REIHE INFORMATIK
TR-04-007

Entwicklung einer Börsensimulation mit der multiagentenbasierten Entwicklungsumgebung NetLogo

Lilli Winschel und Stephan Kopf
Universität Mannheim
– Fakultät für Mathematik und Informatik –
Praktische Informatik IV
L 15,16
D-68131 Mannheim, Germany

Entwicklung einer Börsensimulation mit der multiagentenbasierten Entwicklungsumgebung NetLogo

Lilli Winschel und Stephan Kopf
Praktische Informatik IV
Universität Mannheim
Germany

1. ZUSAMMENFASSUNG

In diesem Bericht werden die Möglichkeiten der Modellierungssumgebung *NetLogo* für die Realisierung von multiagentenbasierten Simulationen für Lehrveranstaltungen evaluiert. In diesem Rahmen wird die Entwicklungsumgebung NetLogo, ihre Skriptsprache und das für die Vernetzung der Agenten zuständige Zusatzpaket *HubNet* anhand der Implementierung einer simulierten Börse getestet.

In der simulierten Börse sollen Marktteilnehmer unterschiedliche Aktien handeln können. Mehrere Benutzer sollen gleichzeitig an der Simulation teilnehmen und dabei die Rolle eines Käufers bzw. Verkäufers einnehmen können. Zusätzlich zu den Benutzern sollen virtuelle Marktteilnehmer implementiert werden, die am Handel beteiligt sind. Ein Administrator soll die Möglichkeit haben, den Benutzern während des Handels zusätzliche Informationen zukommen zu lassen. Schließlich soll das implementierte Modell mit mehreren Teilnehmern getestet werden.

Das der Simulation zugrundeliegende Börsenmodell wird in Kapitel 2 erläutert. Es bildet eine Börse mit drei Aktiengesellschaften nach, deren Kursverläufe sich zufällig entwickeln. Die Rolle der Aktionäre nehmen menschliche Benutzer und computergenerierte Akteure ein.

In Kapitel 3 wird auf die Implementierung des Modells eingegangen. Die entwickelte Simulation, an der beliebig viele Benutzer über ein Netzwerk teilnehmen können, ist in der Skriptsprache *NetLogo* entwickelt. Das darin integrierte *HubNet* unterstützt die Realisierung der Mehrbenutzeraspekte der Simulation.

Darauf folgt in Kapitel 4 eine Beschreibung der Benutzeroberfläche. Zuletzt werden in Kapitel 5 die Ergebnisse des Testlaufes beschrieben.

2. BÖRSENMODELL

2.1 Die Börse in der Realität

Eine Börse ist im weiteren Sinne ein Ort, an dem Käufer und Verkäufer sich treffen um Handel zu betreiben. Wenn im allgemeinen Sprachgebrauch von "der Börse" die Rede ist, sind üblicherweise Wertpapierbörsen gemeint. In dieser Arbeit wird diese Einschränkung erweitert, indem nur bestimmte Wertpapiere, nämlich Aktien, betrachtet werden.

Eine Aktiengesellschaft (AG) ist eine Gesellschaft mit eigener Rechtspersönlichkeit. Sie besitzt ein in Aktien zerlegtes Grundkapital, wobei die Aktien der Gesellschafter (Aktionäre) Anteile am Grundkapital darstellen. Der Besitzer einer Aktie ist demnach Miteigentümer des Untenehmens. Die AG kann über die Dividende, eine pro Aktie an die Besitzer geleistete Zahlung, die Aktionäre am Gewinn des Unternehmens beteiligen.

Theoretisch gehört der gesamte Gewinn einer Aktiengesellschaft den Aktionären, d.h er müßte ihnen vollständig in Form von Dividenden ausgezahlt werden. Alternativ kann die AG die Gewinne gänzlich oder partiell in das Unternehmen investieren. Über die Höhe der ausgeschütteten Dividenden wird auf der Hauptversammlung des Unternehmens entschieden. Als Anlageprodukt ist die Aktie aber nicht hauptsächlich aufgrund der Dividende interessant. Die größten Renditechancen bieten die Kurssteigerungen der Aktie.

Die Hauptaufgabe der Börse ist es, einen "Marktplatz" zur Verfügung zu stellen, auf dem sich Käufer und Verkäufer von Aktien treffen und diese handeln. Als Mittler zwischen allen Händlern legt die Börse den aktuellen Preis jeder Aktie so fest, dass möglichst jeder Kauf- bzw. Verkaufswunsch erfüllt werden kann. Die Ermittlung der Preise auf diese Weise nennt sich Markträumung.

