooperative und nicht kooperative Verhandlungsproblem darf nicht mit der Einteilung in kooperative und nicht kooperative Spieltheorie verwechselt werden.

2.2.2 Modelle des Verhandlungsmechanismus

Nach Beam und Segev kann der Entwurf von Verhandlungsalgorithmen in drei unterschiedliche Ansätze aufgespaltet werden [Beam and Segev, 1996]:

„Human Factors Approach": Diese Schule versucht, Auswirkungen von menschlichen Faktoren wie Stolz, Egoismus und kulturelle Einflüsse zu untersuchen. Die Bedeutung dieses Ansatzes ist eher für das Verständnis von Anwenderpräferenzen und menschlichen Verhandlungen von Bedeutung, nicht jedoch in Hinblick auf automatisierte Verhandlungen. Electronic Commerce verdrängt größtenteils den menschlichen Faktor und versucht, sich auf die reinen Wirtschafts- und Verhandlungsprozesse zu konsentrieren.

Ökonomischer und spieltheoretischer Ansatz: Ökonomie, Spieltheorie und Verhandlungstheorie sehen das Problem von einer mehr mathematischen Seite. Der ökonomische Zugang zielt auf die Gestaltung von optimalen Preisfestsetzungsmechanismen ab. Die Spieltheorie versucht, Handlungsrichtlinien in Verhandlungen zu geben, um ein vorteilhaftes Ergebnis zu erzielen. Dabei nimmt sie Rücksicht auf die Strategie des Verhandlungspartners. Der wesentliche Nachteil des spieltheoretischen Zugangs ist die Annahme des rationalen Handelns, die in der Realität oft nicht erfüllt ist.

Computerwissenschaftlicher Ansatz: Künstliche Intelligenz und intelligente Softwareagenten stehen im Mittelpunkt dieser Denkrichtung. Der Ansatz ist meist eher ein experimenteller, bei typischen Forschungsprojekten werden meist verschiedene Agenten programmiert, die im Wettbewerb gegeneinander antreten.

Der computerwissenschaftliche Zugang bietet den großen Vorteil, dass menschliche Faktoren einen geringen Einfluss nehmen und die Annahme der Superrationalität deswegen nicht so unrealistisch scheint. Weiters können Softwareagenten technisch einfacher in bestehende eCommerce Systeme eingegliedert werden.

Diese Arbeit versucht, nicht kooperative Verhandlungsprobleme aus Sicht der Spieltheorie zu betrachten und daraus Ansätze für Verhandlungsstrategien zu entwickeln. Dabei wird das Modell des bilateralen Verhandlungsspiels mit dem Risiko des Verhandlungsabbruchs nach Osborne und Rubinstein verwendet [Osborne and Rubinstein, 1990]. Die für dieses Spiel maßgebliche Lösungsstrategie bildet dann die Grundlage für einen Algorithmus, nach dem die Softwareagenten handeln. So wird versucht, einen Bogen von der Mikroökonomie und Spieltheorie zum computerwissenschaftlichen Ansatz zu spannen.

2.3 Softwareagenten

Softwareagenten sind von Menschen delegierte Einheiten, die eine Aufgabe für ihren Meister erledigen. Agenten sollen dem Anwender Arbeit abnehmen und diese autonom durchführen [Cheong, 1996, S. 9]. Autonom bedeutet, dass der Agent nicht unter unmittelbarer Kontrolle des Anwenders agiert, sein Verhalten wird durch seine eigene Erfahrung bestimmt.

Ein Agent nimmt seine Umwelt durch Sensoren wahr und reagiert durch seine eigenen Handlungen auf diese Umwelt. Dabei sollen seine Aktionen auf ein bestimmtes Ziel hin ausgerichtet sein [Russell and Norvig, 1995, S. 35].

Russell und Norvig [Russell and Norvig, 1995, S. 40ff] unterscheiden vier verschiedene Typen von Softwareagenten:

Einfache Reflex-Agenten: Einfache Reflex-Agenten handeln nach fest vorgegebenen Regeln. Die Wahrnehmungen werden mit den in den Regeln beschriebenen Zuständen verglichen, bei Übereinstimmung wird die entsprechende Handlung gesetzt.

Agenten, die ihre Umwelt beobachten: Im Gegensatz zu den nach Reflexen handelnden Agenten verfügt diese Form über ein Modell wie sich die Umwelt entwickelt und welche eigenen Aktionen die Umwelt beeinflussen. Durch dieses Modell ist der Agent nun in der Lage, seine Umwelt aktiv zu beobachten, d.h., er kann unterschiedliche Umweltzustände hinsichtlich ihrer Auswirkungen unterscheiden, auch wenn diese Zustände die gleichen Sensorreize produzieren.

Zielgerichtete Agenten: Zielgerichtete Agenten müssen, bevor sie Handlungen setzen, die Auswirkungen dieser Handlungen auf die Umwelt untersuchen. Der potentiell veränderte Umweltzustand wird dann mit den Zielen des Agenten verglichen, so können Aktionen ausgewählt werden, die in Hinblick auf die gestellte Aufgabe zielgerichtet sind.

Nutzenorientierte Agenten: Nutzenorientierte Agenten beurteilen die Auswirkung ihrer Handlung nicht bezogen auf einen bestimmten Wunschzustand, sondern versuchen, eine Nutzenfunktion zu maximieren. Die Nutzenfunktion bildet verschiedene Umweltzustände auf einer Skala ab, die Präferenzen hinsichtlich verschiedener Zustände beschreibt. So wird der Agent in die Lage versetzt, auch Zielkonflikte zwischen mehreren wünschenswerten Zuständen zu beurteilen.

Die in dieser Arbeit entwickelten Agenten beobachten ihre Umwelt, indem sie Reaktionen der Verhandlungspartner verarbeiten. Die Naschmarkt-Agenten sind nutzenorientiert. Die Agenten versuchen, den Nutzen, den sie aus einem verhandelten Preis erzielen, zu maximieren. Zeitabhängige Verhandlungskosten reduzieren dabei den Nutzen. Ein Agent muss daher versuchen, ein Optimum aus dem Nutzen des Preises abzüglich der Nutzeneinbußen durch die Verhandlungsdauer zu erreichen.

2.4 Verhandlungsunterstützung und Automatisation

In den letzten Jahren haben sich drei Ansätze zur Unterstützung bzw. Automatisierung von Verhandlungen herauskristallisiert, Negotiation Support Systems,

kurz NSS, Auktionen und Distributet Artificial Intelligence (DAI). Diese Ansätze werden kurz beschrieben.

2.4.1 Negotiation Support Systems

Negotiation Support Systems (NSS) sind Programme, die den menschlichen Entscheidungsprozess während Verhandlungen unterstützen und so bessere und produktivere Entscheidungen erlauben. Sie stellen eine Untergruppe von Entscheidungsunterstützungssystemen dar. NSS können keine automatisierten Verhandlungen durchführen, können aber als ein Schritt in diese Richtung betrachtet werden [Beam and Segev, 1997, S. 5].

Als Beispiel eines NSS wird kurz das InterNeg Projekt mit dem Negotiation Support System INSPIRE der Carleton University & Concordia University Ottawa-Montreal, Canada vorgestellt (`http://interneg.carleton.ca`) [Kersten and Noronha, 1999]. INSPIRE ist ein Web-basiertes System, somit werden Verhandlungen auch über große geographische Distanzen möglich.
Unterstützt werden drei Verhandlungsphasen, die Vorbereitung, die Verhandlung im eigentlichen Sinne und die Nachbearbeitung.

Verhandlungsvorbereitung: Die Verhandlungsvorbereitung befasst sich mit der Spezifikation des Verhandlungsproblems. Ziel ist es, eine Nutzenfunktion zu entwickeln, die den Nutzen unterschiedlicher Verhandlungsergebnisse bewertet. Der Anwender legt individuell die einzelnen verhandelbaren Punkte fest und bewertet deren mögliche Ausprägungen. Da wegen der großen Anzahl an Kombinationen unterschiedlicher Ausprägungen nicht alle Varianten erfasst werden können, wird eine Nutzenfunktion anhand einer Auswahl vom Benutzer bewerteten Alternativen errechnet.

Verhandlungsphase: Während der eigentlichen Verhandlung hilft das NSS, Angebote zu erstellen und zu übermitteln, sowie Gegenangebote zu bewerten. Bei der Erzeugung von Angeboten wird der Nutzen des Angebots berechnet, um den Verhandler eine Orientierungshilfe zu bieten. Der Nutzenverlauf von Angeboten und Gegengeboten kann zur besseren Übersicht graphisch dargestellt werden. Auch können in dieser Phase die Parameter der Nutzenfunktion geändert werden.
Zusätzlich zu den Angeboten können die Verhandlungspartner Textnachrichten übermitteln, um mit verbalen Argumenten ein günstiges Verhandlungsklima zu schaffen.

Nachbearbeitung: Sollten sich die Parteien auf ein Ergebnis geeignet haben, so übernimmt das NSS die Rolle eines Mediators. Das Ergebnis wird auf seine Pareto-Effizienz hin untersucht. Sollte sich ein Verhandlungspartner jetzt

noch besser stellen können, ohne den anderen zu benachteiligen, schlägt das System verschieden alternative Verhandlungslösungen vor, um einen paretooptimalen Zustand herzustellen.

NSS stellen relativ hohe Anforderungen an den Benutzer. Der Anwender muss sowohl das Problem formulieren als auch das System ständig mit neuen, aktuellen Daten versorgen. Die Entscheidung über den Verhandlungsausgang liegt ebenso beim Anwender [Beam *et al.*, 1997, S. 5].

2.4.2 Auktionen

Eine Auktion stellt einen Mechanismus zur Preisfestsetzung dar. Dabei können meist mehrere Bieter innerhalb einer Frist Angebote abgeben. Am Ende dieser Frist werden Bieter und Verkäufer entsprechend den Angeboten zusammengeführt und so der Markt geräumt [Bichler *et al.*, 1999, S. 2].

Die Auktionsregeln lassen sich in vier Grundtypen einteilen: English, Dutch, First-price sealed-bid und Second-price sealed-bid Auctions, auch als Vickery Auktion bekannt [Lucking-Reiley, 1999].

English Auction: Diese Variante stellt die wohl bekannteste Auktionsregel dar. Bei der English Auction machen die Bieter sich gegenseitig übersteigende Angebote. Wenn kein höheres Angebot mehr gestellt wird, erhält der Meistbieter den Zuschlag zum von ihm gebotenen Preis.

Dutch Auction: Im Gegensatz zur English Auction arbeitet die Dutch Auction mit fallenden Preisen. Eine öffentlich sichtbare „Preisuhrßeigt einen kontinuierlich fallenden Preis. Die Auktion endet, wenn der erste Bieter den Preis nieder genug für ein Gebot hält. Er erhält den Zuschlag zum von der Uhr angezeigten Preis.

First-price sealed-bid Auction: Stellen die English und die Dutch Auction Verfahren dar, die in Echtzeit ablaufen, so sind die Sealed-bid Mechanismen nicht an einen Echtzeitprozess gebunden. Beim First-price sealed-bid Verfahren geben die Bieter ihre Angebote verschlossen ab. Nach dem Ende der Angebotsfrist werden die Gebote gesichtet, der Bieter mit dem höchsten Preis erhält den Zuschlag zum von ihm gebotenen Preis.

Second-price sealed-bid Auction: Dieser Auktionsmechanismus unterscheidet sich nur gering von der First-price sealed-bid Auction. Der Unterschied besteht darin, dass der Meistbieter den Zuschlag zum Preis des nächstbesten Gebots erhält.

> Die Second-price sealed-bid Auction liefert die gleichen Erlöse wie die English Auction, wenn man annimmt, dass der Meistbieter bei einer English Auction nur einen geringfügigen Zuschlag auf das zweitbeste Gebot zahlen muss.
>
> Die Bieter müssen im Falle der Second-price sealed-bid Auction aber nicht über die Angebote der Mitbieter informiert sein. Der Vorteil dieses Verfahrens besteht demnach im wirtschaftlichen Einsatz von Information, die ein Bieter zur Erstellung eines optimalen Gebots benötigt. Es genügt, dass ein Bieter nur seine eigenen Preisvorstellungen kennt, nicht jedoch die der Anderen. Deswegen müssen sich die an der Auktion teilnehmenden weder zeitlich noch örtlich verabreden. [Riley and Samuelson, 1981].

Beam und Segev bieten anhand einer Analyse von 100 Internetauktionshäusern einen Überblick über die aktuelle Entwicklung. Die Ergebnisse dieser Untersuchung sein im folgenden kurz dargestellt [Beam and Segev, 1998]:

Mehr als die Hälfte der beobachteten Auktionsplätze bieten Auktionen von Business-to-Consumer an, ein viertel der Sites bieten Consumer-to-Consumer Auktionen. Business-to-Business Auktionen stehen mit 6% an der letzten Stelle. Als Auktionsgüter kommen meist einfach zu versendende, materielle Güter in Frage, gefolgt von immateriellen Gütern wie Software. Dabei werden mehrheitlich neue und gebrauchte Konsumgüter versteigert, etwa zwei Drittel der Unternehmen haben sich auf bestimmte Produktgruppen spezialisiert.

Fast drei Viertel der Unternehmen haben ausschließlich Internetauktionen als Hauptgeschäftsfeld, 13% setzen sich aus klassischen Auktionshäusern zusammen, die ihre Dienste auch über das Internet anbieten.

Obwohl die technischen Möglichkeiten komplexere Auktionsregeln erlauben würden, werden von nahezu allen Auktionshäusern einfache Regeln angewandt. Der Auktionsschluss wird meist durch ein Zeitlimit bestimmt. Der Zuschlag erfolgt meist in absteigender Reihenfolge der Preise, wenn mehrere Stück eines Gutes angeboten werden. Als häufigster Auktionsmechanismus wird die English-Auction angewandt (85%), dies ist auf die Einfachheit und die allgemeine Bekanntheit des Verfahrens zurückzuführen.

16 der untersuchten English-Auctions erlauben es den Teilnehmern, untereinander Kontakt aufzunehmen. Dies ist insofern überraschend, weil nach der Auktionstheorie gerade Absprachen unter den Bietern ein für den Auktionator schlechteres Ergebnis erwarten lassen. Beam und Segev führen dazu an, dass Auktionshäuser diese Möglichkeiten deshalb bieten, um eine „Communityäufzubauen und so Kunden zu binden.

Ebenfalls bemerkenswert ist der Umstand, dass nur knapp mehr als ein Viertel der Anbieter Verschlüsselungs- und Sicherheitstechnologien bieten, oft auch dann nicht, wenn per Kreditkarte gezahlt werden kann.

Da Auktionsregeln klar definiert und meist auch relativ einfach gestaltet sind, können an der Auktion Teilnehmende ihre Strategien im vorhinein als Softwareagenten programmieren [Beam and Segev, 1997, S. 9]. Diese Einfachheit ist auch der Grund dafür, dass Auktionshäuser wie eBay oder Onsale sich im Internet erfolgreich behaupten können [Bichler *et al.*, 1999, S. 2]. Ein zweiter Erfolgsfaktor scheint die Freude der Anwender zu sein, an einer Internetauktion teilzunehmen. So bietet etwa Onsale die Möglichkeit, Gebote mit Kommentaren zu versehen, nicht zuletzt deswegen an, weil Mitteilungen wie „Do not outbid me or I´ll be back!"den Unterhaltungswert einer Auktion erhöhen [Beam and Segev, 1998, S. 10].

Als großer Nachteil der Auktion erweist sich, dass lediglich um den Preis verhandelt werden kann und andere Attribute wie Qualität, Liefer- und Zahlungskonditionen nicht oder nur schwer verhandelbar sind [Beam and Segev, 1997, S. 2].

2.4.3 Distributed Artificial Intelligence

Der DAI-Ansatz befasst sich mit der Erstellung von Softwareagenten, die Verhandlungen nach vom Anwender spezifizierten Einschränkungen autonom durchführen. Die Entwicklung zeigt zwei Hauptrichtungen, Agenten mit ausschließlich vorprogrammierten Strategien und solche, die während des Verhandlungsprozesses lernen [Beam and Segev, 1997, S. 7].

Nicht lernende Agenten

Nicht lernende Agenten benötigen eine große Anzahl vordefinierter Strategien, um auf jede mögliche Verhandlungssituation reagieren zu können. Die dabei verwendeten Strategien sind meist einfach. Sie folgen einen Gebot-Gegengebot-Zyklus, wobei vom Benutzer festgelegte Nutzenfunktionen als Zielfunktionen verwendet werden [Beam and Segev, 1996].

Ein Beispiel eines solchen Multi-Agent-Systems ist Kasbah (`http://ecommerce.media.mit.edu`), ein Projekt des MIT Media Laboratory. Kasbah arbeitet mit Kauf- und Verkaufagenten, die vom Anwender generiert werden und autonom am Markt agieren.

Ein Kaufagent wird parametrisiert durch eine Frist, bis zu der der Kauf erfolgen soll, den gewünschten Preis und den höchsten akzeptierbaren Preis. Beim Verkaufsagenten ist statt einem Höchstpreis der niedrigste akzeptierbare Preis anzugeben. Als Verhandlungsstrategie stehen dem Auftraggeber drei Varianten zur Verfügung: Der Agent kann den Preis linear, quadratisch oder kubisch über die Verhandlungsdauer erhöhen bzw. verringern. Der Agent beginnt dabei mit dem

Wunschpreis, kann kein Vertragspartner gefunden werden, ändert er den Preis entsprechend der festgelegten Strategie. Während ein Agent am Markt tätig ist kann der Anwender jederzeit die Parameter des Agenten ändern.

In Experimenten mit Kasbah hat sich gezeigt, dass die Akzeptanz von Agenten stark damit zusammen hängt, ob Benutzer die Prinzipien, nach denen ein Agent handelt, verstehen. Problematisch beurteilt wurde der Umstand, dass Agenten oftmals eindeutig „dummeÄktionen setzten, wie ein Angebot zu akzeptieren, obwohl am Markt bessere Gebote vorhanden waren.

Letztlich erwarten sich Benutzer von ihren Agenten mehr Selbständigkeit. So muss dem Agenten die Anweisung „Wechsle von einer kubischen zu einer quadratischen Preisverringerungsfunktion"gegeben werden, soll ein Gut schneller verkauft werden. Im Falle eines menschlichen Agenten würde die Anweisung „Verkaufe das Gut schneller"genügen, die Art und Weise, wie dieses Ziel erreicht werden kann liegt im Verantwortungsbereich des Agenten. Außerdem wünschen sich die Anwender, dass der Agent in der Lage sein sollte, den Wunschpreis zu bestimmen, beispielsweise durch Vergleich mit Agenten, die vergleichbare Güter kaufen oder verkaufen [Maes and Chaves, 1996] [Guttman *et al.*, 1998].

Lernende Agenten

Die Entwicklung bei lernenden Agenten zeigt zwei Hauptansätze auf. Einerseits wird versucht, Lernprozesse mit genetischen Algorithmen umzusetzen, andererseits wird die bayessche Statistik herangezogen.

Genetische Algorithmen basieren auf der Evolutionstheorie. Jeder Softwareagent beginnt zunächst mit einer Population an zufällig generierten, nicht notwendiger Weise erfolgversprechender, Verhandlungsstrategien. Diese Population wird nun in der Verhandlung mit anderen Agenten verwendet. Nach dem Verhandlungsprozess werden die einzelnen Strategien durch den Wert des produzierten Ergebnisses beurteilt. Die besseren Strategien werden anschließend als Eltern für neue Strategien herangezogen, Mutationen können zufällig eingeführt werden [Beam and Segev, 1996]. Eine kurze Einführung zum Thema genetische Algorithmen gibt Sangalli [Sangalli, 1998].

Als wesentlicher Nachteil dieser Methode wird die große Zahl an Versuchen gesehen, bis ein Agent eine geeignete Strategie erlernt hat. Alle Versuche müssen mit Partnern durchgeführt werden, die so realistisch wie möglich sind. Wegen des hohen Aufwands ist es jedoch kaum möglich, einen menschlichen Verhandlungspartner zum Training zu verwenden [Beam and Segev, 1997] [Beam and Segev, 1996]. In einer Multi-Agent-Umgebung kann dieses Verfahren nichts desto Trotz angewandt werden, weil die Agenten gegenseitig voneinander lernen. Oliver [Oliver, 1996] zeigt in verschiedenen Experimenten, wie Softwareagenten durch Verhandlungen voneinander bessere Verhandlungsmethoden

lernen. Dazu werden genetische Algorithmen und genetisches Programmieren verwendet.

Ein auf der bayesschen Methode beruhendes Projekt ist Zeng und Sycaras Bazaar [Zeng and Sycara, 1998]. Zwei Agenten, der Käufer und der Verkäufer, handeln um einen ganzzahligen Preis im Intervall von 0 bis 100. Die Agenten machen abwechselnd Preisvorschläge, der Beginnende wird zufällig ermitteln. Als private Information besitzen die Verhandlungspartner ihren eigenen Reservationspreis. Das ist für den Käufer der Preis, den er maximal bereit ist zu zahlen. Für den Verkäufer ist der Reservationspreis der kleinste Preis, den er bereit ist zu akzeptieren. Die Nutzenfunktionen werden als linear angenommen. Jeder Agent darf seine Angebote nur streng monoton stellen, d.h., der Verkäufer kann in keiner Runde einen höheren Preis verlangen als in einer Runde davor. Für den Käufer gilt dies analog.

Die Vorstellung der Agenten über den Reservationspreis des jeweils anderen wird durch eine diskrete Wahrscheinlichkeitsverteilung ausgedrückt, als Schätzer dieses Reservationspreises wird das arithmetische Mittel verwendet. Die bedingte Verteilung, mit der diese Vorstellungen über den gegnerischen Reservationspreis aktualisiert werden, ist fest vorgegeben. Sie besagt, welchen Vorschlag der Verhandlungspartner unter der Bedingung eines bestimmten Reservationspreises machen würde. Anhand des eigenen Reservationspreises und des Schätzwertes des Reservationspreises des Verhandlungspartners wird ein Angebot errechnet.

Zeng und Sycaras zeigen in ihrem Experiment, dass bei Verhandlungen, an denen zwei lernende Agenten beteiligt sind, für beide Partner bessere Ergebnisse erzielt werden, verglichen mit den Konstellationen: Ein lernender gegen einen nicht lernenden Agenten und zwei nicht lernende Agenten gegeneinander.

Auch die zum Zwecke der Simulation in dieser Arbeit verwendeten Agenten sind lernende Agenten, die nach der bayesschen Methode ihr Wissen über die Umwelt aktualisieren.

2.5 Probleme automatisierter Verhandlungen

Dem praktischen Einsatz bei der Verwendung von Softwareagenten auf künstlichen Märkten stellen sich derzeit noch Probleme in den Weg. Das Fehlen einer gemeinsamen Sprache, geeignete Reputationsmechanismen, unzureichende Methoden der Nutzenmessung und Schwierigkeiten beim Finden einer geeigneten Verhandlungsstrategie bilden die vier Problemfelder.

2.5.1 Ontologie

Eine Ontologie ist eine explizite Beschreibung der Begriffswelt. Der Ausdruck stammt aus der Philosophie, wo damit eine systematische Darstellung des Seins gemeint ist [Gruber, 1993].

Menschen verfügen mit Sprache und Schrift über ein hoch entwickeltes Kommunikationsmedium, mit dem die reale Welt beschrieben werden kann. Als Ontologie bezeichnet man einen Weg, wie Beschreibung und Bedeutung eines realen Objekts semantisch richtig einem Softwareagenten übertragen werden können. Eine Ontologie ist notwendig, um sicher zu stellen, dass miteinander verhandelnde Agenten sich auf die gleichen Dinge beziehen.

Das Problem, eine geeignete Ontologie zu finden, steigt mit der Komplexität des Verhandlungsgegenstandes. Mag es bei einer CD noch relativ simpel sein, eine geeignete Kategorisierung ihrer Merkmale zu finden, so stellt sich die Beschreibung eines Autos oder einer Speise in allen notwendigen Details wesentlich schwieriger dar. Neben der Produktbeschreibung spielen in Verhandlungen noch Attribute wie Lieferzeit, Menge, Qualität oder Finanzierungs- und Zahlungsbedingungen eine wichtige Rolle. Ein Agent muss in der Lage sein, all diese Dinge zu bewerten und Zielkonflikte zwischen den einzelnen Attributen des Vertrags zu beurteilen [Beam and Segev, 1997, S. 4].

2.5.2 Nutzenmodellierung

Ziel einer Verhandlung ist es, den Nutzen des Verhandlungsergebnis zu maximieren. Wenn ein Softwareagent diese Aufgabe übernehmen soll, ist es unerlässlich, dem Agenten die Präferenzen, die der Auftraggeber bezüglich mehrerer Verhandlungsausgänge hat, mitzuteilen. Nutzenmodellierung bedeutet, diese Präferenzen einem Agenten mitzuteilen.

Die größte Schwierigkeit besteht darin, dass Menschen nicht über ausreichend definierte Nutzenfunktionen verfügen. Menschliche Vorstellungen über die Ordnung verschiedener Alternativen sind oft nicht vollkommen logisch erklärbar, so können bei deren Abbildung Konflikte entstehen. Speziell dann, wenn das Verhandlungsresultat nicht nur eindimensional ist, sondern sich aus mehreren Variablen wie etwa Preis, Menge, Qualität und Lieferzeit, zusammensetzt, können die Zusammenhänge zwischen den einzelnen Attributen schwer dargestellt werden.

In diesem Zusammenhang kommt der Gestaltung der Benutzerschnittstelle eine wichtige Bedeutung zu. Die am Bildschirm dargestellte Repräsentation der Anwenderpräferenzen muss so aufgebaut sein, dass einerseits ein geeignetes mathematisches Modell daraus formuliert werden kann und andererseits dem Anwender die Bedeutung der dargestellten Information verständlich ist [Bichler *et al.*, 1999, S. 8].

2.5.3 Reputationsmechanismen

Die Akteure auf virtuellen Marktplätzen haben meist nur unzureichende Informationen über die Identität von anderen Teilnehmern. Und selbst wenn solche Informationen bekannt sind, kann deren Richtigkeit oft nicht überprüft werden. Weiters ist es relativ einfach, die Identität auf einem elektronischen Marktplatz zu wechseln.

Daraus resultiert nun das Problem, dass ein Marktteilnehmer nicht ausreichend prüfen kann, wie verlässlich sein Handelspartner ist. Es besteht die Gefahr der Nichterfüllung von Verträgen, etwa weil der Käufer nicht zahlt oder der Verkäufer eine andere als die versprochene Qualität liefert.

Reputationsmechanismen versuchen, aussagekräftige Beurteilungen über die Verlässlichkeit von Marktteilnehmern zu schaffen, die auf der Einschätzung bisheriger Tauschpartner beruhen.

Zwei Probleme kristallisieren sich bei der Entwicklung solcher Ratings heraus: Zum einen können Marktteilnehmer ihre Identität wechseln, wenn ihre Beurteilung zu schlecht wird, etwa unterhalb des Wertes eines neuen Marktteilnehmers fällt. Zweitens besteht die Möglichkeit, dass Marktteilnehmer untereinander Absprachen über eine große Zahl an Scheintransaktionen treffen. Dabei können sie sich wechselseitig gute Bewertungen ausstellen, um ihre Reputation zu verbessern [Guttman *et al.*, 1998].

2.5.4 Verhandlungsstrategie

Beam und Segev weisen auf das Problem hin, dass ein Verhandlungspartner benachteiligt ist, wenn dem anderen Verhandlungsteilnehmer seine Strategie bekannt ist. Angenommen die Strategie eines Verkäufers ist es, nicht unterhalb eines bestimmten Preises zu verkaufen und jedes Angebot darüber zu akzeptieren. Der Käufer kann nun, sofern er in Kenntnis dieser Strategie ist, einen Preis von 0,- vorschlagen und diesen so lange in kleinen Schritten erhöhen, bis er den Mindestpreis des Verkäufers erreicht hat. In dieser Situation kann der Verkäufer aus der Verhandlung keinen zusätzlichen Nutzen ziehen [Beam and Segev, 1997, S. 5].

Um Handlungsempfehlungen in Verhandlungen zu geben, kann die Spieltheorie herangezogen werden, wie in Kapitel 4.

2.6 Zusammenfassung

Ein Beschaffungsprozess läuft in sechs abgrenzbaren Phasen, Bedarfsidentifikation, Produktsuche, Händlersuche, Verhandlungsführung, Lieferung und Zahlung sowie Service ab. Vor allem in den ersten vier Phasen ist der Einsatz von

Softwareagenten denkbar. Diese Arbeit konzentriert sich auf die Phase der eigentlichen Verhandlungsführung.

Das Verhandlungsproblem kann nach dem “Human Factors Approach”, dem ökonomisch-spieltheoretischen und dem computerwissenschaftlichen Ansatz betrachtet werden. Hier wird nach dem Marktmodell von Osborne und Rubinstein [Osborne and Rubinstein, 1990] ein ökonomisch-spieltheoretischer Lösungsweg gewählt.

Der computerwissenschaftliche Ansatz bleibt nicht unberücksichtigt, da für die Simulation Softwareagenten entwickelt wurden, die ihren Nutzen maximieren und aus Beobachtungen der Umwelt Rückschlüsse auf selbige ziehen.

Als alternative Lösungskonzepte für Verhandlungsprobleme sind Negotiation Support Systems (NSS) und Auktionen zu betrachten. NSS erfordern vom Anwender einen hohen Bedienungs- und Spezifikationsaufwand, die Verhandlungsführung erfolgt nach wie vor manuell. Auktionsmechanismen konnten sich durch die Entwicklung des Internet stark verbreiten.

Die größten Problem im praktischen Einsatz von Softwareagenten zur Verhandlungsführung ergeben sich durch eine unzureichende Ontologie. Damit ist die sinngemäße Beschreibung realer Objekte gemeint. Auch die Nutzenmodellierung erweist sich als schwierig, da menschliche Präferenzen oft nicht mit mathematisch-formalen Beschreibungen erfassbar sind. Ebenfalls problematisch erweißt sich die Entwicklung geeigneter Reputationsmechanismen im anonymen Internet. Das Problem der Verhandlungsstrategie versucht die Spieltheorie zu lösen, indem sie Handlungsempfehlungen in klar definierten Situationen gibt.

Kapitel 3

Mikroökonomische Grundlagen

Ein Markt ist durch das Aufeinandertreffen von Angebot und Nachfrage, welche durch den Preismechanismus ausgeglichen werden, gekennzeichnet.

Im Abschnitt 3.1 wird das Marktmodell der vollkommenen Konkurrenz vorgestellt, nach dem sich ein einheitlicher Marktpreis bildet. Abschnitt 3.2 zeigt, das bei monopolistischer Konkurrenz unterschiedliche Preise durch Monopolmacht entstehen.

Durch den Ansatz der neuen Institutionenökonomie, der in Abschnitt 3.3 erklärt wird, werden Transaktionskosten in das Modell miteinbezogen. Die Transaktionskostentheorie erklärt, warum auf einem Markt unterschiedliche Preise entstehen können.

In Abschnitt 3.4 werden die Grundlagen eines Marktmodells nach Osborne und Rubinstein [Osborne and Rubinstein, 1990] beschrieben, auf das der spieltheoretische Ansatz in Kapitel 4 aufbaut.

3.1 Vollkommene Konkurrenz

Das Modell der vollkommenen Konkurrenz ist durch vier Merkmale charakterisiert [Mansfield, 1996, S. 248]:

Atomistische Angebots- und Nachfragestruktur: Die Anzahl der Anbieter und der Nachfrager ist so groß, dass sowohl ein beliebiger Anbieter als auch ein beliebiger Nachfrager nicht in der Lage ist, durch eigene Maßnahmen den Preis zu beeinflussen.

Homogene Güter: Die am Markt gehandelten Güter unterscheiden sich in den Augen der Marktteilnehmer nicht, oder weder Anbieter noch Nachfrager haben Präferenzen mit bestimmten Nachfragern bzw. Anbietern Kaufverträge abzuschließen [Schuhmann *et al.*, 1999, S. 207].

Vollständige Transparenz des Marktes: Alle Marktteilnehmer sind zu jeder Zeit über den Preis informiert, es gibt keine asymmetrische Informationsverteilung. Oder anders formuliert: Jede Information, die mindestens einem Marktteilnehmer bekannt ist, drückt sich sofort in einer Preisänderung aus, so dass Insider dadurch keine Vorteile haben [Otruba *et al.*, 1996, S. 293].

Vollkommene Mobilität der Ressourcen: Es bestehen keine Markteintrittsbarrieren, die potentielle Marktteilnehmer abhalten, am Handel teil zu nehmen. Die Abwicklung von Transaktionen verursacht keine Kosten.

Unter diesen Annahmen sieht sich ein einzelner Anbieter einer vollkommen elastischen Nachfragefunktion gegenüber. Ein Anbieter kann folglich keinen Einfluss auf den Preis nehmen, Anpassungen sind nur über die angebotene Menge möglich [Samuelson and Nordhaus, 1992, S. 141].

Setzt man einen s-förmigen Verlauf der Gesamtkostenkurve voraus [Lechner *et al.*, 1994, S. 368ff]), und handeln die Anbieter gewinnorientiert, so werden sie die Absatzmenge so anpassen, dass die Grenzkosten dem Marktpreis entsprechen [Mansfield, 1996, S. 249f]. Jeder höhere Preis würde die Absatzmenge auf null reduzieren, jeder geringere Preis verletzt die Annahme des gewinnmaximierenden Handelns.

Schwankungen des Marktpreises werden in diesem Modell dadurch erklärt, dass sich die gesamte Nachfrage, z.B. auf Grund von Geschmacksänderungen aller Nachfrager, oder das gesamte Angebot, z.B. durch Änderungen der Faktorpreise, verändert. Durch die zum Teil sehr strikten Modellprämissen wird also ein Verhandlungsspielraum bezüglich des Preises zwischen einzelnen Anbietern und Nachfragern ausgeschlossen [Mansfield, 1996, S. 266].

3.2 Monopolistische Konkurrenz

Das Marktmodell der monopolistischen Konkurrenz unterscheidet sich von dem der vollkommenen Konkurrenz dadurch, dass die Nachfrager die angebotenen Güter nicht mehr als vollständig gleichwertig erachten, jedoch als starke Substitute [Mansfield, 1996, S. 340]. Durch Produktdifferenzierung versuchen die Nachfrager ein gewisses Maß an Monopolmacht aufzubauen. Diese Differenzierung ist etwa durch Marken möglich. Die Produkte unterscheiden sich hinsichtlich Qualität, Erscheinungsbild oder Reputation [Pindyck and Rubinfeld, 1998, S. 523]. Da sich die Güter nun von denen der Konkurrenten unterscheiden, verläuft die Nachfragekurve nach unten geneigt, die Preise und damit auch die Grenzerlöse sind von der angebotenen Menge abhängig [Pindyck and Rubinfeld, 1998, S. 546].

Ein gewinnorientierter Unternehmer wählt nun die Ausbringungsmenge so, dass die Grenzkosten gleich den Grenzerlösen sind. Bei dieser Menge liegt nun der Marktpreis über den Grenzkosten.

Neben den Grenzkosten ist der Preis nun abhängig von der Preiselastizität der Nachfrage, der sich der Anbieter gegenüber sieht. Die Preiselastizität ist durch Produktdifferenzierung beeinflussbar. Je besser es einem Anbieter gelingt, sich von den Konkurrenten abzuheben, umso größer wird seine Monopolmacht sein, und je höher der von ihm erzielte Preis [Pindyck and Rubinfeld, 1998, S. 523ff].

Das Modell der monopolistischen Konkurrenz erklärt den Marktpreis somit durch das Auftreten von Monopolmacht. Dadurch wird es den einzelnen Anbietern möglich, den Preis zum Teil unabhängig von den Mitbewerbern festzusetzen. Der Markteintritt wird insofern erschwert, als das Markteintrittsbarrieren, wie z.B. hoher Werbeaufwand, bestehen und somit potentielle Konkurrenten abgeschreckt werden.

3.3 Neue Institutionenökonomik

Sowohl unter vollkommener als auch unter monopolistischer Konkurrenz wird die Einrichtung des Marktes als kostenfrei nutzbare Koordinationsplattform angenommen. Die neue Institutionenökonomik lässt diese Annahme fallen und bezieht die Kosten, die durch Bereitstellung und Änderung sowie durch Nutzung des Koordinationssystems entstehen, in das Modell mit ein. Durch Einbeziehen dieser Kosten soll geklärt werden, warum reale Märkte nicht dem Idealtypus des vollkommenen Marktes entsprechen [Richter and Furubon, 1996, S. 340 f].

Als Institution wird ein System von anerkannten Normen und Regeln einschließlich Vorkehrungen zu deren Durchsetzung verstanden. Diese Normen und Regeln kommen durch kodifizierte Gesetze, Sitten, Gebräuche und durch Vereinbarungen zum Ausdruck. Zweck einer Institution ist es, individuelles Verhalten in bestimmte Richtungen zu steuern und so Ordnung, Sicherheit und Berechenbarkeit zu etablieren [Richter and Furubon, 1996, S. 7].

Als Transaktion versteht die neue Institutionenökonomik die Übertragung von Verfügungsrechten an materiellen oder immateriellen Gütern [Brösse, 1999, S. 365]. Transaktionen können unternehmensintern oder extern, d.h. auf dem Markt, durchgeführt werden. Sie sind die Folge einer komplexen, arbeitsteiligen Gesellschaft, in der Güter von Produktionsstufe zu Produktionsstufe weitergeleitet werden und der Zukauf von Halb- und Fertigerzeugnissen aufgrund des hohen Spezialisierungsgrades unumgänglich ist [Richter and Furubon, 1996, S. 48].

Ein Markt kann demnach als Institution betrachtet werden, deren Ausgestaltung durch kodifiziertes Recht, Sitten und Gebräuche sowie individuelle Vereinbarungen erfolgt. Wie Richter und Furubon betonen, besteht der Markt aus den

institutionellen Regeln und den sie anwendenden sowie gestaltenden Menschen [Richter and Furubon, 1996, S. 310]. Im angelsächsischen Raum werden Institutionen unterschieden nach „institiutional environmentsünd „institutional arrangements". Die „institutional arrangements"bezeichnen die individuellen Abmachungen von Wirtschaftssubjekten, also Verträge, die im Rahmen objektiver Regeln, den „institiutional environments"geschlossen werden [Brösse, 1999, 362]. Ein Vertrag ist dann eine durch Wirtschaftssubjekte individuell ausgestaltete Institution, mit der Verfügungsrechte an Gütern zugeordnet werden [Brösse, 1999, S. 361ff]. Ein Markt ist der institutionelle Rahmen, innerhalb dessen Verträge erzeugt werden.

In unserem gegenwärtigen Wirtschaftssystem muss ein aufwendiges System an Institutionen aufrecht erhalten werden. Neben den Kosten von Institutionen im Sinne des objektiven Rechts, das sind Institutionen die ihre Ausprägung in Form von Gesetzen und den damit verbundnen Erzeugungs- und Durchsetzungsmechanismen finden, verursachen auch Institutionen im Sinne des subjektiven Rechts, die Verträge, Kosten. Diese vertragsbezogenen Transaktionskosten sind nun Gegenstand des folgenden Abschnitts.

3.3.1 Transaktionskosten

Mit dem Begriff der Transaktionskosten wird nun zum Ausdruck gebracht, dass die Schaffung, Änderung, Nutzung und Aufrechterhaltung eines institutionellen Rahmens zur Durchführung von Transaktionen mit Kosten verbunden ist [Richter and Furubon, 1996, S. 49]. Nach ihrem Bezug zum Vertrag lassen sich Transaktionskosten wie folgt einteilen [Richter and Furubon, 1996, S. 50f und 318ff]:

Kosten der Vertragsanbahnung: Diese Kosten entstehen vor allem durch Such- und Informationskosten, sind also mit der Informationsbeschaffung über Güterpreise und Produktqualität verbunden. Sie entstehen durch unmittelbare Aufwendungen wie Werbung oder Kundenbesuche oder durch die Schaffung organisierter Märkte, wie beispielsweise Wertpapierbörsen oder Messen. Auch Kommunikationskosten (Porti, Telekommunikationskosten, etc.) aufgrund der Kontaktaufnahme zwischen Vertragspartnern sind hier mit einzubeziehen. Weiters entstehen Kosten wegen Qualitätskontrollen, ein Beispiel dafür ist die Überprüfung von Befähigungen einer Arbeitskraft bei der Personalbeschaffung.

Kosten des Vertragsabschlusses: Sie bestehen aus Verhandlungs- und Entscheidungskosten. Verhandlungskosten können durch Zeitaufwand und juristische Beratungsleistungen entstehen. Entscheidungskosten entstehen durch

die Aufarbeitung von Information als Entscheidungsgrundlage. In diesem Prozess sind oftmals externe Fachleute eingebunden, die ebenfalls zu entlohnen sind.

Überwachungs- und Durchsetzungskosten: Sofern ein Vertrag nicht simultan zum Zeitpunkt des Abschlusses erfüllt wird, kann es zur nicht ordentlichen Erfüllung kommen. Daher muss die Erfüllung überwacht werden, die Verträge müssen im nachhinein adaptiert werden. Zunächst müssen Überwachungsmechanismen geschaffen werden um Lieferfristen, Qualität und Quantität von Produkten zu überprüfen. Im Falle der nicht vertragskonformen Erfüllung entstehen Kosten, um Vertragsbestimmungen durchzusetzen, beispielsweise Gerichtskosten und Anwaltskosten sowie Aufwendungen, die mit der Schaffung von Schiedsgerichten verbunden sind.

Transaktionskosten lassen sich hinsichtlich ihrer bestimmenden Größen nach folgenden drei Kriterien unterscheiden:

Unsicherheit: Für den vollständigen Vertrag wird unterstellt, dass alle Leistungen, Gegenleistungen und Eventualitäten im vorhinein für die Vertragslaufzeit erfasst und geregelt werden können bzw. Zweifelsfälle durch das Rechtssystem gelöst werden [Brösse, 1999, S. 372].

Unsicherheit bedeutet, dass mit einem Vertrag nicht alle zukünftigen Situationen abgedeckt werden können, weil sie beim Abschluss nicht bekannt sind, also auch nicht mit der Wahrscheinlichkeit ihres Eintretens bewertet werden können. Verträge sind demnach immer unvollständig und erfordern eine Anpassung im nachhinein [Williamson, 1990, S. 64f] [Schuhmann *et al.*, 1999, S. 476].

Transaktionshäufigkeit: Die Häufigkeit von Transaktionen ist insofern von Bedeutung, weil die damit verbundenen Kosten pro Transaktion durch Skalenerträge mit höherer Transaktionshäufigkeit fallen [Williamson, 1990, S. 69].