In der Realität sieht diese wie folgt aus: Zunächst geben die Aktionäre sogenannte Orders an die Börse ab. Orders sind, möglicherweise auf den Aktienpreis bedingte, Aufträge zum Kauf oder Verkauf von Aktien. Beabsichtigt der Aktionär Aktien zu erwerben, gibt er in einer Order die Anzahl der gewünschten Aktien und den Höchstpreis an, den er bereit ist für sie zu bezahlen. Hat er die Absicht Aktien zu verkaufen, gibt er die Anzahl und den Mindestpreis an, zu dem er bereit ist zu verkaufen.

Nachdem an der Börse eine Vielzahl an Orders eingegangen sind, wird ein optimal markträumender Preis errechnet. Zu diesem Zweck wird zunächst für jeden möglichen Preis der Überschuß an Nachfrage oder der Überschuß an Angebot ermittelt. Als optimaler Preis ergibt sich derjenige mit dem geringsten Überschuß. Diesen von der Börse errechneten Preis nennt man den Kurs einer Aktie. Wird eine bestimmte Aktie stark nachgefragt, bestehen also mehr Kauf- als Verkauforders, steigt der Kurs der Aktie. Liegen dagegen mehr Verkauf- als Kauforders vor, fällt er. Bedenkt man die enorme Anzahl der Orders die in diese Berechnung einfließen, wird klar, dass der einzelne Aktionär und seine Handelsaktivitäten in der Realität einen verschwindend geringen Einfluß auf den sich ergebenden Kurswert hat.

Rationale Investoren basieren ihre Handelsentscheidungen

auf ihre Einschätzung über den Wert der Aktie. Somit kann der Aktienkurs als die aggregierte Einschätzung aller Marktteilnehmer über den Wert des Aktienunternehmens verstanden werden.

Für eine weitergehende Darstellung der Börse siehe Roth und Beck (2002).

2.2 Modellierung der Börse

Um eine Simulation der Börse zu entwickeln, ist es notwendig, die Komplexität der Realität zu reduzieren. Zu diesem Zweck ist es erforderlich, ein stark vereinfachtes Modell der reellen Börse zu gestalten, auf dem die Simulation basieren kann (siehe Spann, 2002). Das Modell umfaßt die Ausgestaltung der Aktien und deren Preisbildung, sowie der Aktionäre und deren Kauf- und Verkaufverhalten.

2.2.1 Preisbildung der Aktien

Zwei mögliche Varianten der Preisbildung in einem Modell sind denkbar. Zum einen kann die Preisbildung durch Markträumung, zum anderen durch einen zufälligen Kursverlauf gestaltet werden.

Zu beachten ist hier, dass in der hier entwickelten Simulation eine gegenüber der Realität sehr geringe Zahl an Marktteilnehmern am Handel beteiligt sind. Denkt man sich nun eine Börse mit wenigen Marktteilnehmern, zieht die Implementierung der Markträumung eine Vielzahl von Konsequenzen mit sich.

Zunächst wird es, im Vergleich zur Realität, selten möglich sein, die Kauf- bzw. Verkaufswünsche der Aktionäre zu befriedigen. Es ist leicht einzusehen, dass in einem großen Markt mit zahlreichen Angeboten und Nachfragen die Menge der zustandekommenden Transaktionen weit grösser ist als auf einem kleinen Markt. D.h. in einem Modell, in dem weitaus weniger Marktteilnehmer agieren als in der Realität, ist es nicht möglich, durch Markträumung ein realitätsnahes Handelsvolumen nachzubilden.

Sieht man darüber hinweg und errechnet den Kurs dennoch auf Basis von Angebot und Nachfrage, ergibt sich daraus ein unangemessen hoher Einfluss der einzelnen Aktionäre auf den Kursverlauf. Zudem entstünde ein enorm sprunghafter Kursverlauf, welcher den Kursentwicklungen der Realität wenig nahe kommen würde.

Eine weitere Schwierigkeit ergibt sich bei einem kleinen Markt aus der Tatsache, dass der zukünftige Kursverlauf gänzlich vom Verhalten der Spieler abhängt. Für den Erfolg an der Börse ist einer der bedeutendsten Einflußfaktoren Qualität und Quantität von "Insider-" Informationen in Bezug auf den zukünftigen Verlauf des Aktienkurses. Diese Informationen ergeben sich letzlich aus Vermutungen über den zukünftigen Erfolg der Aktiengesellschaft.