Faktorspezifität: Als wichtige Determinante der Transaktionskosten wird in der Literatur die Faktorspezifität hervorgehoben. Faktorspezifität bedeutet, dass Kosten durch Investitionen (z.B. in Produktionsmittel, Werbung oder Ausbildung) entstehen, die vor Abschluss eines Vertrages getätigt werden müssen. Kommt die Transaktion dann nicht zustande, so müssen die Faktoren einer alternativen Verwendung zugeführt werden. Dies ist allerdings oftmals nicht in vollem Ausmaß möglich. Der Teil der „versunkenen Kosten"(vgl. [Schaub, 1997, S. 14, 25, 228]) ist umso höher, je spezieller die Investitionen auf eine bestimmte Transaktion zugeschnitten waren

[Williamson, 1990, S. 37 und 59]. Die Differenz zwischen dem Ertrag der Investition und der alternativen Verwendung wird als Quasi-Rente bezeichnet.

3.3.2 Quasi-Renten

Das auf Williamson zurückgehende Konzept definiert die Quasi-Rente als Differenz zwischen dem Ertrag einer Investition und dem Ertrag ihrer nächstbesten alternativen Verwendung. Williamson führt die Quasi-Rente auf die Faktorspezifität zurück [Williamson, 1990]. Beispielsweise wird für einen Produzenten aufgrund eines Vertrags eine Investition in eine Spezialmaschine zur Produktion des Vertragsgegenstands notwendig. Kommt der Vertrag allerdings nicht zu stande, so können mit der Maschine zwar andere, ähnliche Güter produziert werden, die am Markt allerdings geringere Absatzchancen haben und deswegen kleinere Erträge liefern.

Die Existenz einer Quasi-Rente bietet folglich Gelegenheit für opportunistisches Verhalten. Dabei versucht ein an der Transaktion beteiligter durch „List, Lügen und Betrügen", sich die Quasi-Rente anzueignen. Die Annahme des opportunistischen Verhaltens wird damit begründet, dass diese Art der Ausbeutung immer im Eigeninteresse eines rationalen, nutzenmaximierenden Wirtschaftssubjekt liegt [Williamson, 1990, S. 54].

Der Vertragspartner kann sich die Existenz einer Quasi-Rente nun zu Nutze machen und den Preis der mit der Spezialmaschine hergestellten Güter unter Androhung der Vertragskündigung nach unten drücken [Brösse, 1999, S. 371]. Wie Schaub anhand eines ähnlichen Beispiels erläutert, können Quasi-Renten auf beiden Seiten der an der Transaktion Beteiligten entstehen [Schaub, 1997, S. 235]. So könnte der Produzent nun seinerseits geltend machen, dass die Suche nach einem anderen Produzenten, verbunden mit neuen Vertragsverhandlungen, dem Abnehmer ebenfalls erhebliche Kosten verursachen würde.

In einer Kaufverhandlung kann demnach jeder Verhandlungsteilnehmer folgendes Argument geltend machen: „Schließe jetzt mit mir zu einem für mich günstigeren Preis ab, denn wenn Du Dir einen anderen Vertragspartner suchst, war all der Aufwand für Suche, Informationsgewinnung und Vertragsverhandlungen umsonst, und Du musst all diese Kosten nochmals in Kauf nehmen."

Die Transaktionskostentheorie zeigt, dass die Preisbildung nicht nur durch die Marktform alleine erklärt werden kann, sondern dass auch Kosten der Nutzung des Marktmechanismus mit einbezogen werden müssen. Die dadurch entstehenden Quasi-Renten eröffnen einen Verhandlungsspielraum bei der Preisgestaltung, zur Erklärung der Aufteilung von Quasi-Renten können spieltheoretische Ansätze beitragen [Schaub, 1997, S. 235]. Diese Aufteilung von Quasi-Renten scheint oftmals sehr unterschiedlich zu sein. So zitieren Richter und Furubon

[Richter and Furubon, 1996, S. 57] verschiedene Studien, die erhebliche Preisunterschiede für ähnliche oder sogar gleiche Güter feststellen. Sie begründen diese Preisunterschiede damit, dass Konsumenten zwar bewusst ist, dass das Produkt billiger zu erwerben sei, sie aber oftmals den Aufwand für die Suche nach dem Bestbieter nicht in Kauf nehmen wollen.

3.4 Ein Marktmodell mit dezentralem Handel

Aufbauend auf die mikroökonomische Theorie wird im folgenden ein Marktmodell beschrieben, dass die Rahmenbedingungen für die Ausführungen in Kapitel 4 absteckt. Innerhalb dieses Rahmens können die Händler Preise in bilateralen Verhandlungen bestimmten. Das Modell geht zurück auf [Osborne and Rubinstein, 1990, S.124ff].

3.4.1 Homogene Handelsgüter

Ein nicht teilbares, homogenes Gut wird gegen ein teilbares, homogenes Gut ausgetauscht, wobei letzteres im Folgenden als Geld bezeichnet werden soll.

Homogen bedeutet in diesem Zusammenhang, dass die Güter hinsichtlich ihrer Merkmale in den Augen der Marktteilnehmer gleich bewertet werden. Unterschiedliche Preise aufgrund von Monopolmacht werden damit ausgeschlossen (vgl. Abschnitt 3.2).

3.4.2 Die Händler

Am Markt agieren zwei Gruppen von Händlern: Käufer und Verkäufer. Jeder Verkäufer besitzt ein unteilbares Gut, das er gegen ein teilbares, das Geld, welches der Käufer besitzt, eintauscht. Jede Transaktion wird zu einem bestimmten Preis und zu einem bestimmten Zeitpunkt abgeschlossen.

Jeder Händler kann alle Paarungen aus Preis und Zeitpunkt nach einer Rangordnung ordnen, die durch eine von Neumann-Morgenstern Nutzenfunktion (siehe Abschnitt 4.2) repräsentiert werden kann. Die Händler handeln rational und sind bestrebt, ihren Nutzen zu maximieren.

3.4.3 Matching und Anonymität

Die Gestaltung der Verhandlungsmodalitäten hat einen wesentlichen Einfluss auf die Ergebnisse der Verhandlung. In diesem Zusammenhang ist nicht nur der ei-

gentliche Verhandlungsprozess von Bedeutung, sondern auch die Auswahl der Verhandlungspartner.

Ein Verkäufer hat die Möglichkeit, einen höheren Preis zu erzielen, wenn er damit droht, das Gut an einen alternativen Käufer zu verkaufen. Dies setzt allerdings voraus, dass der Verkäufer

- andere Käufer identifizieren kann und
- in der Lage ist, alternative Käufer selbst zu wählen.

Um solche Verzerrungen auszuschließen, wird angenommen, dass die Marktteilnehmer anonym sind. Die Bestimmung der Paarungen, das Matching, erfolgt zufällig. Die Wahrscheinlichkeit, mit der ein Käufer einen Verhandlungspartner findet, beträgt 1, wenn die Anzahl der Käufer größer oder gleich der Anzahl der Verkäufer ist. Ist die Anzahl der Käufer geringer als die der Verkäufer, entspricht die Wahrscheinlichkeit, mit der ein Käufer einen Verhandlungspartner findet, dem Verhältnis aus Käufern zu Verkäufern.

Ebenfalls zufällig bestimmt wird, wer die Verhandlung beginnen darf. Dabei wird der Beginnende mit einer Wahrscheinlichkeit von 0,5 aus dem Paar ausgewählt [Osborne and Rubinstein, 1990, S. 139.].

Der Ablauf der Verhandlung wird im Detail im Kapitel 4 gemeinsam mit den spieltheoretischen Grundlagen erklärt.

3.4.4 Größe des Verhandlungsspielraums

Osborne und Rubinstein definieren den Verhandlungsspielraum bei Preisverhandlungen im Intervall von 0 bis 1 [Osborne and Rubinstein, 1990, S. 124].

Wie Abschnitt 3.3 zeigt, ist der Verhandlungsspielraum am Markt bei Existenz von Transaktionskosten durch die Quasi-Rente Q bestimmt. P^* ist der Marktpreis, der sich unter vollkommener Konkurrenz (vgl. Abschnitt 3.1), also ohne Transaktionskosten, ergibt. Hat ein Käufer die Quasi-Rente Q_k und der Verkäufer eine Quasi-Rente in der Höhe von Q_v, so beträgt der Verhandlungsspielraum jetzt $Q_k + Q_v$. Kann sich ein Verkäufer durch opportunistisches Verhalten die Quasi-Rente des Käufers vollkommen aneignen, so ergibt sich eine Preisobergrenze von

$$P^{max} = P^* + Q_k \tag{3.1}$$

Diese Preisobergrenze stellt den Reservationspreis des Käufers dar. Es ist jener Preis, den der Käufer maximal bereit ist für das Gut zu zahlen.

Kann sich umgekehrt der Käufer die Quasi-Rente des Verkäufers vollkommen aneignen, so gilt für die Preisuntergrenze:

$$P^{min} = P^* - Q_v \tag{3.2}$$

Der Reservationspreis des Verkäufers ist der Preis, den der Verkäufer als minimales Entgelt für das Gut akzeptiert. Er ist gleich der Preisuntergrenze.

3.5 Zusammenfassung

Nach dem Marktmodell der vollkommenen Konkurrenz kommt auf einem Markt genau ein Preis zustande, der von einzelnen Marktteilnehmern nicht beeinflusst werden kann. Sowohl Käufer und Verkäufer können nur zu diesem Preis Transaktionen tätigen, ein Verhandlungsspielraum ist ausgeschlossen.

Auch die Annahmen der monopolistischen Konkurrenz schließen Verhandlungen der Marktteilnehmer aus, unterschiedliche Preise werden durch Produktdifferenzierung und der damit verbundenen Monopolstellung erklärt.

Erst die neue Institutionenökonomik erklärt durch Transaktionskosten das Zustandekommen von Verhandlungsspielräumen bei der Preisbildung am Markt. Ausgehend vom Marktpreis auf einem perfektem Markt sind noch Quasi-Renten zu berücksichtigen, die durch Faktorspezifität entstehen. Die Marktteilnehmer versuchen dann, sich durch opportunistisches Verhalten die Quasi-Rente des Tauschpartners anzueignen. Wie dieser Aneignungsprozess abläuft, erklärt die Spieltheorie, die den Inhalt des Kapitels 4 bildet.

Kapitel 4

Spieltheoretischer Ansatz

Im vorangegangenen Kapitel wurde aufgezeigt, welche Gestaltungsspielräume bei der Preisfestsetzung durch die Existenz von Transaktionskosten entstehen. Durch die Modellierung eines geeigneten Verhandlungsspiels und der Beschreibung eines Lösungskonzepts für dieses Spiel soll im folgenden die Aufteilung dieser Differenz erklärt werden.

Abschnitt 4.1 erläutert zunächst das bilaterale Verhandlungsspiel und führt in die Spieltheorie ein.

Im Abschnitt 4.2 wird eine Nutzenfunktion entwickelt, die Spieler zur Beurteilung des Spielausgangs heranziehen. Danach wird erläutert, zu welchem Zeitpunkt die Spieler eine Einigung bevorzugen.

Anschließend präsentiert der Abschnitt 4.3 ein Lösungskonzept für das Verhandlungsspiel unter perfekter und vollkommener Information.

Der Abschnitt 4.4 befasst sich mit der Bedeutung von Informationen und deren Verteilung in Spielsituationen. Im anschließenden Abschnitt 4.5 wird gezeigt, welche Lösungskonzepte zur Anwendung kommen, wenn Informationen nicht mehr gleich unter den Spielern verteilt sind.

4.1 Das bilaterale Verhandlungsspiel

Eine Preisverhandlung stellt ein bilaterales Verhandlungsspiel dar, bei dem zwei Spieler, der Käufer k und der Verkäufer v, um die Aufteilung der Differenz aus (höherer) Preisvorstellung des Verkäufers und (niederer) Preisvorstellung des Käufers spielen. Gibt es einen Marktpreis P^*, ist diese Differenz zwischen den Preisvorstellungen der Händler bestimmt durch $Q_k + Q_v$.

Die Menge der möglichen Vereinbarung wird im folgenden dargestellt als Anteil x an $Q_k + Q_v$. Erhält der Verkäufer den Anteil x_v so erhält der Käufer den

Rest $x_k = 1 - x_v$. Der ausgehandelte Preis P ergibt sich somit durch

$$P = P^* - x_k Q_v + x_v Q_k \tag{4.1}$$

Dieses Spiel geht vom „Kuchenteilspieläus, bei dem ein Kuchen zwischen zwei Spielern vollständig aufgeteilt wird [Osborne and Rubinstein, 1990, S. 30].

Beide Spieler haben die Möglichkeit, die Verhandlungen abzubrechen, es besteht keine Vereinbarung darüber, dass sich die Spieler einigen müssen. Sofern die Verhandlung tatsächlich abgebrochen wird, kann von den Spielern die Konfliktauszahlung c realisiert werden. Können sich die Spieler über eine Aufteilung, d.h. auf einen Preis, einigen, so ist dies das Ergebnis der Entscheidungen der Spieler. Das Verhandlungsspiel ist demnach ein nicht kooperatives Spiel [Rieck, 1993, S. 27ff].

Das Spiel selbst läuft alternierend in Runden ab. Zunächst wird aus dem Paar Käufer, Verkäufer zufällig ein Spieler 1 gezogen, der das Spiel beginnt. In der ersten Runde schlägt Spieler 1 einen Preis P_1 vor. Spieler 2 kann P_1 akzeptieren, die Verhandlung abbrechen, wobei c realisiert wird, oder er kann sich entscheiden, das Spiel fortzusetzen und einen Vorschlag P_2 in der 2. Runde zu unterbreiten. Nun kann wieder Spieler 1 akzeptieren, abbrechen oder das Spiel in der dritten Runde mit einem Vorschlag P_3 weiterführen.

In jeder ungeradzahligen Runde ist also Spieler 1 daran, einen Vorschlag zu unterbreiten, in jeder geradzahligen Spieler 2. Beide haben die Alternativen zu akzeptieren, ein Gegengebot zu legen oder die Verhandlung abzubrechen. Jede Runde dieses Spiels konstituiert ein Teilspiel, aus dem das gesamte Verhandlungsspiel zusammengesetzt ist.

Abbildung 4.1 zeigt das Spiel in extensiver Form. An den schwarzen Knoten des Spielbaums trifft Spieler 1 eine Entscheidung, an den weißen Knoten Spieler 2. Wenn ein Spieler einen Preis vorschlägt, so kann er jeden Preis im Intervall P^{min} bis P^{max} wählen. Im Spielbaum werden diese Entscheidungsmöglichkeiten durch Dreiecke dargestellt. In einer Runde kann ein Spieler jeden Preis auf der unteren Kante des Dreiecks wählen [Osborne and Rubinstein, 1990, S. 56].

Bevor auf die verschiedenen Lösungsansätze für das Verhandlungsspiel eingegangen wird, werden zum leichteren Verständnis im Folgenden einige Grundlagen der Spieltheorie erklärt.

4.1.1 Der Strategieraum

Der Strategieraum ist die Menge aller Strategien, die den Spielern zur Verfügung stehen. Eine reine Strategie ist ein Plan, der für alle Entscheidungen innerhalb eines Spiels einen Zug vorschreibt. Eine Strategie enthält auch Pläne für Entscheidungssituationen, die im Spielverlauf nicht erreicht werden.

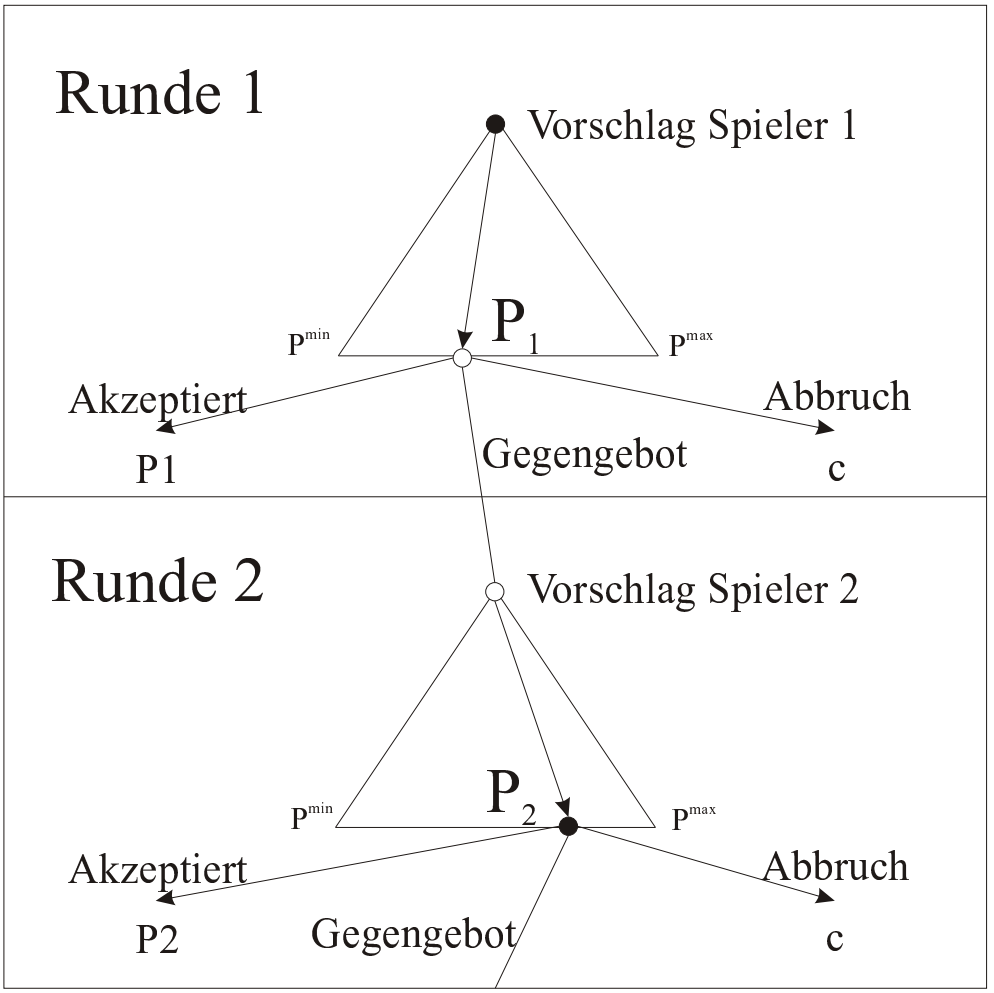

Abbildung 4.1: Darstellung des Verhandlungsspiels in extensiver Form nach Osborne und Rubinstein

Strategien können aus diskreten Zugmöglichkeiten aber auch aus stetigen Zugmöglichkeiten bestehen. Verhandeln zwei Spieler über den Kaufpreis zwischen 0 und 10 Geldeinheit und ist eine Geldeinheit nicht mehr unterteilbar, so kann jeder Spieler aus 11 Zugmöglichkeiten wählen. Es liegt eine diskrete Strategie vor. Wird aber um die Aufteilung einer Torte verhandelt, so gibt es unendlich viele Möglichkeiten, die Torte zu teilen. Die Zugmöglichkeiten sind stetig.

Bei einer gemischten Strategie wählt ein Spieler nach einer bestimmten Wahrscheinlichkeitsverteilung zufällig eine seiner reinen Strategien [Holler and Illing, 1996, S. 33].

4.1.2 Der Auszahlungsraum

Ein Spielausgang x bringt für Spieler 1 und Spieler 2 eine Auszahlung $U(x) = (U_1(x), U_2(x))$. U_1 und U_2 sind die Nutzenfunktionen von Spieler 1 bzw. Spieler 2. Die Menge aller in einem Spiel möglichen Auszahlungsvektoren ist durch den Auszahlungsraum bestimmt.

Sind die Spieler risikoavers, so ist der Auszahlungsraum wegen der konkaven Nutzenfunktionen konvex, d.h., alle Punkte der Verbindungslinien zweier Elemente des Auszahlungsraums sind ebenfalls Elemente des Auszahlungsraums. Die äußere, rechte, obere Grenze des Auszahlungsraums ist die Nutzengrenze, alle Spielausgänge auf dieser Nutzengrenze sind pareto-optimal, weil sich kein Spieler besser stellen kann ohne den anderen schlechter zu stellen (siehe Abbildung 4.2).

Nach links unten ist der Auszahlungsraum durch die Konfliktauszahlung c mit dem Nutzen $U(c) = (U_1(c_1), U_2(c_2))$ begrenzt. Das ist die Auszahlung, die sich die Spieler sichern können, wenn es zu keiner Einigung kommt [Holler and Illing, 1996, S. 40f.] [Holler, 1992, S. 16f.]. Sie ist durch den Nutzen, den ein Spieler erwartet, wenn er in der nächsten Runde eine neue Verhandlung beginnen kann, bestimmt [Osborne and Rubinstein, 1990, S. 126].

Handeln die Spieler individuell rational, so muss das Verhandlungsergebnis immer rechts oberhalb der Konfliktauszahlung liegen. Wird die Differenz zwischen den Preisvorstellungen der Spieler vollständig unter ihnen aufgeteilt ($x_1 + x_2 = 1$), so ist weiters sichergestellt, dass das Ergebnis auf der Nutzengrenze liegt und somit pareto-optimal ist. In diesem Fall kann keiner der Spieler seinen Nutzen erhöhen ohne den anderen Spieler schlechter zu stellen.

Weil jede Verhandlungsrunde Transaktionskosten verursacht, welche die Auszahlung verringern, schrumpft der Auszahlungsraum von Runde zu Runde, wie in Abbildung 4.2 dargestellt.

Unter der Annahme, dass die gesamte Zeit für den erfolgreichen Abschluss einer Transaktion beschränkt ist, ist auch die Konfliktauszahlung zu einem späteren Zeitpunkt geringer. Denn die Wahrscheinlichkeit, einen Verhandlungspartner zu

finden und mit ihm erfolgreich zu verhandeln, ist umso geringer, je kürzer die Zeit dafür ist. Diese Restverhandlungsdauer wird von Runde zu Runde kleiner, somit auch die Erfolgswahrscheinlichkeit einer positiven Verhandlungslösung. Der erwartete Nutzen im Fall des Abbruchs muss somit geringer werden.

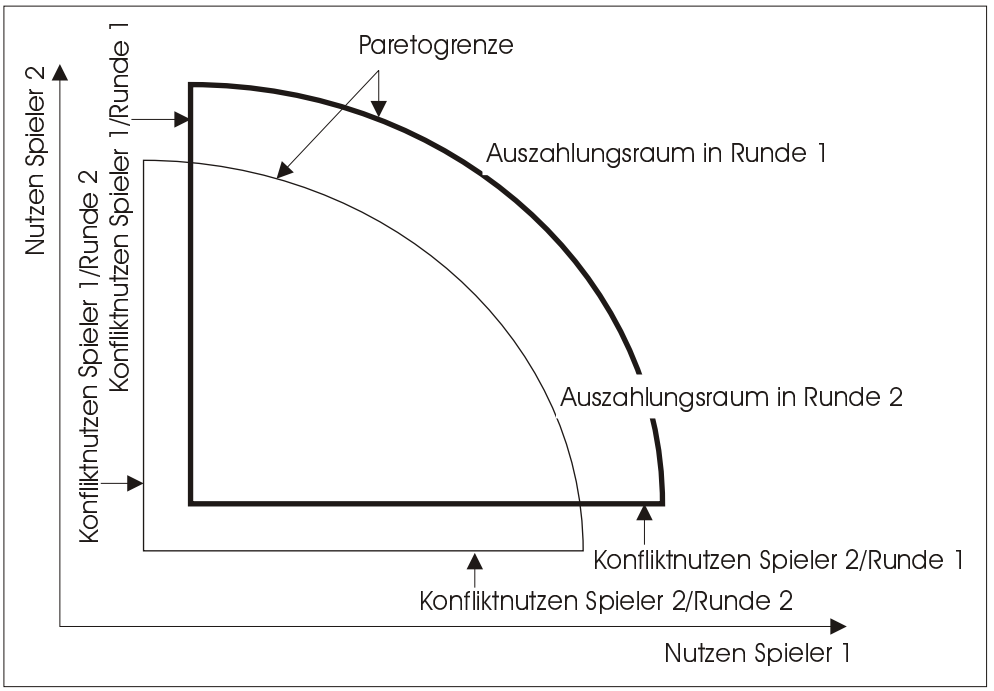

Abbildung 4.2: Der Auszahlungsraum in Runde 1 und Runde 2

4.2 Präferenzen der Spieler

Die Spieltheorie stellt das Ergebnis eines Spiels in Form von Auszahlungen dar. Diese Auszahlungen werden in Nutzeneinheiten gemessen, die die individuelle Rangordnung eines Spielers über verschiedene Alternativen widerspiegelt. Sofern die Auszahlungen nicht sicher sind, sondern bloß mit einer bestimmten Wahrscheinlichkeit eintreten, genügt eine ordinale Nutzenskala nicht mehr. Vielmehr muss ein kardinales System zur Bewertung herangezogen werden, das auch konsistente Aussagen über die Bewertung von Alternativen zulässt, wenn deren Eintreten mit einer bestimmten Wahrscheinlichkeit erwartet wird [Rieck, 1993, S. 132].

Die Erwartungsnutzentheorie verwendet die von Neumann-Morgenstern Nutzenfunktion, die diese Eigenschaften erfüllt.

Um den Nutzen in Bezug auf die Wahrscheinlichkeit, mit der ein Elementarereignis $x_i \in X = \{x_1, x_2, \ldots, x_n\}$ eintritt bewerten zu können, wird zunächst eine

Lotterie $L = (x_1, p_1, x_2, p_2, \ldots, x_n, p_n)$ definiert, bei der jedes Elementarereignis $x_j \in X$ mit der Wahrscheinlichkeit $0 \leq p_j \leq 1$ mit $j = 1 \ldots n$ eintritt.

Die folgenden Axiome charakterisieren nun die geforderten Eigenschaften der von Neumann - Morgenstern Nutzenfunktion [Rieck, 1993, S. 133f]:

1. Es existiert eine schwache Ordnung über die Elementarereignisse.

   Für jedes Paar $x_i, x_j \in X$ gilt $x_i \preceq x_j$ oder $x_i \succeq x_j$.

   Dieses Axiom besagt, dass ein Spieler in der Lage ist, alle Alternativen in einer monotonen Reihenfolge nach seinen Präferenzen zu ordnen. Weil die Lotterie $(x_i, 1, x_j, 0)$ identisch ist mit dem Elementarereignis x_i gelten alle folgenden Forderungen auch für Elementarereignisse.

2. Eine zusammengesetzte Lotterie ist gleich zu bewerten wie eine Lotterie über Elementarereignisse, wenn die Elementarereignisse in beiden Fällen mit den gleichen Wahrscheinlichkeiten eintreten. Es gilt:

   $$[(x_1, p, x_2, (1-p)), w, x_2, (1-w)] \sim [x_1, p \times w, x_2, (1 - p \times w)]$$

   Dieses Axiom fordert, dass ein Spieler die mathematischen Prinzipien der Wahrscheinlichkeitsrechnung anwendet und sich nicht durch die Anordnung von Spielen und Subspielen täuschen lässt.

3. Stetigkeit: Auch noch so große Unterschiede zwischen den Elementarereignissen können durch eine beliebig kleine Eintrittswahrscheinlichkeit kompensiert werden.

   Wenn $x_1 \prec x_i$ und $x_i \prec x_3$ gilt, gibt es immer eine Wahrscheinlichkeit p, für die gilt: $x_i \sim (x_1, p, x_3, (1-p))$.

   x_i wird auch als Sicherheitsäquivalent zur Lotterie bezeichnet.

4. Unabhängigkeit von irrelevanten Alternativen

   Es sei $x_1 \sim x_2$. Dann gilt auch $(x_1, p, x_i, (1-p)) \sim (x_2, p, x_i(1-p))$.

   Wenn ein Elementarereignis x_i besser oder schlechter ist wie $x_1 \sim x_2$ soll das immer gelten, auch dann, wenn x_i Bestandteil einer Lotterie ist, die auch andere Elementarereignisse hervorbringen kann. Diese anderen Elementarereignisse sind aber für die Beurteilung von x_i irrelevant.

5. Transitivität der Ordnung über Lotterien

   Wenn $L_1 \succ L_2$ und $L_2 \succ L_3$ muss auch $L_1 \succ L_3$ gelten. Durch eine Konstruktion einer Lotterie mit $p = 1$ kann gezeigt werden, dass dann auch die Ordnung von Elementarereignissen transitiv ist.

6. Ein Individuum bevorzugt die Lotterie, die das bevorzugte Elementarereignis mit größerer Wahrscheinlichkeit hervorbringt.

   Ist $x_1 \succ x_2$, dann gilt $(x_1, p, x_2(1-p)) \succeq (x_1, w, x_2, (1-w))$ genau dann, wenn $p \geq w$.

Für eine Nutzenfunktion U, die den Axiomen entspricht, gilt nun

$$U(x_i, p, x_j, (1-p)) = pU(x_i) + (1-p)U(x_j) \text{ sofern } 0 < p \leq 1.$$

Nun ist sicher gestellt, dass der Nutzen aus einem erwarteten Ereignis gleich dem Nutzen des Ereignis mal seiner Eintrittswahrscheinlichkeit ist.

Eine von Neumann-Morgenstern Nutzenfunktion lässt keinen interpersonellen Vergleich zu, da Kardinalskalen keinen absoluten Nullpunkt aufweisen [Rieck, 1993, S.141f]. Sie repräsentiert lediglich eine Präferenzordnung über verschiedene Alternativen, d.h., auch eine ordnungserhaltende, lineare Transformation spiegelt die gleichen Präferenzen wieder. Eine Funktion $V(x) = aU(x) + b$ mit $a > 0$ beschreibt die gleichen Präferenzen wie $U(x)$ [Holler and Illing, 1996, S. 39].

4.2.1 Risikoeinstellung der Spieler

Wie ein Spieler eine unsichere Auszahlung einer Lotterie in Bezug zu einer sicheren Auszahlung bewertet, wird durch den Verlauf seiner Nutzenfunktion bestimmt. Ein risikoaverser Spieler ist indifferent zwischen einer sicheren Auszahlung, die kleiner als der Erwartungswert der Lotterie ist, seine Nutzenfunktion verläuft konkav. Diese sichere Auszahlung ist das Sicherheitsäquivalent der Lotterie. Ist das Sicherheitsäquivalent größer als der Erwartungswert der Lotterie, ist die Nutzenfunktion konvex und der Spieler risikofreudig. Im Falle einer linearen Nutzenfunktion ist der Spieler risikoneutral, das Sicherheitsäquivalent ist gleich dem Erwartungswert der Lotterie [Rieck, 1993, S. 137].

Im Allgemeinen kann angenommen werden, dass die Spieler risikoavers bzw. risikoneutral sind und die Risikoaversion mit höherem Einkommensniveau abnimmt. Eine von Neumann Morgenstern Nutzenfunktion, die diesen Annahmen entspricht ist [Biswas, 1997, 20f.]:

$$U = x^{\alpha} \text{ mit } 0 < \alpha \leq 1$$

Dabei gibt α den Grad der Risikoaversion wieder, ein $\alpha < 1$ bedeutet, dass der Spieler sich risikoavers verhält, $\alpha = 1$ charakterisiert einen risikoneutralen Spieler.

Die Erwartungsnutzentheorie erlaubt es, Präferenzordnungen aus bestimmten Axiomen abzuleiten. Anzumerken ist allerdings, dass individuelles Verhalten

oftmals nicht damit in Einklang steht [Holler and Illing, 1996, S. 39]. Geht man beispielsweise davon aus, dass Gewinne anders bewertet werden als Verluste, gemessen an einem gemeinsamen Referenzpunkt, erhält man eine S-förmige Nutzenfunktion. Sie verläuft im Verlustbereich risikofreudig, im Gewinnbereich risikoavers. Weil eine solche Nutzenfunktion nicht mehr transformierbar ist, werden Spiele, die unter den Annahmen nach von Neumann und Morgenstern ident sind, jetzt nicht mehr als ident beurteilt [Rieck, 1993, S. 141f]. Einen Überblick zu dieser Thematik bieten Biswas und Kreps [Biswas, 1997] [Kreps, 1990, S. 116].

4.2.2 Die Konstruktion einer Nutzenfunktion

Wie bei Mansfield beschrieben, kann eine von Neumann-Morgenstern Nutzenfunktion dadurch entwickelt werden, indem ein Individuum mit einer Lotterie konfrontiert wird, bei der es gilt, die Wahrscheinlichkeit zu finden, bei der das Individuum indifferent zwischen einem sicheren Ergebnis und der Lotterie ist [Mansfield, 1996, S. 159]. In Anlehnung an dieses Verfahren kann nun eine Nutzenfunktion gefunden werden, welche die Präferenzen eines Marktteilnehmers hinsichtlich des Preises widerspiegelt. Im Folgenden wird zunächst eine geeignete Nutzenfunktion für einen Käufer entwickelt.

Der Käufer hat das sichere Einkommen Y, es ist jenes Vermögen, das der Käufer beibehält, wenn er nicht in den Markt eintritt. Er ist bereit, einen Preis in der Höhe von P^{max} für den Besitz des Gutes zu zahlen. P^{max} ist jener Preis, bei dem der Käufer indifferent zwischen dem Besitz des Gutes und dem Verlust der Geldmenge P^{max} ist. Bei jedem geringeren Preis wird er sich für den Kauf entscheiden, bei jedem höheren Preis wird er den Besitz von P^{max} Geldeinheiten vorziehen.

Gelingt es, das Gut sofort zum Marktpreis $P^* \leq P^{max}$ zu erwerben, so steigt das Nutzenniveau um $U(P^{max} - P^*)$, denn der Verlust von P^{max} Geldeinheiten wird gerade durch den Besitz des Gutes ausgeglichen. Zeitabhängige Transaktionskosten, etwa für die Suche nach alternativen Anbietern oder Verhandlungskosten, fallen keine an, da das Gut sofort erworben werden kann. Tritt der Käufer erfolglos in den Markt ein, d.h., ist er nicht in der Lage, das Gut innerhalb einer bestimmten Frist t^{max} zu erstehen, so muss er trotzdem Transaktionskosten hinnehmen. Diese Kosten sind zum Teil von der Verweildauer am Markt abhängig (Verhandlungskosten, Suchkosten) und können also als Funktion $T(t^{max})$ ausgedrückt werden. Sein Nutzen beträgt in diesem Falle $U(Y - T(t^{max}))$, sein Einkommen $Y - T(t^{max})$.

Aus Erfahrung weiß der Käufer, dass er mit einer Wahrscheinlichkeit ω das Gut zum Marktpreis erstehen kann. Mit der Wahrscheinlichkeit $1 - \omega$ kann das

Gut nicht erworben werden. Nun kann folgende Gleichung formuliert werden, um den Nutzen von U(Y) zu berechnen:

$$U(Y) = (1 - \omega)U(Y - T(t^{max})) + \omega U(Y + P^{max} - P^*) \tag{4.2}$$

Durch festlegen zweier beliebiger Werte für $U(Y - T(t^{max}))$ und $U(Y + P^{max} - P^*)$, wobei $U(Y - T(t^{max})) < U(Y + P^{max} - P^*)$ gelten muss, kann $U(Y)$ berechnet werden.

Mit den so gefundenen drei Punkten lässt sich jetzt die Nutzenfunktion durch

$$U(Y) = a \times Y^{\alpha} + b \tag{4.3}$$

annähern.

Um den Nutzen einer Transaktion zum Preis P zu berechnen, wird in die Nutzenfunktion das Einkommen nach der Transaktion $Y + P^{max} - P$ eingesetzt:

$$U(P) = a \times (Y + P^{max} - P)^{\alpha} + b \quad \text{mit} \quad P \leq P^{max} \tag{4.4}$$

Dazu ein Beispiel: Der Marktpreis P^* beträgt 100 GE (Geldeinheiten). Die zeitabhängigen Transaktionskosten betragen 2% vom Marktpreis pro Verhandlungsrunde. Die Wahrscheinlichkeit ω beträgt 0,5. Unter diesen Umständen ist der Käufer bereit, maximal 140 GE für das Gut zu bezahlen und will maximal 10 Verhandlungsrunden für den Beschaffungsprozess aufwenden. Demnach ist $P^{max} = 140$ und $t^{max} = 10$. (Der Unterschied zum von Mansfield ([Mansfield, 1996, S. 159f]) angeführten Beispiel besteht darin: Das Individuum, dessen Nutzenfunktion gesucht wird, wird nicht nach der Wahrscheinlichkeit gefragt, die eine Lotterie mit gegebenen Elementarereignissen haben müsste, um gleich wie ein sicheres Ereignis beurteilt zu werden. Hier wird nach den Elementarereignissen der Lotterie bei gegebener Wahrscheinlichkeit gefragt. Diese Elementarereignisse, $Y - T(t^{max})$ und $Y + P^{max} - P^*$, kann der Käufer durch die Wahl von P^{max} und t^{max} bestimmen.)

Um die Ableitung der Nutzenfunktion zu vereinfachen, sei das sichere Einkommen Y gleich $T(t^{max})$. Es beträgt dann $T = P^* \times t^{max} \times 2\%$, also 20 GE. Der Nutzen von $U(Y - T(t^{max}))$ ist dann gleich $U(0)$ und wird mit 1 festgesetzt. $U(Y + P^{max} - P^*) = U(60)$ wird mit 101 festgesetzt. Durch einsetzen in die Gleichung 4.2 kann $U(Y)$ ausgerechnet werden. Folglich beträgt $U(20) = 51$. Mit diesen drei Punkten können nun die Parameter a, b sowie α der Gleichung 4.3 errechnet werden. Als Ergebnis erhält man die Nutzenfunktion

$$U(Y) = 7,55 \times Y^{0,63} + 1 \tag{4.5}$$

für das Einkommen und

$$U(P) = 7,55 \times (160 - P)^{0,63} + 1 \tag{4.6}$$

Tabelle 4.1: Risikoaversion bei unterschiedlicher maximaler Verhandlungsdauer und Reservationspreis

Verhandlungsdauer t^{max}	Reservationspreis P^{max}	Risikoaversion $1-\alpha$
10	140	0,37
8	140	0,45
10	145	0,41

für den Preis.

Um den Grad der Risikoaversion zu messen, kann die relative Risikoaversion $(1-\alpha)$ verwendet werden [Biswas, 1997, S. 20]. Dies ermöglicht, die Werte für t^{max} und P^{max} bei der Ableitung einer Nutzenfunktion zu variieren, um deren Auswirkung auf die Risikoaversion der Marktteilnehmer darzustellen, wie in Tabelle 4.1.

Das Ergebnis zeigt deutlich, dass ein Marktteilnehmer umso risikoaverser ist, je größer sein Reservationspreis ist, was mit der im Reservationspreis enthaltenen Risikoprämie zu erklären ist. Auch ein ungeduldiger Käufer erweist sich -bei gleichem Reservationspreis- risikoaverser, da die zeitabhängigen Transaktionskosten geringer sind und die Risikoprämie steigt.

Analog lässt sich die Nutzenfunktion für den Verkäufer ermitteln, wenn man für das Einkommen nach der Transaktion $Y + P^* - P^{min}$ verwendet.

Eine von Neumann-Morgenstern Nutzenfunktion kann für einen Marktteilnehmer bei gegebenen Marktpreis und Transaktionskosten anhand seines Reservationspreises und dem Zeitlimit für den Beschaffungsprozess t^{max} angenähert werden. Der Reservationspreis ist für den Käufer die Preisobergrenze P^{max}, für den Verkäufer die Untergrenze P^{min}. Diese Nutzenfunktion zeigt die Präferenzen hinsichtlich des Einkommensniveau bzw. des erzielten Preises. Sie lässt aber keine Aussage darüber zu, welche Präferenzen ein Individuum hinsichtlich des Zeitpunktes der Einigung hat, da in der zur Konstruktion verwendeten Lotterie lediglich der beste Fall, die sofortige Einigung zum Marktpreis, mit dem schlechtesten Fall, keine Einigung am Schluss der Verhandlungsfrist, verglichen wurde. Diese Zeitpräferenz der Marktteilnehmer wird im folgenden Abschnitt erläutert.

4.2.3 Zeitpräferenz der Spieler

Die Zeitpräferenzen der Spieler spiegeln deren Bewertung eines Verhandlungsergebnis in unterschiedlichen Verhandlungsperioden wider.

Der Abschluss von Verträgen am Markt ist mit Kosten verbunden. Sie entstehen durch Aufwendungen in Zusammenhang mit Vertragsanbahnung und Abschluss.

Die Transaktionskosten können als Prozentsatz des Transaktionsvolumens ausgedrückt werden [Richter and Furubon, 1996]. Die Transaktionskosten sind von der Dauer der Such- und Informationsphase sowie von der Dauer der Verhandlungsphase abhängig. Da diese Kosten mit zunehmender Verhandlungsdauer steigen, sind die Verhandlungspartner bestrebt, möglichst rasch eine Einigung zu finden [Osborne and Rubinstein, 1990, S. 29ff].

Werden all diese Kosten als Prozentsatz des Preises P pro Verhandlungsrunde ausgerückt, so hat das Verhandlungsergebnis einen Nutzen von

$$u = U(P)\delta^t \qquad \text{mit} \qquad 0 < \delta < 1 \tag{4.7}$$

wobei t die Dauer der Verhandlung ist. Die Variable δ bezeichnet den Diskontfaktor, mit dem der Nutzen pro Verhandlungsrunde abnimmt. Die Nutzeneinbuße pro Runde beträgt somit $P(1-\delta)$. Die Funktion $U(P)$ ist die Nutzenfunktion des Marktteilnehmers hinsichtlich des Preises [Osborne and Rubinstein, 1990, S. 36].

Für Osborne und Rubinstein ist der Diskontfaktor einerseits ein Maß für die Verhandlungskosten, andererseits zeigt er die Geduld der Verhandlungsteilnehmer. Auf eine konkrete Herausarbeitung dieser Komponenten verzichten Osborne und Rubinstein jedoch (vgl. [Osborne and Rubinstein, 1990, S. 37]).

Eine genaue Aufteilung all dieser Verhandlungskosten erscheint allerdings schwierig: Zum einen wird sie durch quantifizierbare Verhandlungskosten geprägt. Andererseits zeigt die Erwartungsnutzentheorie, dass die Risikoeinstellung der Marktteilnehmer Einfluss auf Reservationspreise hat (siehe Abbildung 4.3 auf Seite 40). Als dritte Größe ist nun noch die Geduld der Verhandlungsteilnehmer zu nennen.

Daraus kann der Schluss gezogen werden, dass δ sowohl von der Höhe der Transaktionskosten, der Risikoeinstellung, und auch von der für die Verhandlung zur Verfügung stehenden Zeit abhängt.