Da hier keine "Erfolgsgeschichte" der Aktiengesellschaften als Entscheidungshilfe vorhanden ist, sind die Informationen auf andere Weise zu simulieren. Mit einem Kursverlauf, der vom Spielerverhalten abhängt, ist es nun aber unmöglich, Vorhersagen über ihn zu treffen. Dem in der Realität äußerst wichtigen Aspekt "Information" könnte somit nicht Rechnung getragen werden.

Die obigen Ausführung zeigen, dass eine Entscheidung für die Implementierung der Markträumung eine Vielzahl an realitätsverzerrenden Konsequenzen mit sich bringt. Aus diesem Grund implementieren gängige Börsensimulationen üblicherweise die Preisbildung nicht über Markträumung, sondern als exogene Größe. Einige, vor allem internetbasierte Simulationen, spielen echte Börsenkurse ein.[1] Andere erzeugen hypothetische Kurse auf eine zufällige Art.[2] Dies ist auch die Strategie, die in dieser Arbeit verfolgt wird. Es wird eine normalverteilte Rendite mit vorgegebenen Standardabweichungen und Mittelwerten gewählt (siehe Kapitel 3.3.3).

2.2.2 Aktionäre

Die Ausgestaltung der Anleger ist ein weiterer wesentlicher Punkt der Modellierung. Die Gruppe der Anleger besteht aus "menschlichen" und "virtuellen" Aktionären. Alle Aktionäre erhalten zu Beginn der Simulation ein Startkapital und sind berechtigt, nach Belieben darüber zu verfügen. Soweit es Ihr Bargeldguthaben erlaubt, können sie jederzeit Aktien kaufen bzw. verkaufen.

Die menschlichen Anleger werden durch die über Netzwerk verbundenen Spieler vertreten. Im Großen und Ganzen verfügen alle über die gleichen Voraussetzungen und Rechte. Lediglich in der ungleichen Informationensversorgung unterscheiden sie sich. Ein Teil der Anleger, die "informierten Anleger", erhalten zeitweise "Insiderinformationen" über den zukünftigen Kursverlauf einer Aktie. Der andere Teil ist auf seine Intuition bzw. auf sein Glück angewiesen. Welche der Spieler den "informierten" Anlegern zugeordnet wird, wird zufällig vom System festgelegt und ist weder durch den Spielleiter noch den Spielenden zu beeinflussen.

Die virtuellen Anleger dienen den menschlichen Spielern zum Vergleich und sind vollständig computergeneriert. Die Strategie, nach der sie Aktien handeln, kann wie folgt motiviert werden: Eine entscheidende Größe für das Investitionsverhalten von Anlegern ist die Rendite. Die Rendite der Aktie j von Periode t bis Periode $t+1$ ist definiert als

$$r_{jt} = \frac{p_{jt+1}}{p_{jt}} - 1, \tag{1}$$

wobei p_{jt} der Preis der Aktie j in Periode t ist. Bei einer Investitionsentscheidung in Periode t ist der Aktienpreis in der Folgeperiode p_{jt+1} und somit auch die Rendite ungewiss. Der Investor baut seine Entscheidung auf seine Erwartung bezüglich dieser unsicheren Größe auf.

Wie bereits in Kapitel 2.2.1 erläutert, sind die Renditen in diesem Börsenmodell unabhängig normalverteilt. Daher genügt es den Investoren, sich ein Bild über die Erwartungswerte und Volatilitäten der Aktien zu bilden, die durch die Mittelwerte und die Varianzen abgebildet werden. Im Folgenden bezeichnet also $\overline{r}_{ijt}$ den von Investor i vermuteten Erwartungswert der Rendite von Aktie j zum Zeitpunkt t. Die entsprechende Varianz sei als σ^2_{ijt} bezeichnet.

Generell wünscht sich ein risikoaverser Anleger eine möglichst hohe erwartete Rendite und eine möglichst geringe Unsicherheit. Wie wichtig ihm dabei diese beiden Aspekte sind, kann durch einen Risikopräferenzparameter a_i abgebildet werden. Je höher dieser Parameter ist, desto geringer ist die Risikobereitschaft und damit der in Aktien investierte Teil seines Vermögens V_i.

Die gerade beschriebenen Aspekte werden für die computergenerierten Spieler berücksichtigt. Ihre Investitionsentscheidung kann als die Zahl der Aktien vom Typ j dargestellt werden, die Investor i zum Zeitpunkt t besitzen möchte. Diese ist spezifiziert als

[1] Siehe z.B Börsen-Star (`www.boersen-star.de`) oder ESM-Börsensimulation (`www.die-prognose.de/download/ESM4.pdf`)

[2] Siehe z.B Die Börsianer (`www.tigerclan.de`)

$$x^*_{ijt} = V_{it} \frac{\overline{r}_{ijt}}{p_{jt} a_i \sigma^2_{ij}} \quad (2)$$

Diese Investitionsstrategie kann entscheidungstheoretisch abgeleitet werden. Ingersoll (1987, S. 98) zeigt, dass ein Investor mit bestimmten Präferenzen bei normalverteilten Renditen optimalerweise genauso investiert.

Die virtuellen Anleger unterscheiden sich in ihrem Investitionsverhalten in Risikoparameter a_i und, ebenso wie die "menschlichen" Anleger, in der ungleichen Informationsversorgung. Um die Abhängigkeit der Gewinne bzw. Verluste von unterschiedlichen Risikopräferenzen und Informationen sichtbar zu machen, sind den vier Computerspielern jeweils die Parameter risikoscheu/risikofreudig und informiert/nicht-informiert zugewiesen.

2.2.3 *Weitere Aspekte des Modells*

Auf folgende, in der Realität wichtige Aspekte der Börse, wurde in diesem Modell zugunsten der Komplexitätsreduzierung und der Übersichtlichkeit verzichtet:

- Die Ausschüttung von Dividenden.
- Transaktionskosten, die in der Realität beim Handel mit Aktien entstehen, z.B. Gebühren beim Kauf bzw. Verkauf von Aktien.
- Eine Simulation von Umwelteinflüssen, wie beispielsweise Inflation, Ölpreise, etc.
- Eine Simulation der Entwicklung des Geschäftserfolges der Aktiengesellschaften.

3. IMPLEMENTIERUNG

3.1 Entwicklungsumgebung: NetLogo und Hubnet

Für die Implementierung der Börsensimulation wurde NetLogo 2.0.1 verwendet. NetLogo ist eine in Java programmierte Entwicklungsumgebung für Modelle, die einfache Phänomene der Realität simulieren. Sie wurde am CCL (Northwestern's Center for Connected Learning and Computerbased-Modeling) entwickelt, das sich laut ihrer Homepage mit der "kreativen Ausnutzung von Technologien" befaßt um besseres Lernen zu ermöglichen. NetLogo basiert auf StarLogoT, einem früheren Projekt von CCL, das allerdings nur auf Apple Macintoshs lauffähig ist.

NetLogo ermöglicht es, in einer Simulation individuellen Subjekten einfache Regeln zu geben und das Resultat aus ihrem Verhalten und der entsprechenden Interaktionen zu beobachten. Alle Subjekte können gleichzeitig und selbstständig agieren.

In einem weiteren Projekt, dem "Participatory Simulations Project" entwickelte CCL *Hubnet*, eine in NetLogo eingebundene Technologie, die ein vernetztes agieren beliebig vieler Nutzer erlaubt[3]. Damit werden sogenannte "Participatory Simulations" ermöglicht. Hauptaugenmerk liegt hierbei auf der Unterstützung von technologiebasiertem, interaktivem Experimentieren und Lernen in Unterrichtsräumen. Jeder "Lernende" (Client) kontrolliert einen Teil des Modells, bzw. ein Subjekt, und nutzt hierfür ein eigenes Endgerät.

Die Mehrbenutzerfunktion der hier implementierten Börsensimulation wurde mit *Hubnet* realisiert. Im Moment stehen zwei Typen von *Hubnet* zur Verfügung, *Calculator HubNet* und *Computer HubNet*. *Calculator HubNet* wurde in Zusammenarbeit mit Texas Instruments entwickelt und nutzt ein TI-Navigator System. *HubNet* verwendet hierbei einen von TI entwickelten Rechner als Endgerät für die Clients. Die zweite, hier verwendete HubNet-Variante *Computer HubNet* nutzt gewöhnliche Desktop Rechner und Notebooks als Endgeräte. Zum gegenwärtigen Zeitpunkt befaßt sich CCL mit der Entwicklung eines weiteren HubNet Typens, der die Verwendung von PDAs als Endgerät unterstützt.

Im Internet ist NetLogo zum freien Download verfügbar[4]. Neben allen bisher erschienenen NetLogo Versionen findet man hier auch das User Manual (Wilensky, 1999) und alle Dokumentationen der bisher veröffentlichen Modelle aus den Bereichen Kunst, Soziologie, Wirtschaft, Biologie, Erdkunde, Spiele, Medizin, Physik, Chemie, Mathematik und Informatik.