Ein Spieler wird einen immer größer werdenden Anteil an der Quasi-Rente opfern, je länger die Verhandlung dauert, um die Chancen auf einen Abschluss zu erhöhen. In der letzten Runde ist er schließlich bereit, die Transaktion zu seinem Reservationspreis durchzuführen. Bezeichnet man mit t^r die Restverhandlungsdauer, so kann die Quasi-Rente Q durch die Verhandlungskosten im Sinne von Osborne und Rubinstein ausgedrückt werden [Osborne and Rubinstein, 1990, vgl. S. 36]:

$$Q = P^*(1 - \delta^{t^r}) \tag{4.8}$$

Durch Umformung dieser Gleichung kann jetzt δ mit

$$\delta = \sqrt[t^r]{(1 - \frac{Q}{P^*})} \tag{4.9}$$

bestimmt werden.

Für jede Restverhandlungsdauer kann jetzt bei gegebenen Marktpreis und Quasi-Rente ein Diskontfaktor ermittelt werden. Der Diskontfaktor δ nimmt mit kleinerer Restverhandlungsdauer t^r ab: Das Verhandlungsergebnis wird zu einem späteren Zeitpunkt schlechter bewertet.

Auch die Höhe der Quasi-Rente nimmt Einfluss auf δ: Je Höher Q ist, umso kleiner wird δ. Muss ein Händler höhere zeitabhängige Transaktionskosten in Kauf nehmen, so sind seine Nutzeneinbußen pro weiterer Verhandlungsrunde größer.

Die Quasi-Rente setzt sich aus zeitabhängigen Transaktionskosten und der Risikoprämie eines Händlers zusammen. Die zeitabhängigen Transaktionskosten am Markt werden als gegeben angenommen. In einer Verhandlung besteht immer das Risiko des Abbruchs. In diesem Fall muss ein Spieler weitere Transaktionskosten für die Suche nach einem neuen Partner und die Verhandlungsführung mit ihm in Kauf nehmen. Ist der Spieler nun risikoavers, wird er das durch drohende Transaktionskosten bestehende Risiko der Nutzeneinbuße durch das Akzeptieren eines schlechteren Verhandlungsergebnis verringern. Je nach der Risikoeinstellung ist also zu den maximalen Transaktionskosten noch eine Risikoprämie aufzuschlagen, soll der Verhandlungsspielraum vollkommen erklärt werden. Abbildung 4.3 zeigt die Zusammensetzung der Quasi-Rente unter der Annahme risikoaverser Spieler.

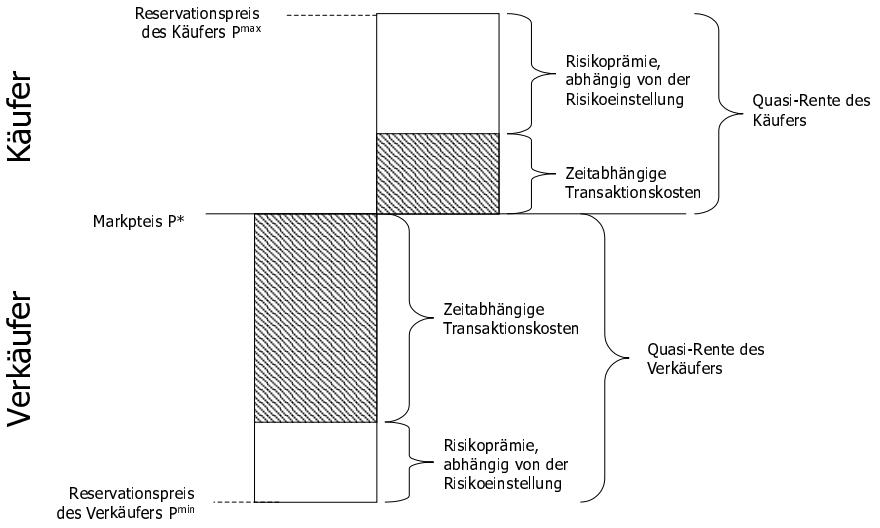

Abbildung 4.3: Zusammensetzung des Verhandlungsspielraums

Der Diskontfaktor δ bringt über die Restverhandlungszeit die Zeitpräferenzen

der Spieler in Zusammenhang mit der Quasi-Rente. Wie im Abschnitt 4.3 gezeigt wird, nimmt der Diskontfaktor wesentlichen Einfluss auf den Verhandlungsausgang.

4.3 Lösungskonzepte

Um Aussagen über das tatsächliche Spielergebnis treffen zu können, werden Lösungskonzepte entwickelt, die aus dem Auszahlungsraum bestimmte Auszahlungen auswählen, die sich aus Annahmen über rationales Verhalten oder Axiomen ableiten. Erstere werden als strategische Konzepte, letztere als axiomatische Lösungen bezeichnet.

Als Beispiel eines axiomatischen Lösungskonzeptes kann die Nash-Verhandlungslösung angeführt werden. Sie fordert, dass sich die Spieler immer auf ein pareto-optimales Ergebnis einigen und sich nicht durch die Form des Auszahlungsraumes täuschen lassen, wenn die Konfliktpunkte die gleichen sind. Weiters bestimmt die Nash-Lösung, dass der Nutzen des Ergebnisses für beide Spieler gleich ist, wenn der Auszahlungsraum symmetrisch ist [Holler, 1992, S. 25]. Die Nash-Verhandlungslösung darf nicht mit dem unten erklärten Nash-Gleichgewicht verwechselt werden.

Die in Folge beschriebenen Lösungskonzepte zählen zum strategischen Ansatz.

Lösungen werden als gleichgewichtig bezeichnet, wenn sie sich dadurch auszeichnen, dass die Spieler ihre Strategieentscheidung auch im nachhinein nicht revidieren wollen [Holler and Illing, 1996, S. 53].

4.3.1 Nash-Gleichgewicht

Das Nash-Gleichgewicht stellt eine Standardlösung in der Spieltheorie dar. Es fordert, dass jeder Spieler eine Strategie wählt, die bei gegebener Strategie des anderen die höchste Auszahlung sichert.

Die Strategie von Spieler 1 sei gegeben mit dem Vorschlag x_1^*, wobei die hochgestellten Sterne die Gleichgewichtslösungen kennzeichnen, x sei wieder der Anteil an den Quasi-Renten $Q_k + Q_v$. Er akzeptiert jedes Angebot von Spieler 2 für das gilt $(1 - x_2) \geq x_1^*$. Die Verhandlung wird abgebrochen wenn $(1 - x_2) < c_1$, die Spieler erhalten dann die Konfliktauszahlung c_1 und c_2. Umgekehrt sieht die Strategie von Spieler 2 aus, wie in Tabelle 4.2 beschrieben.

Wenn also $(1 - x_1^*) < c_2$ oder $(1 - x_2^*) < c_1$ ist, ist c_1, c_2 ein Nash-Gleichgewicht.

Tabelle 4.2: Ein Nash-Gleichgewicht des Verhandlungsspiels

Spieler 1	Vorschlag	x_1^*
	Akzeptiert	$x_1 \geq x_1^*$
	Abbruch	$x_1 < c_1$
Spieler 2	Vorschlag	x_2^*
	Akzeptiert	$x_2 \geq x_2^*$
	Abbruch	$x_2 < c_2$

In allen anderen Fällen ist jeder Vorschlag x_1^* bzw. x_2^* ein Nash-Gleichgewicht: Wenn Spieler 1 immer x_1^* vorschlägt, und jedes Angebot, das schlechter als x_1^* ist, ablehnt, kann Spieler 2 sich nicht besser stellen, wenn er den Vorschlag von Spieler 1 ablehnt. Umgekehrt kann sich Spieler 1 nicht besser stellen, wenn Spieler 2 immer x_2^* vorschlägt und jedes schlechtere Angebot ablehnt.

Im beschriebenen Verhandlungsspiel liefert folglich das Nash-Gleichgewicht kein eindeutiges Verhandlungsergebnis. Jeder Preisvorschlag, der besser als die Konfliktauszahlung ist, stellt ein Nash-Gleichgewicht dar [Osborne and Rubinstein, 1990, S. 41f].

4.3.2 Teilspielperfektes Gleichgewicht

Da das Nash-Gleichgewicht keine eindeutige Aussage über das Verhandlungsergebnis treffen kann, wird das Lösungskonzept verfeinert. Das Verfeinern eines Gleichgewichts bedeutet, aus der Menge der Nash-Gleichgewichtsstrategien aufgrund weiterer Annahmen bestimmte Strategien als unplausibel auszuscheiden [Holler and Illing, 1996, S. 108].

Ausgangspunkt sind jetzt Nash-Gleichgewichte, die keine „glaubhafte Drohung"darstellen.

Schlägt Spieler 1 in der Runde t x_1 vor, so kann Spieler 2 damit drohen, das Angebot auszuschlagen und in der nächsten Runde $t+1$ x_2 vorzuschlagen. Wegen des Diskontfaktors von Spieler 2 kann nun der Nutzen von x_2 in $t+1$ kleiner sein als der von $x_2 = 1 - x_1$ in der Runde t. Die Drohung, x_1 abzulehnen und x_2 zu spielen, ist dann nicht glaubhaft.

Ein Beispiel: Käufer k hat einen Reservationspreis P^{max} von 150 GE und den Diskontfaktor δ_k von 0,8. Der Verkäufer v erwartet mindestens eine Preis von $P^{min} = 60$ GE und sein Diskontfaktor δ_v beträgt 0,6. Bei einem Marktpreis von $P^* = 100$ GE ist der Preisspielraum $Q_k + Q_v$ folglich 90 GE.

Schlägt der Käufer nun eine Preis von $P_1 = 80$ GE vor, so ist die Drohung des Verkäufers, $P_2 = 90$ GE zu verlangen, nicht glaubhaft: Der Verkäufer erhält vom

Käufer einen Anteil an den Quasi-Renten von 20 GE ($P_1 - P^{min}$). Diskontiert man nun den Anteil von 30 GE ($P_2 - P^{min}$), den der Verkäufer für sich reklamiert, mit seinem Diskontfaktor für eine Verhandlungsrunde ab, so erhält man $30 \times 0,6 = 18$, was einem Preis von 78 GE in Runde 1 entspricht. Der Verkäufer kann also eine höhere Auszahlung erzielen, wenn er das Angebot des Käufers annimmt, statt auf seinem Gegenvorschlag zu beharren.

Ein Nash-Gleichgewicht schließt solche „unglaubwürdigen Drohungennicht aus, da es die Strategien der Spieler nur vom Ausgangspunkt jedes Teilspiels beurteilt [Osborne and Rubinstein, 1990, S.43].

Ein teilspielperfektes Gleichgewicht fordert von den Strategien der Spieler, dass in jedem Teilspiel, also auch in der nächsten Verhandlungsrunde, wiederum ein Nash-Gleichgewicht erreicht wird. Das bedeutet, dass der Vorschlag eines Spielers dem entsprechen muss, was der andere Spieler in der nächsten Runde vorschlagen würde.

Damit eine Drohung von Spieler 2 glaubwürdig ist, muss nun der Nutzen von x_2 in Runde $t+1$ größer oder gleich dem Nutzen von $1 - x_1$ in t sein, also $U_2(x_2, t+1) \succeq U_2(1 - x_1, t)$.

Für ein Gleichgewicht muss weiters $U_2(1 - x_1, t) \succeq U_2(x_2, t+1)$ gelten, da sonst Spieler 2 einen Anreiz hätte, x_1 abzulehnen und in der nächsten Runde x_2 zu fordern.

Auf obiges Beispiel bezogen bedeutet das folgendes: Bietet der Käufer eine Preis von 70 GE, so wird der Verkäufer dies ablehnen und einen höheren Preis fordern. Er erhält vom Käufer 10 GE von den Quasi-Renten. Verlangt der Verkäufer in der nächsten Runde 20 GE von den Quasi-Renten, also einen Preis von 80 GE, ist er besser gestellt, denn $10 < 20 \times 0,6 = 12$, was einem Preis von 72 GE in Runde 1 entspricht.

Eine analoge Überlegung gilt für die Regel, mit der Spieler 1 Angebote von Spieler 2 in Runde $t+1$ annimmt bzw. ablehnt. Somit müssen folgende Bedingungen erfüllt sein, damit ein teilspielperfektes Gleichgewicht erreicht wird:

1. Der Nutzen, den Spieler 1 aus dem Angebot x_2 von Spieler 2 erzielt, muss gleich sein dem Nutzen, den Spieler 1 erzielt, wenn er in der nächsten Runde x_1 vorschlägt.

2. Der Nutzen, den Spieler 2 aus dem Angebot x_1 von Spieler 1 erzielt, muss gleich sein dem Nutzen, den Spieler 2 erzielt, wenn er in der nächsten Runde x_2 vorschlägt.

Wenn die Vorschläge beider Spieler größer 0 sind, gilt dann [Osborne and Rubinstein, 1990, S. 45]:

$$U_1(1 - x_2, t) = U_1(x_1, t+1) \qquad \text{und} \qquad U_2(x_2, t+1) = U_2(1 - x_1, t) \quad (4.10)$$

Nimmt man für die Nutzenfunktion der Spieler 1 und 2 nun

$$U_1 = x\delta_1^t \qquad \text{und} \qquad U_2 = x\delta_2^t \tag{4.11}$$

an so existiert ein teilspielperfektes Gleichgewicht, bei dem Spieler 1 in der ersten Runde immer x_1^* vorschlägt und Spieler 2 immer x_2^* akzeptiert mit [Osborne and Rubinstein, 1990, S. 49]:

$$x_1^* = \frac{1-\delta_2}{1-\delta_1\delta_2} \qquad \text{und} \qquad x_2^* = \frac{\delta_2(1-\delta_1)}{1-\delta_1\delta_2}. \tag{4.12}$$

Zurück zum Beispiel: Schlägt der Käufer nun einen Preis von 80,8 GE vor, so erhält der Verkäufer 20,8 GE von den Quasi-Renten, was einem Anteil von 23% an den Quasi-Renten entspricht. Dieser Anteil kann durch Gleichung 4.12 mit $\delta_1 = 0,8$ und $\delta_2 = 0,6$ errechnet werden. Um den Preis auszurechnen, wird Gleichung 4.1 verwendet.

Der Verkäufer müsste in der nächsten Runde mehr als einen Preis von $60 + \frac{20,8}{0,6} = 94,7$ GE verlangen, um besser gestellt zu sein. Bei diesem Preis erhält der Käufer 55,3 GE an den Quasi-Renten.

Dieses Angebot wird der Käufer allerdings ablehnen, da er sich wiederum in der nächsten Runde einen Anteil an den Quasi-Renten von $(150 - 80,8) \times 0,8 = 55,36$ sichern kann. Der Verkäufer kann demnach seine Situation nicht verbessern, wenn der Käufer einen Preis von 80,8 GE bietet. Umgekehrt kann der Käufer auch keinen geringeren Preis als 80,8 GE fordern. Diese Drohung wäre nicht glaubwürdig.

Ein Preis von 80,8 GE stellt das teilspielperfekte Gleichgewicht in der beschriebenen Situation dar.

Eigenschaften des teilspielperfekten Gleichgewichts

Die Präferenzen der Spieler werden durch ihren Diskontfaktor δ beschrieben. Wie die Gleichung 4.9 zeigt, nimmt δ mit höherer Quasi-Rente und geringerer Restverhandlungsdauer ab. Risikoaversion und Transaktionskosten nehmen demnach über δ Einfluss auf den Verhandlungsausgang. Je kleiner der Diskontfaktor δ eines Spielers ist, umso schlechter ist das erzielte Verhandlungsergebnis für ihn.

Je geringer die Risikoaversion eines Spielers ist, desto weniger wird ein Spieler bereit sein, ein für ihn schlechteres Ergebnis statt eines unsicheren, weil in weiteren Verhandlungsrunden erzielbaren, in Kauf zu nehmen.

Ebenso muss ein ungeduldiger Spieler ein schlechteres Ergebnis als ein geduldiger Spieler in Kauf nehmen, da die Verhandlungskosten für ihn höher sind.

Weiters ist das Verhandlungsergebnis asymmetrisch, weil es den Spieler der zuerst einen Vorschlag zu machen hat, bevorzugt. Diese Asymmetrie entsteht

allerdings künstlich, nimmt man an, dass die Zeit zwischen den Verhandlungsrunden gegen 0 strebt, so ist dass resultierende teilspielperfekte Gleichgewicht symmetrisch, der Beginnende kann dadurch keinen Vorteil mehr erzielen. Das entsprechende teilspielperfekte Gleichgewicht ist dann

$$x_1 = \frac{log(\delta_2)}{log(\delta_1) + log(\delta_2)} \qquad \text{und} \qquad x_2 = \frac{log(\delta_1)}{log(\delta_1) + log(\delta_2)} \tag{4.13}$$

[Osborne and Rubinstein, 1990, S. 52, 81ff].

4.4 Informationsstände

Die oben dargestellten Lösungsansätze gehen davon aus, dass alle beteiligten Spieler über alle Informationen verfügen. Diese Situation ist in der Realität aber nicht gegeben.

Für den Ausgang eines Spieles mit rational handelnden Spielern ist es wichtig, festzulegen über welche Informationen die Beteiligten verfügen und wie dieses Wissen unter den Spielern verteilt ist. Information kann von den Spielern immer dann ausgenützt werden um sich einen Vorteil zu verschaffen, wenn sie diese Information alleine besitzen.

4.4.1 Common Knowledge

Common Knowledge ist das gemeinsame Vorwissen der Spieler. Dazu zählen meist die Spielregeln oder bestimmte Eigenschaften der Spieler. Eine Information ist dann Common Knowledge, wenn sie jedem Spieler bekannt ist und wenn alle Spieler wissen, dass diese Information jedem bekannt ist, was wiederum alle wissen, usw. Common Knowledege liegt erst dann vor, wenn diese Kette unendlich oft fortgesetzt werden kann und somit niemals eine Situation erreicht wird, wo einem Spieler nicht mehr bekannt ist, dass ein anderer weiß, dass der Spieler eine Information besitzt [Holler, 1992, S. 44]. Ein illustratives Beispiel dieses Prinzips findet sich bei [Rieck, 1993, S. 177].

Unter das gemeinsame Vorwissen fallen im speziellen Informationen über die Spielregeln des Verhandlungsspiels und Informationen über die Marktsituation:

- Die Spieler kennen den Durchschnittspreis am Markt P^*, d.h., die Marktteilnehmer haben die gleichen Preiserwartungen.
- Die Anzahl der Marktteilnehmer ist bekannt, so dass jeder das Verhältnis von Käufern und Verkäufern kennt. Somit sind die Spieler über die zu erwartende Dauer bis zum Finden eines Verhandlungspartners informiert.

4.4.2 Perfekte und imperfekte Information

Beim Spiel mit perfekter Information ist jeder Spieler in Kenntnis der vorangegangenen Spielzüge der anderen Spieler. Im Falle von imperfekter Information sind nicht alle Züge der Mitspieler beobachtbar.

Sofern ein Spiel über mehrere Stufen gespielt wird, hat aber ein Spieler die Möglichkeit, von beobachtbaren Zügen Rückschlüsse auf nicht beobachtbare Züge zu ziehen [Holler and Illing, 1996, S. 46f] und [Rieck, 1993, S. 92].

Eine weitere Möglichkeit, Wissen über die Mitspieler zu erlangen, besteht im Screening. Dabei wird ein Spielzug gewählt, auf den unterschiedliche Typen von Spielern unterschiedlich reagieren [Feess, 1997, S. 590].

Da die Marktteilnehmer anonym sind, können vorangegangene Spielzüge nur innerhalb einer Verhandlung beobachtet werden. Spielzüge in vorangegangenen Verhandlungen können zwar beobachtet, aber nicht mehr zugeordnet werden.

4.4.3 Vollständige und unvollständige Information

Vollständige Information bedeutet, dass kein Spieler private Informationen besitzt, oder anders gesagt, jede Information ist Common Knowledge. Bei unvollständiger Information besitzen die Spieler private Informationen, die den anderen nicht bekannt sind, Informationen sind asymmetrisch verteilt [Holler and Illing, 1996, S48ff]. Eigenschaften der Spieler, die ihr Verhalten bestimmen und als private Information angesehen werden können sind:

- Jeder Spieler kennt seine eigenen Verhandlungskosten bzw. Zeitpräferenzen, nicht aber die der anderen. Es liegt im Interesse der Spieler, diese geheim zu halten, da sonst die Gefahr besteht, dass ein Verhandlungspartner dieses Wissen ausnützt, indem er vorgibt, selbst niedere Kosten zu haben bzw. weniger ungeduldig zu sein.

- Die Risikoaversion ist ebenfalls eine private Information, gemeinsam mit Verhandlungskosten und Zeitpräferenzen bestimmt sie das Verhandlungsergebnis.

- Jeder Spieler hat Erwartungen über Verhandlungskosten, Zeitpräferenzen und Risikoaversion der anderen Marktteilnehmer. Diese Erwartungen drücken aus, was ein Marktteilnehmer über die Eigenschaften der anderen vermutet. Sie entstehen vor allem durch Erfahrung.

- Jeder Spieler hat Erwartungen hinsichtlich der Dauer einer Verhandlung und der Erfolgswahrscheinlichkeit ω. Die Erwartungen bildet sich der Spieler durch eigene Erfahrungen.

Nach [Harsanyi, 1967] kann ein Spiel mit unvollständiger Information in ein Spiel mit vollständiger, aber imperfekter Information umgewandelt werden. Statt einem Spieler mit privater Information werden mehrer Spielertypen mit bekannten Charakteristika angenommen, aus denen die Natur nach einer bestimmten Wahrscheinlichkeitsverteilung einen als Spieler auswählt. Da der Zug der Natur nicht beobachtbar ist, entsteht die gleiche Situation wie beim Spiel unter imperfekter Information. Die Wahrscheinlichkeitsverteilung über die Typen spiegelt die Verteilung über die private Information wider [Holler and Illing, 1996, S. 49].

Für den gut informierten Spieler besteht die Möglichkeit, durch Signale den schlecht informierten davon zu überzeugen, dass er von einem bestimmten, für ihn vorteilhaften, Typ ist. Dabei werden Spielzüge so gewählt, dass sie vom schlecht informierten Mitspielern als Indiz für einen bestimmten Typ interpretiert werden.

Ob eine Trennung unterschiedlicher Typen durch Signaling möglich ist, also in einem Signalspiel ein „separating equilibriumërreicht wird, hängt von der apriori Einschätzung der Spieler über die Wahrscheinlichkeit ab, einem bestimmten Typ gegenüber zu stehen. Anderenfalls wird ein „pooling equilibriumërreicht, bei dem alle Typen das gleiche Signal senden [Feess, 1997, S. 599] [Kreps, 1990, S. 463ff].

4.4.4 Perfect Recall

Vollkommene Erinnerung unterstellt, dass ein Spieler sich immer an alle vorangegangenen Spielzüge erinnern kann. Er vergisst demnach nie, welche Entscheidungen er im Verlauf des Spiels selbst getroffen hat. Vor allem wenn ein Spieler in der Lage sein soll, aus dem bisherigen Spielverlauf Rückschlüsse auf die Spielsituation zu ziehen, ist die Annahme von Perfect Recall unumgänglich, dies gilt speziell bei strategischen Lösungskonzepten [Holler and Illing, 1996, S. 45].

4.5 Lösungskonzept bei asymmetrischer Information

Die Spieler kennen den Typ ihres Gegenübers bei unvollkommener Information nicht. Das Lösungskonzept der Teilspielperfektion kann nicht angewandt werden, weil sich nun kein Teilspiel mehr konstruieren lässt [Osborne and Rubinstein, 1990, S. 95].

Die Spieler können aber Erwartungen hinsichtlich Verhandlungskosten, Zeitpräferenz und Risikoeinstellung ihres Gegenübers bilden. Aus diesen Erwartungen kann dann eine Erwartung über den Diskontfaktor δ abgeleitet werden. Das Konzept des sequentiellen Gleichgewichts versucht, die-

se Erwartung entsprechend dem bisherigen Spielverlauf zu rationalisieren [Osborne and Rubinstein, 1990, S. 95].

4.5.1 Sequentielles Gleichgewicht

Als Verfeinerung der Teilspielperfektion ist ein sequentielles Gleichgewicht ein Paar aus einer Strategie und einer Wahrscheinlichkeitseinschätzung über die Spielsituation, das folgende Eigenschaften besitzt [Holler and Illing, 1996, S.113]:

- Jede Handlung eines Spielers in einer Spielsituation ist optimal, gegeben die Strategie des anderen und die Wahrscheinlichkeitseinschätzung über die Situation.

  Für das beschriebene Verhandlungsmodell bedeutet das, dass jeder Spieler seine eigenen Präferenzen δ_1 und den Erwartungswert von δ_2 als Schätzer für die Präferenzen der anderen Marktteilnehmer kennt.

  - Wenn Spieler 1 ein Angebot macht, gilt nun: Wenn $\delta_1 \geq E(\delta_2)$ ist, ist das optimale Angebot ein teilspielperfektes Gleichgewicht mit δ_1 und $E(\delta_2)$ als Diskontfaktoren für Spieler 1 und 2, im folgenden als $TSP(\delta_1, E(\delta_2))$ bezeichnet. Ist $\delta_1 < E(\delta_2)$, so wird Spieler 1 $E(\delta_2)$ statt dem eigenen Diskontfaktor δ_1 verwenden, da er andernfalls verraten würde, dass seine Präferenzen ungünstiger sind als die des Verhandlungspartners.
  - Ist Spieler 1 daran, ein Angebot zu akzeptieren oder abzulehnen, dann gilt: Ein Angebot P wird immer angenommen, wenn $P \succeq TSP(\delta_1, E(\delta_2))$. Gilt $TSP(\delta_1, E(\delta_2)) \succ P \succeq c$, dann wird ein Gegenvorschlag unterbreitet. Sollte $c \succ P$ gelten, dann wird die Verhandlung abgebrochen und die Konfliktauszahlung c realisiert.

- Die Wahrscheinlichkeitseinschätzungen sind konsistent mit den im Spielverlauf gemachten Erfahrungen. Spieler 1 muss demnach den Erwartungswert von δ_2 entsprechend der Reaktion auf sein Angebot anpassen.

  - Akzeptiert Spieler 2 das Angebot, so gilt $\delta_2 \leq E(\delta_2)$, die tatsächlichen Präferenzen wurden durch $E(\delta_2)$ überschätzt, $E(\delta_2)$ muss nach unten revidiert werden.
  - Wenn sich Spieler 2 für eine Gegenvorschlag entscheidet muss Spieler 1 seine Einschätzung hinsichtlich δ_2 nach oben revidieren. Der Strategie des sequentiellen Gleichgewichts folgend wird Spieler 2 nur dann

einen Gegenvorschlag unterbreiten, wenn seine Drohung auch glaubhaft ist. Damit das der Fall ist, ist es notwendig, dass Spieler 2 einen höheren Diskontfaktor hat.

– Sollten die Verhandlungen abgebrochen werden, so muss $E(\delta_2)$ nach oben revidiert werden. Weil das Angebot von Spieler 1 nun den Konfliktpunkt von Spieler 2 überschritten hat, muss c_2 nieder sein, was nur bei größerem δ_2 der Fall sein kann.

(vgl. [Osborne and Rubinstein, 1990, S. 95 ff] sowie [Osborne and Rubinstein, 1990, S. 113])

4.5.2 Lernen mit der Bayesschen Regel

Lernen bedeutet, aus den bisher gemachten Beobachtungen die Wahrscheinlichkeitseinschätzung über die Eigenschaften der Mitspieler zu revidieren. Weil das Spiel durch das Vorhandensein von Unsicherheiten wegen unvollständiger Information gekennzeichnet ist, können die Spieler nur dann lernen, wenn sie bereits eine Einschätzung über die Eigenschaften ihrer Mitspieler haben. Diese Einschätzungen sind die „Beliefs"der Spieler [Holler and Illing, 1996, S. 50].

Die Konsistenzeigenschaft des sequentiellen Gleichgewichts fordert, dass die Spieler ihre Wahrscheinlichkeitseinschätzungen anhand der bayesschen Regel bilden [Holler and Illing, 1996, S. 113] [Osborne and Rubinstein, 1990, S. 95].

Jeder Spieler hat zunächst eine Verteilungsfunktion, die die Vorstellung über die unterschiedlichen Typen an Mitspielern in der Grundgesamtheit widerspiegelt. Diese Verteilung ist die sogenannte a-priori Verteilung.

Die „Beliefs"der Spieler sind eine Funktion, die anhand des bisherigen Spielverlaufs eine Wahrscheinlichkeit dafür ermittelt, dass der Mitspieler eine bestimmte Reaktion zeigt, wenn er von einem bestimmten Typ ist.

Mathematisch können „Beliefs"durch eine Likelihood-Funktion dargestellt werden. Eine Likelihood-Funktion gibt an, wie stark eine Wahrscheinlichkeit sich unter einer bestimmten Bedingung ändert.

Die a-posteriori Verteilung ist jene Verteilung über die Typen in der Grundgesamtheit, die dadurch entsteht, weil der Spieler aufgrund der Reaktion des Mitspielers seine a-priori Verteilung angepasst hat.

Die Wahrscheinlichkeit, dass ein Spieler von einem bestimmten Typ T_i ist, nachdem das Ereignis E eingetreten ist, lässt sich nun folgendermaßen errechnen:

$$W(T_i|E) = \frac{W(E|T_i) \times W(T_i)}{W(E)} \tag{4.14}$$

$W(E|T_i)$ ist die Wahrscheinlichkeit, mit der ein Ereignis E eintritt, wenn ein Spieler vom Typ T_i ist. $W(T_i)$ ist aufgrund der a-priori Verteilung bekannt. Die

Eintrittswahrscheinlichkeit von E lässt sich durch den Satz von der totalen Wahrscheinlichkeit berechnen:

$$W(E) = \sum_{i=1}^{I} W(E|T_i) \times W(T_i) \tag{4.15}$$

Die angewandten mathematischen Methoden der Wahrscheinlichkeitsrechnung sind bei Antelman [Antelman, 1997] und Heike und Tarcdea [Heike and Tarcdea, 2000] beschrieben.

Eine Konkretisierung der Gleichungen 4.14 und 4.15 in Hinblick auf die gegebene Spielsituation erfolgt im Kapitel 5.

4.5.3 Spielausgang

Wenn die Spieler nicht vollständig über die Eigenschaften ihrer Verhandlungspartner informiert sind, so benutzen sie ihre Spielzüge, um miteinander zu kommunizieren. Jeder Spieler wird versuchen, aus den Spielzügen des anderen Spielers Rückschlüsse auf dessen Eigenschaften zu ziehen. Gleichzeitig wird jeder Spieler versuchen, sein Gegenüber davon zu überzeugen, dass er in einer besseren Verhandlungsposition ist [Osborne and Rubinstein, 1990, S. 91].

Beobachtet ein Spieler einen Spielzug, der inkonsistent mit allen Strategien der im Spiel möglichen Spielertypen ist, so können auch keine konsistenten Erwartungen hinsichtlich der Eigenschaften des Verhandlungspartners mehr gebildet werden. In solchen Situationen ist ein Spieler grundsätzlich frei in der Erwartungsbildung. Das sequentielle Gleichgewicht fordert lediglich, dass die Erwartungsbildung ab diesem Zeitpunkt konsistent mit dem weiteren Spielverlauf ist [Osborne and Rubinstein, 1990, S. 96].

Eine solche Situation entsteht beispielsweise, wenn ein Käufer das Angebot P macht, dass sowohl von Verkäufern mit höherem als auch niedrigeren Diskontfaktor angenommen werden sollte. Der Verkäufer lehnt das Angebot allerdings ab und stellt ein Gegengebot.

Diese Freiheitsgrade in der Erwartungsbildung, hervorgerufen durch unerwartete Ereignisse im Spielverlauf, führen dazu, dass es eine große Zahl an Gleichgewichtslösungen gibt [Osborne and Rubinstein, 1990, S. 98].

Die Lösungsmenge der sequentiellen Gleichgewichte hängt bei asymmetrischer Informationsverteilung demnach wesentlich von der Einschätzung der Spieler über ihre Mitspieler ab.

Wenn die Spieler aber voneinander lernen können, so sollten die beobachteten Lösungen mit andauerndem Spielverlauf geringere Abweichungen voneinander aufweisen, weil sich die Informationsasymmetrien verringern:

Die Verhandlungsstärke eines Spielers ist bestimmt durch die Höhe der Quasi-Rente und durch seine Geduld, ausgedrückt durch das Zeitlimit. Je höher die Quasi-Rente ist, umso mehr wird ein Spieler ausbeutbar sein.

Ein geduldiger Spieler wird ein besseres Verhandlungsergebnis erzielen, weil sein Diskontfaktor δ größer ist [Osborne and Rubinstein, 1990, S. 51].

Spieler mit einer relativ zum Marktdurchschnitt schlechteren Verhandlungsposition (höhere Quasi-Renten und kleineres Zeitlimit) sollten diesen Umstand erkennen, und durch Vorgeben einer besseren Verhandlungsposition auch bessere Ergebnisse erzielen.

Spieler mit einer guten Verhandlungsposition (niedere Quasi-Renten und längeres Zeitlimit) sollten wegen der „Täuschmanöver"der schlechter gestellten Spieler ihre Vorteile nicht so stark ausspielen können.

Gleichzeitig sollten die Verhandlungszeiten geringer werden. Beim Spiel mit perfekter und vollkommener Information wird eine eindeutige Lösung immer in der ersten Runde erreicht. Sind Informationen asymmetrisch verteilt, muss das nicht zwingend der Fall sein. Fehleinschätzungen der Spieler führen zu Angeboten, die vom Anderen nicht akzeptiert werden. Wenn aber Spieler voneinander lernen, sollten diese Fehleinschätzungen geringer werden und sich die Verhandlungszeit der unter perfekter und vollkommener Information nähern. Kürzere Verhandlungszeiten sind wirtschaftlicher, da weniger zeitabhängige Transaktionskosten anfallen.

Diese Aussagen sollen durch die Naschmarkt-Simulation überprüft werden. Die Lösungsmenge des Verhandlungsspiels wird dabei begrenzt durch den größten und kleinsten Preis, der in einer Verhandlungsrunde gebildet wird. Diese Preisspanne sollte folglich in späteren Marktphasen geringer werden, ebenso wie die Verhandlungszeit.

4.6 Zusammenfassung

Im bilateralen Verhandlungsspiel gleichen Käufer und Verkäufer ihre entgegengesetzten Interessen aus. Jeder Händler kann den Vorschlag seines Gegenübers akzeptieren, ablehnen und einen Gegenvorschlag unterbreiten oder die Verhandlung abbrechen. In diesem Fall muss er mit weiteren Kosten, verursacht durch Suche und Verhandlungen mit alternativen Partnern, rechnen.

Wie die verschiedenen Spielausgänge bewertet werden, versucht die Nutzentheorie zu erklären. Anhand des Reservationspreis und des Zeitlimits für die Verhandlung wurde nach den Axiomen von Neumann und Morgenstern eine Nutzenfunktion modelliert. Geht man von risikoaversen Händlern aus, so trachten diese, dass Abbruchrisiko und die damit weiter verbundenen Transaktionskosten zu verringern, indem sie ein schlechteres Verhandlungsergebnis akzeptieren. Deswegen

ist in den Quasi-Renten eine Risikoprämie enthalten.

Um den Verhandlungsausgang zu klären, wurde das Konzept des teilspielperfekten Gleichgewichts herangezogen. Das teilspielperfekte Gleichgewicht liefert beim Spiel mit perfekter und vollkommener Information ein eindeutiges Ergebnis.

Sind Informationen asymmetrisch verteilt, kann das teilspielperfekte Gleichgewicht nicht mehr angewandt werden. Als Verfeinerung des teilspielperfekten Gleichgewichts wurde das sequentielle Gleichgewicht präsentiert. Es fordert, dass die Strategien der Spieler konsistent mit den Einschätzungen über die Spielsituation sind. Weil aber in der Bildung von Erwartungen über die Spielsituation Freiheitsgrade bestehen, kann das sequentielle Gleichgewicht keine eindeutige Aussage über den Spielausgang treffen.

Wenn die Informationsasymmetrien aber durch lernfähige Agenten verringert werden können, sollten auch die erzielten Verhandlungslösungen geringere Abweichungen untereinander aufweisen. Die Verhandlungslösungen sollten sich dann den des teilspielperfektem Gleichgewichts unter perfekter und vollkommener Information annähern.

Kapitel 5

Dokumentation der Simulation

Aus den in den Kapiteln 3 und 4 gefundenen Erkenntnissen wird nun ein Simulationsmodell entwickelt, das im Anschluss beschrieben wird. Die Implementierung dieses Modells erfolgt in der objektorientierten Programmiersprache C++. Zum Thema C++ sei auf [Willms, 1999] und [Willms, 1997] verwiesen, die C++ Standardbibliothek beschreiben [Kuhlins and Schader, 1999].

Die Simulation greift auf das bereits im Kapitel 3 im Abschnitt 3.4 vorgestellte und in Kapitel 4 näher erläuterte Marktmodell nach Osborne und Rubinstein [Osborne and Rubinstein, 1990] zurück.

Eine große Zahl anonymer Käufer und Verkäufer handeln mit einem homogenen Gut. Die Marktteilnahme ist mit Transaktionskosten verbunden. Außerdem besteht die Gefahr, eine Transaktion nicht in der dafür vorgesehenen Zeit abschließen zu können. Risikoaverse Händlern berücksichtigen diese Gefahr durch eine Risikoprämie in ihren Reservationspreisen.

Der durch Quasi-Renten entstehende Verhandlungsspielraum zwischen Käufern und Verkäufern wird in dem in Kapitel 4 erläuterten bilateralen Verhandlungsspiel aufgeteilt. Dabei machen Käufer und Verkäufer alternierend Vorschläge über den Preis des Gutes. Der ein Preisgebot erhaltende Händler kann dieses entweder annehmen, ein Gegengebot unterbreiten oder die Verhandlung abbrechen.

Die Verhandlungsstrategie der Marktteilnehmer ist durch das Konzept des sequentiellen Gleichgewichts gegeben. Jeder Spieler verfügt über eigene Präferenzen und über eine Erwartung hinsichtlich der Präferenzen des Verhandlungspartners, die er im Verhandlungsverlauf anpassen muss.

5.1 Aufgabe, Ziel und Bewertung

Die Aufgabe der Softwareagenten besteht darin, eine bestimmte Anzahl von Transaktionen am Markt durchzuführen. Für Käufer bedeutet das, eine bestimmte Stückzahl zu erwerben, Verkäufer sollen eine bestimmte Anzahl an Gütern verkaufen. Dabei verfolgen die Händler folgende Ziele:

- Die Händler sollen den bestmöglichen nominellen Preis erzielen. Der nominelle Preis ist der Preis, zu dem ein Vertrag abgeschlossen wird, er berücksichtigt keine Transaktionskosten. Verkäufer müssen versuchen, einen möglichst hohen Preis zu erreichen, Käufer einen möglichst geringen Preis.

- Die Transaktionskosten sollen gering sein. Da ein Teil der Transaktionskosten von der Dauer der Verhandlung abhängig ist, müssen die Agenten versuchen, möglichst rasch zu einer Verhandlungslösung zu kommen.

Um den Erfolg eines Agenten bewerten zu können, werden folgende Daten protokolliert:

Nomineller Preis: Durch den Vergleich des nominellen Preises mit dem vorherrschenden Marktpreis kann ermittelt werden, inwieweit ein Agent den bestmöglichen Preis erreicht hat.

Effektiver Preis: Der effektive Preis ergibt sich aus dem nominellen Preis zuzüglich der angefallenen zeitabhängigen Transaktionskosten für den Käufer. Beim Verkäufer müssen diese Transaktionskosten vom nominellen Preis abgezogen werden.

Verhandlungszeit: Ein Agent ist umso erfolgreicher, je kürzer die Zeit ist, die er benötigt, um eine bestimmte Anzahl von Transaktionen durchzuführen. Je kürzer die Gesamtdauer ist, umso geringer sind die Transaktionskosten.

5.2 Informationsstände

Ebenso wie der Marktmechanismus übt die Informationsverteilung wesentlichen Einfluss auf den Verhandlungsausgang aus.

Unter die Common Knowledge-Annahme fällt, dass jeder Marktteilnehmer über den Marktmechanismus informiert ist. Er kennt den vorherrschenden Marktpreis P^*. Die Anzahl der in den Markt eingetretenen Käufer und Verkäufer, K und V, ist jedem Agenten bekannt.

Das Verhandlungsspiel ist ein Spiel mit imperfekter Information. Ein Spieler kann zwar alle Züge des Anderen innerhalb einer Verhandlung beobachten.

Tabelle 5.1: Marktvariablen

Variable	Beschreibung
P^*	vorherrschender Marktpreis
K	Anzahl der Käufer im Markt
V	Anzahl der Verkäufer im Markt

Jedoch ist es keinem der Spieler möglich, die Züge des anderen Verhandlungspartners zu beobachten, die dieser in vorangegangenen Verhandlungen gemacht hat. Dies nicht einmal dann, wenn ein Spieler schon einmal mit dem selben Partner verhandelt hat. Denn wegen der Anonymität der Spieler können die Spielzüge dem Verhandlungspartner nicht mehr zugeordnet werden. Ein Käufer weiß also nicht, dass er mit diesem Verkäufer schon einmal verhandelt hat.

Alle Informationen, die nicht unter die Common Knowledge - Annahme fallen, sind privat. Dies gilt insbesondere für die bestimmenden Größen Zeitlimit pro Transaktion t^{max} und Quasi-Rente Q. Es liegt somit ein Spiel mit unvollständiger Information vor, in dem die Verhandlungspartner Erwartungen hinsichtlich der privaten Informationen des Anderen bilden müssen.

5.3 Modellvariablen

Die das Modell bestimmenden Variablen werden in Marktvariablen und Variablen, die die Eigenschaften der Händler festlegen unterteilt. Marktvariablen unterscheiden sich von den anderen Variablen dadurch, dass sie für alle Marktteilnehmer die gleichen Ausprägungen haben, sie stellen das Common Knowledge dar. Die Tabellen 5.1 und 5.2 geben einen Überblick.

Die im Folgenden verwendeten Indizes k und v zeigen, ob es sich bei der bezeichneten Größe um die eines Käufers oder eines Verkäufers handelt.

5.3.1 Vorherrschender Marktpreis

Der Marktpreis P^* wird zu Beginn der Simulation mit 100 Geldeinheiten festgesetzt. Danach bestimmt sich P^* durch das arithmetische Mittel aller Preise, die durch erfolgreiche Verhandlungen in einer Verhandlungsrunde zustande gekommen sind. Sollten keine Vereinbarungen getroffen werden, so wird der Marktpreis aus der vorrangegangenen Runde übernommen.