3.2 Systemvoraussetzungen

Um NetLogo installieren und nutzen zu können sind folgende Systemvoraussetzungen zu beachten:[5]

- Windows: Es besteht die Möglichkeit, mit NetLogo eine passende Java Virtual Machine herunterzuladen. Falls man eine selbst installierte Java Virtual Machine nutzen möchte, benötigt Version 1.4.1 oder neuer. Empfohlen wird Version 1.4.2.
- MAC OS X: Auf OS X ist die Java Virtual Machine von Apple als Teil des Betriebssystems mitgeliefert. OS X 10.3 umfasst eine passende Java Virtual Machine. OS X 10.2 Nutzer müssen das Java 1.4.1 Update 1 installieren, welches Apple als Software Update zur Verfügung stellt.
- Andere Betriebssysteme: NetLogo sollte auf allen Betriebssystemen laufen auf denen eine Java Virtual Machine, Version 1.4.1 oder neuer, installiert ist. Empfohlen wird Version 1.4.2 oder neuer.

Desweiteren sind für die Nutzung von *Computer HubNet* folgende Voraussetzungen erforderlich:

- Ein vernetzter Computer auf dem NetLogo installiert ist, der als Server für die HubNet Clients dient.
- Für jeden Client ein Endgerät (Desktop oder Notebook) auf dem NetLogo installiert ist. *Computer HubNet* wurde bisher nur über LAN's getestet. Die Funktionsfähigkeit über WLAN's oder Einwahlverbindungen ist nicht gesichert.

[3]Siehe `http://ccl.northwestern.edu/partsims.html`.

[4]Siehe `http://ccl.northwestern.edu/netlogo/`

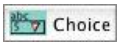
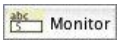
[5]Siehe `http://ccl.northwestern.edu/netlogo/requirements.html`

3.3 Das Modell im Einbenutzerbetrieb

Die in dieser Arbeit implementierte NetLogo Simulation lehnt sich an das in Kapitel 2 beschriebene Börsenmodell an. Sie simuliert Aktienkurse dreier Aktienunternehmen und Handelsaktivitäten von Aktionären. Die Gruppe der Aktionäre besteht aus vier virtuellen, computergenerierten Spielern, den eingeloggten Clients und dem Administrator, der den Server bedient. In diesem Kapitel wird zunächst die Implementierung des Modells im Einbenutzerbetrieb erläutert, d.h. es handeln zunächst allein die vier virtuellen Spieler und der Administrator.

3.3.1 Interface

Um das Interface zu erstellen, steht dem Entwickler in der NetLogo-Umgebung ein Interface-Builder zur Verfügung[6], den man im NetLogo Hauptfenster durch das Wählen der Registerkarte *Interface* erreicht. Hier können die Oberflächenelemente per "drag and drop" aus einer Symbolleiste in die Oberfläche eingefügt werden. Über diese Elemente kann der Benutzer später mit dem Modell kommunizieren. NetLogo stellt folgende Oberflächenelemente bereit:

Symbol	Beschreibung
	Buttons können ein einmaliges Ereignis auslösen oder eines, das solange wiederholt wird, bis er abermals betätigt wird. In den Einstellungen eines Buttons hat man die Möglichkeit auszuwählen, ob er ein *once* oder ein *forever* Button sein soll. *forever* Button lösen das Ereignis in einer Schleife etwa 500 Mal pro Sekunde aus.
	Slider sind globale Variablen. Der Benutzer verändert den Inhalt der Variable durch das Bewegen eines Schiebereglers.
	Choices sind globale Variablen, deren Wert man anhand einer vorgegebenen Liste auswählen kann. Die Liste erscheint in einem Drop Down Menü.
	Switches sind globale Variablen, die nur die boolean Werte true und false umfassen und werden durch einen On/Off Schalter betätigt.
	Plots dienen der graphischen Visualisierung von im Programmcode generierten Werten in Kurven oder Histogrammen.
	Mit einer Textbox kann man Beschriftungen in das Interface einfügen.
	Monitore zeigen Ergebnisse beliebiger Ausdrücke an. Sie aktualisieren ihren Wert automatisch mehrmals in der Sekunde.

[6]Die Benutzeroberfläche nutzt das Java Swing toolkit.

Graphics Windows visualisieren *turtles* und *patches*. Beide sind sogenannte Agenten, die sich in der zweidimensionalen Welt des Graphics Windows befinden und über Koordinaten in dieser Welt platziert sind. *turtles* sind beliebig geformte Figuren, die sich innerhalb dieser Welt bewegen können. *patches* stellen den unbeweglichen Hintergrund dieser Welt dar.

Ein Rechtsklick auf die Elemente eröffnet die Optionen "Select", "Delete" und "Edit". Wählt man die Option "Select" kann man Größe und Position des Elementes ändern. "Delete" löscht das Element aus dem Interface. In der Option "Edit" kann man verschiedene Einstellungen vornehmen, wie beispielsweise dem Element einen Namen oder Ereignis zuweisen. In Abbildung 1 sind beispielhaft Einstellungsoptionen eines Buttons dargestellt.

Figure 1: Button editieren

Hier wird beim Klicken des Buttons *buy DC shares* die Prozedur *buyRP* mit dem Argument 1 aufgerufen. Das Argument gibt Auskunft darüber, welche Aktie gekauft werden soll. Die Prozeduren, die durch die diversen Buttons aufgerufen werden, werden in den folgenden Unterkapiteln erläutert. Analog kann man für alle Elementen diverse Einstellungen editieren.

Das hier vorgestellte Modell enthält im Einbenutzerbetrieb folgende Interface Elemente (siehe Abbildung 4 auf Seite):

- Ein Plot zur Visualisierung der Kursverläufe.
- Monitore, die das Vermögen der Spieler, den aktuellen Preis der Aktien, einen Countdown der verbleibenden Zeit zur nächsten Kursveränderung und die verbleibenden Perioden bis zum Spielende anzeigen.
- Einen Button um ein neues Spiel zu starten und einen "forever" Button um den Kursverlauf zu starten und zu stoppen. Weitere Buttons zum Kauf bzw. Verkauf von Aktien und zur Auswahl der Spielerdaten die angezeigt werden sollen.
- Ein Graphics Window, welches anhand von Pfeilen Informationen über die zukünftigen Kurswerte anzeigt.

3.3.2 Programmstruktur

Wählt man im Hauptfenster die Registerkarte *Procedures*, gelangt man zum Programmeditor von NetLogo, in

dem man den Programmcode der Modelle erstellen und bearbeiten kann. In diesem Kapitel soll zunächst ein grober Überblick über die Programmstruktur gegeben werden.

Die Prozeduren *startup*, *setup* und *start* dienen der Steuerung des Programmablaufes. Die Funktion *startup* wird automatisch einmal beim Start des Modells vom System aufgerufen. Diese Prozedur bewirkt lediglich die Anbindung der Clients und den Start von HubNet (siehe Kap 3.4).

Die Prozedur *setup* wird durch einen Mausklick des Benutzers auf den Button *setup* aufgerufen. Hier werden sowohl die Unterprozeduren *setup-values*, *setup-prices* und *setup-infos* aufgerufen, die sämtliche Variablen auf ihre Startwerte setzen, als auch eine Unterprozedur *setInformations*, die für die Bereitstellung von Informationen für die Spieler zuständig ist.

Die Funktion *start* wird vom Benutzer durch ein Anklicken von *start/stop* aufgerufen. Diese Prozedur reguliert den gesamten Programmablauf des Modells. In Abbildung 2 wird der Programmablauf der Prozedur *start* bildhaft anhand eines Diagrames dargestellt.

Da *start/stop* ein "forever" Button ist, wird die Prozedur *start* in einer Schleife einige hundert Male in der Sekunde aufgerufen. Demnach wird die Prozedur *listenClients* genauso oft aufgerufen, wohingegen der Countdown-Zähler nur einmal pro Sekunde um eins reduziert wird. Alle fünf Sekunden beginnt eine neue Periode, in der neue Preise errechnet und an die Spieler gesendet, die Investitionen der Spieler bearbeitet und gegebenenfalls Informationen über die zukünftigen Kursverläufe erfasst und versendet werden. Schließlich wird am Ende des 50sten Durchlaufes die Prozedur *end-of-game* und eine Anweisung *stop* aufgerufen, die die alles umfassende Schleife und somit die gesamte Prozedur *start* beendet. In *end-of-game* wird am Ende der Simulation eine Highscore-Liste nach dem Gesamtvermögen erzeugt und dem Administrator angezeigt.

Die einzelnen Unterprozeduren werden im Folgenden erläutert.

3.3.3 Kursentwicklung

Es werden drei Aktien simuliert. Ihre Kursverläufe sind in der Prozedur *updatePrices* implementiert:

Eine Liste *corporation-lst* enthält die Bezeichnungen der drei Aktiengesellschaften. Sie werden aus Gründen der Anschaulichkeit im Code auf "BASF", "DaimlerChrysler" und "SAP" gesetzt. Da wir den Aktionären Informationen über den zukünftigen Kursverlauf der Aktien liefern möchten, ist es notwendig, den Kurs für die nächste Periode schon in der aktuellen Periode zu berechnen.