Tabelle 5.2: Variablen der Händler

Variable	Beschreibung
t^{max}	maximale Verhandlungszeit des Händlers
Q	Quasi-Rente
R	Risikoprämie
Θ	Transaktionskostenminimum pro Verhandlungsrunde
δ	Diskontfaktor
ω	Wahrscheinlichkeit einer Einigung zu P^* innerhalb von t^{max}
c	Konfliktnutzen
P_c	Konfliktpreis
Y	Einkommen
Q_e	Zufallsvariable, Erwartung über die Quasi-Rente
t_e^{max}	Zufallsvariable, Erwartung über das Zeitlimit
P^{min}	Reservationspreis eines Käufers
P^{max}	Reservationspreis eines Verkäufers
x	Anteil eines Händlers am Verhandlungsspielraum
P	In Verhandlung erzielter Preis
T	Transaktionskosten

5.3.2 Transaktionskostenminimum pro Runde

Jeder Marktteilnehmer muss ein Mindestmaß an Transaktionskosten hinnehmen, wenn er am Marktmechanismus teilnimmt.

Das Transaktionskostenminimum Θ wird in Prozent des Marktpreises pro Verhandlungsrunde festgelegt.

5.3.3 Maximale Verhandlungszeit

Käufer und Verkäufer haben das Ziel, eine Transaktion innerhalb einer gewissen Frist zu tätigen. Sie ist durch t^{max} Verhandlungsrunden festgelegt.

5.3.4 Quasi-Rente

Um die Höhe der Quasi-Rente Q modellieren zu können, müssen die folgenden Einflussfaktoren berücksichtigt werden:

Zeitlimit: Ziel der Marktteilnehmer ist es, Transaktionen innerhalb einer gewissen Frist t^{max} durchzuführen. Je länger diese Frist ist, umso größer sind die Transaktionskosten, die aufgrund der Nutzung des Marktes entstehen. Diese Kosten sind durch Θ in Prozent des Marktpreises pro Runde gegeben.

Dauert eine Verhandlung t Runden, so entstehen Transaktionskosten in der Höhe von

$$T = P^* \times \Theta \times t \tag{5.1}$$

Risikoeinstellung: Der Markteintritt unter der Bedingung, eine Transaktion innerhalb einer gewissen Frist durchzuführen, ist immer mit dem Risiko verbunden, die Transaktion nicht Fristgerecht durchführen zu können. Unter der Annahme von risikoaversen Wirtschaftssubjekten werden diese bei ihrer Preisober- bzw. Untergrenze noch eine Risikoprämie R berücksichtigen. Ein Marktteilnehmer wird umso eher eine Transaktion abschließen können, je günstiger sein Angebot ist, d.h., je höher der Preisvorschlag des Käufers bzw. je kleiner der des Verkäufers ist.

Die Höhe der Risikoprämie ist bestimmt durch die Risikoeinstellung eines Marktteilnehmers (vgl. Kapitel 4 Absnitt 4.2).

Somit kann die Größe des Verhandlungsspielraums ausgedrückt werden durch die Quasi-Rente, für die gilt (vgl. Abbildung 4.3, Seite 40):

$$Q = T(t^{max}, P^*, \Theta) + R \tag{5.2}$$

5.3.5 Konfliktnutzen

Der Konfliktnutzen ist jener Nutzen, den ein Spieler erwartet, wenn er die Verhandlung abbricht um mit einem anderen Partner eine Einigung zu erzielen. Er ist bestimmt durch [Osborne and Rubinstein, 1990, S. 126]:

$$c = \omega \times U(P^*)\delta^{t^r} \tag{5.3}$$

Dabei bezeichnet ω die Wahrscheinlichkeit, mit der eine Verhandlung erfolgreich zum Marktpreis P^* abgeschlossen werden kann. Ist ψ die Wahrscheinlichkeit, mit einem Verhandlungspartner eine Einigung in einer Runde zu erzielen, so kann ω durch die Summe der geometrischen Reihe

$$\omega = \psi + (1-\psi)^1 \times \psi + (1-\psi)^2 \times \psi + \ldots + (1-\psi)^{t^r} \times \psi = 1 - (1-\psi)^{t^r} \tag{5.4}$$

bestimmt werden. Die Zeit t^r ist die Restverhandlungsdauer.

Neben der Wahrscheinlichkeit ω, die angibt, ob eine Verhandlung mit einem Partner erfolgreich abgeschlossen werden kann, muss noch die Wahrscheinlichkeit, einen Verhandlungspartner zu finden, berücksichtigt werden (vgl. Abschnitt 3.4).

Der Konfliktnutzen für den Käufer ist dann

$$c_k = Min(1, \frac{V}{K}) \times \omega \times U(P^*)\delta^{t^r} \tag{5.5}$$

und für den Verkäufer

$$c_v = Min(1, \frac{K}{V}) \times \omega \times U(P^*)\delta^{t^r} \tag{5.6}$$

5.3.6 Konfliktpreis

Um den Preis zu berechnen, bei dessen Überschreiten ein Käufer die Verhandlung abbricht bzw. bei dessen Unterschreiten ein Verkäufer abbricht, muss der Konfliktpunkt c, der in Nutzeneinheiten gemessen wird, in Geldeinheiten umgerechnet werden.

Bei gegebenen Transaktionskosten Θ sind die Präferenzen eines Händlers durch Q und t^{max} bestimmt.

Wie bereits in Kapitel 4 beschrieben, kann nun eine Nutzenfunktion bestimmt werden.

Sind die Nutzenfunktionen mit

$$U(P) = a \times (Y + P^{max} - P)^{\alpha} + b \tag{5.7}$$

für den Käufer und

$$U(P) = a \times (Y + P - P^{min})^{\alpha} + b \tag{5.8}$$

für den Verkäufer bestimmt, kann der Konfliktpreis berechnet werden.

Wird der Konfliktnutzen c_k in die Gleichung 5.7 bzw. c_v in Gleichung 5.8 statt $U(P)$ eingesetzt und diese Gleichung nach P aufgelöst, so erhält man den Konfliktpreis, der dem Konfliktnutzen entspricht.

Für den Käufer ist der Konfliktpreis

$$P_c^k = Y + P^{max} - \sqrt[\alpha]{\frac{c-b}{a}} \tag{5.9}$$

und für den Verkäufers

$$P_c^v = P^{min} - Y + \sqrt[\alpha]{\frac{c-b}{a}} \tag{5.10}$$

.

5.4 Erwartungen der Marktteilnehmer

Um eine Verhandlungslösung finden zu können, müssen die Erwartungen der Marktteilnehmer bestimmt werden. Die Informationsverteilung im Verhandlungsspiel ist asymmetrisch, Zeitlimit und Quasi-Rente sind private Informationen der Verhandlungsteilnehmer.

Jeder Händler hat Erwartungen über die Quasi-Rente und das Zeitlimit der anderen Marktteilnehmer. Sie werden durch zwei Zufallsvariablen beschrieben, für die erwartete Quasi-Rente Q_e und t_e^{max} für das erwartete Zeitlimit. Als Schätzer für die Präferenzen des Verhandlungspartners können jetzt die Erwartungswerte $E(Q_e)$ und $E(t_e^{max})$ verwendet werden. Diese Erwartungen müssen während der Verhandlungen entsprechend der Aktionen des Verhandlungspartners aktualisiert werden. Dabei wird der Satz von Bayes angewandt.

5.5 Ablauf einer Verhandlung

Zusammenfassend wird nun der Ablauf einer Verhandlung aus der Sicht eines Käufers dargestellt. Die Eigenschaften des Käufers sind durch das Zeitlimit t_k^{max} und seine Risikoprämie Q_k gegeben. Der Käufer hat die Erwartungen $E(Q_e)$ hinsichtlich der Quasi-Rente und $E(t_e^{max})$ hinsichtlich des Zeitlimits des Verkäufers. Diese Erwartungen werden durch ursprünglich binomialverteilte Zufallsvariablen

dargestellt. Durch das Lernen aus Verhandlungen sind diese Zufallsvariablen nach erfolgtem Lernprozess empirisch verteilt.

Der Käufer beginnt mit dem ersten Angebot. Als Verhandlungsstrategie verwenden die Marktteilnehmer das aus dem Kapitel 4 bekannte sequentielle Gleichgewicht.

Zu Beginn der Verhandlung muss der Käufer entscheiden, ob er seine tatsächlichen Eigenschaften für die Erstellung des Angebots heranzieht oder ob er vorgibt, günstigere Eigenschaften zu besitzen. Er verwendet für die Angebotserstellung deswegen immer das Maximum aus seinem eigenen Zeitlimit und dem Erwartungswert von t_e^{max} bzw. das Minimum aus Q_k und $E(Q_e)$.

Danach bestimmt er seinen eigenen Konfliktnutzen c_k (Gleichung 5.5) und seinen Konfliktpreis P_c^k (Gleichung 5.9). Mit den Erwartungswerten $E(Q_e)$ und $E(t_e^{max})$ kann der Käufer den Konfliktnutzen, den Konfliktpreis und den Diskontfaktor des Verkäufers schätzen.

Nun kann der Käufer den eigenen Diskontfaktor errechnen und den des Verkäufers schätzen, dazu wird Gleichung 4.9 verwendet:

$$\delta = \sqrt[t^r]{(1 - \frac{Q}{P^*})} \tag{5.11}$$

Damit kann ein teilspielperfektes Angebot errechnet werden (vgl. Gleichung 4.13 Seite 45):

$$x_k = \frac{log(\delta_v)}{log(\delta_k) + log(\delta_v)} \qquad \text{und} \qquad x_v = \frac{log(\delta_k)}{log(\delta_k) + log(\delta_v)} \tag{5.12}$$

Damit wird ein systematischer Vorteil für den die Verhandlung beginnenden Spieler ausgeschlossen. Der angebotene Preis in Runde 1 wird errechnet mit

$$P_1 = P^* - x_k(P^* - P_c^v) + x_v(P_c^k - P^*) \tag{5.13}$$

Dies entspricht der Gleichung 4.1 auf Seite 30, mit dem Unterschied, dass nicht die gesamte Differenz zwischen den Reservationspreisen $P^{max} - P^{min} = Q_k + Q_v$ aufgeteilt wird sondern die Differenz zwischen den Konfliktpreisen $P_c^k - P_c^v$. Dies geschieht, weil der Strategieraum begrenzt ist durch den Konfliktnutzen, wie in Abschnitt 4.1 dargestellt, der sich vom Nutzen des Reservationspreises unterscheiden kann. Der Konfliktnutzen ist der Nutzen, den ein Händler erwartet, wenn er die Verhandlung abbricht um mit einem anderen Partner eine Lösung zu suchen [Osborne and Rubinstein, 1990, S. 126]. Der Konfliktpreis ist jener Preis, der dem Konfliktnutzen entspricht (siehe Gleichung 5.9 und 5.10).

Als Reaktion auf das Angebot P_1 erhält der Verkäufer nun das Gegengebot P_2. Er muss nun seine Erwartungen Q_e und t_e^{max} anpassen. Die Entscheidung,

die Erwartung über die Quasi-Rente oder das Zeitlimit zu revidieren, wird anhand der Wahrscheinlichkeiten $W(Q_v < E(Q_e))$ und $W(t_v^{max} > E(t_e^{max}))$ getroffen. Ist die Wahrscheinlichkeit, dass die tatsächliche Quasi-Rente des Verkäufers kleiner als der Erwartungswert ist, größer als die Wahrscheinlichkeit, dass das tatsächliche Zeitlimit den Schätzwert übersteigt, so revidiert er Q_e wie unten beschrieben durch den Satz von Bayes. Andernfalls revidiert er t_e^{max} nach oben.

Die Abbildung 5.1 und 5.2 zeigen, wie mittels der Dichtefunktionen für t_e^{max} und Q_e die Wahrscheinlichkeiten $W(t_v^{max} > E(t_e^{max}))$ bzw. $W(Q_v < E(Q_e))$ bestimmt werden können. Die Größe der schraffierten Flächen entsprechen den jeweiligen Wahrscheinlichkeiten. Da die Zufallsvariablen ursprünglich binomialverteilt sind, und diese Verteilung nicht symmetrisch ist, kann dieses Verfahren verwendet werden. In späteren Verhandlungsrunden sind diese Zufallsvariablen empirisch verteilt, weil durch Lernen Erfahrungen gewonnen werden konnten.

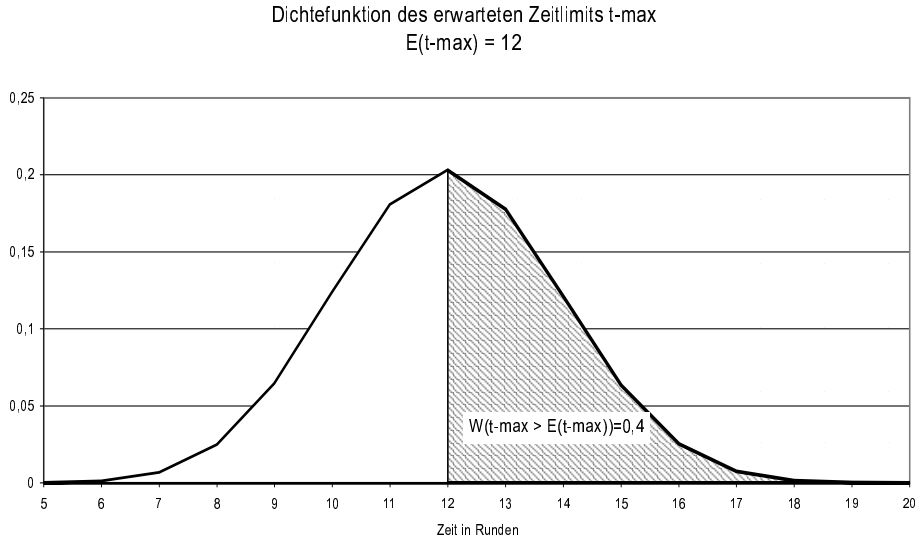

Abbildung 5.1: Ermittlung von $W(t_v^{max} > E(t_e^{max}))$

Im Anschluss prüft der Käufer, ob das Gegengebot seinen eigenen Konfliktpreis übersteigt. Wenn ja, so bricht er die Verhandlung ab. Übersteigt das P_2 den eigenen Konfliktpreis nicht, so prüft der Käufer, ob das Gegengebot günstiger ist als das Angebot, welches er in der nächsten Runde stellen würde. Ist P_2 besser, so nimmt er an und die Transaktion kann abgewickelt werden.

Sofern P_2 nicht besser als das nächste Angebot, aber besser als der Konfliktpreis ist, wird der Käufer weiter verhandeln.

Es ist wichtig anzumerken, dass der Käufer für die Ermittlung des eigenen Konfliktpreises und des Vergleichsgebots immer die eigenen Werte für Q und t^{max} verwendet. Nur bei der Angebotserstellung selbst verwendet der Spieler die je-

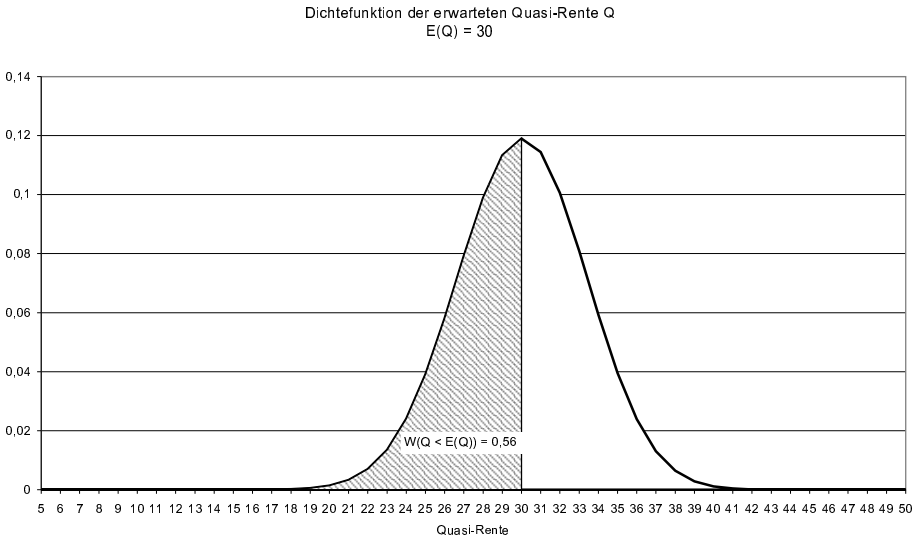

Abbildung 5.2: Ermittlung von $W(Q_v < E(Q_e))$

weils für ihn günstigeren Werte aus Q und $E(Q_e)$ bzw. t^{max} und $E(t_e^{max})$.

5.5.1 Lernen während der Verhandlung

Die Strategie des sequentiellen Gleichgewichts (vgl. Abschnitt 4.4) schreibt vor, dass die Wahrscheinlichkeitseinschätzungen konsistent mit den im Spielverlauf gemachten Erfahrungen sind. Um eine Konsistenz der Wahrscheinlichkeitseinschätzungen zu erreichen, wird vorgeschlagen, diese Einschätzungen mit dem Satz von Bayes anhand des Spielverlaufs zu revidieren [Osborne and Rubinstein, 1990, S. 95] und [Holler and Illing, 1996, S. 113]. Um die Schätzwerte $E(Q_e)$ und $E(t_e^{max})$ entsprechend der Reaktion des Verhandlungspartners anpassen zu können, muss eine geeignete Likelihood-Funktion gefunden werden, mit der die a-priori Verteilungen von Q_e und t_e^{max} verändert werden.

In einer Verhandlung macht der Käufer k ein Angebot zum Preis P_1. Der Verkäufer v entscheidet, dieses Angebot abzulehnen und P_2 als Gegengebot zu machen. Der Käufer entscheidet sich dafür, den Erwartungswert der Quasi-Rente des Verkäufers nach unten zu revidieren.

Aufgrund der a-priori Verteilung von Q_e kann der Verkäufer die Wahrscheinlichkeit $W(Q_e = q_i)$ bestimmen. Die a-posteriori Wahrscheinlichkeit $W(Q_e = q_i|E)$, wobei das Ereignis E das Ablehnen von P_1 ist, kann nun wie folgt ermittelt werden: Die bedingte Wahrscheinlichkeit $W(E|Q_e = q_i)$ ist die Wahrscheinlichkeit, mit der ein Verkäufer P_1 ablehnt, wenn seine Quasi-Rente q_i ist. Sie ist bestimmt durch $W(t_e^{max} > t_{krit}|Q_e = q_i)$:

Für einen Verkäufer, der die Quasi-Rente q_i hat und das Angebot P_1 ablehnt, lässt sich eine Untergrenze für das Zeitlimit t_{krit} durch Umformen der Gleichung 5.11 errechnen, mit dem er P_1 gerade noch angenommen hätte:

$$t_{krit} = \frac{lg(1 - \frac{Q}{P^*})}{lg(\delta_v)} \tag{5.14}$$

Lehnt er aber ab, so muss das tatsächliche Limit größer sein. Die Wahrscheinlichkeit dafür kann durch die Verteilung von t_e^{max} bestimmt werden und beträgt $W(t_e^{max} > t_{krit})$. Die dafür notwendige Bestimmung eines kritischen Wertes für δ ist möglich, weil bekannt ist, welchen Preis der Verkäufer v abgelehnt hat, somit ist bekannt, wie groß der Anteil am Verhandlungsspielraum $P_c^k - P_c^v$ ist, den v abgelehnt hat. Durch Umformen der Gleichung 5.12, mit der das teilspielperfekte Gleichgewicht errechnet wird, kann δ berechnet werden:

$$\delta_v = 10^{\frac{x_k lg(\delta_k)}{x_v}}$$

Die Gleichungen 4.14 und 4.15 können jetzt wie folgt formuliert werden:

$$W(Q_e = q_i|E) = \frac{W(E|Q_e = q_i) \times W(Q_e = q_i)}{W(E)} \tag{5.15}$$

mit

$$W(E) = \sum_{i=1}^{I} W(E|Q_e = q_i) \times W(Q_e = q_i) \tag{5.16}$$

Zum Satz von Bayes sei auf [Antelman, 1997] und [Heike and Tarcdea, 2000] verwiesen.

Die Abbildung 5.3 zeigt die a-priori Verteilung von Q_e mit $E(Q_e) = 28$ und die a-posterior Verteilung mit $E(Q_e) = 25,2$ sowie die dazugehörige Likelihood-Funktion.

Die Likelihood-Funktion ist bestimmt durch

$$L = \frac{W(E|Q_e = q_i)}{W(E)} \tag{5.17}$$

In analoger Weise ist es möglich, eine geeignete Likelihood-Funktion für den Fall, in dem der Käufer glaubt, der Verkäufer hat P_1 abgelehnt, weil sein Zeitlimit größer ist, zu finden. Unter der Bedingung $t^{max} = t_i$ kann jetzt eine kritische Quasi-Rente q_{krit} errechnet werden, bei der der Verkäufer P_1 gerade noch annehmen hätte müssen. Da der Verkäufer aber abgelehnt hat, muss seine tatsächliche Quasi-Rente kleiner als q_{krit} sein. Diese Wahrscheinlichkeit $W(Q_e < q_{krit})$ kann für die Erzeugung der Likelihood-Funktion verwendet werden.

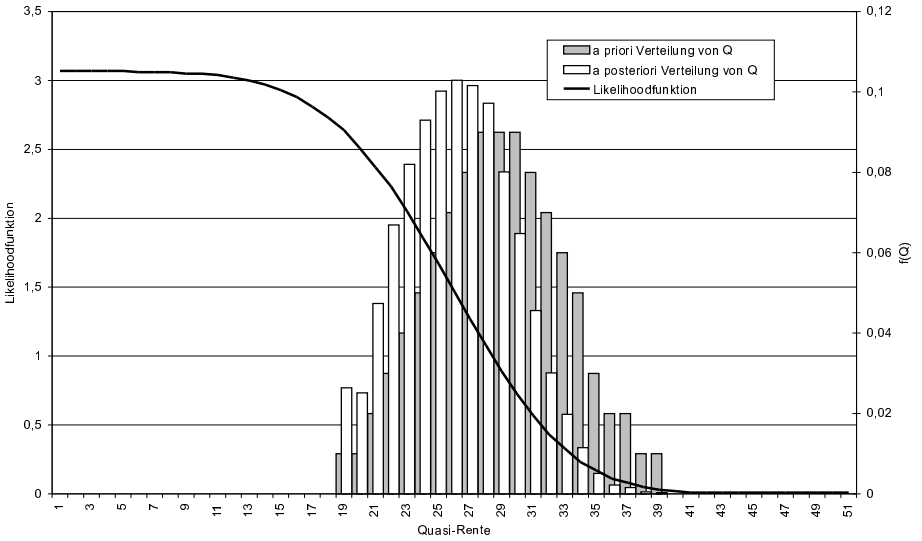

Abbildung 5.3: Lernen über die Quasi-Rente

5.5.2 Ende einer Verhandlung

Eine Verhandlung endet, wenn sich die Partner auf einen Preis geeinigt haben oder das Angebot eines Verhandlungsteilnehmers den Konfliktpreis des anderen übersteigt. Neben diesen beiden Gründen können allerdings noch andere Umstände eine Verhandlung beenden:

- Die Frist t^{max} für den erfolgreichen Abschluss der Verhandlung kann bei längerem Verhandlungsverlauf auslaufen.
- Auch falsches Lernen stellt einen Grund für den Abbruch dar. Sollte ein Händler nicht entsprechend über das Zeitlimit des anderen gelernt haben, so kann es passieren, dass er glaubt, die Restverhandlungsdauer seines Partners beträgt 0. Die Angebotsberechnung ist jetzt nicht mehr möglich, die Verhandlung muss erfolglos abgebrochen werden.

5.6 Lernen aus einer Verhandlung

Innerhalb einer Verhandlung kann ein Händler nur über die Präferenz seines Verhandlungspartners lernen. Um über die vorherrschenden Präferenzen der Marktteilnehmer als Gruppe lernen zu können, muss am Ende einer Verhandlung das Ergebnis beurteilt werden.

Ausgangspunkt sind die Erwartungen hinsichtlich Quasi-Rente und Zeitlimit, die ein Agent zu Beginn der Verhandlung hatte. Das Lernverfahren ist wieder-

um der Satz von Bayes, wie oben beschrieben, mit dem Unterschied, dass nicht aus dem durch den Partner abgelehnte Preis gelernt wird sondern aus dem in der Verhandlung erzielten Preis. Dabei wird wie folgt vorgegangen:

1. Das Verhandlungsergebnis ist schlechter als der Marktpreis (aus Käufersicht ist der erzielte Preis höher als der Marktpreis, aus Verkäufersicht geringer):

   Der Verhandlungspartner muss demnach eine kleinere Quasi-Rente und/oder ein größeres Zeitlimit haben.

   (a) Die Verhandlungsdauer war länger als erwartet:
   Es ist anzunehmen, dass das Zeitlimit des Verhandlungspartners t^{max} größer als ursprünglich erwartet ist.

   (b) Die Verhandlungsdauer war nicht länger als erwartet:
   Die Quasi-Rente des Verhandlungspartners ist kleiner. Die erwartete Quasi-Rente darf allerdings nur dann nach unten revidiert werden, wenn die Annahme der Risikoaversion nicht verletzt wird. Sollte diese Annahme verletzt werden, so wird die Quasi-Rente trotzdem nach oben revidiert, denn es ist möglich, dass der Verhandlungspartner ungünstigere Eigenschaften besitzt als er vorgibt.

2. Das Verhandlungsergebnis ist nicht schlechter als der Marktpreis (aus Käufersicht ist der erzielte Preis geringer als der Marktpreis, aus Verkäufersicht größer):

   Der Verhandlungspartner muss demnach eine größere Quasi-Rente und/oder ein kleineres Zeitlimit haben. Ist die Verhandlungszeit kleiner als erwartet und wird die Annahme der Risikoaversion nicht verletzt, so wird das Zeitlimit nach unten revidiert. Andernfalls wird die Quasi-Rente nach oben revidiert.

Gelernt wird immer nur aus einem positivem Verhandlungsabschluss. Aus dem Verhandlungsabbruch können keine Rückschlüsse gezogen werden. Der Grund für den Abbruch ist immer nur dem abbrechenden Verhandlungspartner bekannt. Der den Abbruch hinnehmende Verhandlungsteilnehmer kann nicht aus der Aktion des anderen lernen, weil er den Grund für den Abbruch nicht kennt.

Neben den Erwartungen über Quasi-Rente und Zeitlimit der anderen Marktteilnehmer benötigt ein Agent noch eine Erwartung hinsichtlich der Verhandlungsdauer. Dies ist notwendig für die Berechnung des Konfliktnutzens, weil dafür eine Annahme über die zu erwartende Verhandlungsdauer notwendig ist. Diese Erwartung wird durch einen gleitenden Mittelwert gebildet.

Der Mittelwert der Verhandlungszeit t_m wird am Ende einer Verhandlung errechnet:

$$t_m = t_{n-1}(1-\gamma) + t_n \times \gamma \tag{5.18}$$

Dabei bezeichnet der Index $n-1$ den Wert aus der letzten Verhandlung, der Index n den aktuellen Wert, γ ist die Lernrate.

Kapitel 6

Ergebnisse der Simulation

In diesem Kapitel werden die Ergebnisse verschiedener Simulationen mit den in Kapitel 5 beschriebenen Agenten erläutert. Dabei wird untersucht, wie sich der Markt entwickelt und inwieweit die Ergebnisse mit den theoretischen Erkenntnissen in Einklang stehen.

Nach einer kurzen Erklärung des Simulationsaufbaus in Abschnitt 6.1 wird in Abschnitt 6.2 der typische Ablauf einer Simulation beschrieben, der in allen Experimenten zu beobachten ist. Dabei wird vorrangig darauf eingegangen, ob die spieltheoretischen Erkenntnisse in der Simulation gelten.

In einem ersten Experiment werden zwei Märkte mit unterschiedlichen Transaktionskosten verglichen. Die Ergebnisse sind in Abschnitt 6.3 wiedergegeben.

Im Abschnitt 6.4 werden auf einem Markt Agenten mit unterschiedlichen Transaktionskosten simuliert, um zu untersuchen, wie sich diese Unterschiede auf die Verhandlungsergebnisse auswirken.

Abschnitt 6.5 fasst die wichtigsten Erkenntnisse aus den Simulationen noch einmal zusammen.

Das Kapitel schließt mit Beispielen konkreter Verhandlungen, um zu prüfen, inwieweit die entwickelten Softwareagenten plausibles Verhalten an den Tag legen.

6.1 Simulationsaufbau

Für die im Folgenden beschriebenen Simulationen wurden immer 250 Käufer und 250 Verkäufer mit der Aufgabe, jeweils 400 Transaktionen abzuschließen, verwendet. Die Agenten werden anhand der zeitabhängigen Mindesttransaktionskosten in zwei Gruppen eingeteilt:

- Agenten mit niederen Transaktionskosten haben Mindesttransaktionskosten von 1,5% des Marktpreises pro Verhandlungsrunde.

- Agenten mit hohen Transaktionskosten haben Mindesttransaktionskosten von 3% des Marktpreises pro Verhandlungsrunde.

Die Risikoprämie und das maximale Zeitlimit pro Transaktion werden durch Zufallswerte, deren Verteilung sich an eine Binomialverteilung annähert, bestimmt.

Die Population der Käufer ist symmetrisch zu jener der Verkäufer. Es gibt zu jedem Käufer mit einer bestimmten Quasirente und einem bestimmten Zeitlimit einen Verkäufer mit der gleichen Quasirente und dem gleichen Zeitlimit.

Eine Simulation wird abgebrochen, wenn nur noch 2% der Käufer und Verkäufer am Markt sind, d.h., wenn mindestens 245 Käufer und 245 Verkäufer ihre Aufgabe erfüllt haben.

6.2 Typischer Simulationsablauf

Der Ablauf einer Simulation kann anhand des Transaktionsvolumens pro Runde und der Spannweite der in einer Runde entstandenen Preise in drei Phasen gegliedert werden. Die Abbildung 6.1 zeigt diese Abschnitte.

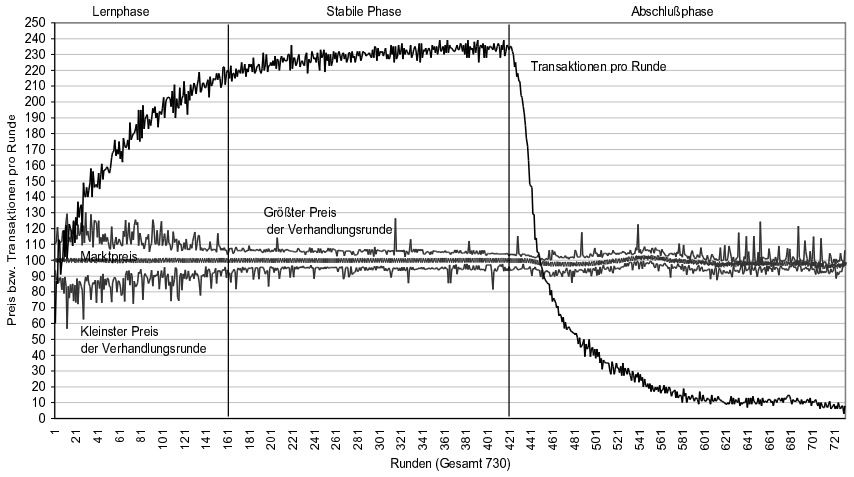

Abbildung 6.1: Einteilung des Simulationsablaufs in Phasen

6.2.1 Lernphase

Rund die ersten 160 Verhandlungsrunden bilden die Lernphase. Die Spannweite der in den Verhandlungen zustande gekommenen Preise nimmt mit zunehmender Simulationsdauer stetig ab. Das Transaktionsvolumen steigt stark an. Der Marktpreis weist leichte Fluktuationen auf.

6.2.2 Stabile Phase

Über etwa die nächsten 270 Runden bleibt der Markt nahezu stabil. Die Preisspannweiten verändern sich kaum, ebenso wenig steigt das Transaktionsvolumen nur noch geringfügig.

Die Länge dieser Phase hängt davon ab, wie viele Transaktionen die Agenten tätigen müssen.

6.2.3 Abschlußphase

Nach zirka 420 Runden haben die ersten Agenten ihre Aufgabe erfüllt und verlassen den Markt. In den folgenden 70 bis 90 Runden erreicht der Großteil der Agenten das Ziel, 400 Transaktionen zu tätigen, erkennbar am stark fallenden Transaktionsvolumen. Gleichzeitig beginnt sich der Marktpreis zu ändern.

Nach etwa 500 Runden sind nur noch rund ein Fünftel der Marktteilnehmer am Markt.

Die Preisspannweiten werden gegen Simulationsende wieder größer, was aber eher auf die geringe Zahl der Marktteilnehmer zurückzuführen ist als auf bestimmte Lerneffekte der Agenten.

Die Abschlußphase wird im Folgenden aus der Beurteilung der Ergebnisse ausgenommen, da in einem realen Markt immer wieder neue Händler hinzukommen und nicht alle Marktteilnehmer den Markt verlassen.

6.2.4 Spieltheoretische Beurteilung

Dieser typische Verlauf der Simulation bestätigt die zu Ende des Kapitel 4 getroffenen Aussagen:

Händler mit ungünstigen Eigenschaften können ihre Verhandlungsergebnisse verbessern, während Händler mit günstigen Eigenschaften ihre Vorteile nicht mehr so stark nutzen können. Deswegen werden die Preisspannweiten geringer.

Abbildung 6.2 zeigt die durchschnittlichen Nominalpreise der Verkäufer in Bezug zum Diskontfaktor δ. Dabei werden die Nominalpreise nach 50 Runden mit den Nominalpreisen nach 400 Runden verglichen. Nach den ersten 50 Runden ist eine deutliche Korrelation zwischen kleinem Diskontfaktor und schlechtem Verhandlungsergebnis zu erkennen. Nach 400 Runden erzielen die Verkäufer mit geringerem Diskontfaktor deutlich bessere Ergebnisse während jene mit hohem Diskontfaktor nicht mehr so extrem gute Verhandlungsergebnisse erzielen.

Auch Hinsichtlich der Verhandlungszeit zeigt sich, dass die Agenten mit zunehmender Simulationsdauer effizientere Ergebnisse erzielen: Da jede Verhandlungsrunde mit zeitabhängigen Transaktionskosten verbunden ist, sind kürzere Verhandlungszeiten wirtschaftlich effizienter.

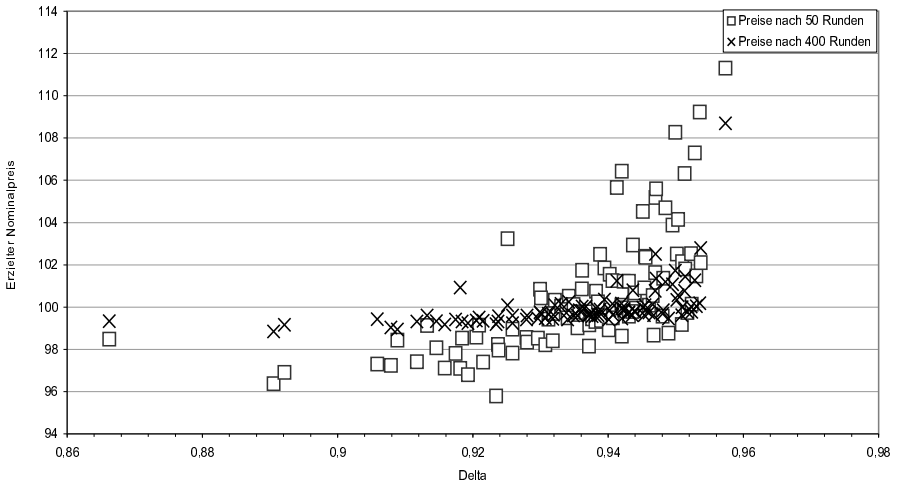

Abbildung 6.2: Durchschnittliche Nominalpreise in Runde 50 und 400

Das Abnehmen der Verhandlungszeiten ist in allen Experimenten zu beobachten, eine Darstellung der Verhandlungszeiten zeigt Abbildung 6.4.

6.3 Zwei Märkte mit unterschiedlichen Transaktionskosten

Im folgenden werden zwei Märkte miteinander verglichen, die unterschiedliche zeitabhängige Transaktionskosten aufweisen. In einer ersten Simulation müssen alle Agenten 3% des Marktpreises pro Runde als Transaktionskosten in Kauf nehmen, in einer zweiten Simulation sind es 1,5%.

Abbildung 6.3 zeigt die Population der beiden Simulationen nach der Anzahl der Agenten mit bestimmter Quasi-Rente und bestimmten Zeitlimit. Die Fläche der Kreise ist proportional zur Anzahl der Agenten.

Die Verteilung der Risikoprämien und des Zeitlimits ist auf beiden Märkten gleich. Die durchschnittliche Risikoprämie beträgt 31 GE, das durchschnittliche Zeitlimit 10 Runden. Die mittlere Quasi-Rente beträgt 61 GE auf dem Markt mit hohen Transaktionskosten bzw. 46 GE auf dem Markt mit niederen Transaktionskosten.

Deutliche Unterschiede zwischen dem Markt mit hohen und dem Markt mit niederen Transaktionskosten zeigen sich in der durchschnittlichen Verhandlungsdauer und der Preisspannweite.

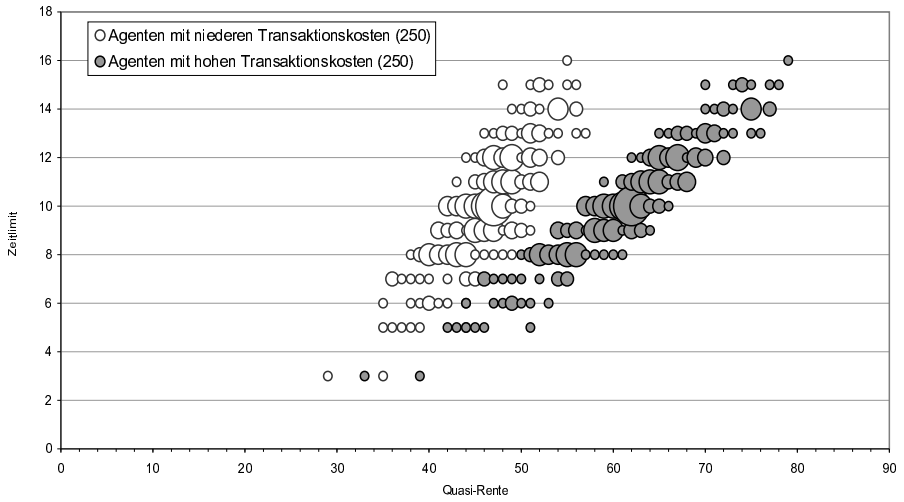

Abbildung 6.3: Population auf Markt mit niederen bzw. hohen Transaktionskosten

6.3.1 Verhandlungsdauer

Auf dem Markt mit hohen Transaktionskosten ist die Verhandlungsdauer kürzer. Abbildung 6.4 stellt die durchschnittlichen Verhandlungszeiten auf den beiden Märkten dar.

Zu Beginn der Simulation liegt die durchschnittliche Verhandlungsdauer auf dem Markt mit hohen Transaktionskosten noch über der durchschnittlichen Dauer am Markt mit niederen Transaktionskosten. Am Ende der Lernphase kristallisiert sich aber deutlich heraus, dass die Verhandlungsdauer bei niederen Transaktionskosten länger ist. Das Ansteigen der durchschnittlichen Verhandlungsdauer nach 400 Runden entsteht dadurch, weil die rasch verhandelnden Agenten den Markt bereits verlassen haben.

Die kürzeren Verhandlungszeiten kommen zum Teil durch weniger Verhandlungsabbrüche zustande. Abbildung 6.5 vergleicht die kumulierte Anzahl der Verhandlungsabbrüche.

Dabei zeigt sich deutlich, dass auf dem Markt mit hohen Transaktionskosten die Anzahl der Verhandlungsabbrüche geringer ist. Am Markt mit hohen Transaktionskosten gibt es in der stabilen Phase kaum noch Verhandlungsabbrüche. Am Markt mit niederen Transaktionskosten treten Verhandlungsabbrüche auch noch in der stabilen Phase auf, zu erkennen am Anstieg der oberen Kurve in Abbildung 6.5.

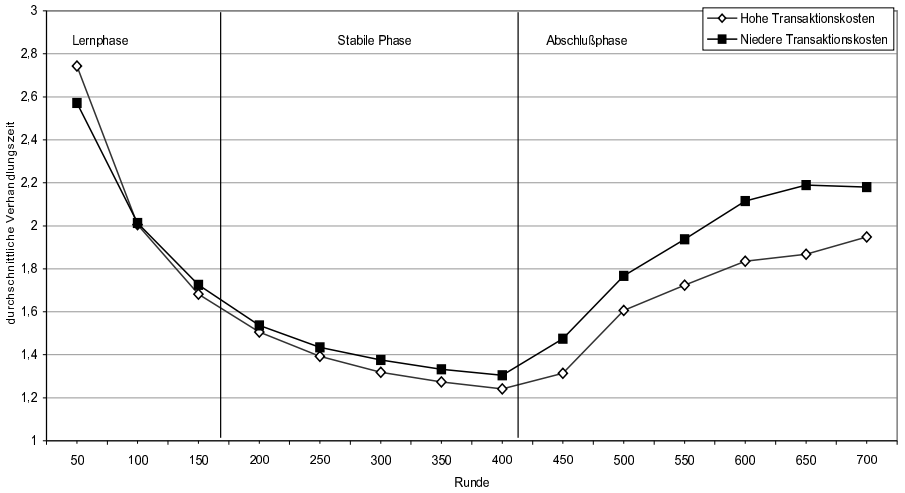

Abbildung 6.4: Verhandlungszeiten auf Märkten mit niederen bzw. hohen Transaktionskosten

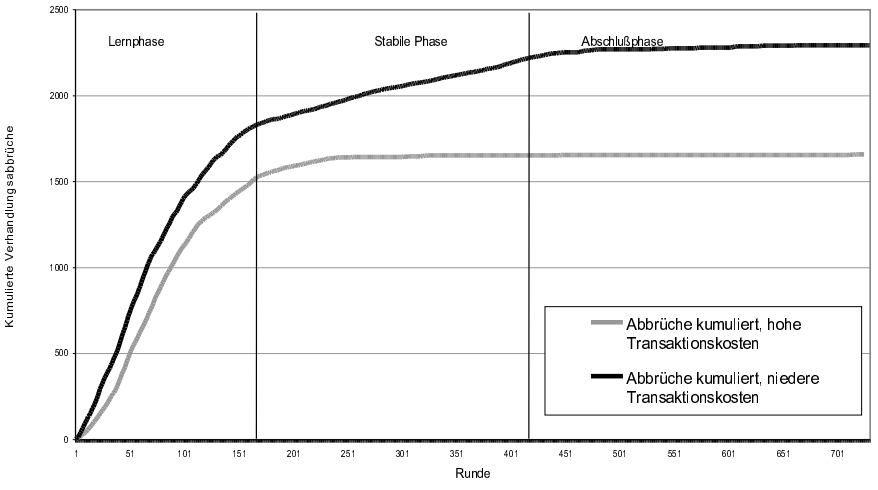

Abbildung 6.5: Kumulierte Verhandlungsabbrüche

6.3.2 Preisentwicklung

Abbildung 6.6 zeigt, dass die Preisunterschiede bei hohen Transaktionskosten deutlich größer ausfallen. Die Abbildung vergleicht einen gleitenden Mittelwert der Preisspannweite in Prozent des Marktpreises.

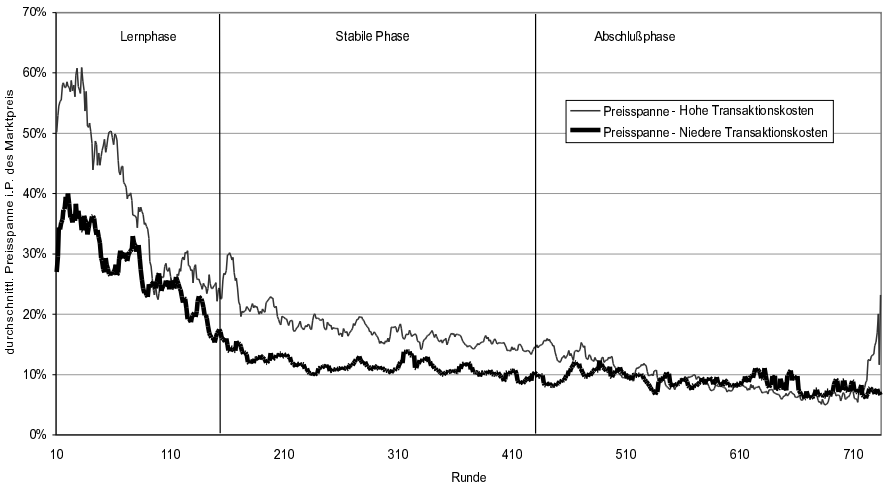

Abbildung 6.6: Gleitender Durchschnitt der Preisspannweite

Bei gleichem mittlerem Marktpreis ist die Streuung der Preise auf dem Markt mit hohen zeitabhängigen Transaktionskosten größer. Das muss schon alleine deshalb der Fall sein, weil auch die Streuung der Quasi-Renten in der Population der Marktteilnehmer größer ist.

6.3.3 Schlussfolgerung

Das Ergebnis lässt folgenden Schluss zu: Die Händler versuchen, die hohen zeitabhängigen Transaktionskosten durch rascheres Verhandeln zu kompensieren. Verhandlungen werden auf Märkten mit hohen Transaktionskosten seltener abgebrochen. Statt den Verhandlungspartner zu wechseln, wird ein schlechteres Verhandlungsergebnis akzeptiert.

Agenten auf Märkten mit niederen Transaktionskosten sind besser über den Markt informiert. Sie wechseln häufiger die Verhandlungspartner und lernen dadurch den Markt besser kennen.

Der Vergleich der beiden Simulationen bestätigt die in Kapitel 3 getroffenen Feststellungen: Die Preisbildung auf Märkten mit Transaktionskosten wird durch die Aufteilung der Quasi-Renten beeinflusst. Unterschiedliche Preise entstehen, weil die Marktteilnehmer einen schlechteren Preis in Kauf nehmen, um Such-,

Informations- und Verhandlungskosten zu sparen. Je höher die Transaktionskosten sind, umso weniger Aufwand wird der Informationssuche gewidmet. Die Folge davon sind größere Preisunterschiede.

Die Größe der Preisunterschiede auf einem Markt ist direkt proportional zur Höhe der Transaktionskosten.

6.4 Unterschiedliche Transaktionskosten auf einem Markt

In den beschriebenen Experimenten wurden die Händler in zwei Gruppen geteilt: Eine Gruppe mit hohen und eine Gruppe mit niederen zeitabhängigen Transaktionskosten.

In einer ersten Simulation beträgt die Gruppengröße der Agenten mit hohen Transaktionskosten auf beiden Marktseiten 50 bei jeweils insgesamt 250 Käufern bzw. Verkäufern. In einer weiteren Simulation beträgt die Gruppengröße der Händler mit hohen Transaktionskosten jeweils 200.

Die Abbildungen 6.7 und 6.8 zeigen die Population der Käufer. Die Population der Verkäufer ist wieder symmetrisch dazu aufgebaut.

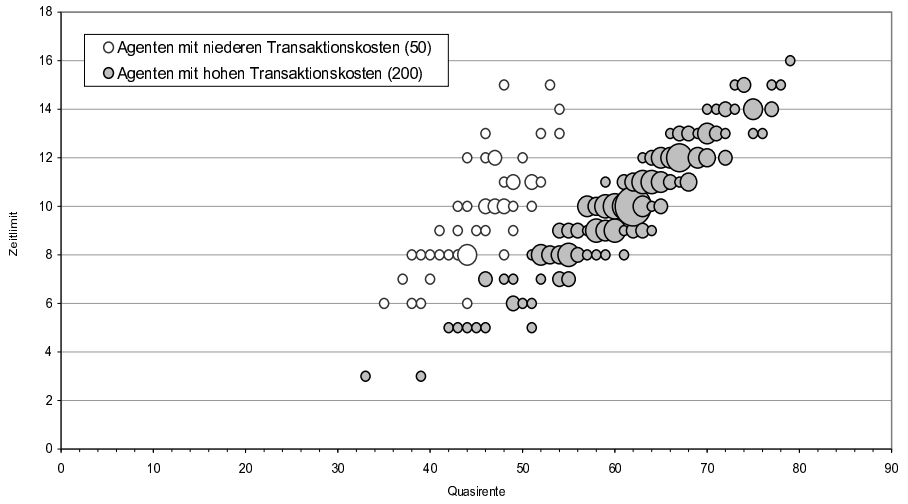

Abbildung 6.7: Marktteilnehmer, die Agenten mit niederen Transaktionskosten sind in der Minderheit

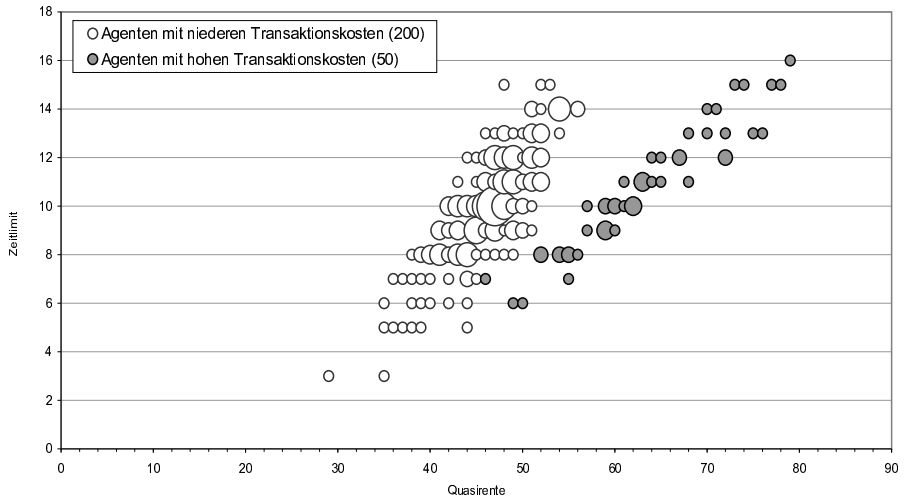

Abbildung 6.8: Marktteilnehmer, die Agenten mit hohen Transaktionskosten sind in der Minderheit

6.4.1 Verhandlungszeit

Wieder zeigt sich der schon in Abschnitt 6.3 beobachtete Verlauf der durchschnittlichen Verhandlungsdauer: Agenten mit höheren Transaktionskosten müssen zunächst längere Verhandlungsdauern in Kauf nehmen. Bereits nach 100 Runden liegt die Verhandlungsdauer der Händler mit hohen Transaktionskosten unter der der Händler mit niedrigen Transaktionskosten. In der stabilen Phase beträgt der Unterschied in den Verhandlungsdauern etwa 6%.

Abbildung 6.9 vergleicht die durchschnittlichen Verhandlungszeiten der Agenten mit hohen und niederen Transaktionskosten.

Die beiden Kurven stellen die Verhandlungszeit der Agenten mit hohen Transaktionskosten relativ zu der Dauer der Verhandlungen der Händler mit niederen Transaktionskosten dar. Die extremen unterschiede in den Verhandlungszeit zwischen Agenten mit hohen und niederen Transaktionskosten nach 400 Runden ist wegen der geringen Zahl der noch am Markt befindlichen Händler nicht schlüssig zu interpretieren.

6.4.2 Preisentwicklung

Die Entwicklung der Nominalpreise zeigt Unterschiede in Abhängigkeit der Mehrheitsverhältnisse auf dem Markt: Sind die Agenten mit niederen Transaktionskosten in der Minderheit, so können sie ihren Vorteil besser ausnutzen. Wenn Händler mit hohen Transaktionskosten in der Minderheit sind, so können sie die-

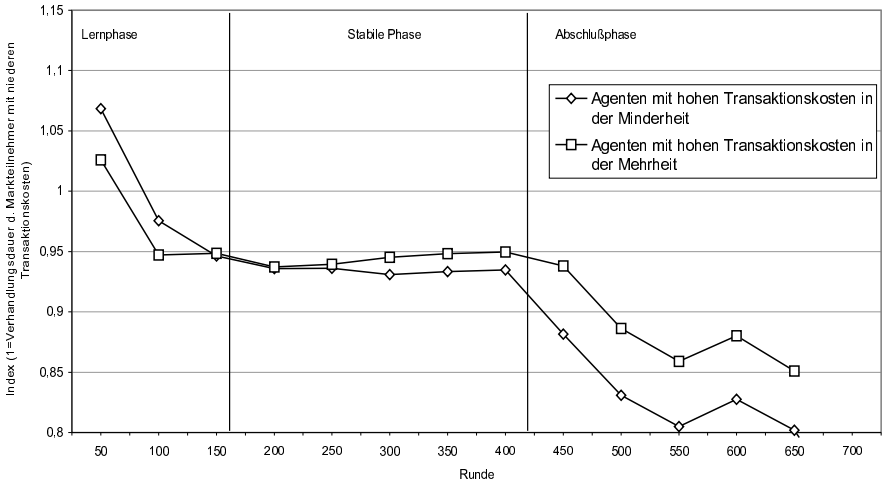

Abbildung 6.9: Relative Verhandlungszeiten der Agenten mit hohen Transaktionskosten

sen Nachteil besser kompensieren.

Abbildung 6.10 stellt die Preisentwicklung der Händler mit hohen Transaktionskosten relativ zu den Preisen der Agenten mit niederen Transaktionskosten dar. Dabei werden wieder zwei Märkte mit unterschiedlichen Mehrheitsverhältnissen verglichen.

Die Preise der wegen hoher Transaktionskosten benachteiligten Käufer liegen generell oberhalb der Preise der Käufer mit niederen Transaktionskosten. Im Falle der Verkäufer ergibt sich ein umgekehrtes Bild.

Vor allem zu Beginn der Simulation, also in der Phase, in der Markteilnehmer noch nicht viele Erfahrungen sammeln konnten, können die Agenten mit niederen Transaktionskosten ihre Vorteile am stärksten ausnutzen, wenn sie in der Minderheit sind. Sind jedoch die Agenten mit hohen Transaktionskosten in der Minderheit, so können sie diesen Umstand anfänglich vollkommen kompensieren. Erst mit fortschreitender Simulationsdauer entstehen in diesem Fall unterschiedliche Nominalpreise, die Unterschiede werden aber ab der Runde 100 wieder kleiner.

Für die Preisentwicklung nach 400 Runden gilt wieder, dass sie wegen der geringen Händlerzahlen nicht mehr repräsentativ ist.

6.4.3 Schlussfolgerung

Inwieweit Agent mit niederen Transaktionskosten ihren Vorteil ausnutzen können, hängt von den Mehrheitsverhältnissen am Markt ab, was sich an der Preisentwicklung erkennen lässt. Agenten mit niederen Transaktionskosten erzielen umso

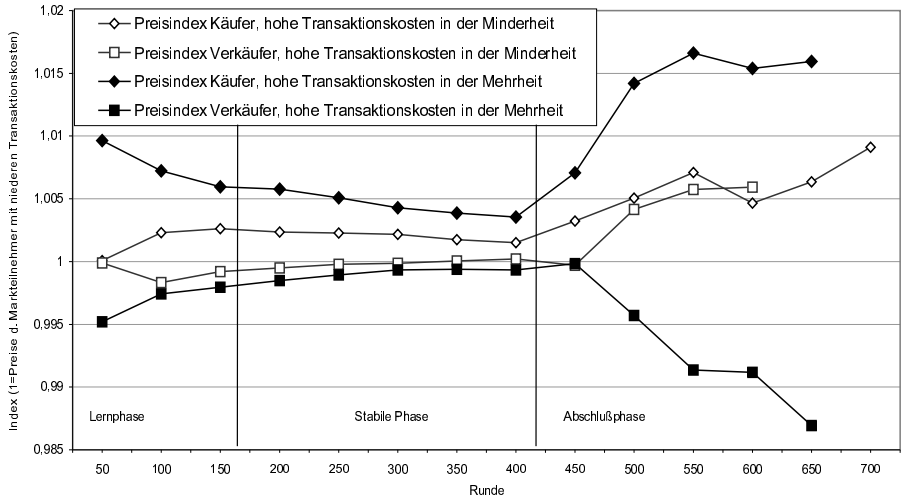

Abbildung 6.10: Relative Preise der Agenten mit hohen Transaktionskosten

bessere Verhandlungsergebnisse, je kleiner ihr Anteil an der Gesamtpopulation des Marktes ist. Dieser Vorteil wird noch verstärkt, wenn die Erfahrungen der Marktteilnehmer über andere Händler gering sind.

6.4.4 Effektivpreisentwicklung

Hinsichtlich der effektiven Preise müssen die Ergebnisse der Händler mit hohen Transaktionskosten schon per Definition schlechter sein als die der Agenten mit niederen Transaktionskosten.

Es ist zu beobachten, dass Agenten mit niederen Transaktionskosten durch längere Verhandlungen bessere Effektivpreise erzielen. Bei Agenten mit hohen Transaktionskosten wirken sich längere Verhandlungszeiten kaum positiv auf die Effektivpreise aus. Die besseren Nominalpreise bei längerer Verhandlungsdauer werden durch die höheren zeitabhängigen Transaktionskosten wett gemacht. Nur ein geringer Teil der Agenten mit hohen Transaktionskosten kann eine wesentliche Effektivpreisbesserung durch längeres Verhandeln erzielen. Diese Gruppe hat eine durchschnittliche Verhandlungszeit zwischen 2 und 2,5 Runden. Diese Agenten erreichen etwa die gleichen Effektivpreise wie Agenten mit niederen Transaktionskosten, deren durchschnittliche Verhandlungszeit zwischen 1,5 und 1,8 Runden liegt.

Abbildung 6.11 und 6.12 vergleichen Nominalpreise und Effektivpreise von Händlern mit den durchschnittlichen Verhandlungszeiten. Für dieses Simulation wurde die Population so eingeteilt, dass die Marktteilnehmer je zur Hälfte hohe

bzw. niedere Transaktionskosten haben. Die Ergebnisse sind Mittelwerte, die durch 90 Simulationsläufe bestimmt wurden.

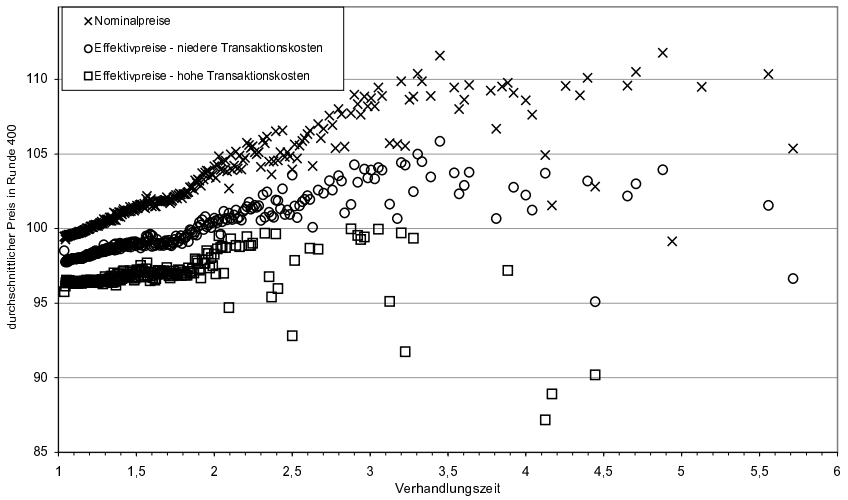

Abbildung 6.11: Preise der Verkäufer zur durchschnittlichen Verhandlungszeit nach 400 Runden

6.5 Zusammenfassung der Ergebnisse

Insgesamt ist zu beobachten, das die Nominalpreise nicht nur von den Transaktionskosten abhängig sind, sondern auch von der Verhandlungszeit. Agenten mit längerer Verhandlungszeit erzielen bessere Nominalpreise, was auch die in Kapitel 4 getroffenen Feststellungen hinsichtlich der Geduld der Händler bestätigt.

Agenten mit niederen zeitabhängigen Transaktionskosten können diesen Umstand auf zwei Arten zu ihren Gunsten ausnützen:

- Sie können mehr Zeit für die Suche und Information aufwenden. In der Naschmarkt-Simulation spiegelt sich dieser Umstand im Vergleich der Verhandlungsabbrüche wieder.
- Sie können ihre Geduld besser nutzen: Agenten mit niederen Transaktionskosten erzielen bessere Effektivpreise, wenn sie lange verhandeln.

Agenten mit hohen zeitabhängigen Transaktionskosten versuchen diesen Nachteil durch kürzere Verhandlungszeiten zu kompensieren. Das bringt mit sich, dass sie weniger Zeit für Suche und Information aufwenden, sie sind daher im Vergleich zu Agenten mit niederen Transaktionskosten schlechter informiert.

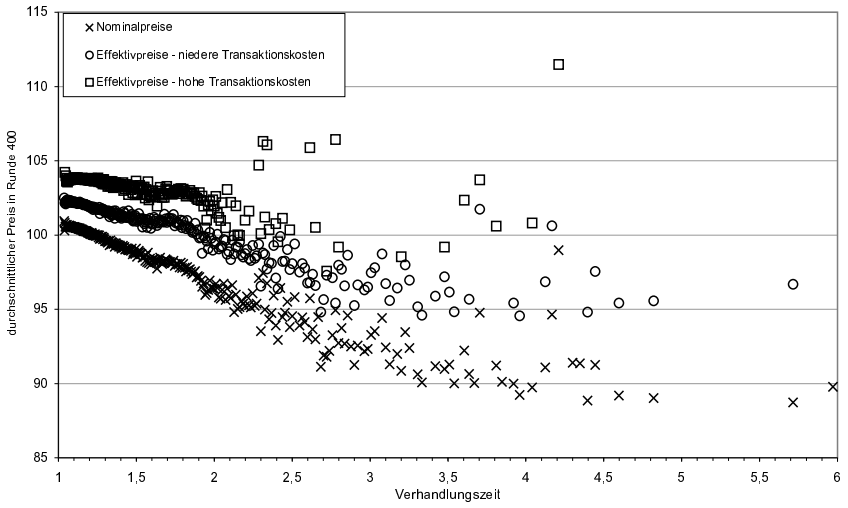

Abbildung 6.12: Preise der Käufer zur durchschnittlichen Verhandlungszeit nach 400 Runden

Auch wenn sie geduldig sind, also länger Zeit für den Abschluss einer Transaktion zur Verfügung haben, können sie diesen Umstand hinsichtlich der Effektivpreise nicht nutzen.

Misst man die Effizienz an den Kosten, die den Händlern durch die Teilnahme am Markt entstehen, so können folgende Aussagen getroffen werden:

- Kosten, die einzelnen Händlern dadurch entstehen, weil Händler schlecht informiert über den Markt sind, werden im Simulationsverlauf geringer. Diese Kosten drücken sich in den Preisspannweiten aus, die zu Beginn einer Simulation am größten sind und dann abnehmen. Hohe Preisspannweiten entstehen, weil schlecht informierte Händler bereit sind, einen ungünstigeren Preis zu akzeptieren, da sie nicht wissen, dass sie mit einem anderen Partner eine bessere Lösung erzielen können. Je länger die Simulation dauert, desto besser wird die Informationslage der einzelnen Händler über den Markt. Sind die Händler besser informiert, so werden sie ungünstige Preise nicht mehr annehmen, weil sie damit rechnen können, eine bessere Lösung mit einem anderen Partner zu erzielen. Die Preisspannweiten nehmen daher ab.

- Die Verhandlungszeiten sinken und mit ihnen die Transaktionskosten. Je rascher eine Lösung erreicht werden kann, umso weniger Transaktionskosten müssen dafür aufgewandt werden. Da die durchschnittliche Verhandlungszeit sich in allen Simulationen einer Runde annähert, können Kosten gespart werden.

6.6 Verhandlungsbeispiele

Um einen kurzen Einblick in den Verlauf von Verhandlungen zwischen den Naschmarkt-Agenten zu geben, werden zwei Verhandlungsbeispiele näher beschrieben.

Obwohl die meisten Verhandlungen relativ rasch nach einer oder zwei Runden abgeschlossen werden können, finden sich immer wieder länger andauernde Verhandlungen.

Abbildung 6.13 zeigt eine über 4 Runden andauernde Verhandlung, in der der Verkäufer beginnt.

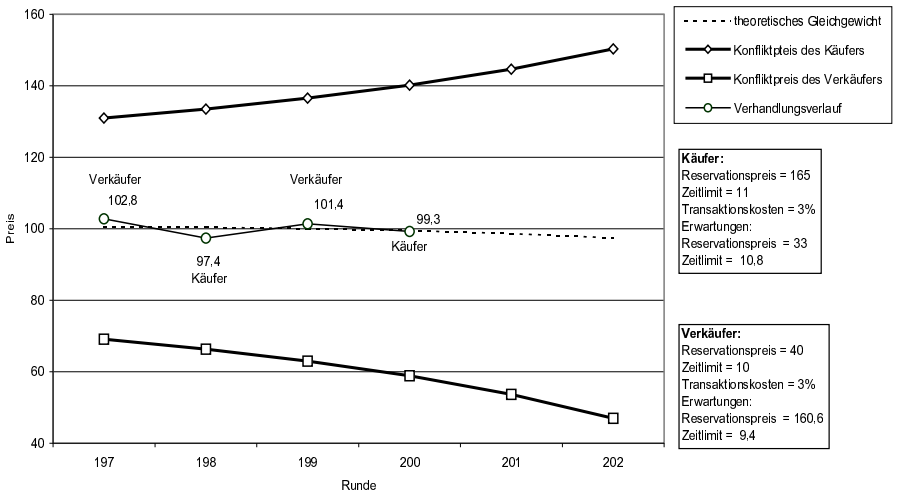

Abbildung 6.13: Beispiel einer Verhandlung, in der sich beide Händler entgegen kommen

Der Käufer unterschätzt zu Beginn der Verhandlung den Reservationspreis des Verkäufers. Der Verkäufer unterschätzt das Zeitlimit des Käufers.

Beide Agenten geben von Runde zu Runde in ihren Preisforderungen nach, sie handeln kooperativ. Nach 4 Runden einigen sie sich auf einen Preis, der dem theoretischen teilspielperfektem Gleichgewicht entspricht.

Wie sehr der Verhandlungsverlauf von den Erwartungen der Agenten geprägt ist, zeigt das Beispiel aus Abbildung 6.14.

Der Käufer überschätzt vor allem den Reservationspreis des Verkäufers. Der Verkäufer überschätzt den Reservationspreis des Käufers und unterschätzt das Zeitlimit. Das hat zur Folge, dass alle Angebote in der Verhandlung über dem theoretischen teilspielperfektem Gleichgewicht liegen. Letztendlich überzahlt der Käufer den Verkäufer um nahezu 20 Geldeinheiten.

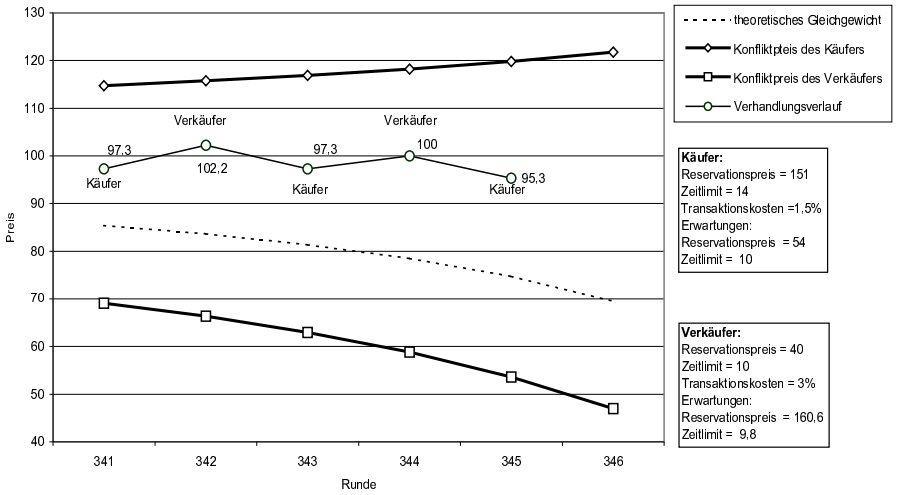

Abbildung 6.14: Beispiel einer Verhandlung, in der sich beide Händler falsch einschätzen

Weiters zeigt dieses Beispiel, dass Verhandlungen zwischen Naschmarkt-Agenten nicht immer durch wechselseitiges Entgegenkommen gekennzeichnet sein müssen. Der Käufer verfolgt hier eine Verhandlungsstrategie, die man, an menschlichen Maßstäben gemessen, als hart bezeichnen könnte. Er weicht zunächst nicht von seiner Preisforderung ab, um in der letzten Runde einen noch niederen Preis zu fordern, womit er letztendlich erfolgreich ist. Dazu ist er in der Lage, weil er die Geduld des Verkäufers richtig einschätzt und selber ein Zeitlimit von 14 Runden hat. Diesen Vorteil kann er hier ausnützen.

Kapitel 7

Zusammenfassung

Am Ende dieser Arbeit soll, nachdem die Ergebnisse verschiedener Simulationen analysiert wurden, die Beurteilung aus einer eher praxisorientierten Sichtweise stehen. Außerdem werden Probleme aufgezeigt, die während der Entwicklung des Simulationsmodells auftraten.

7.1 Spieltheorie als Lösungsansatz

Die Spieltheorie bietet einen günstigen Weg, Verhandlungsprobleme einzuordnen und Lösungskonzepte zu entwickeln. Werden solche Ansätze in die Praxis übertragen, treten dabei jedoch zwei Probleme auf:

- Die Spieltheorie geht von sehr einschränkenden Annahmen aus. Um Lösungskonzepte zu entwickeln, muss zunächst das zu untersuchende Spiel exakt beschrieben werden. Spielregeln und Informationsstände sind genau festgelegt. Eine Veränderung der einem Spiel zugrunde liegenden Annahmen konstituiert unweigerlich ein anderes Spiel, das andere Lösungskonzepte fordert. In der Praxis sind aber nicht alle Verhandlungsprobleme gleich, weil Verhandlungspartner unterschiedliche Informationen besitzen oder Mechanismen divergieren. Dem ist entgegenzuhalten, dass künstliche Märkte jedoch die Möglichkeit bieten, die Verhandlungsmechanismen zu standardisieren, wie es zum Teil bei Auktionshäusern (siehe auch Kapitel 2) oder Börsen der Fall ist.

  Die European Energy Exchange normiert (http://www.eex.de) vier verschiedene Orderformen, wobei grundsätzlich limitierte und nicht limitierte Orders möglich sind. Der Markt ist in drei Handelsphasen gegliedert, die Vorhandelsphase, Haupthandelsphase und die Nachhandelsphase.

  Die Regeln der Preisbildung sind ebenfalls klar definiert. Grundsätzlich

werden die Orders zunächst nach Preis und nach Zeit gereiht. Der Auktionspreis ist derjenige Preis, zu dem das höchste ausführbare Ordervolumen und der niedrigste Überhang je im Orderbuch vorhandenem Limit besteht.

Solche klar definierten Regeln würden es erleichtern, Softwareagenten mit dem Handel zu betrauen.

- Lösungskonzepte sind in der Sprache der Mathematik formuliert. Wie sich auch in dieser Arbeit zeigt, ist der Umgang mit der Mathematik ein unumgänglicher Bestandteil bei der Ableitung von Lösungskonzepten. Da aber diese Sprache nicht für jedermann transparent und verständlich ist, stellt sich die Frage, ob Anwender genügend Vertrauen in einen Softwareagenten haben, um ihn mit Verhandlungsaufgaben zu beauftragen, wenn sie nicht verstehen, wie er arbeitet. Selbst wenn diese Agenten plausible Ergebnisse liefern, könnte ein Akzeptanzproblem deren Einsatz im Wege stehen.

Trotz dieser Schwierigkeiten sollten spieltheoretische Erkenntnisse in die Entwicklung künstlicher Märkte einfließen, um einen systematischen und fundierten Zugang zur Problemstellung zu ermöglichen. Vergleicht man den hier beschriebenen Lösungsweg mit dem Ansatz, den das Kasbha-Projekt [Maes and Chaves, 1996] wählt, so zeigt sich, dass für zwei Schwächen der Kasbha-Agenten ein Lösungsansatz gefunden wurde:

- Im Vergleich zu den Kasbha-Agenten handeln die Naschmarkt-Agenten selbstständiger. Die Kasbha-Agenten arbeiten mit einer vom Anwender spezifizierten Preisänderungsfunktionen. Die Verhandlungsstrategie ist damit vom Anwender fest vorgegeben. Die Naschmarkt-Agenten können in jeder Verhandlungsrunde autonom über den gebotenen Preis entscheiden. Dabei orientieren sie sich an den auf dem Markt gesammelten Informationen und am Verhalten des Verhandlungspartners.

  Die in Abschnitt 6.6 dargestellten Verhandlungsbeispiele zeigen, dass Verhandlungen zwischen den Naschmarkt-Agenten sehr unterschiedlich ablaufen können. Es ist nicht zu erwarten, dass ein Agent automatisch in seinen Preisforderungen nachgibt. Auch grobe Fehleinschätzungen können den Ausgang der Verhandlung prägen. Sofern menschliches Handeln als Maßstab verwendet werden soll, kann gesagt werden, dass solche Verhandlungsverläufe durchaus auch zwischen Menschen zu beobachten sind.

- Die Parametrisierung erfolgt bei den Kasbha-Agenten durch Wunschpreis, Höchst- bzw. Tiefstpreis und Zeitlimit. Gerade die Angabe eines Wunschpreises scheint für den Anwender oft schwierig. Insofern sind die

Naschmarkt-Agenten einfacher bedienbar, da ein Wunschpreis nicht angegeben werden muss. Die Naschmarkt-Agenten orientieren sich statt dessen am vorherrschenden Marktpreis.

7.2 Lernen nach der bayesschen Regel

Der Satz von Bayes stellt eine mathematisch haltbare Lernmethode basierend auf der Wahrscheinlichkeitsrechnung dar. Es ist aber nicht anzunehmen, dass menschliche Lernprozesse nach exakten mathematischen Modellen ablaufen. Auch die Vielzahl unterschiedlicher Wahrnehmungen, die ein Mensch zur Meinungsbildung verwendet, kann schwer in einem formalen Modell berücksichtigt werden.

Ein Kernstück der verwendeten Lernmethode ist die Likelihood-Funktion. Die hier verwendete Methode der Entwicklung einer Likelihood-Funktion ist zumindest konsistent mit den getroffenen Modellannahmen. Jedoch ist nicht auszuschließen, dass andere Funktionen bessere Ergebnisse liefern.

Vor allem die Frage, was ein Agent zu einem bestimmten Zeitpunkt lernen soll, lässt sich nur äußerst schwer beantworten. Dabei stehen mehrere Möglichkeiten zur Auswahl: Ein Agent kann lernen, das die Quasi-Rente des Partners höher, nieder oder gleich ist, das gleiche gilt für das erwartete Zeitlimit. Diese Varianten können miteinander kombiniert werden, so dass sich insgesamt neun Möglichkeiten ergeben.

Die Auswahl des in Kapitel 5 beschriebenen Verfahrens erfolgte unter zwei Gesichtspunkten:

- Zum einen darf die Entscheidung, ob die Erwartung über die Quasi-Rente oder das Zeitlimit zu revidieren ist, keinem Muster folgen (beispielsweise alternierend Quasi-Rente und Zeitlimit). Eine solche Regelmäßigkeit wäre Teil der Spielregeln und würde das Lösungskonzept des Verhandlungsspiels erheblich verkomplizieren.

- Andererseits müssen die Entscheidungen anhand eines plausiblen Kriteriums getroffen werden, wenn man die Annahme des rationalen Verhaltens nicht verletzen möchte.

Obwohl das Resultat dieser Bemühungen plausible Ergebnisse liefert, wäre eine Leistungssteigerung der Agenten nicht auszuschließen, würden diese starren Lernregeln durch flexiblere Methoden ersetzt. Beispielsweise könnten diese Entscheidungen durch einen genetischen Algorithmus getroffen werden. Denkbar wäre auch, die Muster von Angebotsverläufen durch neuronale Netze zu klassifizieren um daraus auf den Typ des Verhandlungspartners zu schließen.

7.3 Einsatzmöglichkeiten

Schon anhand der Literatur und anderer, vergleichbarer Projekte zeigt sich, dass die Entwicklung auf dem Gebiet der automatisierten Verhandlungen erst am Beginn steht. Auch zahlreiche damit in Zusammenhang stehende Probleme, allen voran die Nutzenmodellierung, stellen große Hemmnisse für den kommerziellen Einsatz dar.

Da bei vielen Gütern zahlreiche Qualitätsmerkmale bewertet werden müssen und menschliche Präferenzordnungen oft nicht transitiv sind, lassen sich maschinell verarbeitbare Bewertungsskalen nur schwer konstruieren. Auch bei dieser Arbeit musste auf die Annahme homogener Güter zurückgegriffen werden, selbst der Handel mit unterschiedlich großen Mengen wurde ausgeschlossen.

Für ein gemeinsames Verständigungsprotokoll zwischen softwareagenten zeichnen sich Lösungsmöglichkeiten ab: Für den Datenaustausch zwischen Unternehmen sind zum Teil schon Standards wie die EDI-Protokolle [Krcmar, 1995] geschaffen, außerdem gibt es mit KIF [Genesereth and Fikes, 1992] und KQML [Finin *et al.*, 1994] Protokolle für den Informationsaustausch zwischen Agenten. Für zahlreiche Güter existieren außerdem detaillierte Spezifikationen durch Industrienormen wie die DIN.

Die Verbreitung von digitalen Signaturen erleichtert die Authentifizierung und Identifikation von über das Internet in Verbindung tretenden Parteien, so dass auf Basis dieser Technologie auch wirksame Reputationsmechanismen geschaffen werden könnten.

Die aufgezeigten Probleme lassen jedoch erkennen, dass die Vision, einem Softwareagenten einen Kaufauftrag für eine CD oder einen Automobilersatzteil zu erteilen, den dieser dann selbständig durchführt und abwickelt, derzeit noch nicht umsetzbar ist.

Literaturverzeichnis

[Antelman, 1997] Gordon Antelman. *Elementary Bayesian Statistics*. Edward Elgar Publishing Limited, Cheltenham, UK, 1997.

[Beam and Segev, 1996] Carrie Beam and Arie Segev. *Electronic Catalogs and Negotiations*. CITM Working Paper 96-WP-1016. The Fisher Center for Management and Information Technology, Berkely, August 1996.

[Beam and Segev, 1997] Carrie Beam and Arie Segev. Automated negotiations: a survey of the state of the art. *Wirtschaftsinformatik*, 39(3):263–268, 1997. WI-Schwerpunktaufsatz.

[Beam and Segev, 1998] Carrie Beam and Arie Segev. *Auctions on the Internet: A Field Study*. CITM Working Paper 98-WP-1032. The Fisher Center for Management and Information Technology, Berkely, November 1998.

[Beam *et al.*, 1997] Carrie Beam, Arie Segev, and George Shanthikumar. *Electronic Negotiation through Internet-Based Auctions*. CITM Working Paper 97-WP-1022. The Fisher Center for Management and Information Technology, Berkely, Mai 1997.

[Bichler *et al.*, 1999] Martin Bichler, Marion Kaukal, and Arie Segev. Multi-attribute auctions for electronic procurement. In *Proceedings of the First IBM IAC Workshop on Internet Based Negotiation Technologies*, Yorktown Heights, NY, März 1999.

[Binmore and Vulkan, 1997] Ken Binmore and Nir Vulkan. Applying game theory to automated negotiation, 1997. DIMACS Workshop on Economics, Game Theory and the Internet, Rutgers University, New Brunswick, New Jersey.

[Biswas, 1997] Tapan Biswas. *Decision-making under uncertainty*. Macmillan, Basingstoke, 1997.

[Brenner *et al.*, 1998] Walter Brenner, Rüdiger Zarnekow, and Hartmut Wittig. *Intelligent Sortware Agents*. Springer, Berlin Heidelberg New York, 1998.

[Brösse, 1999] Ulrich Brösse. *Einführung in die Volkswirtschaftslehre Mikroökonomie*. Oldenburg Verlag, München, 3 edition, 1999.

[Cheong, 1996] Fah-Chun Cheong. *Internet Agents: Spiders, Wanderers, Brokers, and Bots*. New Riders Publishing, Indianapolis, 1996.

[Feess, 1997] Eberhard Feess. *Mikroökonomie: Eine spieltheoretische Einführung*. Metropolis-Verlag, Marburg, 1997.

[Finin *et al.*, 1994] T. Finin, R. Fritzson, D. McKay, and R. McEntire. Kqml as an agent communication language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM94)*, pages 456–463, Gaithersburg, Maryland, 1994. ACM Press.

[Genesereth and Fikes, 1992] M. R. Genesereth and R. E. Fikes. Knowledge interchange format, version 3.0 reference manual, 1992.

[Goldberg *et al.*, 1992] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[Gruber, 1993] Tom R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5/2, 1993.

[Guttman *et al.*, 1998] R. Guttman, A. Moukas, and P. Maes. Agent-mediated electronic commerce: A survey. *Knowledge Engineering Review*, 1998.

[Harsanyi, 1967] J.C. Harsanyi. Games with incomplete information played by bayesian players. *Management Science*, 14, 8 1967.

[Heike and Tarcdea, 2000] Hans Dieter Heike and Constantin Tarcdea. *Grundlagen der Statistik und Wahrscheinlichkeitsrechnung*. Oldenburg Verlag, München, Wien, 2000.

[Holler and Illing, 1996] Manfred Holler and Illing. *Einführung in die Spieltheorie*. Springer Verlag, Berlin, 1996.

[Holler, 1992] Manfred Holler. *Ökonomische Theorie der Verhandlungen*. Oldenburg Verlag, Wien, München, 1992.

[Kersten and Noronha, 1999] Gregory E. Kersten and Sunil J. Noronha. Www-based negotiation support: Design, implementation and use. *Decision Support Systems*, 25(2), März 1999.

[Krcmar, 1995] Helmut Krcmar. *EDI in Europe*. Wiley, Chichester, 1995.

[Kreps, 1990] David M. Kreps. *Microeconomic Theory*. Harvester Wheatsheaf, New York, 1990.

[Kuhlins and Schader, 1999] Stefan Kuhlins and Martin Schader. *Die C++ Standardbibliothek*. Springer Verlag, Berlin, Heidelberg, New York, 1999.

[Lechner *et al.*, 1994] Karl Lechner, Anton Egger, and Reinbert Schauer. *Einführung in die Allgemeine Betriebswirtschaftslehre*. Linde Verlag, Wien, 15 edition, 1994.

[Lucking-Reiley, 1999] David Lucking-Reiley. Using field experiments to test equivalence between auction formats: Magic on the internet. *The American Economic Review*, 89(5), Dezember 1999.

[Maes and Chaves, 1996] Pattie Maes and Anthony Chaves. Kasbah: An agent marketplace for buying and selling goods, April 1996. Proceedings of th First International Conference on Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96).London, UK.

[Maes *et al.*, 1999] Pattie Maes, Robert Guttman, and alexandros Moukas. Agents that buy and sell: Transforming commerce as we know it. *Communications of the ACM*, 42:81 – 91, März 1999.

[Mansfield, 1996] Edwin Mansfield. *Microeconomics*. W. W. Norton, New York, 9 edition, 1996.

[Oestereich, 1999] Bernd Oestereich. *Objektorientierte Softwareentwicklung*. Oldenbourg Wissenschaftsverlag, 1999.

[Oliver, 1996] J. R. Oliver. A machine learning approach to automated negotiation and prospects for electronic commerce. *Journal of Management Information Systems*, 13/3, 1996.

[Osborne and Rubinstein, 1990] Martin J. Osborne and Ariel Rubinstein. *Bargaining and Markets*. Academic Press, Inc., San Diego, 1990.

[Otruba *et al.*, 1996] Heinrich Otruba, Gerhard Munduch, and Alfred Stiassny. *Makroökonomik*. Springer Verlag, Wien, New York, 2 edition, 1996.

[Pindyck and Rubinfeld, 1998] Robert S. Pindyck and David L. Rubinfeld. *Mikroökonomie*. Oldenburg Verlag, München, 1998.

[Richter and Furubon, 1996] Rudolf Richter and Eirik Furubon. *Neue Institutionenökonomik: Eine Einführung und kritische Würdigung*. Mohr, Tübingen, 1996.

[Rieck, 1993] Christian Rieck. *Spieltheorie Einführung für Wirtschafts- und Sozialwissenschafter*. Gabler, Wiesbanden, 1993.

[Riley and Samuelson, 1981] John G. Riley and William F. Samuelson. Optimal auctions. *American Economic Review*, 71/3, 1981.

[Russell and Norvig, 1995] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Upper Saddle River, 1995.

[Samuelson and Nordhaus, 1992] Paul A. Samuelson and William D. Nordhaus. *Economics*. McGraw-Hill, Inc., New York, 14 edition, 1992.

[Sangalli, 1998] Arturo Sangalli. *The Importance of being Fuzzy*. Princeton University Press, Princeton, New Jersey, 1998.

[Schaub, 1997] Harald Schaub. *Sunk Costs, Rationalität und ökonomische Theorie*. Schäffer-Poeschl-Verlag, Stuttgart, 1997.

[Schuhmann *et al.*, 1999] Jochen Schuhmann, Ulrich Meyer, and Wolfgang Ströbele. *Grundzüge der mikroökonomischen Theorie*. Springer Verlag, Berlin, 1999.

[Williamson, 1990] Oliver E. Williamson. *Die ökonomischen Institutionen des Kapitalismus*. Mohr, Tüebingen, 1990.

[Willms, 1997] Andre Willms. *C++ Programmierung*. Addison-Wesley, Bonn, 1997.

[Willms, 1999] Gerhard Willms. *C++ - Das Grundlagenbuch*. Data Becker, Düsseldorf, 1999.