Für jede AG wird in einer Schleife der aktuelle Kurs und der Kurs zum Zeitpunkt t+1 berechnet. Der Kurs zum Zeitpunkt t wird in einer Liste *price-lst*, der zum Zeitpunkt t+1 in einer Liste *pricetomorrow-lst* gespeichert. Die relative Veränderung des Kurswertes von Aktie j wird zufällig aus einer Normalverteilung gezogen:

$$Kurswert_{t+1} = Kurswert_t \cdot (1 + d_{tj}(\mu_j, \sigma_j)) \qquad (3)$$

wobei d_{tj} eine normalverteilte Zufallsvariable mit Mittelwert μ_j und Standardabweichung σ_j ist.

Der Mittelwert μ_j entspricht der erwarteten Rendite und wurde für die unterschiedlichen Aktien auf Werte zwischen 0,5 und 2 Prozent gesetzt. Die Standardabweichung σ_j beeinflusst die Volatilität der Kurse und hat einen Wert zwischen 0,04 und 0,07. Die Mittelwerte und Standardabweichungen werden den Listen *expreturn-lst* und *stddev-lst* entnommen.

Nach der Berechnung der Kurswerte werden diese anhand der Befehle `set-current-plot-pen` und `plot` an das im Interface vorhandene Plot zur Darstellung der Kursverläufe übergeben.

3.3.4 Handel

Die Handelsaktivitäten der menschlichen und virtuellen Anleger werden in den Prozeduren *tradingVP*, *buyRP*, *sellRP* und *updateTotalassets* realisiert.

Die Prozedur *tradingVP* behandelt ausschließlich den Handel der virtuellen Computerspieler. Nachdem zu Beginn jeder Periode die aktuellen Kurse festgelegt wurden, wird diese Prozedur aufgerufen. Auf Basis der in Kapitel 2.2.2 erläuterten Entscheidungsformel werden die Aktivitäten dieser Anleger generiert. Zwei der vier virtuellen Spieler sind informierte Spieler. Bei diesen Spielern wird zufällig in einigen Perioden der erwartete geschätzte Return durch denjenigen ersetzt, der tatsächlich zum Zeitpunkt `t+1` eintreten wird.

Eine zweidimensionale Liste *share-lst* beinhaltet in der ersten Dimension die vier virtuellen Aktionäre und die eingeloggten Clients und in der zweiten Dimension die drei unterschiedlichen Aktien. Für jeden virtuellen Anleger und für jede Aktie wird hier in einer geschachtelten Schleife die Anzahl der Aktien errechnet, die der jeweilige Anleger kaufen möchte. Anschließend wird anhand der Liste *liquidassets-lst*, die das Bargeldvermögen der Handelnden beinhaltet, geprüft ob sich der Anleger die gewünschte Anzahl an Aktien leisten kann. Bzw. es wird anhand der Liste *share-lst* geprüft ob er genug Aktien besitzt, um die gewünschte Anzahl zu verkaufen. Sind diese Voraussetzungen nicht erfüllt, erhält bzw. verkauft der virtuelle Spieler die maximal mögliche Anzahl an Aktien.

In der Prozedur *buyRP* wird der Kaufwunsch des Administrators bearbeitet. Sie wird durch das Betätigen einer der drei Kauf-Buttons im Interface des Benutzers aufgerufen und enthält die gewünschte Aktie als Argument. Innerhalb der Prozedur wird zunächst mit der Funktion `user-input` ein Eingabefenster erzeugt, in dem der Benutzer die Anzahl der Aktien eingeben kann, die er kaufen möchte. Bevor der Kauf vollzogen wird, wird geprüft, ob die Eingabe eine Zahl ist und ob der Anleger über hinreichende liquide Mittel verfügt. Sollte dies nicht der Fall sein, wird dem Benutzer eine Fehlermeldung ausgegeben. Wird der Kauf vollzogen, so werden die Variablen, die das Bargeld- und Aktienvermögen speichern, aktualisiert. Die Prozedur *sellRP* bearbeitet analog hierzu die Verkäufe des Administrators.

Sind die Handelsaktivitäten für die laufende Periode abgeschlossen, wird in der Prozedur *updateTotalassets* das Gesamtvermögen des Administrators und der Clients aktualisiert. Die Variable *Totalasset-rp* sichert das des Administrators und die Liste *Totalassets-lst* das der virtuellen Aktionäre und der Clients.