[Zeng and Sycara, 1998] Dajun Zeng and Katia Sycara. Bayesian learning in negotiation. *International Journal of Human-Computer Studies*, Oktober 1998.

Anhang A

Benutzerhinweise zu NASCHMARKT

Die Marktsimulation Naschmarkt erzeugt je 250 Käufer und Verkäufer, je 249 davon werden durch einen Zufallsgenerator initialisiert. Die Eigenschaften des 250. Verkäufers werden vom Anwender festgelegt. Dieser Verkäufer wird über die gesamte Marktperiode hinweg beobachtet, er ist im Folgenden als Trace-Agent bezeichnet. Um keine systematische Verzerrung zu erzeugen, wird ein von den Eigenschaften her zu diesem 250. Verkäufer symetrischer 250. Käufer automatisch erzeugt.

Die Agenten werden in zwei Gruppen eingeteilt, eine Gruppe hat geringe zeitabhängige Transaktionskosten (1,5%), die andere Gruppe hohe zeitabhängige Transaktionskosten (3%). Die Gruppengröße kann vom Benutzer eingestellt werden.

Der Marktpreis wird zu Beginn der Simulation mit 100 GE festgelegt.

A.1 Simulationseinstellungen

Zunächst müssen die Aufträge für Käufer und Verkäufer festgelegt werden. Alle Käufer und Verkäufer müssen eine bestimmte Anzahl an Gütern kaufen bzw. verkaufen (z.B. 400/400).

Danach wird die Gruppengröße der Käufer mit hohen zeitabhängigen Transaktionskosten festgelegt.

Jetzt werden die Kaufagenten parametrisiert. Für jeden Parameter wird ein Mittelwert festgelegt, anhand dessen ein Zufallsgenerator binomialverteilte Zahlen erzeugt, deren Verteilung einen Mittelwert hat, der dem angegebenen Parameter entspricht.

Tabelle A.1: Beispiel - Parameter der Käufer

Parameter des Agenten	Wert
Zeitlimit	12
Reservationspreis	150
erwartetes Zeitlimit	12
erwarteter Reservationspreis	50

Tabelle A.2: Beispiel - Parameter der Verkäufer

Parameter des Agenten	Wert
Zeitlimit	12
Reservationspreis	50
erwartetes Zeitlimit	12
erwarteter Reservationspreis	150

Zeitlimit: Durchschnittliche maximale Verhandlungszeit pro Transaktion (min. 5 Runden, max. 20 Runden)

Reservationspreis: Der Reservationspreis beinhaltet die zeitabhängigen Transaktionskosten, so dass sich ein Mindestreservationspreis in Abhängigkeit vom Zeitlimit ergibt. Eine Mindestrisikoprämie von 2 wird ebenfalls berücksichtigt.

Dann werden die Erwartungen der Käufer über die Verkäufer festgelegt, dabei gilt für Zeitlimit und Reservationspreis oben erwähntes.

In gleicher Weise werden anschließend die Parameter der Verkäufer bestimmt. Zunächst wird wieder die Gruppengröße der Agenten mit hohen zeitabhängigen Transaktionskosten festgelegt, im Anschluss die eigenen Parameter der Verkäufer und dann die Erwartungen der Verkäufer über die Käufer.

Die Parameter sind folgender Maßen zu interpretieren: Jeder Agent hat durchschnittlich 12 Verhandlungsrunden Zeit, eine Transaktion abzuschließen. Ein durchschnittlicher Agent mit hohen zeitabhängigen Transaktionskosten muss mit Transaktionskosten in der Höhe von

$$36 = 12 \times 100 \times 0,03$$

Tabelle A.3: Beispiel - Parameter des Trace-Agenten

Parameter des Agenten	Wert
Hohe/Niedere Transaktionskosten	H
Zeitlimit	12
Reservationspreis	150
erwartetes Zeitlimit	12
erwarteter Reservationspreis	50

rechnen. Die durchschnittliche Risikoprämie ist dann

$$24 = 50 - 36$$

Bei den Agenten mit niederen Transaktionskosten ist die durchschnittliche Risikoprämie gleich. Einen niedere durchschnittliche Quasi-Rente ergibt sich wegen der geringeren zeitabhängigen Transaktionskosten. In diesem Beispiel gilt für Agenten mit niederen Transaktionskosten im Durchschnitt eine Quasi-Rente von

$$42 = 12 \times 100 \times 0,015 + 24$$

Zuletzt wird der Trace-Agent parametrisiert. Zunächst wird festgelegt, ob der Trace-Agent hohe oder niedere zeitabhängige Transaktionskosten besitzt. Danach werden die Parameter wie oben beschrieben eingegeben.

Jetzt startet die Simulation, die Ergebnisse werden in folgenden Dateien protokolliert:

mktdat.txt: Daten einer Verhandlungsrunde, werden auch am Bildschirm ausgegeben.

agtdat.txt: Zu Beginn der Simulation und alle 50 Runden werden die Daten aller Agenten protokolliert.

trcdat.txt: Protokolliert die einzelnen Verhandlungen des Trace-Agenten und seiner Verhandlungspartner, inklusive der Agenten-Daten.

Die Dateien enthalten Einträge im Textformat, durch Semikolon (;) getrennt. Bei numerischen Einträgen wird der Punkt als Dezimaltrennzeichen verwendet, Tausendertrennzeichen kommen nicht vor. Die Daten können in ein Tabellenverarbeitungsprogramm (z.B. MS-Excel) importiert und dort ausgewertet werden.

A.2 Erläuterung der Dateien

Im folgenden Abschnitt wird der Inhalt der Dateien beschrieben. Die Reihenfolge der Aufzählungen entspricht der Reihenfolge der Spalten innerhalb der Dateien.

A.2.1 Datei mktdat.txt

Die Datei mktdat.txt enthält Informationen über das Geschehen in jeder einzelnen Marktrunde. Diese Daten werden während der Simulation auch am Bildschirm angezeigt. Die Nummerierung entspricht den Spaltennummern.

1. Runde der Simulation, beginnend mit 1
2. Vorherrschender Marktpreis, arithmetisches Mittel aller in einer Runde zustande gekommener Preise
3. Kleinster Preis, der in dieser Runde zustande gekommen ist
4. Größter Preis in dieser Runde
5. Anzahl der Käufer, die ihren Auftrag noch nicht erfüllt haben und somit am Markt sind *)
6. Anzahl der Verkäufer, die ihren Auftrag noch nicht erfüllt haben und somit am Markt sind *)
7. Anzahl der in einer Runde zustande gekommenen Transaktionen (Vertragsabschlüsse)
8. Differenz zwischen all jenen Käufern und Verkäufern, die einem Vertragsabschluss in dieser Runde zugestimmt haben
9. Anzahl der Agenten, die Verhandlungen abgebrochen haben, weil die max. Verhandlungszeit zu Ende war
10. Anzahl der Käufer, die Verhandlungen abgebrochen haben, weil ihr Partner den Konfliktpreis überschritten hat
11. Anzahl der Verkäufer, die wegen einer Konfliktpreisüberschreitung abgebrochen haben und sich einen neuen Partner suchen
12. Anzahl der Agenten, die Verhandlungen abbrechen mussten, weil sie erwarten, dass die max. Verhandlungszeit ihres Partners zu verstrichen ist.

13. Anzahl der Abbrüche, weil das Gebot des Verhandlungspartner den Reservationspreis überschritten hat
14. Durchschnittliche Erwartung der Käufer über die Quasi-Rente der Verkäufer
15. Durchschnittliche Erwartung der Verkäufer über die Quasi-Rente der Käufer
16. Durchschnittliche Erwartung der Käufer über das Zeitlimit der Verkäufer
17. Durchschnittliche Erwartung der Verkäufer über das Zeitlimit der Käufer

*) Am Bildschirm wird nur die Differenz zwischen Käufer und Verkäufern angezeigt, um Platz zu sparen.

A.2.2 Datei agtdat.txt

Diese Datei protokolliert die Daten aller Agenten zu Beginn der Simulation sowie alle 50 Simulationsrunden.

1. Status

   NiM: Agent ist nicht mehr im Markt, weil er Auftrag erfüllt hat

   iV: Agent verhandelt zur Zeit

   NMat: Not matched, Agent konnte keinen Partner finden
2. In welcher Runde ist der Agent
3. Käufer oder Verkäufer
4. Identifikationsnummer des Agenten
5. Eigene Quasi-Rente
6. Eigenes Zeitlimit
7. Erwartete Quasi-Rente
8. Erwartetes Zeitlimit
9. Quasi-Rente, die Agent vorgibt zu haben
10. Zeitlimit, das Agent vorgibt zu haben

11. Verhandlungszeit, die sich ein Agent erwartet (Initialwert = 3)
12. Gesamte Verweildauer am Markt (wenn Auftrag erfüllt wurde, wird die Zeit nicht mehr weiter gezählt)
13. Summe aller bis jetzt erzielter nomineller Preise (ohne Transaktionskosten)
14. Summe aller bis jetzt erzielter effektiver Preise
15. Transaktionen, die bis jetzt abgeschlossen wurden

A.2.3 Datei trcdat.txt

Die in dieser Datei enthaltenen Daten zeigen die Daten des Trace-Agenten sowie seiner Verhandlungspartner. Anhand dieser Daten können alle Verhandlungen des Trace-Agenten verfolgt werden.

1. Simulationsrunde
2. Aktueller vorherrschender Marktpreis
3. Aktuelles Gebot in der Verhandlung
4. Erläutert aktuelles Gebot:

   0: Preis ist ein Angebot

   1: Abbruch, weil Zeitlimit erreicht

   2: Preis wird angenommen, Verhandlung abgeschlossen

   3: Abbruch, weil Konfliktpreis überschritten wurde

   4: Abbruch, weil Zeitlimit des Partners falsch eingeschätzt

   5 Abbruch, weil Reservationspreis überschritten
5. Käufer oder Verkäufer
6. Identifikationsnummer des Agenten
7. Eigene Quasi-Rente
8. Eigenes Zeitlimit
9. Erwartete Quasi-Rente
10. Erwartetes Zeitlimit

11. Quasi-Rente, die Agent vorgibt zu haben
12. Zeitlimit, das Agent vorgibt zu haben
13. Verhandlungszeit, die sich ein Agent erwartet (Initialwert 3)
14. Gesamte Verweildauer am Markt (wenn Auftrag erfüllt wurde, wird die Zeit nicht mehr weiter gezählt)
15. Summe aller bis jetzt erzielter nomineller Preise (ohne Transaktionskosten)
16. Summe aller bis jetzt erzielter effektiver Preise
17. Transaktionen, die bis jetzt abgeschlossen wurden

Anhang B

Dokumentation des Quellcodes

Im folgenden Abschnitt werden die wichtigsten Elemente der Marktsimulation Naschmarkt vorgestellt, um einen Einblick in den Quellcode zu geben. Auf eine detaillierte Dokumentation wird an dieser Stelle aus Gründen des Umfangs verzichtet, der im Anhang enthaltene Quellcode ist allerdings mit zahlreichen Kommentaren versehen, so dass im Zusammenwirken mit diesem Abschnitt eine ausreichende Dokumentation gegeben ist.

Die abgebildeten Diagramme verwenden die Notation der „Unified Modelling Language"(UML). Beschrieben wird die UML in [Oestereich, 1999]. Zum Thema C++ bieten sich [Willms, 1999] und [Willms, 1997] an, die C++ Standardbibliothek beschreiben [Kuhlins and Schader, 1999].

Das Kernstück der Simulation bildet die Klasse `agent`, aus der die Klassen `kaufagent` und `verkaufagent` abgeleitet sind, im Modul `agent.cpp`. Da sowohl Käufer als auch Verkäufer im wesentlichen die gleichen Algorithmen zur Angebotserstellung und Bewertung verwenden, sind die meisten Elemente und Methoden bereits in der Basisklasse `agent` deklariert und definiert. Die abgeleiteten Klassen `kaufagent` und `verkaufagent` definieren diejenigen Methoden, die für den Unterschied zwischen Käufer und Verkäufer verantwortlich sind. Das sind

`m_pc()`: Errechnet den eigenen Konfliktpreis

`s_pc()`: Errechnet den Konfliktpreis des Verhandlungspartners

`score()`: Errechnet den Effektivpreis eines Verhandlungsergebnis

`istbesser()`: Vergleicht zwei Preise

`p_res()`: Errechnet den Reservationspreis

Die Klasse `zvar` im Modul `meinmath.cpp` bildet eine Zufallsvariable ab, deren Dichtefunktion einer bestimmten Verteilung gehorcht. Sie stellt Methoden zur Bestimmung eines Erwartungswerts und für die Wahrscheinlichkeitsrechnung notwendige mathematische Funktionen zur Verfügung. Die Klasse `zvar` wird von `agent` für die Schätzwerte für Quasi-Rente und Zeitlimit der anderen Marktteilnehmer verwendet.

Das Modul `markt.cpp` steuert die Simulation und erzeugt drei Textdateien, in denen die Ergebnisse protokolliert werden.

Abbildung B.1 zeigt das Klassendiagramm mit den wichtigsten Eigenschaften und Methoden.

Bei den Bezeichnungen der im Code verwendeten Variablen wurde versucht, die im Text verwendeten mathematischen Symbole und Variablen nach Möglichkeit in eine der C++-Syntax gerechte Form zu übersetzen. Griechische Symbole werden dabei ausgeschrieben.

Innerhalb des Moduls `agent.cpp` wurden folgende voran gestellte Abkürzungen verwendet, um Variablen näher zu beschreiben:

`my_`**:** Bringt zum Ausdruck, das es sich dabei um ein `private` bzw. `protected` Element handelt.

`m_` **und** `s_`**:** Bei einigen Datenelementen, wie z.B. dem letzten gültigen Angebot, muss unterschieden werden, ob es das eigene (`m`ein) oder das des Verhandlungspartners ist (`s`ein).

`e_`**:** Bringt zum Ausdruck, dass es sich beim jeweiligen Element um einen Erwartungswert handelt.

B.1 Der Verhandlungsablauf

Jede Verhandlungsrunde läuft nach dem hier beispielsweise dargestelltem Ablauf ab, den auch Abbildung B.2 zeigt:

Der Kaufagent ist daran, ein Angebot zu erstellen. Er übergibt an das die Verhandlung steuernde Modul `markt` einen Preisvorschlag und einen Parameter (`aktion`), der Auskunft darüber gibt, wie der Preisvorschlag zu interpretieren ist. Dazu wird die Schnittstelle `angebot()` verwendet. Das Steuerungsmodul übergibt diesen Preisvorschlag mit der Schnittstelle
`getgebot()` an den Verkaufsagenten.

Die Schnittstelle `getgebot()` interpretiert diesen Vorschlag. Je nachdem, wie dieser Vorschlag interpretiert wird, bestimmt `getgebot()` die nächste Aktion des Verkaufsagenten (Annehmen, Abbrechen oder Gegengebot erstellen) in der nächsten Verhandlungsrunde.

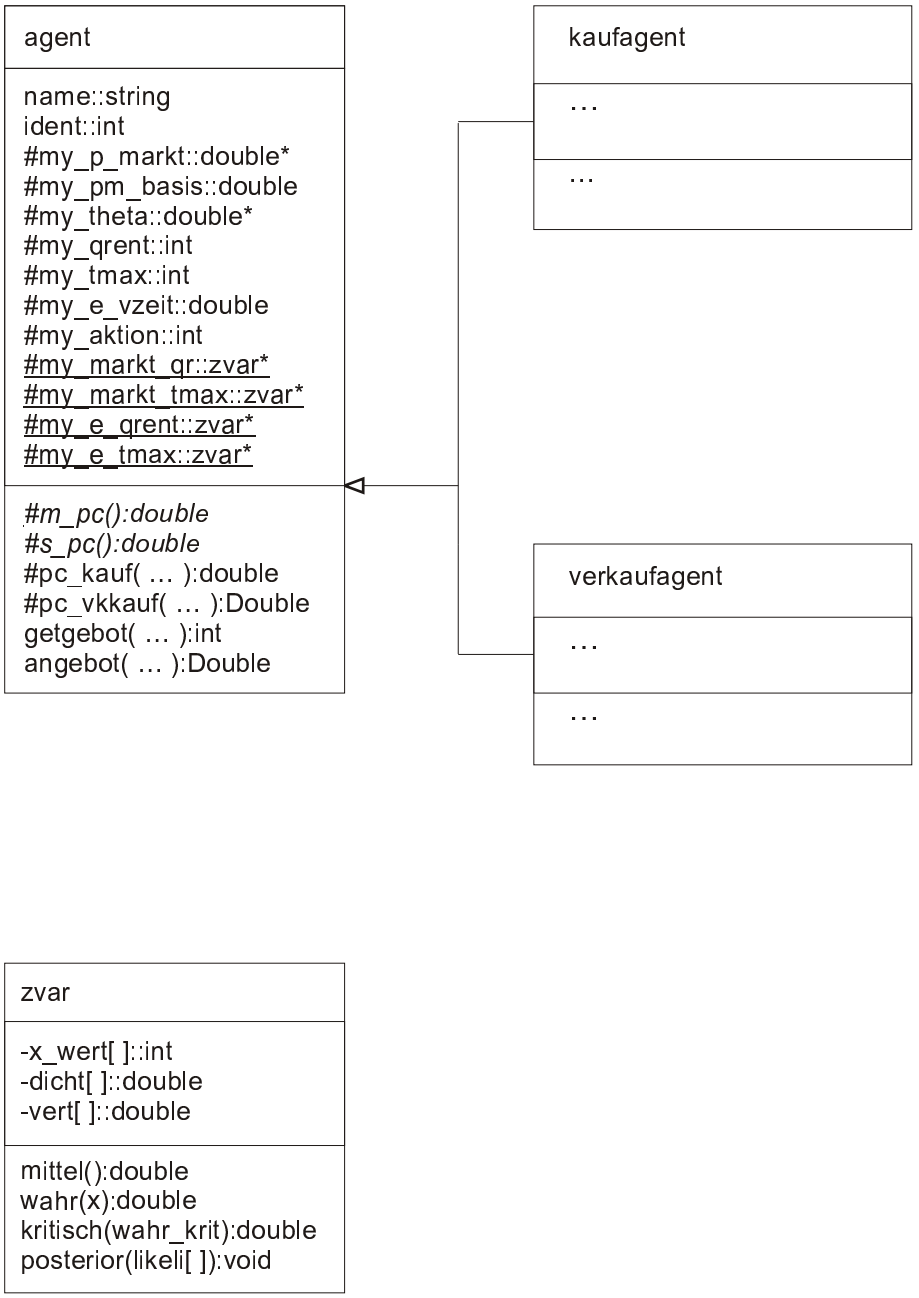

Abbildung B.1: Übersicht über das Klassendiagramm

Nur in dem Falle, in dem der Kaufagent ein Angebot stellt und der Verkaufagent sich auf Grund dieses Angebots zum Abbruch entscheidet, wird der Kaufagent noch in der selben Verhandlungsrunde vom Abbruch informiert. Notwendig ist das deswegen, damit die Verhandlung nicht in eine neue Runde gehen kann, wenn die Verhandlungszeit des Verkaufsagenten bereits abgelaufen ist.

Hat der Kaufagent ein in der Vorrunde erstelltes Angebot angenommen oder entschließt er sich, die Verhandlung abzubrechen, so ist der Verhandlungsprozess beendet.

Entschließt sich der Verkaufsagent, weiter zu verhandeln, ist diese Verhandlungsrunde beendet. In der nächsten Verhandlungsrunde tauschen der Kaufagent und der Verkaufagent die Rollen, so dass der Verkaufagent ein Gebot erstellt.

B.1.1 Die Klasse `agent`

Nach außen hin verfügt die Klasse `agent` über zwei wichtige Schnittstellen `angebot()` für die Berechnung eines Preisvorschlags und `getgebot()` zum Entgegennehmen eines Gebots. Neben dem Preis gibt `angebot()` einen Referenzparameter `m_aktion` zurück, der darüber Auskunft gibt ob der Preis als Gegengebot oder Annahmepreis interpretiert werden soll bzw. ob die Verhandlung abgebrochen wird. Der detaillierte Ablauf der Angebotsberechnung ist in Abbildung B.3 wiedergegeben.

Die Methode `getgebot()` nimmt einen Preisvorschlag und die Aktion des Partners entgegen und entscheidet, was aus dem Gegengebot gelernt werden soll. Für die Steuerung des Verhandlungsablaufs gibt `getgebot()` einen Wert (`weiter`) zurück, der angibt, ob die Verhandlung in eine weitere Runde geht. Dies ist nur dann der Fall, wenn sich der Agent entscheidet, ein Gegengebot zu stellen. Abbildung B.4 beschreibt die Aufgaben von `getgebot()` und beschreibt, welche Lernprozesse unter bestimmten Bedingungen gestartet werden.

Bluff-Variablen

Da ein Agent gemäß seiner Strategie vorgeben muss, bessere als tatsächlich bestehende Eigenschaften zu haben, wenn er vermutet, schlechter als sein Partner gestellt zu sein, werden zu Beginn einer Verhandlung Bluff-Werte festgelegt. Für die Quasi-Rente ist der Bluff-Wert das Minimum aus der eigenen Quasi-Rente und der erwarteten Quasi-Rente. Im Falle des Zeitlimits ist der Bluff-Wert das Maximum aus dem eigenen Zeitlimit und dem erwarteten Zeitlimit.

Diese Bluff-Werte für Quasi-Rente und Zeitlimit müssen auch beim Lernen verwendet werden, da sonst die kritischen Werte für Q und t^{max} bei der Bestimmung der Likelihood-Funktion verzerrt wären. Daraus resultiert das Zwischenspeichern der Variablen `my_qrent` sowie `my_tmax` für Quasi-Rente und Zeit-

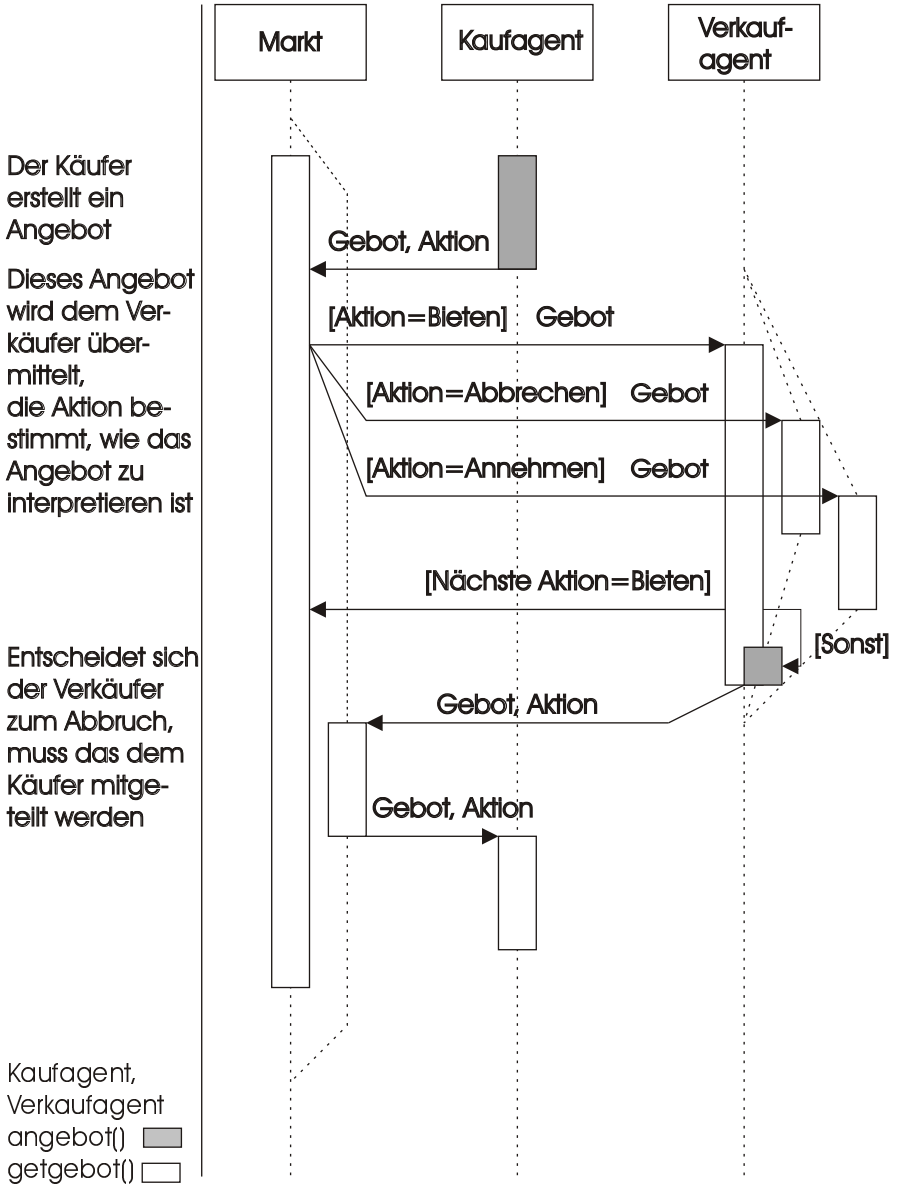

Abbildung B.2: Ablauf einer Verhandlungsrunde

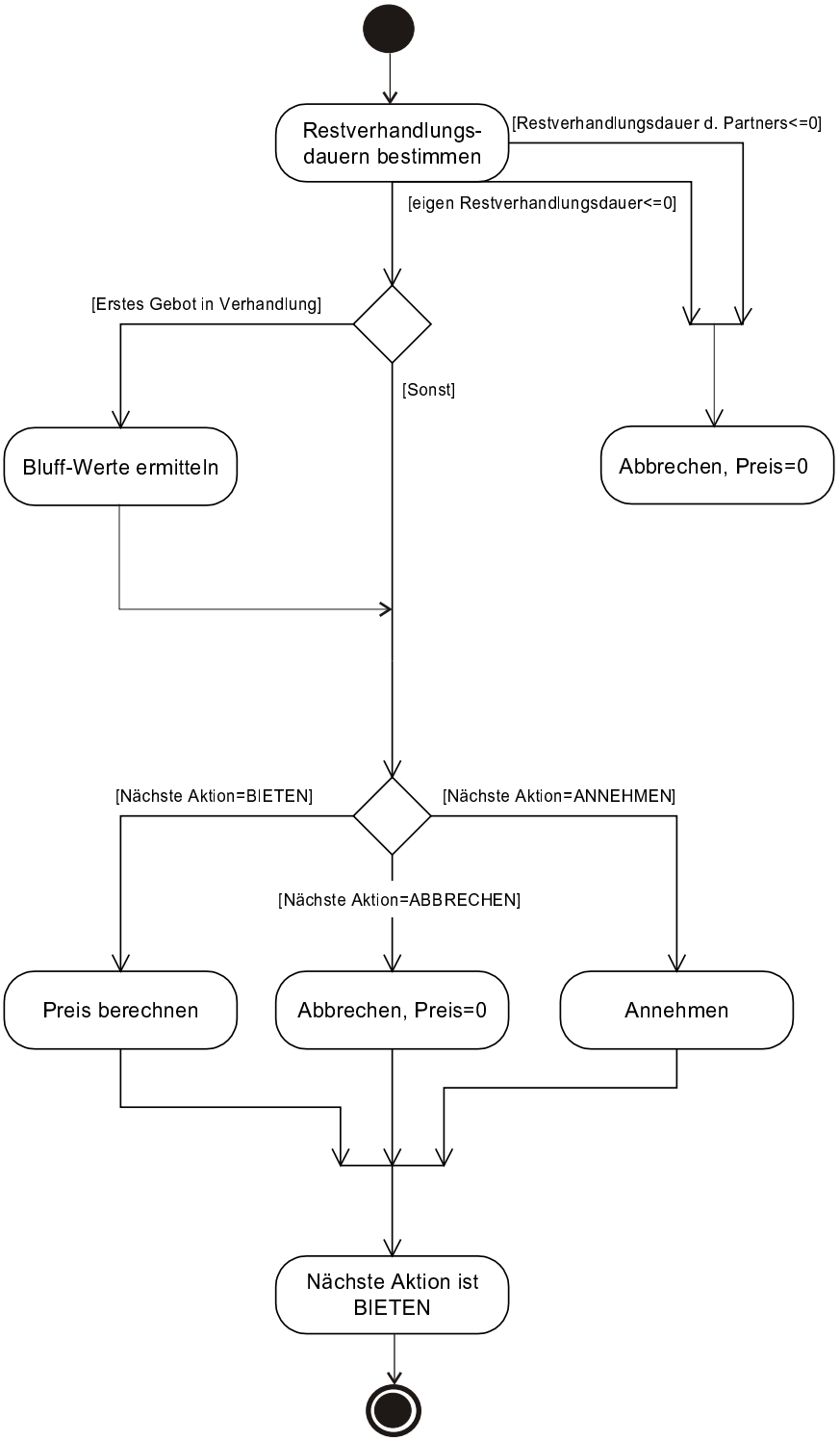

Abbildung B.3: Ablauf der Angebotserrechnung

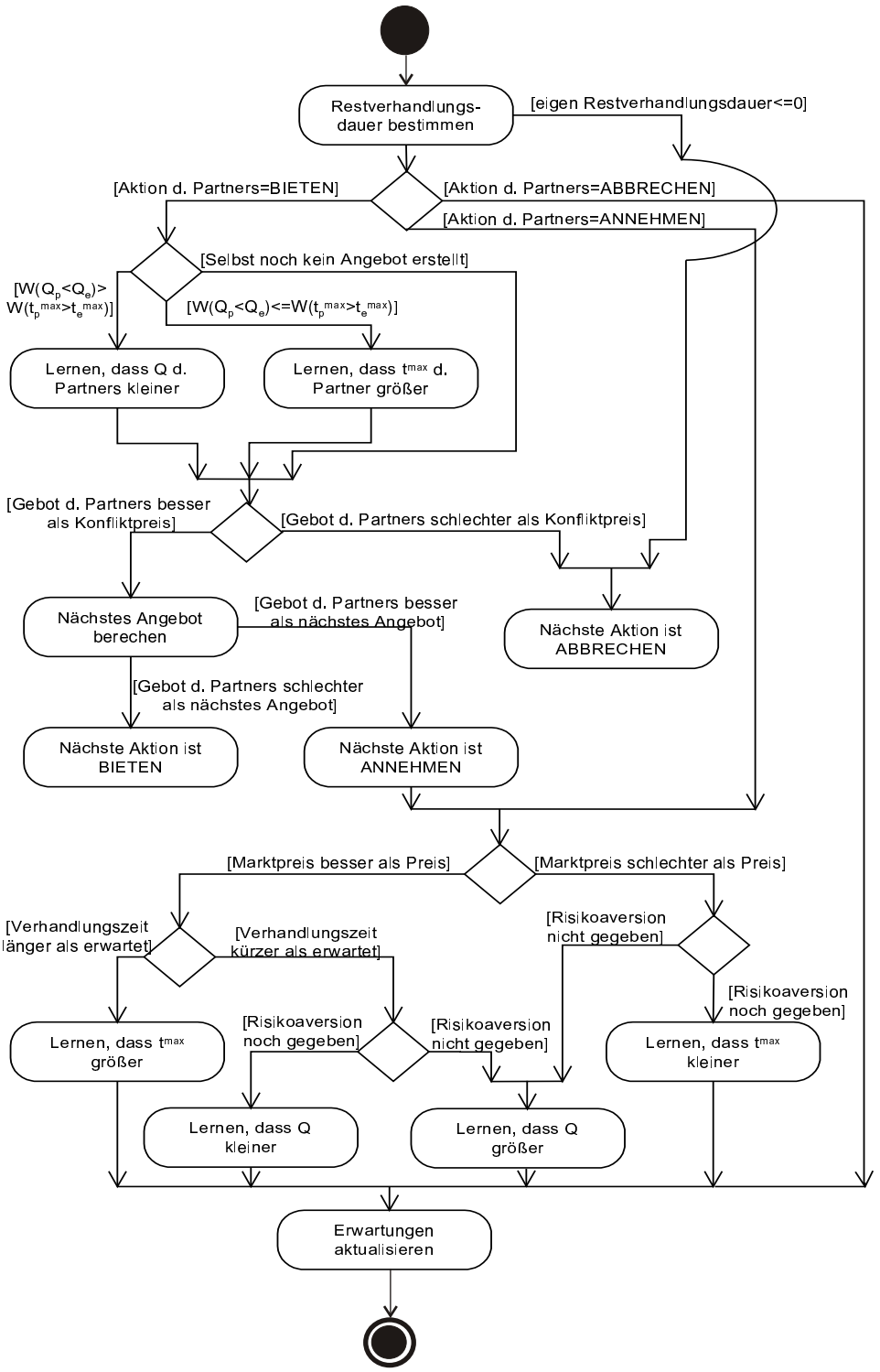

Abbildung B.4: Beurteilung eines Gegengebots

limit an zahlreichen Stellen in den Methoden `angebot()` und `getgebot()`. Die entsprechenden Bluff-Werte werden in `my_bluff_qr` und `my_bluff_tmax` gespeichert und durch die Methode `set_bluff()` vor dem Erstellen des ersten Angebots in einer Verhandlung ermittelt.

B.1.2 Konfliktpreisberechnung

Um die Methoden für die Konfliktpreisermittlung eines Käufers und eines Verkäufers nur zweimal kodieren zu müssen, wurden auf Basisklassenebene die Methoden `pc_kauf()` und `pc_vkkauf()` für den Konfliktpreis des Käufers und des Verkäufers implementiert. Die virtuell abgeleitete Methode `m_pc()` für den eigenen Konfliktpreis ruft im Falle von `kaufagent::m_pc()` die Methode `agent::pc_kauf()` mit den entsprechenden Parametern auf. Für den Konfliktpreis des Verhandlungspartners ruft `kaufagent::s_pc()` `agent::pc_vkkauf()` auf. Umgegehrt ruft `verkaufagent::m_pc()` `agent::pc_vkkauf()` auf bzw. `verkaufagent::s_pc()` `agent::pc_kauf()` auf.

B.2 Die Klasse `zvar`

Für die Lernmethoden eines Agenten müssen Zufallsvariablen verwaltet werden, die einer bestimmten Verteilung gehorchen. Die Klasse `zvar` stellt ein entsprechendes Objekt zur Verfügung. Die x-Werte einer Dichtefunktion $f(x)$ werden in dem dynamisch erzeugten Feld `x_wert` gespeichert, wobei $f(x_0) = 0$ gilt. Analog werden die Werte der Dichtefunktion im Feld `dicht` sowie die Verteilungsfunktion im Feld `vert` gespeichert.

Ein `zvar`-Objekt stellt eine stetige Zufallsvariable dar. Programmtechnisch wurde aber annäherungsweise eine diskrete Zufallsvariable realisiert. Die möglichen Ausprägungen werden im Feld `x_wert` gespeichert. Das Intervall zwischen den einzelnen Ausprägungen beträgt 1.

Um sich einer stetigen Verteilung anzunähern, wurde lineare Interpolation verwendet.

Für die Wahrscheinlichkeitsrechnung stellt `zvar` folgende Methoden zur Verfügung:

`mittel()`: Liefert den Mittelwert (Erwartungswert) eines `zvar`-Objekts.

`wahr(x)`: Stellt die Wahrscheinlichkeit $W(X \leq x)$ zur Verfügung.

`kritisch(w)`: Berechnet jene Ausprägung x der Zufallsvariable X, für die gilt: $w = W(X \leq x)$

`posterior(*l)`: Der Parameter `*l` ist ein Feld, das die Werte einer Likelihood-Funktion beinhaltet. Die Methode `posterior` ermittelt die a-posteri-Verteilung eines `zvar`-Objekts mit der Likelihood-Funktion in `*l`.

B.3 Simulationsablauf

Das Modul `markt.cpp` beinhaltet das Hauptprogramm und ist für die Steuerung der Simulation und die Protokollierung der Ergebnisse verantwortlich. Nach dem Einlesen der Simulationsparameter werden die entsprechenden Objekte der Klassen `kaufagent` und `verkaufagent` erzeugt, die in `list`-Containern der C++ Standard Template Libary verwaltet werden. Eine Verhandlungsrunde läuft wie folgt ab:

1. Aus den Containern `kaufer` und `verkaufer`, die Zeiger auf alle Kauf- bzw. Verkaufsagenten beinhalten, werden diejenigen in die Container `in_kaufer` bzw. `in_verkaufer` übertragen, die noch Güter zu kaufen bzw. zu verkaufen haben.

2. Durch die Funtktion `match()` werden zufällig und paarweise Agenten in die Container `m_kaufer` sowie `m_verkaufer` gestellt. Im Container `start` wird für jedes Paar vermerkt, ob ein Käufer oder ein Verkäufer eine Verhandlung beginnt. Wer die Verhandlung beginnt, wird zufällig ermittelt. Jeder Händler kann mit einer Wahrscheinlichkeit von 0,5 die Verhandlung beginnen.

3. Die Funktion `verhandeln()` wickelt eine Verhandlungsrunde mit allen in den Listen `m_kaufer` und `m_verkaufer` enthaltenen Objekten ab. Dabei stellt ein Agent mit der Schnittstelle `angebot()` ein Angebot, das dem Verhandlungspartner über `getgebot()` übermittelt wird. Nimmt dieser den Preis an oder bricht die Verhandlung ab, so endet die Verhandlung. Sofern eine Verhandlung endet, werden die Verhandlungspartner in die Listen `kaufer` und `verkaufer` zurückgestellt, um erneut mit anderen Verhandlungspartnern gematcht werden zu können.

   Geht die Verhandlung weiter, so bleiben die Agenten in den Listen `m_kaufer` und `m_verkaufer`.

Die Simulation endet, wenn nur noch fünf Käufer und Verkäufer ihre Aufgabenstellung nicht erfüllt haben.

B.4 Parameter

Sowohl für die Population der Kaufagenten als auch für Verkaufagenten werden Zeitlimit und Reservationspreis sowie Erwartungen über Zeitlimit und Reservationspreis der anderen Marktteilnehmer festgelegt. Dabei werden Mittelwerte bestimmt, nach denen für jeden einzelnen Agenten mit einem Zufallszahlengenerator die Eigenschaften und Erwartungen bestimmt werden. Die Verteilung dieser Zufallszahlen gehorcht einer Binomialverteilung mit einem Mittelwert, der den eingegebenen Parametern entspricht.

Die Funktion `binomwert()` im Modul `meinmath.cpp` erstellt diese Zufallswerte.

Anhang C

C++ Quellcode

C.1 Mathematische Funktionen

C.1.1 Header-Datei meinmath.h

```
/* Mathematische Funktionen fuer Naschmarkt                    */
/* Autor: Bernhard E. Beran                                    */
/* Datum: 27.10.2000                                           */
# ifndef MEINMATHDEF
# define MEINMATHDEF

#define GENAU 10                                               // fuer Standard-
                                                               // konstruktor
#define hoch(basis,expo) exp(log(basis)*(expo))                // Potenzrechnen
#define max(z1,z2) ((z1)>(z2)?(z1):(z2))                       // Max zweier Zahlen
#define min(z1,z2) ((z1)<(z2)?(z1):(z2))                       // Min zweier Zahlen

/* Runden: Rundet eine Zahl auf Stellen, wobei                 */
/* 0 ganzzahlig                                                */
/* Stellen > 0 : Auf Nachkommastellen runden                   */
/* Stellen < 0 : Auf Vorkommastellen runden                    */
double runden(double zahl, int stellen=0);

/* Fakultaet einer Zahl berechnen                               */
double fak(int zahl);

/* Dichte einer binominalverteilten Zufallsvariable x          */
/* u_grenze <= Z <= o_grenze mit Mittelwert mittel             */
double fdicht(int    u_grenze, int o_grenze, double mittel, int x);

/* Erzeugen einer binominalverteilten Zufallszahl zwischen*/
/* u_grenze und o_grenze mit Mittelwert mittel                 */
int binomwert(int u_grenze, int o_grenze, double mittel);
```

```
                                                                        30
/* Klasse zvar fuer Zufallsvariablen                                  */
/* erster Wert von x_wert: Untergrenze, W(X<=x_wert[0])=0 */
/* letzter Wert von x_wert: Obergrenze,                               */
/* W(X<=x_wert[grenze]=1)                                             */
class zvar
{
    private:
      int    *x_wert;                                  // x_Werte
      double *dicht;                                   // Dichte-
                                                       // funktion f(x)   40
      double *vert;                                    // Verteilung-
                                                       // sfunktion F(X)
      int grenze;                                      // Obergrenze
                                                       // Index f. Arrays
                                                       // Arraygroesse
                                                       //    grenze+1


    public:
      zvar();                                          // Std.-konstruktor  50
      /* Konstruktor mit Unter und Obergrenze, Gleichverteilt           */
      zvar(int u_grenze, int o_grenze);
      /* Konstruktor mit Unter- und Obergrenze, Verteilung nach Anzahl d. */
      /* Auspraegungen                                                  */
      zvar(int u_grenze, int o_grenze, double *apraeg);
      /* Konstruktor mit Unter- und Obergrenze, Mittelwert              */
      /* nach Binominalverteilung                                       */
      zvar(int u_grenze, int o_grenze, double mittel);
      /* Destruktor: freigeben der dynamisch erzeugten Arrays           */
      ~zvar()                                                            60
      {
        delete [] x_wert;
        delete [] dicht;
        delete [] vert;
      }
      /* Abfragen der Obergrenze der Arrays                             */
      int getgrenze()
        {
          return(grenze);
        }                                                                70
      /* Mittelwert von x                                               */
      double mittel();
      /* Wahrscheinlichkeit, dass X<=x        */
      double wahr(double x);
      /* x fuer das gilt W(x<=wahr_krit) */
      double kritisch(double wahr_krit);
      /* Bestimmen einer a-posterior Verteilung mittels einer Likelihood */
```

```
      /* Durch Multiplikationd der Dichtefunktion mit Likelihood-Funktion*/
      void posterior(double *likeli);
     /* Liefert die Auspraegung entsprechend der Dichte von x          */
     /* einer Variable mit der Grundgesamtheit gesamt                  */
     void auspraeg(double *apraeg, int gesamt);
     /* Operator[i] liefert den i-ten x_wert                           */
     int operator[](int index);
     /* Zuweisungsoperator                                             */
     zvar& operator=(zvar &z);
      /* Anzeige der Werte                                             */
      void show();
};

# endif
```

C.1.2 C++ Code meinmath.cpp

```
/* Mathematische Funktionen fur Naschmarkt - Objektsource */
/* Autor: Bernhard E. Beran                               */
/* Datum: 27.10.2000                                      */

#include <iostream.h>
#include <iomanip.h>
#include <math.h>
#include <stdlib.h>
#include "meinmath.h"

/* Runden aus stellen Stellen                             */
/* stellen >= 0 ... Nachkommastellen                      */
double runden(double zahl, int stellen=0)
{
  double ganz;
  double dezimal;
  int vz=1;
  zahl<0?vz=-1:vz=1;
  zahl=fabs(zahl);
  ganz = zahl*hoch(10,stellen)-fmod(zahl*hoch(10,stellen),1);
  dezimal = fmod(zahl*hoch(10,stellen),1);
  dezimal>=0.5?ganz++:ganz;
  return(vz*ganz/hoch(10,stellen));
}

/* Fakultaet von zahl berechnen                           */
double fak(int zahl)
{
  int i = 1;
  double myzahl=1;
```

```
    if (zahl>=0)
     for (i=2;i<=zahl;i++) myzahl*=i;
    else
     {
       cerr << "\nfak: Zahl negativ " << zahl << "\n";
       exit(1);
     }
  return(myzahl);
}

/* Dichte einer binomialverteilten Zufallsvariable */
/* u_grenze <= z <= o_grenze */
/* Mittelwert = mittel */
double fdicht(int u_grenze, int o_grenze, double mittel, int x)
{
 int grenze;
 double anteil;
 double f;
 o_grenze-=u_grenze;
 mittel-=u_grenze;
 x-=u_grenze;
 grenze = o_grenze;
 anteil=(mittel)/(grenze);
 f= fak(grenze)/(fak(x)*fak(grenze-x))*
    hoch(anteil,x)*hoch(1-anteil,grenze-x);
 return(f);
}

/* Binominalverteilte Zufallsvariable erzeugen */
/* u_grenez <= z <= o_grenze */
/* Mittelwert = mittel */
int binomwert(int u_grenze, int o_grenze, double mittel)
{
 double zzahl,v=0;
 int i=u_grenze;
 zzahl=double(rand()%(1000))/1000;
      while(zzahl>v)
       {
        v+=fdicht(u_grenze,o_grenze, mittel,i);
        i++;
       }
 return(zzahl==0?i:i-1);
}


/* Definition Standardkonstruktor */
/* erzeugt gleichverteilte Zufallsvariable zwischen 0 und GENAU-1*/
zvar::zvar()
{
```

```
                                                                    80
  int i=0;
  int ok=-1;

  grenze=GENAU-1;
  x_wert= new int[grenze+1];
  ok&=(x_wert!=NULL);
  dicht = new double[grenze+1];
  ok&=(dicht!=NULL);
  vert = new double[grenze+1];
  ok&=(vert!=NULL);                                                 90
      if (!ok)
        {
         cerr << "\nzvar::zvar: Allokationsfehler\n";
         exit(1);
        }
  x_wert[0]=0;
  dicht[0]=0;
  vert[0]=0;
  for (i=1;i<=grenze;i++)
    {                                                               100
      x_wert[i]=i;
      dicht[i]= 1.0/(grenze);
      vert[i]=double(i)/(grenze);
    }
}

/* Konstruktor mit unter und obergrenze */
/* erzeugt gleichverteilte Zufallswahriable zwischen u_greneze und o_grenze)*/
zvar::zvar(int      u_grenze, int o_grenze)
{                                                                   110
  int i=0;
  int ok=-1;


      if (u_grenze>=o_grenze)
        {
          cerr << "\nzvar::zvar: Untergrenze groesser(gleich) Obergrenze\n";
          exit(1);
        }
                                                                    120
  grenze=o_grenze-u_grenze;
  x_wert= new int[grenze+1];
  ok&=(x_wert!=NULL);
  dicht = new double[grenze+1];
  ok&=(dicht!=NULL);
  vert = new double[grenze+1];
  ok&=(vert!=NULL);
      if (!ok)
```

```
        {
         cerr << "\nzvar::zvar: Allokationsfehler\n";
         exit(1);
        }

   x_wert[0]=u_grenze;
   dicht[0]=0;
   vert[0]=0;

   for (i=1;i<=grenze;i++)
     {
       x_wert[i]=u_grenze + i;
       dicht[i]= 1.0/(grenze);
       vert[i]=double(i)/(grenze);
     }
}

/* Konstruktor mit Auspraegungen                                    */
/* erzeugt Zufallsvariable zwischen u_grenze und ogrenze            */
/* Verteilung lt. *auspraeg, wobei auspraeg[0] ignoriert wird */
/* W(x_wert[0])=0                                                   */
zvar::zvar(int u_grenze, int o_grenze, double *apraeg)
{
   int i=0;
   double sum=0;
   int ok=-1;
     if (u_grenze>=o_grenze)
     {
      cerr << "\nzvar::zvar: Untergrenze groesser(gleich) Obergrenze\n";
      exit(1);
     }
   grenze=o_grenze-u_grenze;
   x_wert= new int[grenze+1];
   ok&=(x_wert!=NULL);
   dicht = new double[grenze+1];
   ok&=(dicht!=NULL);
   vert = new double[grenze+1];
   ok&=(vert!=NULL);
        if (!ok)
         {
          cerr << "\nzvar::zvar: Allokationsfehler\n";
          exit(1);
         }

   x_wert[0]=u_grenze;
   vert[0]=0;
   dicht[0]=0;
   for (i=1;i<=grenze;i++)          // Summe der Merkmalsauspraegungen errechnen
     sum+=apraeg[i];
```

```
  for (i=1;i<=grenze;i++)
    {
      x_wert[i]=u_grenze + i;          // x-Intervallgrenze errechnen
      dicht[i]= apraeg[i]/sum;         // Wahrscheinlichkeit von x berechnen
      vert[i]=vert[i-1]+dicht[i];      // Verteilung= aufsummierte Dichte
    }
}

/* Konstruktor mit Unter- und Obergrenze, Mittelwert                    */
/* nach Binominalverteilung                                             */
/* W(x_wert[0])=0                                                       */
zvar::zvar(int u_grenze, int o_grenze, double mittel)
{
  int i=0;
  double anteil=0;
  int ok=-1;

    if (u_grenze>=o_grenze)
     {
      cerr << "\nzvar::zvar: Untergrenze groesser(gleich) Obergrenze\n";
      exit(1);
     }
  grenze=o_grenze-u_grenze;
  x_wert= new int[grenze+1];
  ok&=(x_wert!=NULL);
  dicht = new double[grenze+1];
  ok&=(dicht!=NULL);
  vert = new double[grenze+1];
  ok&=(vert!=NULL);
      if (!ok)
       {
        cerr << "\nzvar::zvar: Allokationsfehler\n";
        exit(1);
       }

  x_wert[0]=u_grenze;
  vert[0]=0;
  dicht[0]=0;

  anteil=(mittel-u_grenze-1)
         /(o_grenze-u_grenze-1);

  for (i=1;i<=grenze;i++)
    {
      x_wert[i]=u_grenze + i;          // x-Intervallgrenze errechnen
      // Dichte der Bionomialverteilung:
      dicht[i]= fak(grenze-1)/(fak(i-1)*fak(grenze-i))*
                hoch(anteil,i-1)*hoch(1-anteil,grenze-i);
      vert[i]=vert[i-1]+dicht[i];      // Verteilung= aufsummierte Dichte
```

```
    }
 }

/* Mittelwert von x */
double zvar::mittel()
{
 double mymittel=0.0;
 int i;
      for (i=1;i<=grenze;i++)
       mymittel+=x_wert[i]*dicht[i];
 return(mymittel);
}

/* Wahrscheinlichkeit, dass X<=x */
double zvar::wahr(double x)
{
  int i=0;
  double mywahr=0.0;

       if (x<x_wert[0])
            mywahr=0.0;
       else
            if (x>=x_wert[grenze])
                 mywahr=1.0;
            else
                 {
                  for (i=0;x_wert[i+1]<x && i<=grenze-1;i++);

                  mywahr = vert[i] + (vert[i+1]-vert[i]) *
                           (x-x_wert[i])/(x_wert[i+1]-x_wert[i]);
                 }
  return(mywahr);
}

/* x fuer das gilt W(x<=wahr_krit) */
/* liefert x_wert, bei dem W(X<=x_wert)=wahr_krit */
double zvar::kritisch(double wahr_krit)
{
  int i=0;
  double mykrit=0.0;

       if (wahr_krit<0 || wahr_krit>1)
        {
         cerr << "\nzvar::kritisch: Wahrscheinlichkeit ungueltig!\n";
         exit(1);
        }
       else
         {
              for (i=0;vert[i+1]<wahr_krit && i<=grenze-1;i++);
```

```
                mykrit = x_wert[i] + (x_wert[i+1]-x_wert[i]) *
                        (wahr_krit-vert[i])/(vert[i+1]-vert[i]);
            }
   return(mykrit);
}                                                                    280

/* Bestimmen einer a-posterior Verteilung mittels einer Likelihood */
/* durch Multiplizieren der Dichte-Funktion mit der Likelihood-     */
/* Funktion im Feld likeli                                          */
void zvar::posterior(double *likeli)
{
     int i=0;

     dicht[i]=0;
     vert[i]=0;                                                      290
     for (i=1;i<=grenze;i++)
      {
       dicht[i]*=likeli[i];
       vert[i]=vert[i-1]+dicht[i];
      }
}

/* Liefert die Auspraegung entsprechend der Dichte von x */
/* einer Variable mit der Grundgesamtheit gesamt         */
void zvar::auspraeg(double *apraeg, int gesamt)                      300
{
 int i=0;
 for (i=0;i<=grenze;i++)
 apraeg[i]=gesamt*dicht[i];
}

/* Operator[i] liefert den i-ten x-Wert                             */
int zvar::operator[](int index)
{
     if (index<0 || index>grenze)                                    310
      {
       cerr << "\nzvar::[]: ungueltiger Index\n";
       exit(1);
      }
 return(x_wert[index]);
}


/* Zuweisungsoperator                                               */
zvar& zvar::operator=(zvar &z)                                       320
{
 int i, ok=-1;

 grenze=z.grenze;
```

```
 delete [] x_wert;
 delete [] dicht;
 delete [] vert;
 x_wert = new int[grenze+1];
 ok&=(x_wert!=NULL);
 dicht = new double[grenze+1];                                          330
 ok&=(x_wert!=NULL);
 vert = new double[grenze+1];
 ok&=(x_wert!=NULL);
      if (!ok)
       {
        cerr << "\nzvar::operator=: Allokationsfehler \n";
        exit(1);
       }
      for (i=0;i<=grenze;i++)
       {                                                                340
        x_wert[i]=z.x_wert[i];
        dicht[i]=z.dicht[i];
        vert[i]=z.vert[i];
       }
 return(*this);
}

/* Definition show() */
void zvar::show()
{                                                                       350
    int i=0;

    cout << "\nx-Werte:\n" << setiosflags(0x1000)<< setprecision(2);
    for (i=0;i<=grenze;i++)
     cout << setw(5)<< x_wert[i] << ";";

    cout << "\nDichte x:\n";
    for (i=0;i<=grenze;i++)
     cout << setw(5) << dicht[i] << ";";
                                                                        360
    cout << "\nVerteilung x:\n";
    for (i=0;i<=grenze;i++)
     cout << setw(5) << vert[i] << ";";
    cout <<"\nMittelwert x:\n" << mittel();
}
```

C.2 Agenten

C.2.1 Header-Datei agentpt.h

```
/* Agent.h - Deklarationen für Agent.cpp,                    */
/* Klassen agent, kaufagent und verkaufagent                 */
/* Autor: Bernhard E. Beran                                  */
/* Datum: 07.04.2001                                         */
#include "meinmath.h"                                        // fuer Zufallsvariablen
                                                             // erforderlich

#include "marktpt.h"
#define FNIVEAU 0.002                                        // Fehlerniveau fuer
                                                             // die Lernenroutinen
                                                             // qrentklein(),
                                                             // qrentgross(),
                                                             // tmaxklein(), tmaxgross()

/* Lernrate                                                  */
#define GEWICHT 0.2                                          // Neue Erfahrungen werden
                                                             // bei marktupdate()
                                                             // mit GEWICHT gelernt

#include <iostream.h>                                        // Transferoperator

/* Agentenklasse                                             */
class agent
{
  protected:
      double *const my_p_markt;                              // Zeiger akt. Marktpreis
      double my_pm_basis;                                    // Marktpreis, von dem bei
                                                             // Verhandlungen
                                                             // ausgegangen wird
      double my_theta;                                       // Zeiger auf
                                                             // Transaktionskosten
      double my_psi;                                         // Wahrscheinlichkeit,
                                                             // mit der P* in
                                                             // t=0 erreicht werde kann
      int *const my_anz_kauf;                                // Zeiger Kaeuferzahl
      int *const my_anz_vkkauf;                              // Zeiger Verkaeuferzahl
      int my_tmax;                                           // Zeitlimit fuer
                                                             // Vertragsabschluss
      int my_qrent;                                          // Quasirente (T+R)
      int my_bluff_qr;                                       // Quasirente zum Bluffen
      int my_bluff_t;                                        // Zeitlimit zum Bluffen
      double my_score;                                       // Bewertung des Agenten
      double my_score_eff;                                   // Bewertung des Agenten
      double my_p_eff;                                       // erzielter Effektivpreis
      int my_anz_trans;                                      // Anzahl der Abschluesse
```

```
int my_tzeit;                          // Gesamte Verweildauer
                                       // am Markt
int my_vzeit;                          // Gesamte verstrichene
                                       // Verhandlungszeit
double my_e_vzeit;                     // durchschn. erwartete
                                       // Verhandlungszeit
int my_aktion;                         // Bestimmt naechste
                                       // Aktion, wird von
                                       // getgebot() festgelegt
zvar *my_markt_qr;                     // Zvar mit Auspraegungen
                                       // der MarktQuasirente
zvar *my_markt_tmax;                   // Zvar mit Auspraegungen
                                       // des Zeitlimits
                                       // der Marktteilnehmer
zvar *my_e_qrent;                      // Quasirente des Partners
zvar *my_e_tmax;                       // Zeitlimit des
                                       // Verhandlungspartners
double my_s_gebot;                     // Letztes Gegengebot
double my_m_gebot;                     // eigenes letztes Angebot

/* Parameter einer Nutzenfunktion errechnen      */
void nutzenpar
     (double &a, double &b, double &alpha,double qr,double tmax);
/* Bluff-Variablen setzen                        */
void set_bluff();
/* Eigenen Konfliktpreis errechnen               */
virtual double m_pc()=0;
/* Konfliktpreis des Partners errechnen          */
virtual double s_pc(double qr, double tmax)=0;
/* wahr, wenn p1 besser als p2 ist               */
virtual int istbesser(double p1, double p2)=0;
/* Reservationspreis berechnen                   */
virtual double p_res()=0;
/* Konfliktpreis eines Kaeufers errechnen        */
double pc_kauf
       (double a, double b, double alpha, double qr, double tmax);
/* Konfliktpreis eines Verkaeufers errechnen     */
double pc_vkkauf
       (double a, double b, double alpha, double qr, double tmax);
/* Errechnen der kritschen Zeit                  */
/* t, das Partner bei qrent und my_s_gebot hat*/
double t_krit(double qrent);
/* Errechnen der kritischen Quasirente           */
/* Q, die Partner bei tmax und my_s_gebot hat */
double q_krit(double tmax);
/* Lernen, dass QuasiRente groesser              */
void qrentgross();
/* Lernen, dass QuasiRente kleiner               */
void qrentklein();
```

```
    /* Lernen, dass Zeitlimit groesser                    */
    void tmaxgross();
    /* Lernen, dass Zeitlimit kleiner                     */
    void tmaxklein();
    /* Markterfahrung aendern, nach dem                   */
    /* Verhandlung zu ende                                */
    void marktupdate();                                                  100
    /* Berechnung des Scores                              */
    virtual void score(double preis)=0;


public:
    char name[12];                          // Kaeufer bzw. Verkaeufer
    int ident;                              // Identifikation
    /* Standardkonstruktor                                */
    agent();
    /* Erzeugt Agent mit qrent und tmax sowie             */          110
    /* mit E(Q)=e_qrent und E(tmax)=e_tmax u.theta*/
    agent(int qrent, int tmax, int e_qrent, int e_tmax,
          double theta);
    /* Destruktor                                         */
    ~agent()
       {
         delete my_markt_qr;                // Freigeben d. Speichers
         delete my_markt_tmax;              // d. zvars, die Markt-
                                            // struktur enthalten
         delete my_e_qrent;                 // Freigeben d, Speichers   120
         delete my_e_tmax;                  // beider Zufallsvariablen
       }
    /* Gebot entgegennehemen und Aktion bestimmen */
    /* Lernen                                             */
    int getgebot(double gebot, int s_aktion);
    /* Angebot erstellen, wenn bluff=0 werden             */
    /* immer die eigenen Parameter statt den Bluff*/
    /* Werten verwendet                                   */
    double angebot(int &m_aktion, int bluff=-1);
    /* Zeitintervall entgegen nehmen                      */          130
    void ticker()
    {
     my_vzeit++;
     my_tzeit++;
    }

    /* Starten einer neuen Verhandlung                    */
    virtual void neustart()=0;
    /* Ausgeben der erwarteten Quasi-Rente                */
    double get_e_qrent()                                                 140
    {
     return(my_markt_qr->mittel());
```

```
        }
        /* Ausgeben des erwarteten Zeitlimits                */
        double get_e_tmax()
        {
         return(my_markt_tmax->mittel());
        }
        /* Eintretten in denMarkt                            */
        virtual int eintritt()=0;                                          150

       /* friend-Deklaration desTransferoperator             */
       /* für Agent                                          */
       friend ostream& operator<<(ostream& ostr, const agent& agt);

};


/* Klasse der Kaeufer                                        */
class kaufagent : public agent                                             160
{
  protected:
        /* Eigenen Konfliktpreis errechnen                   */
        virtual double m_pc();
        /* Konfliktpreis des Partners errechnen              */
        virtual double s_pc(double qr, double tmax);
        /* berechnen des scores                              */
        virtual void score(double preis);
        /* wahr wenn p1 besser als p2 ist                    */
        virtual int istbesser(double p1, double p2)                        170
        {
         return(p1<=p2);
        }
        /* Reservationspreis berechnen                       */
        virtual double p_res()
        {
         return(my_pm_basis*(1+my_qrent/my_pm_basis));
        }

     public:                                                               180
        /* Standardkonstruktor                               */
        kaufagent();
        /* Erzeugt Kaeufer mit qrent und tmax sowie          */
        /* mit E(Q)=e_qrent und E(tmax)=e_tmax u. theta      */
        kaufagent(int qrent, int tmax, int e_qrent, int e_tmax,
        double theta);
        /* Starten einer neuen Verhandlung                   */
        virtual void neustart();
        /* Eintretten in den Markt?                          */
        virtual int eintritt();                                            190
};
```

```
/* Klasse der Verkaeufer                                    */
class verkaufagent : public agent
{
  protected:
      /* Eigenen Konfliktpreis errechnen                    */
      virtual double m_pc();
      /* Konfliktpreis des Partners errechnen               */                    200
      virtual double s_pc(double qr, double tmax);
      /* berechnen des scores                               */
      virtual void score(double preis);
      /* wahr wenn p1 besser als p2 ist                     */
      virtual int istbesser(double p1, double p2)
      {
       return(p1>=p2);
      }
      /* Reservationspreis berechnen                        */
      virtual double p_res()                                                     210
      {
       return(my_pm_basis*(1-my_qrent/my_pm_basis));
      }


  public:
      /* Standardkonstruktor                                */
      verkaufagent();
      /* Erzeugt Verkaeufer mit qrent und tmax sowie */
      /* mit E(Q)=e_qrent und E(tmax)=e_tmax u. theta       */                    220
      verkaufagent(int qrent, int tmax, int e_qrent, int e_tmax,
                   double theta);
      /* Starten einer neuen Verhandlung                    */
      virtual void neustart();
      /* Eintretten in den Markt?                           */
      virtual int eintritt();
};

/* Transferoperator fuer Agent                              */
ostream& operator<<(ostream& ostr, const agent& agt);                            230
```

C.2.2 C++ Code agentpt.cpp

```
/* Agent.cpp - Definition der Klassen agent, kauf-          */
/* agent und verkaufagent                                   */
/* Autor: Bernhard E. Beran                                 */
/* Datum: 07.04.2001                                        */
```

```
#include "marktpt.h"
#include "agentpt.h"
#include "meinmath.h"

#include <math.h>                                                    10
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>

/* Deklarationen fuer die Marktvariablen                 */
extern double p_markt;                                  // Marktpreis
extern double theta;                                    // Transaktionskosten i.P
extern int anz_kauf;                                    // Kaeuferzahl          20
extern int anz_vkkauf;                                  // Verkaeuferzahl

/* Deklarationen fuer die Agentenparameter             */
extern int r_unten;                                     // Untergrenze Risikopraemie
extern int r_oben ;                                     // Obergrenze Risikopraemie
extern int r_schnitt ;                                  // Durchschn. Risikopraemie
extern int tmax_unten ;                                 // Untergrenze Zeitlimit
extern int tmax_oben ;                                  // Obergrenze Zeitlimit
extern int tmax_schnitt ;                               // Durchschn. Zeitlimit
extern int k_auftrag ;                                  // Auftrag Kaeufer      30
extern int v_auftrag ;                                  // Auftrag Verkaeufer


/* **********************AGENT********************************************** */


/* Standardkonstruktor fuer Agent                        */
agent::agent()
{
 cerr << "\n agent::agent(): unzulaessiger Konstruktor";                     40
}

/* Konstruktor mit Quasirente und t-max sowie          */
/* E(Q) und E(t-max) und theta                         */
agent::agent(int qrent, int tmax, int e_qrent, int e_tmax,
             double theta) :
     my_p_markt(&p_markt),                              // Pointer auf Marktpreis
     my_anz_kauf(&anz_kauf),                            // Pointer auf Kaeuferzahl
                                                        // am Markt
     my_anz_vkkauf(&anz_vkkauf)                         // Pointer auf Verkaeuferzahl   50
                                                        // am Markt
{
 my_psi=PSI;                                            // PSI=0,5
 my_tmax=tmax;                                          // Zeitlimit lt. Parameter
```

```
 my_qrent=qrent;                                    // Quasi-Rente lt. Parameter
 my_pm_basis=PM_BASIS;                              // Ausgangspreis ist 100
 my_theta = theta;                                  // Transaktionskostenminimum
 my_tzeit=0;                                        // Verweildauer am Markt und
 my_vzeit=0;                                        // Verhandlungszeit = 0
 my_e_vzeit = E_VZEIT;                              // Erwartete                    60
                                                    // Verhandlungsdauer
 my_aktion=BIETE;                                   // Aktion ist Bieten
 /* Allokation erwartete Quasi-Rente des Partners   */
 my_e_qrent= new zvar(Q_UNTEN,Q_OBEN,e_qrent);
     if (my_e_qrent==NULL)
      {
       cerr << "\nagent::agent: Allokationsfehler my_e_qrent\n";
       exit(1);
      }
/* Allokation erwartetes Zeitlimit des Partners     */                              70
my_e_tmax= new zvar(tmax_unten,tmax_oben,e_tmax);
      if (my_e_tmax==NULL)
     {
      cerr << "\nagent::agent: Allokationsfehler my_e_tmax\n";
      exit(1);
     }
 /* Allokation erwartete Quasi-Rente am Markt       */
 my_markt_qr = new zvar(Q_UNTEN,Q_OBEN,e_qrent);
      if (my_markt_qr==NULL)
       {                                                                            80
        cerr << "\nagent::agent: Allokationsfehler my_markt_qr\n";
        exit(1);
       }
 /* Allokation erwartetes Zeitlimit am Markt        */
 my_markt_tmax = new zvar(tmax_unten,tmax_oben,e_tmax);
      if (my_markt_tmax==NULL)
       {
        cerr << "\nagent::agent: Allokationsfehler my_markt_tmax\n";
        exit(1);
       }                                                                            90
 my_s_gebot=0;                                      // Gegengebot = 0
 my_m_gebot=0;                                      // Letztes Angebot = 0
 my_score=0;                                        // Score
 my_score_eff=0;                                    // Score zum Effektivpreis
 my_anz_trans=0;                                    // Transaktionsanzahl
 set_bluff();
}

/* Berechnen der Parameter der Nutzenfunktion      */
/* U=a*Y^alpha+b                                   */                              100
void agent::nutzenpar
      (double &a, double&b, double &alpha, double qr, double tmax)
{
```

```
 double y=my_theta*my_pm_basis*tmax;                    // Einkommensniveau = T
 double u_y=my_psi*U_P_MARKT+(1-my_psi)*U_NULL; // Nutzenniveau von Y
 b=U_NULL;                                              // b entspricht U(0)
 alpha=log((U_P_MARKT-b)/(u_y-b))/log((y+qr)/y);        // alpha berechnen
 a=(u_y-b)/hoch(y,alpha);                               // a berechnen
}

/* Bluff-Variablen setzen                               */
/* Minimum aus eigenm Q und E(Q) bzw.                   */
/* Maximum aus eigenm t und E(t)                        */
void agent::set_bluff()
{
 my_bluff_qr=min(my_qrent,int(runden(my_e_qrent->mittel())));
 my_bluff_t=max(my_tmax,int(runden(my_e_tmax->mittel())));
}


/* Konfliktpreis eines Kaeufers errechnen               */
double agent::pc_kauf
        (double a, double b, double alpha, double qr, double tmax)
{
 double trest;                                          // Restverhandlungsdauer
 double omega,delta;                                    // W(P* innerhalb t-rest)
                                                        // Diskontfaktor
 double tmp_psi,c;                                      // W(P* sofort),
                                                        // Konfliktnutzen
 double pc,y=tmax*my_theta*my_pm_basis;                 // Konfliktpreis,
                                                        // Einkommen
 double u_pm;                                           // Nutzen des akt.
                                                        // Marktpreis
 trest = tmax-my_vzeit;                                 // Berechnung
                                                        // Restverhandlungsdauer

 delta = hoch(1-qr/my_pm_basis,1.0/trest);              // Diskontfaktor berechnen
 /* Psi muss an die Situation am Markt angepasst werden */
 /* ist abhaengig vom Verhaeltnis Verkaeufer/Kaeufer    */
 tmp_psi = my_psi*min(1,double(*my_anz_vkkauf)/double(*my_anz_kauf));
 omega = 1-hoch(1-tmp_psi,trest);                       // Wahrscheinlichkeit,
                                                        // in t-rest
                                                        // ein Ergebnis zu erzielen
 /* Nutzen des Marktpreises berechnen                   */
 u_pm = a*hoch((y+my_pm_basis+my_qrent-*my_p_markt),alpha)+b;
 trest= my_e_vzeit;
 c=omega*u_pm*hoch(delta,trest);                        // Konfliktnutzen errechnen
 pc=y+my_pm_basis+qr-hoch((c-b)/a,1/alpha);             // Konfliktpreis errechnen
 return(pc);
}

/* Konfliktpreis eines Verkaeufers errechnen            */
```

```
double agent::pc_vkkauf
        (double a, double b, double alpha, double qr, double tmax)
{
 double trest;                                  // Restverhandlungsdauer
 double omega,delta;                            // W(P* innerhalb
                                                //von t-rest),
                                                // Diskontfaktor
 double tmp_psi,c;                              // W(P* sofort),
                                                // Konfliktnutzen
 double pc,y=tmax*my_theta*my_pm_basis;         // Konfliktpreis, Einkommen
 double u_pm;                                   // Nutzen des aktuellen
                                                // Marktpreis
 trest = tmax-my_vzeit;                         // Berechnung
                                                // Restverhandlungsdauer

 delta = hoch(1-qr/my_pm_basis,1.0/trest);      // Diskontfaktor berechnen
 /* Psi muss an die Situation am Markt angepasst werden */
 /* ist abhaengig vom Verhaeltnis Verkaeufer/Kaeufer      */
 tmp_psi = my_psi*min(1,double(*my_anz_kauf)/double(*my_anz_vkkauf));
 omega = 1-hoch(1-tmp_psi,trest);               // Wahrscheinlichkeit,
                                                // in t-rest
                                                // ein Ergebnis zu erzielen
 /* Nutzen des Marktpreises berechnen            */
 u_pm = a*hoch((y+my_pm_basis+my_qrent-*my_p_markt),alpha)+b;
 trest= my_e_vzeit;
 c=omega*u_pm*hoch(delta,trest);                // Konfliktnutzen errechnen
 pc=my_pm_basis-qr-y+hoch((c-b)/a,1/alpha);     // Konfliktpreis errechnen
 return(pc);
}

/* Errechnen der kritschen Zeit t, die Partner bei      */
/* qrent und my_m_gebot hat                             */
double agent::t_krit(double qrent)
{
 /* Eigenen Diskontfaktor berechnen                     */
 double m_delta=hoch(1-my_qrent/(my_pm_basis),1.0/(my_tmax-my_vzeit));
 double log_s_delta;                            // Log Diskontfaktor des
                                                // Partners
 double m_x;                                    // eigener Anteil am
                                                // Verhandlungsspielraum

 /* berechnet eignen Anteil am Verhandlungsspielraum     */
 /* ausgehend von den E(Q) und E(t-max) und dem zuletzt */
 /* gemachten Angebot                                    */
  m_x=(m_pc()-my_m_gebot)/
       (m_pc()-s_pc(my_e_qrent->mittel(),my_e_tmax->mittel()));
 log_s_delta=m_x*log(m_delta)/(1-m_x);          // Log des Diskontfaktors
                                                // des Verhandlungspartners
 // t-kritsch zurueckgeben
```

```
 return(log(1-qrent/(my_pm_basis))/log_s_delta+my_vzeit);
}

/* Errechnen der kritischen Quasi-Rente Q, die Partner      */
/* bei tmax und my_m_gebot hat                              */
double agent::q_krit(double tmax)
{
 /* Eigenen Diskontfaktor berechnen                         */
 double m_delta=hoch(1-my_qrent/(my_pm_basis),1.0/(my_tmax-my_vzeit));   210
 double s_delta;                                  // Diskontfaktor d. Partner
 double m_x;                                      // eigener Anteil am
                                                  // Verhandlungsspielraum

 /* berechnet eigenen Anteil am Verhandlungsspielraum       */
 /* ausgehend von den E(Q) und E(t-max) und dem zuletzt */
 /* gemachten Angebot                                       */
 m_x=(m_pc()-my_m_gebot)/
      (m_pc()-s_pc(my_e_qrent->mittel(),my_e_tmax->mittel()));
 s_delta=hoch(M_E,m_x*log(m_delta)/(1-m_x));      // Diskontfaktor Partner   220
 // Q-kritsich zurueckgeben
 return(my_pm_basis*(1-hoch(s_delta,tmax-my_vzeit)));
}

/* Lernen, dass QuasiRente kleiner als erwartet ist         */
void agent::qrentklein()
{
 double *likeli=new double[my_e_qrent->getgrenze()+1];  // Array L-Funktion
 double *dicht=new double[my_e_qrent->getgrenze()+1];   // temp Dichte-Funktion
 double w_e=0;                                    // Wahrscheinlichkeit      230
                                                  // des Ereignis E W(E)
                                                  // E:Gegengebot
 int i;
      if (likeli==NULL || dicht==NULL)
       {
        cerr << "\n::qrentklein: Allokationsfehler\n";
        exit(1);
       }

 my_e_qrent->auspraeg(dicht, 1);                  // Dichte von Q in         240
                                                  // dicht speichern
      for(i=0;i<=my_e_qrent->getgrenze();i++)
      {
       /* Wahrscheinlichkeit, dass t>t-kritisch,          */
       /* wenn Agent qr[i] hat, entspricht W(ablehnen|qr)*/
       likeli[i]=max(FNIVEAU,1-my_e_tmax->wahr(t_krit((*my_e_qrent)[i])));
       w_e+=likeli[i]*dicht[i];                   // W(E) nach Satz d.
                                                  // totalen
                                                  // Wahrscheinlichkeit
      }                                                                    250
```

```
      for (i=0;i<=my_e_qrent->getgrenze();i++)
       likeli[i]/=w_e;                                  // Likelihood-Funktion:
                                                        // L=W(E|A)/W(E)
 my_e_qrent->posterior(likeli);                         // f-aposterior=f-apriori*L
 delete [] likeli;                                      // Speicher f. temp
 delete [] dicht;                                       // Arrays freigeben
}

/* Lernen, dass QuasiRente groesser als erwartet ist          */
void agent::qrentgross()                                                    260
{
 double *likeli=new double[my_e_qrent->getgrenze()+1];  // Array f. L-Funktion
 double *dicht=new double[my_e_qrent->getgrenze()+1];   // temp Dichte-Funktion
 double w_e=0;                                          // Wahrscheinlichkeit
                                                        // des Ereignis E W(E)
                                                        // E:Angebot angenommen
 int i;
      if (likeli==NULL || dicht==NULL)
       {
        cerr << "\n::qrentgross: Allokationsfehler\n";                       270
        exit(1);
       }
 my_e_qrent->auspraeg(dicht, 1);                        // Dichte von Q
                                                        // in dicht speichern
     for(i=0;i<=my_e_qrent->getgrenze();i++)
      {
       /* Wahrscheinlichkeit, dass t<t-kritisch,           */
       /* wenn Agent qr[i] hat entspricht W(annehmen|qr)*/
       likeli[i]=max(FNIVEAU,my_e_tmax->wahr(t_krit((*my_e_qrent)[i])));
       w_e+=likeli[i]*dicht[i];                         // W(E) nach Satz d.   280
                                                        // totalen
                                                        // Wahrscheinlichkeit
      }
     for (i=0;i<=my_e_qrent->getgrenze();i++)
      likeli[i]/=w_e;                                   // Likelihood-Funktion:
                                                        // L=W(E|A)/W(E)
 my_e_qrent->posterior(likeli);                         // f-aposterior=f-apriori*L
 delete [] likeli;                                      // Speicher f.temp
 delete [] dicht;                                       // Arrays freigebe
}                                                                           290

/* Lernen, dass Zeitlimit groesser als erwartet ist           */
void agent::tmaxgross()
{
 double *likeli= new double[my_e_tmax->getgrenze()+1]; // Array fuer L-Funktion
 double *dicht= new double[my_e_tmax->getgrenze()+1];   // temp Dichte-Funktion
 double w_e=0;                                          // Wahrscheinlichkeit
                                                        // des Ereignis E W(E)
                                                        // E:Gegengebot erhalten
```

```
 int i;                                                                    300
      if (likeli==NULL || dicht==NULL)
       {
        cerr << "\n::tmaxgross: Allokationsfehler\n";
        exit(1);
       }
 my_e_tmax->auspraeg(dicht, 1);                    // Dichte von Q
                                                   // in dicht speichern
     for(i=0;i<=my_e_tmax->getgrenze();i++)
      {
       /* Wahrscheinlichkeit, dass Q<Q-kritisch,  */                       310
       /* wenn Agent tmax[i] hat, entspricht      */
       /* W(ablehnen|t)                           */
       likeli[i]=max(FNIVEAU,my_e_qrent->wahr(q_krit((*my_e_tmax)[i])));
       w_e+=likeli[i]*dicht[i];                    // W(E) nach Satz d.
                                                   //totalen
                                                   // Wahrscheinlichkeit
      }
      for (i=0;i<=my_e_tmax->getgrenze();i++)
       likeli[i]/=w_e;                             // Likelihood-Funktion:
                                                   // L=W(E|A)/W(E)         320
 my_e_tmax->posterior(likeli);                     // f-aposterior=f-apriori*L
 delete [] likeli;                                 // Speicher f. temp
 delete [] dicht;                                  // Arrays freigeben
}

/* Lernen, dass Zeitlimit kleiner                 */
void agent::tmaxklein()
{
 double *likeli= new double[my_e_tmax->getgrenze()+1]; // Array f. L-Funktion
 double *dicht= new double[my_e_tmax->getgrenze()+1];  // temp Dichte-Funktion  330
 double w_e=0;                                     // Wahrscheinlichkeit
                                                   // des Ereignis E W(E)
                                                   // E: Angebot angenommen
 int i;
      if (likeli==NULL || dicht==NULL)
       {
        cerr << "\n::tmaxklein: Allokationsfehler\n";
        exit(1);
       }
 my_e_tmax->auspraeg(dicht, 1);                    // Dichte von Q          340
                                                   // in dicht speichern
      for(i=0;i<=my_e_tmax->getgrenze();i++)
      {
       /* Wahrscheinlichkeit, dass Q>Q-kritisch, wenn  */
       /* Agent tmax[i] hat, entspricht W(annehmen|t)  */
       likeli[i]=max(FNIVEAU,1-my_e_qrent->wahr(q_krit((*my_e_tmax)[i])));
       w_e+=likeli[i]*dicht[i];                    // W(E) nach Satz d.
                                                   // totalen
```

```
                                                    // Wahrscheinlichkeit
      }                                                                    350
      for (i=0;i<=my_e_tmax->getgrenze();i++)
       likeli[i]/=w_e;                               // Likelihood-Funktion:
                                                     // L=W(E|A)/W(E)
 my_e_tmax->posterior(likeli);                       // f-aposterior=f-apriori*L
 delete [] likeli;                                   // Speicher f. temp
 delete [] dicht;                                    // Arrays freigeben
}

/* Angebot errechnen                                 */
/* int m_aktion: Beschreibung der Aktion             */                   360
double agent::angebot(int &m_aktion, int bluff=-1)
{
 double m_delta,s_delta;                             // eigener, Partners
                                                     // Diskontfaktor
 double m_x,s_x;                                     // eigener Anteil,
                                                     // Anteil d. Partners
                                                     // am Verhandlungsspielraum
 double trest;                                       // Restverhandlungsdauer
 double p;                                           // Preis (Angebot)
 int tmp_qr, tmp_t;                                  // temp Q und t-max     370

      if (bluff && my_aktion!=ANNEHME)
         if (my_tmax-my_vzeit<=0)                    // Pruefen, ob Zeitlimit
           my_aktion=ABBRUCH_T;                      // noch weiter
                                                     // verhandeln erlaubt
         else
           if (my_e_tmax->mittel()-my_vzeit<=0)      // Pruefen, ob Zeitlimit
             my_aktion=ABBRUCH_NL;                   // des Partners weitere
                                                     // Verhandlungen erlaubt,
                                                     // wenn nicht, Abbruch,  380
                                                     // weil Zeitlimit
                                                     // nicht richtig gelernt
  m_aktion=my_aktion;                                // Aktion

   if(my_m_gebot==0)                                 // Bluff-Werte setzen,
    set_bluff();                                     // wenn neue
                                                     // Verhandlung begonnen
                                                     // wurde

      switch (my_aktion)                             // Entsprechend der Aktion 390
      {
       case BIETE:                                   // Angebot erstellen
            tmp_qr=my_qrent;                         // Q, tmax zwischen-
            tmp_t=my_tmax;                           // speichern, da
                 if (bluff)                          // Bluff-WErte ver-
                                                     // wendet werden,
                                                     // wenn Partner
```

```
                                        // getaeuscht werden soll
        {
         my_qrent=my_bluff_qr;          // Bluff-Werte verwenden
         my_tmax=my_bluff_t;
        }
    /* Eigene Restverhandlungsdauer und */
    /* eigenen Diskontfaktor ermitteln */
    trest=my_tmax-my_vzeit;
    m_delta=hoch(1-my_qrent/my_pm_basis,1.0/trest);
    /* Restverhandlungsdauer und Diskontfaktor*/
    /* des Partners errechenen */
    trest=my_e_tmax->mittel()-my_vzeit;
    s_delta=hoch(1-my_e_qrent->mittel()/my_pm_basis,1.0/trest);
    /* Eigener Anteil am Verhandlungsspielraum*/
    /* gemaess teilspielperfektem Gleichgewicht */
    m_x=log(s_delta)/(log(m_delta)+log(s_delta));
    s_x=1-m_x;                          // Anteil des Partners
    /* Aufteilung des Verhandlungsspielraums */
    p=my_pm_basis-m_x*(my_pm_basis-s_pc(my_e_qrent->mittel(),
      my_e_tmax->mittel()))+
      s_x*(m_pc()-my_pm_basis);
    p=runden(p,1);
    my_qrent=tmp_qr;                    // eigene Quasi-Rente und
    my_tmax=tmp_t;                      // t-max wieder herstellen
    my_m_gebot=p;
    p=runden(p,1);                      // Preis runden
    break;
case ABBRUCH_P:                         // Abbrechen, weil
                                        // Gegengebot den
                                        // Reservationspreis
                                        // ueberschreitet
case ABBRUCH_NL:                        // Abbrechen, weil
                                        // t-max d. Partners
                                        // nicht gelernt wurde
case ABBRUCH_C:                         // Abbrechen, weil
                                        // Gegengebot den
                                        // Konflikltpreis
                                        // ueberschreitet
    p=0;
    my_m_gebot=0;                       // Letzte
    my_s_gebot=0;                       // Angebote wieder 0
    break;
case ABBRUCH_T:                         // Abbrechen, weil
                                        // Transaktion nicht
                                        // in vorgesehener Frist
                                        // erreicht
    score(0);                           // Bewertung des Ergebnis
    p=0;
    my_m_gebot=0;                       // Letzte Gebot und des
```

```
            my_s_gebot=0;                                   // Partners wieder 0
            my_vzeit=0;                                     // Verhandlungszeit 0
            break;
        case ANNEHME:                                       // Gebot wird angenommen
            score(my_s_gebot);                              // Bewertung des Ergebnis
            p=my_s_gebot;                                   // Annahmepreis ist Gebot
                                                            // des Partners
            my_s_gebot=0;                                   // Letzte
            my_m_gebot=0;                                   // Gebote wieder 0
            my_vzeit=0;                                     // verhandlungszeit 0
            break;
        default:
            cerr << "\nkaufagent::angebot(): undefinierte Aktion";
            exit(1);
      }
 my_aktion=BIETE;                                           // Naechste Aktion
                                                            // immer anbieten
 return(p);
}

/* Gebot entgegen nehmen, Lernen und bestimmen             */
/* der naechsten Aktion                                    */
int agent::getgebot(double gebot, int s_aktion)
{
 /* Wahrscheinlichkeiten, mit der Q des Partners           */
 /* kleiner Erwartung bzw. t-max groesser ist              */
 double w_qr_klein, w_t_gross;
 int tmp_aktion=BIETE;                                      // f. Erstellen eins
                                                            // hypotetischen Angebots
 int weiter =0;                                             // f. Verhandlungssteuerung
 int tmp_qr;                                                // Zwischenspeicher f.
 int tmp_t;                                                 // eigenes Q und t-max

    if (my_tmax-my_vzeit<=0)                                // Gebot darf nicht mehr
     {                                                      // angenommen werden, wenn
      my_aktion=ABBRUCH_T;                                  // t-max ueberschritten
      return(weiter);                                       // Info ueber
     }                                                      // Abbruch an Partner

 my_s_gebot=runden(gebot,1);                                // Gebot entgegennehmen
 /* Wahrscheinlichkeit, das Q des Partners kleiner         */
 /* als erwartet                                           */
 w_t_gross=1-my_markt_tmax->wahr(my_e_tmax->mittel());
 /* Wahrscheinlichkeit, das t-max d. Partners groesser*/
 /* als erwartet                                           */
 w_qr_klein=my_markt_qr->wahr(my_e_qrent->mittel());
    switch (s_aktion)                                       // Handeln entspr. der
    {                                                       // Aktion d. Partners
     case BIETE:                                            // Partner macht Gegengebot
```

```
if (my_m_gebot!=0)                    // Lernen nur, wenn bereits
 {                                    // ein eigenes Angebot
                                      // vorlag
  tmp_qr=my_qrent;                    // Eigene Parameter
  tmp_t=my_tmax;                      // zwischenspeichern
  my_qrent=my_bluff_qr;               // Bluff-Werte verwenden
  my_tmax=my_bluff_t;
    /* Entscheiden, was gelernt wird  */
    if (w_qr_klein>w_t_gross && my_e_tmax->mittel()-my_vzeit>2)
      qrentklein();                   // Lernen, das Q kleiner
    else
      tmaxgross();                    // Lernen, das tmax grosser
  my_qrent=tmp_qr;                    // Wieder eigene Parameter
  my_tmax=tmp_t;                      // verwenden
 }
if (istbesser(my_s_gebot,m_pc()))     // Pruefen, ob Gebot
 {                                    // besser als Konfliktpreis
  tmp_t=my_vzeit;                     // Verhandlungszeit
                                      // zwischenspeichern
  /* Verhandlungszeit erhoehen und  */
  /* erstellen eines hypotetischen  */
  /* Angebots der naechsten Runde   */
  my_vzeit=(my_tmax-my_vzeit>=2?++my_vzeit:my_vzeit);
  angebot(tmp_aktion,0);
  my_vzeit=tmp_t;
      /* Annahmeentscheidung        */
      if ((istbesser(my_s_gebot,my_m_gebot)
          &&istbesser(gebot,p_res()))
          || (istbesser(gebot,p_res())&&my_tmax-my_vzeit<=1))
       {
        my_aktion=ANNEHME;            // Annehmen
        *my_e_qrent=*my_markt_qr;     // Erwartungen ueber
        *my_e_tmax=*my_markt_tmax;    // den Markt werden
                                      // revidiert
        /* Anpassen des Gegengebots   */
        my_m_gebot=2*my_pm_basis-my_s_gebot;
        tmp_qr=my_qrent;              // Zwischenspeichern
        tmp_t=my_tmax;                // der Parameter und
        my_qrent=my_bluff_qr;         // Verwenden d. Bluff-Werte
        my_tmax=my_bluff_t;
            /* Entscheiden, was       */
            /* gelernt werden soll    */
            /* bei schlechtem Preis:  */
            if (istbesser(*my_p_markt,gebot))
                /* Bei langer Dauer   */
                if (my_vzeit>runden(my_e_vzeit)+1)
                 tmaxgross();         // Lernen, das Q
                                      // groesser als erwartet
                else
```

```
                    {
                        // sonst lernen, dass Q kleiner,
                        // sofern Risikoaversion noch gegeben
                        if (my_e_qrent->mittel()>
                            my_theta*my_e_tmax->mittel()*gebot)
                          qrentklein();                                    550
                        else
                         qrentgross();
                     }
                else                        // lernen wenn Preis besser P*
                    if (my_e_tmax->mittel()>5
                       && my_vzeit<runden(my_e_vzeit))
                       tmaxklein();         // Lernen das t kleiner
                    else
                     qrentgross();          // Lernen das Q grroesser
                                                                           560
                  my_qrent=tmp_qr;          // Wieder eigene
                  my_tmax=tmp_t;            // Parameter verwenden
                  marktupdate();            // Markterwartungen
                                            // aktualisieren
                  my_s_gebot=gebot;         // angenommenen Preis in
                 }                          // my_s_gebot speichern
                else
                 weiter=-1;                 // Verhandlung geht in
                                            // naechste Runde, wenn
                                            // Gebot nicht angenommen      570
                                            // wird
           }
          else                              // Abbruch
           {                                // weil der
            my_aktion=ABBRUCH_C;            // Konfliktpreis erreicht
            *my_e_qrent=*my_markt_qr;       // kein lernen moeglich
            *my_e_tmax=*my_markt_tmax;
            marktupdate();                  // Markterwartungen
                                            // aktualisieren
           }                                                               580
      break;
case  ANNEHME:                              // Partner nimmt an
      score(gebot);                         // Bewertung des Ergebnis
      *my_e_qrent=*my_markt_qr;             // Erwartungen ueber den
      *my_e_tmax=*my_markt_tmax;            // den Markt revidieren
      /* Anpassen des Gegengebots     */
      my_m_gebot=2*my_pm_basis-my_s_gebot;
      tmp_qr=my_qrent;                      // Zwischenspeichern
      tmp_t=my_tmax;                        // eigener Parameter und
      my_qrent=my_bluff_qr;                 // Verwenden der               590
      my_tmax=my_bluff_t;                   // Bluff-Werte
          /* Entscheidungen, was gelernt werden */
          /* soll bei schlechtem Preis:              */
```

```
                if (istbesser(*my_p_markt,gebot))
                    /* Bei langer Dauer                    */
                    if (my_vzeit>runden(my_e_vzeit)+1)
                     tmaxgross();              // Lernen, das Q groesser als erwartet
                    else
                     {
                          // sonst lernen, dass Q kleiner, wenn                    600
                          // Risikoaversion noch gegeben
                          if (my_e_qrent->mittel()>
                              my_theta*my_e_tmax->mittel()*gebot)
                           qrentklein();
                          else
                           qrentgross();
                      }
                else                                    // lernen wenn Preis besser P*
                     if (my_e_tmax->mittel()>5 && my_vzeit<runden(my_e_vzeit))
                      tmaxklein();                      // Lernen das t kleiner  610
                     else
                      qrentgross();                     // Lernen das Q groesser
            my_qrent=tmp_qr;                            // Wieder eigene Parameter
            my_tmax=tmp_t;                              // verwenden
            marktupdate();                              // Markterwartungen
                                                        // aktualisieren
            my_aktion=BIETE;                            // Alles zuruecksetzen
            my_m_gebot=0;
            my_s_gebot=0;
            my_vzeit=0;                                                       620
            break;                                      // switch verlassen
        case ABBRUCH_P:                                 // Wenn Partner abbricht,
        case ABBRUCH_NL:                                // kann nicht
        case ABBRUCH_C:                                 // gelernt werden
        case ABBRUCH_T:
            *my_e_qrent=*my_markt_qr;
            *my_e_tmax=*my_markt_tmax;
            marktupdate();                              // Markterwartungen
                                                        // aktualisieren
            my_aktion=BIETE;                            // Alles zuruecksetzen   630
            my_m_gebot=0;
            my_s_gebot=0;
            break;                                      // switch verlassen
        default:
            cerr <<"\n:: gebot: undefinierte Aktion\n";
            exit(1);
      }
  return(weiter);
}
                                                                              640
/* Aus den in einer Verhandlung gemachten Erfahrungen*/
/* Rueckschluesse auf den Markt ziehen                */
```

```
void agent::marktupdate()
{
 double *dicht_mq, *dicht_mt;                    // Felder f. temp Dichte
 double *dicht_q, *dicht_t;
 int i, ok=-1;


 dicht_mq = new double[my_e_qrent->getgrenze()+1];   // Array f. Markt-Q
 dicht_q = new double[my_e_qrent->getgrenze()+1];    // Array f. Q aus
                                                     // Verhandlung
 dicht_mt = new double[my_e_tmax->getgrenze()+1];    // Array f. Markt-t-max
 dicht_t = new double[my_e_tmax->getgrenze()+1];     // Array f. t-max
                                                     // aus Verhandlung
 ok=(dicht_mq!=NULL)&&(dicht_q!=NULL)&&(dicht_mt!=NULL)&&(dicht_t!=NULL);
     if (!ok)
     {
      cerr << "\n:marktupdate: Allokationsfehler1\n";
      exit(1);
     }

 /* Um verzerrungen wegen Binaerarithmetik gering zu halten: */
 my_markt_qr->auspraeg(dicht_mq,100);                // alte Dichte
                                                     // von Q-Markt * 100
 my_e_qrent->auspraeg(dicht_q,100);                  // neue Dichte
                                                     // von Q-Markt * 100
 my_markt_tmax->auspraeg(dicht_mt,100);              // alte Dichte
                                                     // von t-Markt * 100
 my_e_tmax->auspraeg(dicht_t,100);                   // neue Dichte
                                                     // von t-Markt * 100
     /* Alte Dichte mit neuer Dichte aktualisieren      */
     for(i=0;i<=my_e_qrent->getgrenze();i++)
      dicht_mq[i]+=(dicht_q[i]-dicht_mq[i])*GEWICHT; // Quasi-Rente
     for(i=0;i<=my_e_tmax->getgrenze();i++)
      dicht_mt[i]+=(dicht_t[i]-dicht_mt[i])*GEWICHT; // Zeitlimit
 /* Alte Erwartungen loeschen                       */
 delete my_markt_qr;
 delete my_markt_tmax;
 delete my_e_qrent;
 delete my_e_tmax;
 /* Neue Erwartungen erzeugen mit aktualisierter    */
 /* Auspraegung der Zufallsvariablen                */
 /* Quasi-Renten:                                   */
 my_markt_qr= new zvar(Q_UNTEN,Q_OBEN,dicht_mq);
 ok&=my_markt_qr!=NULL;
 my_e_qrent= new zvar(Q_UNTEN,Q_OBEN,dicht_mq);
 ok&=my_e_qrent!=NULL;
 /* Zeitlimits:                                     */
 my_markt_tmax=new zvar(TMAX_UNTEN,TMAX_OBEN,dicht_mt);
 ok&=my_markt_tmax!=NULL;
```

```
 my_e_tmax= new zvar(TMAX_UNTEN,TMAX_OBEN,dicht_mt);
 ok&=my_e_tmax!=NULL;
      if (!ok)
      {
       cerr << "\n:marktupdate: Allokationsfehler2\n";
       exit(1);
      }
 /* Speicher f. temp Arrays freigeben                    */
 delete [] dicht_mq;                                                    700
 delete [] dicht_mt;
 delete [] dicht_q;
 delete [] dicht_t;
}


/* ******************KAUFAGENT******************************************** */

/* Konstruktoren f. den Kaeufer                            */
/* Standardkonstruktor                                     */            710
kaufagent::kaufagent() : agent()
{
 my_p_eff=my_pm_basis*(1+my_e_vzeit*my_theta);          // Anfaenglicher
                                                        // Effektivpreis
}


/* Konstruktor mit QR, Zeitlimit und E(QR), E(Zeit)        */
kaufagent::kaufagent(int qrent, int tmax, int e_qrent, int e_tmax,
                     double theta) :                                    720
                    agent(qrent, tmax, e_qrent, e_tmax, theta)
{
 my_p_eff=my_pm_basis*(1+my_e_vzeit*my_theta);          // Anfaenglicher
                                                        // Effektivpreis
}

/* Berechnen des eignen Konfliktpreises                    */
double kaufagent::m_pc()
{
 double a,b,alpha;                                      // Parameter      730
                                                        // Nutzenfunktion
 /* Parameter der Nutzenfunktion muessen neu berechnet   */
 /* werden, da u.U. die Bluff-Werte verwendet werden     */
 nutzenpar(a,b,alpha,my_qrent,my_tmax);
 return(pc_kauf(a, b, alpha, my_qrent, my_tmax));       // pc_kauf aufrufen
}

/* Konfliktpreis des Verhandlungspartners berechnen        */
double kaufagent::s_pc(double qr, double tmax)
{                                                                       740
```

```
 double a, b, alpha, pc, tmp_vzeit;                    // Parameter d. Nutzen-
 nutzenpar(a,b,alpha,qr, tmax);                        // funktion errechnen
 /* Verhandlungszeit muss an das Verhaeltnis Verkaeufer */
 /* zu Kaeufer angepasst werden                          */
 tmp_vzeit=my_e_vzeit;                                 // Zwischenspeichern
 my_e_vzeit*=min(1,double(*my_anz_vkkauf)/double(*my_anz_kauf));
 pc=pc_vkkauf(a, b, alpha, qr, tmax);                  // pc_vkkauf aufrufen
 my_e_vzeit=tmp_vzeit;                                 // E(V-Zeit) wieder
                                                       // wieder herstellen
 return(pc);                                                                  750
}

/* Score der Verhandlung berechnen                     */
void kaufagent::score(double preis)
{
 /* Erwartete Verhandlungszeit aktualisieren          */
 my_e_vzeit=my_e_vzeit*(1-GEWICHT)+my_vzeit*GEWICHT;
      if (preis>0)
        {
         my_score_eff+=preis*(1+my_vzeit*my_theta);                           760
         my_score+=preis;
         my_anz_trans++;
        }
      else
       my_score_eff+=my_pm_basis*my_vzeit*my_theta;
}

/* Neustarten der Verhandlung                          */
void kaufagent::neustart()
{                                                                             770
 my_pm_basis=runden(my_pm_basis*(1-GEWICHT)+*my_p_markt*(GEWICHT),1);
}

/* Pruefen, ob in den Markt eingetreten werden soll    */
int kaufagent::eintritt()
{
 return(k_auftrag>my_anz_trans);
}

/* ******************VERKAUFAGENT***************************************** */ 780

/* Konstruktoren f. den Verkaeufer                     */
/* Standardkonstruktor                                 */
verkaufagent::verkaufagent() : agent()
{
 my_p_eff=my_pm_basis*(1-my_e_vzeit*my_theta);         // Anfaenglicher
                                                       // Effektivpreis
}
```

```
                                                                        790
/* Konstruktor mit QR, Zeitlimit und E(QR), E(Zeit)          */
verkaufagent::verkaufagent(int qrent, int tmax, int e_qrent, int e_tmax,
                           double theta) :
                      agent(qrent, tmax, e_qrent, e_tmax, theta)
{
 my_p_eff=my_pm_basis*(1-my_e_vzeit*my_theta);        // Anfaenglicher
                                                      // Effektivpreis
}

/* Berechnen des eignen Konfliktpreises                      */      800
double verkaufagent::m_pc()
{
 double a,b,alpha;                                    // Parameter der
                                                      // Nutzenfunktion
 /* Parameter der Nutzenfunktion muessen neu berechnet     */
 /* werden, da u.U. die Bluff-Werte verwendet werden       */
 nutzenpar(a,b,alpha,my_qrent,my_tmax);
 return(pc_vkkauf(a, b, alpha, my_qrent, my_tmax)); // Aufruf von pc_vkkauf
}
                                                                        810
/* Konfliktpreis Verhandlungspartners (Kaeufer)berrechen */
double verkaufagent::s_pc(double qr, double tmax)
{
 double a, b, alpha, pc, tmp_vzeit;                   // Parameter der Nutzen-
 nutzenpar(a,b,alpha,qr, tmax);                       // funktion errechnen
 /* Verhandlungszeit muss an das Verhaeltnis Verkaeufer */
 /* zu Kaeufer angepasst werden                            */
 tmp_vzeit=my_e_vzeit;                                // Zwischenspeichern
 my_e_vzeit*=min(1,double(*my_anz_kauf)/double(*my_anz_vkkauf));
 pc=pc_kauf(a, b, alpha, qr, tmax);                    // pc_kauf aufrufen  820
 my_e_vzeit=tmp_vzeit;                                 // E(V-Zeit)wieder
                                                       // herstellen
 return(pc);
}

/* Score der Verhandlung berechnen                           */
void verkaufagent::score(double preis)
{
 /* Erwartete Verhandlungszeit aktualisieren              */
 my_e_vzeit=my_e_vzeit*(1-GEWICHT)+my_vzeit*GEWICHT;                    830
      if (preis>0)
       {
        my_score_eff+=preis*(1-my_vzeit*my_theta);
        my_score+=preis;
        my_anz_trans++;
       }
      else
       my_score_eff-=my_pm_basis*my_vzeit*my_theta;
```

```
}
                                                                                    840
/* Neustarten der Verhandlung                                          */
void verkaufagent::neustart()
{
 my_pm_basis=runden(my_pm_basis*(1−GEWICHT)+*my_p_markt*(GEWICHT),1);
}

/* Pruefen, ob in den Markt eingetreten werden soll                    */
int verkaufagent::eintritt()
{
 return(v_auftrag>my_anz_trans);                                                    850
}

/* Transferoperator f. Agent                                           */
ostream& operator<<(ostream& ostr, const agent& agt)
{
 ostr << setiosflags(0x1000);
 ostr << setprecision(2);
 ostr << setw(10);
 ostr << setprecision(4);
 ostr << agt.name << ";" << setw(3) << agt.ident;                                   860
 ostr << ";" << setw(2) << agt.my_qrent << ";" << setw(2) << agt.my_tmax;
 ostr << ";"<< setw(8)<<agt.my_e_qrent−>mittel();
 ostr << ";" << setw(8)<<agt.my_e_tmax−>mittel();
 ostr << ";" << setw(2)<<agt.my_bluff_qr << ";" << setw(2)<< agt.my_bluff_t;
 ostr << ";" << setw(8) << agt.my_e_vzeit << ";" << setw(2)<< agt.my_tzeit;
 ostr << ";" << setw(8) << agt.my_score;
 ostr << ";" << setw(8) << agt.my_score_eff;
 ostr << ";" << setw(4) << agt.my_anz_trans << "\n";
}
```

C.3 Verhandlungssteuerung

C.3.1 Header-Datei marktpt.h

```
/* Deklarationen allgemein für Naschmarkt                */
/* Autor: Bernhard E. Beran                              */
/* Datum: 01.09.2000                                     */


/* Konstanten                                            */
#define THETA_H 0.03                                     // hohe Transaktionskosten
#define THETA_N 0.015                                    // niedere Transaktions-
                                                         // kosten in Prozent
                                                         // des Marktpreises

#define PSI 0.5                                          // Wahrscheinlichkeit
                                                         // von P* sofort
#define PM_BASIS 100.0                                   // Basismarktpreis

#define ANZ_KAUF 249                                     // Anzahl der Kaeufer-1
#define ANZ_VKKAUF 249                                   // Anzahl der Verkaefuer-1
                                                         // wegen Trace-Agenten!

#define K_AUFTRAG 50                                     // Auftrag Kaufer
#define V_AUFTRAG 50                                     // Auftrag Verkaufer

#define E_VZEIT 3                                        // initiale erwartete
                                                         // Verhandlungsdauer

/* Konstanten für die Nutzenberechnungen                 */
#define U_NULL 1                                         // Nutzen von U(Y-T)=U(0)
#define U_P_MARKT 101                                    // Nutzen von U(PM_BASIS)


/* Konstanten für die Aktion eines Agenten               */
#define BIETE 0                                          // Agent macht ein Angebot
#define ABBRUCH_T 1                                      // Abbruch wegen Zeitlimit
#define ANNEHME 2                                        // Agent nimmt Gebot an
#define ABBRUCH_C 3                                      // Abruch weil
                                                         // Konfliktpunkt erreicht
#define ABBRUCH_NL 4                                     // Abbruch wegen
                                                         // nicht lernen
#define ABBRUCH_P 5                                      // Abbruch, weil
                                                         // Reservationspreis
                                                         // nicht erreicht
                                                         // werden kann
```

```
/* Konstanten, die den Wertebereich der Quasirente */
/* festlegen                                       */
#define R_UNTEN 0                                  // kleinseteRisikopraemie
#define R_OBEN 40                                  // groesste Risikopraemie
#define R_SCHNITT 20                               // Durchschn.Risikopraemie
#define TMAX_UNTEN 0                               // Untergrenze Zeitlimit
#define TMAX_OBEN 25                               // Obergrenze Zeitlimit
#define TMAX_SCHNITT 12                            // Durchschn. Zeitlimit
#define Q_UNTEN 0                                  // Minimale Quasirente
#define Q_OBEN 100                                 // Maximale Quasirente
```

C.3.2 C++ Code marktpt.cpp

```
/* Markt - Prototyp f. Naschmarkt Objektcode */
/* Marktsimulation                           */
/* Autor: Bernhard E. Beran                  */
/* Datum: 28.10.2000                         */

#include <iostream.h>
#include <iomanip.h>
#include "meinmath.h"
#include "agentpt.h"
#include "marktpt.h"
#include <list>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <string.h>
#include <conio.h>
#include <fstream.h>

ofstream agtdat;                           // Ausgabedatei f.
                                           // Agentendaten
ofstream mktdat;                           // Ausgabedatei f.
                                           // Marktdaten
ofstream trcdat;                           // Ausgabedatei f.
                                           // Einzelverhandlung

using namespace std;
typedef list<agent*> agentenliste;         // Listentyp f. Agenten
typedef list<agent*>::iterator agenteniter; // Iterator f.
                                           // Agentenliste
```

```
/* Deklarationen f.      die Marktvariablen                    */

double p_markt = PM_BASIS;                                    // Marktpreis
double theta = THETA_H;                                       // Transaktionskosten i.P
int anz_kauf = ANZ_KAUF;                                      // Kaeuferzahl
int anz_vkkauf = ANZ_VKKAUF;                                  // Verkaeuferzahl

/* Deklarationen f.      die Agentenparameter                  */
int r_unten = R_UNTEN;                                        // UntergrenzeRisikopraemie
int r_oben = R_OBEN;                                          // Obergrenze Risikopraemie
int r_schnitt = R_SCHNITT;                                    // Durchschn. Risikopraemie
int tmax_unten = TMAX_UNTEN;                                  // Untergrenze Zeitlimit
int tmax_oben = TMAX_OBEN;                                    // Obergrenze Zeitlimit
int tmax_schnitt = TMAX_SCHNITT;                              // Durchschn. Zeitlimit

int k_auftrag = K_AUFTRAG;                                    // Auftrag Kaeufer
int v_auftrag = V_AUFTRAG;                                    // Auftrag Verkaeufer


/* Variablen f. Marktdaten                                    */
int zeit=0;                                                   // Zeitzaehler

/* Container f. Agenten                                       */
agentenliste kaufer;                                          // Kaufer
agentenliste verkaufer;                                       // Verkaufer
agentenliste in_kaufer;                                       // Kaeufer im Markt
agentenliste in_verkaufer;                                    // Verkaeufer im Markt
agentenliste m_kaufer;                                        // gematchte Kaeufer
agentenliste m_verkaufer;                                     // gematchte Verkaeufer
agentenliste neustarter;                                      // Agenten, die Trans.
                                                              // abgeschlossen haben

list<int> starter;                                            // 0=Kaeufer begind,
                                                              // 1=Verkaeufer begind


/* Hilfsfunktion f. rand()-Zufallszahlen                      */
inline int zzahl(int u_grenze, int o_grenze)
{
 return(u_grenze+(abs(rand() % (o_grenze-u_grenze+1))));
}


/* Einlesen des Zeitlimits                                    */
int lese_tmax()
{
 int t=tmax_schnitt;
 cout << setiosflags(0x1000) << setprecision(0);
      do
```

```
      {                                                                    80
       cout << "Zeitlimit pro Transaktion (5 <= t <= "
            << setw(2) << tmax_oben-5 <<"): ";
       cin >> t;
      }
     while(t<5 || t>tmax_oben-5);
 return(t);
}

/* Einlesen des Reservationspreises                                  */
/* t ... Zeitlimit pro Transaktion                                   */   90
/* wer = 0 ... f. Kaeufer                                            */
/* wer <> 0 ... f. Verkaeufer                                        */
/* Liefert Risikopraemie als Ergebnis                                */
int lese_pres(int t, int wer)
{
 int p_u, p_o, p, r;
     if (wer)
      {                                                 // Limits Verkaeufer
       p_u=int(runden(PM_BASIS*(1-theta*t)-r_oben+2));
       p_o=int(runden(PM_BASIS*(1-theta*t)-2));                            100
      }
     else
      {                                                 // Limits Kaeufer
       p_u=int(runden(PM_BASIS*(1+theta*t)+2));
       p_o=int(runden(PM_BASIS*(1+theta*t)+r_oben-2));
      }
 cout << setiosflags(0x1000) << setprecision(0);
     do
      {
       cout << "Reservationspreis (" << setw(3) << p_u << " <= p <= "      110
            << setw(3) << p_o << "): ";
       cin >> p;
      }
     while(p<p_u || p>p_o);
     if (wer)
      r=int(runden(PM_BASIS*(1-theta*t)-p));          // Risikopraemie VKaufer
     else
      r=int(runden(p-PM_BASIS*(1+theta*t)));          // Risikopraemie Kaufer
 return(r);
}                                                                          120

/* Aufbau der Kaeufer                                                */
void bau_kaufer(int anzahl)
{
 agent* kauf;
 int i, anz;
 int q, t, eq, et;
 int mr, mt, mer, met;
```

```
 cout << "Parameter der Kaeufer\n";
 cout << "Gruppengroesze mit hohen Transaktionskosten\n";
   do
    {
      cout << "(0 <= Grosze <= "<< anzahl << "): ";
      cin >> anz;
    }
   while (anz < 0 || anz > anzahl);
 theta=THETA_H;
 cout << "Eigen Parameter\n";
 mt=lese_tmax();
 mr=lese_pres(mt,0);
 cout << "Erwartungen ueber Verkaeufer\n";
 met=lese_tmax();
 mer=lese_pres(met,-1);
     for (i=1;i<=anz;i++)                          // anz Kaeufer aufbauen
      {
       q=int(binomwert(r_unten,r_oben,mr));
       eq=int(binomwert(r_unten,r_oben,mer));
       t=int(binomwert(tmax_unten,tmax_oben,mt));
       et=int(binomwert(tmax_unten,tmax_oben,met));
       q+=int(runden(t*theta*p_markt)+1);
       eq+=int(runden(et*theta*p_markt)+1);
       kauf=new kaufagent(q,t,eq,et,theta);
       kauf->ident=i;
       strcpy(kauf->name,"KaufH");
       kauf->neustart();
       kaufer.push_back(kauf);
      }
     theta=THETA_N;
     for (i=anz+1;i<=anzahl;i++)                    // restl. Kaeufer
      {                                             // mit niederem Theta
       q=int(binomwert(r_unten,r_oben,mr));
       eq=int(binomwert(r_unten,r_oben,mer));
       t=int(binomwert(tmax_unten,tmax_oben,mt));
       et=int(binomwert(tmax_unten,tmax_oben,met));
       q+=int(runden(t*theta*p_markt)+1);
       eq+=int(runden(et*theta*p_markt)+1);
       kauf=new kaufagent(q,t,eq,et,theta);
       kauf->ident=i;
       strcpy(kauf->name,"KaufN");
       kauf->neustart();
       kaufer.push_back(kauf);
      }
}

/* Aufbau der Verkaeufer */
void bau_verkaufer(int anzahl)
{
```

```
agent* verkauf;
int i, anz;
int q, t, eq, et;                                                    180
int mr, mt, mer, met;
cout << "Parameter der Verkaeufer\n";
cout << "Gruppengroesze mit hohen Transaktionskosten\n";
  do
   {
     cout << "(0 <= Groesze <= "<< anzahl << "): ";
     cin >> anz;
   }
  while (anz < 0 || anz > anzahl);
theta=THETA_H;                                                       190
cout << "Eigen Parameter\n";
mt=lese_tmax();
mr=lese_pres(mt,-1);
cout << "Erwartungen ueber Kaeufer\n";
met=lese_tmax();
mer=lese_pres(met,0);
    for (i=1;i<=anz;i++)                          // anz Verkaeufer aufbauen
     {
       q=int(binomwert(r_unten,r_oben,mr));
       eq=int(binomwert(r_unten,r_oben,mer));                        200
       t=int(binomwert(tmax_unten,tmax_oben,mt));
       et=int(binomwert(tmax_unten,tmax_oben,met));
       q+=int(runden(t*theta*p_markt)+1);
       eq+=int(runden(et*theta*p_markt)+1);
       verkauf=new verkaufagent(q,t,eq,et,theta);
       verkauf->ident=i;
       strcpy(verkauf->name,"VerkH");
       verkauf->neustart();
       verkaufer.push_back(verkauf);
     }                                                               210
    theta=THETA_N;
    for (i=anz+1;i<=anzahl;i++)                    // restl. verkaeufer
     {                                             // mit niederem Theta
       q=int(binomwert(r_unten,r_oben,mr));
       eq=int(binomwert(r_unten,r_oben,mer));
       t=int(binomwert(tmax_unten,tmax_oben,mt));
       et=int(binomwert(tmax_unten,tmax_oben,met));
       q+=int(runden(t*theta*p_markt)+1);
       eq+=int(runden(et*theta*p_markt)+1);
       verkauf=new verkaufagent(q,t,eq,et,theta);                    220
       verkauf->ident=i;
       strcpy(verkauf->name,"VerkN");
       verkauf->neustart();
       verkaufer.push_back(verkauf);
     }
}
```

```
/* Agenten, die bereit sind, in den markt schicken          */
void eintreten(agentenliste& ausmarkt, agentenliste& inmarkt)
{
 agenteniter ausagent=ausmarkt.begin();          // Iterator erster Agent
 agenteniter n_ausagent;                         // Iterator naechster

     while(ausagent!=ausmarkt.end())
      {
       n_ausagent=ausagent;
       n_ausagent++;
            if ((*ausagent)->eintritt())         // Agent eintrittswillig?
             inmarkt.splice(inmarkt.end(),ausmarkt,ausagent);
       ausagent=n_ausagent;

      }
}

/* Agenten zufaellig auswaehlen und matchen                 */
void match()
{
 agenteniter haendler;                           // Iterator auf Agent
 int i, pos, j=0;
      /* Solange matchen, bis mindestens eine Liste         */
      /* (Kaeufer oder Verkaufer) leer ist                  */
      while(in_kaufer.size()>0 && in_verkaufer.size()>0)
       {
        pos=zzahl(0,in_kaufer.size()-1);             // Kaeufer waehlen
        haendler=in_kaufer.begin();
             for (i=0;i<pos;i++)
              haendler++;
        /* Kaeufer zu den gematchten stellen                */
        m_kaufer.splice(m_kaufer.end(),in_kaufer,haendler);
        pos=zzahl(0,in_verkaufer.size()-1);          // Verkaeufer waehlen
        haendler=in_verkaufer.begin();
             for(i=0;i<pos;i++)
              haendler++;
        /* Veraeufer zu den gematchten stellen              */
        m_verkaufer.splice(m_verkaufer.end(),in_verkaufer,haendler);
        starter.push_back(zzahl(0,1));               // Beginner auswaehlen
       }
}

/* Agendendaten protokollieren                             */
void agentenstat(agentenliste& liste, int r, char* text)
{
 agenteniter iter=liste.begin();
   while(iter!=liste.end())
```

```
      {
       agtdat << text << ";" << r << ";";
       agtdat << **iter;
       iter++;
      }                                                              280
}

/* Erwartete Quasirente aufsummieren f. Protokoll        */
double eqrent(agentenliste& liste)
{
 agenteniter iter=liste.begin();
 double summe=0;
       while(iter!=liste.end())
        {
         summe+=(*iter)->get_e_qrent();                              290
         iter++;
        }
 return(summe);
}

/* Erwartetes Zeitlimit aufsummieren f. Protokoll        */
double etmax(agentenliste& liste)
{
 agenteniter iter=liste.begin();
 double summe=0;                                                     300
       while(iter!=liste.end())
        {
         summe+=(*iter)->get_e_tmax();
         iter++;
        }
 return(summe);
}

/* Steuerung einer Verhandlungsrunde                     */
void verhandeln()                                                    310
{
 double tmp_p_markt=0;
 double gebot;
 double eqrent_k=0, eqrent_v=0;
 double etmax_k=0, etmax_v=0;
 int a_k=0, a_v=0;
 int c_k=0, c_v=0;
 int wer=1;
 double anzahl;
 int aktion, tmp_aktion;                                             320
 agenteniter anbieter, anfrager;                  // anbieter: stellt Angebot
                                                  // anfrager: nimmt Angebot
                                                  // entgegen
 agenteniter kauf=m_kaufer.begin(), n_kauf;
```

```
agenteniter verkauf=m_verkaufer.begin(), n_verkauf;
list<int>::iterator start=starter.begin(), n_start;
/* Variablen f. Protokoll der Aktionen pro Runde        */
double min_p_markt=0;                                   // Kleinster Preis p. Runde
double max_p_markt=0;                                   // Groesster Preis p. Runde
int s_k=0, s_v=0;                                                                    330
int abbr_t=0;                                           // Abbruch wegen Zeitlimit
int abbr_c=0;                                           // Abbruche: Pc erreicht
int abbr_nl=0;                                          // Abbruch nicht Lernen
int abbr_p=0;                                           // Abbruch PRes erreicht
int anz_trans=0;                                        // Anzahl der Transaktionen

n_kauf=kauf;
n_verkauf=verkauf;
n_start=start;
anz_kauf=m_kaufer.size()+in_kaufer.size();              // akt. Anzahl der          340
                                                        // Kaeufer im Markt
anz_vkkauf=m_verkaufer.size()+in_verkaufer.size(); // akt. Anzahl der
                                                        // Verkaufer im Markt
    while(kauf!=m_kaufer.end() && verkauf!=m_verkaufer.end()
          && start!=starter.end())
     {
      // Inkrementieren der next-Iteratoren
      n_kauf++;
      n_verkauf++;
      n_start++;                                                                     350
           if ((zeit % 2) == *start)                  // bestimmen, wer anbietet
            {
             anbieter=kauf;                           // Kaeufer macht Angebot
             anfrager=verkauf;
             s_k++;
             wer=1;
            }
           else
            {
             anbieter=verkauf;                        // Verkaeufer macht Angebot 360
             anfrager=kauf;
             s_v++;
             wer=-1;
            }
      gebot=(*anbieter)->angebot(aktion);           // Angebot machen

      // TRACE - Wenn Traceagent an der Reihe ist
      //              Daten Protokollieren
           if (!strcmp((*anbieter)->name,"VerkTR")&&
               (*anbieter)->ident==ANZ_VKKAUF+1 ||                                  370
               !strcmp((*anfrager)->name,"VerkTR")&&
               (*anfrager)->ident==ANZ_VKKAUF+1 )
            {
```

```
        trcdat << zeit << ";" << p_markt << ";" << gebot
               << ";" << aktion << ";" << **anbieter;
      }
//ENDTRACE

    if (!(*anfrager)->getgebot(gebot,aktion))// Wenn anfrager nicht
     {                                       // nicht weiter verhandelt      380
        if (aktion==0)                       // wenn sich anfrager zum
                                             // Abbruch entscheidet
         {                                   // den Partner informieren
           gebot=(*anfrager)->angebot(aktion);

// TRACE - Daten des Traceagents protokollieren
    if (!strcmp((*anbieter)->name,"VerkTR")&&
        (*anbieter)->ident==ANZ_VKKAUF+1 ||
        !strcmp((*anfrager)->name,"VerkTR")&&
        (*anfrager)->ident==ANZ_VKKAUF+1 )   390
     {
        trcdat << zeit << ";" << p_markt << ";" << gebot
               << ";" << aktion << ";" << **anfrager;
      }
//ENDTRACE
           (*anbieter)->getgebot(gebot,aktion);
           wer*=-1;
         }
        switch (aktion)                      // Was ist passiert
         {                                                                   400
          case ANNEHME:                      // Anfrager hat angenommen
               anz_trans++;                  // Transaktionszahl erhohen
               tmp_p_markt+=gebot;           // Marktpreis aufsummieren
               min_p_markt=min_p_markt==0?gebot:min(gebot,min_p_markt);
               max_p_markt=max(gebot,max_p_markt);
                    if (wer>0)
                      a_k++;
                    else
                      a_v++;
               (*anbieter)->neustart();      // Neue Verhandlung             410
               (*anfrager)->neustart();      // Neue Verhandlung
               break;
          case ABBRUCH_T:                    // Zeitlimit ueberschritten
               abbr_t++;
               break;
          case ABBRUCH_C:                    // Pc ueberschritten
               abbr_c++;
               if (wer>0)
                 c_k++;
               else                                                          420
                 c_v++;
               break;
```

```
                    case ABBRUCH_NL:                    // nicht genug gelernt
                        abbr_nl++;
                        break;
                    case ABBRUCH_P:                     // Preis>Pres
                        abbr_p++;
                        break;
                    default:
                        cerr << "\nverhandeln: Ungueltige Aktion: "          430
                             << aktion << "\n";
                        cerr << **anbieter;
                        cerr << **anfrager;
                        exit(1);
                    }
                // Agenten zurueckstellen, damit neu gematcht werden kann
                kaufer.splice(kaufer.end(),m_kaufer,kauf);
                verkaufer.splice(verkaufer.end(),m_verkaufer,verkauf);
                starter.erase(start);
                }                                                               440
            kauf=n_kauf;
            verkauf=n_verkauf;
            start=n_start;
           }
/* Errechnen der Protokolldaten                        */
       if (anz_trans>0)
         p_markt=tmp_p_markt/anz_trans;
p_markt=runden(p_markt,2);
cout << setiosflags(0x1000) << setprecision (2);
eqrent_k=eqrent(kaufer)+eqrent(in_kaufer)+eqrent(m_kaufer);                     450
etmax_k=etmax(kaufer)+etmax(in_kaufer)+etmax(m_kaufer);
anzahl=kaufer.size()+in_kaufer.size()+m_kaufer.size();
eqrent_k=eqrent_k/anzahl;
etmax_k=etmax_k/anzahl;
eqrent_v=eqrent(verkaufer)+eqrent(in_verkaufer)+eqrent(m_verkaufer);
etmax_v=etmax(verkaufer)+etmax(in_verkaufer)+etmax(m_verkaufer);
anzahl=verkaufer.size()+in_verkaufer.size()+m_verkaufer.size();
eqrent_v=eqrent_v/anzahl;
etmax_v=etmax_v/anzahl;
/* Ausgeben der Daten                                  */                      460
     cout << setw(3) << zeit % 1000 << ";" << setw(6) << p_markt
          << ";" << setw(6) << min_p_markt << ";" << setw(6) << max_p_markt
          << ";" << setw(2) << anz_kauf - anz_vkkauf
          << ";" << setw(3) << anz_trans
          << ";" << setw(3) << a_k-a_v
          << ";" << setw(2) << abbr_t
          << ";" << setw(2) << c_k << ";" << setw(2) << c_v << ";" << setw(2)
          << abbr_nl
          << ";" << setw(1) << abbr_p
          << ";" << setw(6) << eqrent_k << ";" << setw(6) << eqrent_v        470
          << ";" << setw(6) << etmax_k << ";" << setw(6) << etmax_v
```

```
          << "\n";
  mktdat << setw(3) << zeit << ";" << setw(6) << p_markt
         << ";" << setw(6) << min_p_markt << ";" << setw(6) << max_p_markt
         << ";" << setw(3) << anz_kauf << ";" << setw(3) << anz_vkkauf
         << ";" << setw(3) << anz_trans
         << ";" << setw(3) << a_k-a_v
         << ";" << setw(2) << abbr_t
         << ";" << setw(2) << c_k << ";" << setw(2) << c_v << ";" << setw(2)
         << abbr_nl << ";" << setw(1) << abbr_p                          480
         << ";" << setw(6) << eqrent_k << ";" << setw(6) << eqrent_v
         << ";" << setw(6) << etmax_k << ";" << setw(6) << etmax_v
         << "\n";
}

/* Zeit erhoehen (Verhandlungs u. Totalzeit)                    */
void tick(agentenliste& liste)
{
 agenteniter iter=liste.begin();
   while (iter!=liste.end())                                            490
    {
     (*iter)->ticker();
     iter++;
    }
}

void main()
{
 int stop=0;
 int tq, tt, teq, tet;                          // f. Trace-Agent      500
 char thl;
 agent* trc_agent;
 agtdat.open("agtdat.txt", ios::out|ios::trunc);   // Datei Agentendaten
 mktdat.open("mktdat.txt", ios::out|ios::trunc);   // Datei Marktdaten
 trcdat.open("trcdat.txt", ios::out|ios::trunc);   // Datei Verhandlung

 cout << "NASCHMARKT - MARKTSIMULATION\n";
 cout << "Diplomarbeit von Bernhard E. Beran MatNr.9250011 2001\n\n";
 cout << "Auftrag Kaeufer - zu kaufende Stueckzahl: ";
 cin >> k_auftrag;                                                      510
 cout << "Auftrag Verkaeufer - zu verkaufende Stueckzahl: ";
 cin >> v_auftrag;
 cout << "\n";
 srand(729);
 bau_kaufer(ANZ_KAUF);
 srand(729);
 bau_verkaufer(ANZ_VKKAUF);
 cout << "Eigenschaften des Trace-Agent (Verkaeufer)\n";
  do
   {                                                                    520
```

```
    while (cin.get() != '\n');
    cout << "Hohe oder niedere Transaktionskosten? (H,N): ";
    thl=cin.get();
   }
  while (!(thl=='h' || thl=='H' || thl=='N' || thl=='n'));
  if (thl=='h' || thl=='H')
    theta=THETA_H;
  else
    theta=THETA_N;
tt=lese_tmax();                                                          530
tq=int(runden(PM_BASIS*theta*tt))+lese_pres(tt,-1);
cout << "Erwartungen des Trace-Agent ueber Kaeufer\n";
tet=lese_tmax();
teq=int(runden(PM_BASIS*theta*tet))+lese_pres(tet,0);
trc_agent = new verkaufagent(tq,tt,teq,tet,theta);
trc_agent->ident=ANZ_VKKAUF+1;
strcpy(trc_agent->name,"VerkTR");
trc_agent->neustart();
verkaufer.push_back(trc_agent);
trc_agent = new kaufagent(tq,tt,teq,tet,theta);                          540
trc_agent->ident=ANZ_KAUF+1;
strcpy(trc_agent->name,"KaufTR");
trc_agent->neustart();
kaufer.push_back(trc_agent);

cout << "\nNASCHMARKT - Straten der Simulation";
cout << "\nKaeufer:    " << kaufer.size() << "\n";
cout << "Verkaeufer: " << verkaufer.size() << "\n";

srand(time(NULL));                                                       550
agentenstat(kaufer,zeit, "NiM");
agentenstat(verkaufer,zeit,"NiM");
agentenstat(in_kaufer,zeit,"NMat");
agentenstat(in_verkaufer,zeit,"NMat");
agentenstat(m_kaufer,zeit,"iV");
agentenstat(m_verkaufer,zeit,"iV");
    while (!stop)
     {

       eintreten(kaufer, in_kaufer);                                     560
       eintreten(verkaufer, in_verkaufer);
       stop=in_kaufer.size()<=5 && in_verkaufer.size()<=5 ||
            in_kaufer.size()<=0 || in_verkaufer.size()<=0;
       tick(in_kaufer);
       tick(in_verkaufer);
       tick(m_kaufer);
       tick(m_verkaufer);
       zeit++;
```

```
                    if (zeit%50==0)                                          570
                      {
                       agentenstat(kaufer,zeit, "NiM");
                       agentenstat(verkaufer,zeit,"NiM");
                       agentenstat(in_kaufer,zeit,"NMat");
                       agentenstat(in_verkaufer,zeit,"NMat");
                       agentenstat(m_kaufer,zeit,"iV");
                       agentenstat(m_verkaufer,zeit,"iV");
                      }

          match();                                                           580
          verhandeln();
         }
   agtdat.close();
   mktdat.close();
   trcdat.close();
}
```