3.3.5 Informationen

Die Realisierung der "informierten" virtuellen Spieler wurde bereits in Kapitel 3.3.4 erläutert. In diesem Kapitel werden die Informationen des Administrators besprochen.

Der Administrator ist ein vollständig informierter Aktionär. Er erhält in jeder Periode Informationen zu den Kursentwicklungen zum Zeitpunkt `t+1`. Diese Informationen wer-

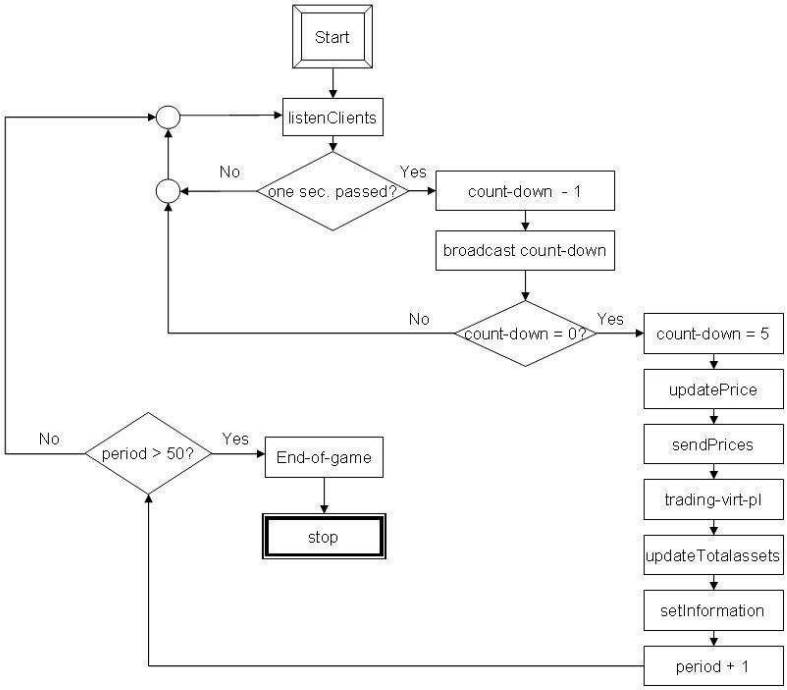


Figure 2: Prozedur start

den in Form von Pfeilen veranschaulicht, die durch ihre Richtung und ihre Farbe den zukünftigen Trend der Aktien anzeigen. Die Pfeile sind im Graphics Window als *Turtles* realisiert. Wie schon erwähnt stellt NetLogo *Turtles* als Formen zur Verfügung, die sich im Graphics Window bewegen können. Zudem gibt es die Möglichkeit, mehrere *Turtles* in einer *Brut* zusammenzufassen, deren Mitglieder gemeinsame Eigenschaften aufweisen. In der Prozedur *setup-infos* wird die hier existierende *Brut* von Pfeilen initialisiert, in der Funktion *setInformations* wird in jeder Periode die Richtung und die Farbe der Pfeile aktualisiert.

Die Informationsversorgung der Clients wird im Kapitel "Das Modell im Mehrbenutzerbetrieb" (3.4) erläutert.

3.3.6 Weitere Funktionen

Für den Einbenutzerbetrieb sind zwei weitere Prozeduren implementiert, *setDisplayedPlayer* und *end-of-game*. *setDisplayedPlayer* wird durch die *Display Player* Buttons des Interfaces ausgelöst. Aus einem, nach dem Auslösen des Buttons, erscheinenden Auswahlfenster kann der Benutzer einen Aktionären auswählen, über dessen Vermögen er Auskunft haben möchte. Er hat die Möglichkeit, aus allen Aktionären, die in der Liste *user-lst* gespeicherten sind, also sowohl die virtuellen Spieler als auch die eingeloggten Clients, vier auszuwählen.

Die ID des Spielers, der vom Benutzer aus dem Auswahlfenster gewählt wurde, wird innerhalb der Prozedur in einer der vier vorhandenen Positionen der Liste *displayed-player-lst* gespeichert. Die Monitore des Interfaces können nun wie folgt die ausgewählten Spieler anzeigen:

```
item (item 0 displayed-player-lst) liquidassets-lst
```

Das heißt: zeige das Bargeldvermögen des Spielers an, dessen ID an der ersten Stelle der *displayed-player-lst* steht. Analog dazu sind die Monitore editiert, die das Aktienvermögen, das Gesamtvermögen und den Namen der Spieler anzeigen.

Die Prozedur *end-of-game* erzeugt am Ende des Simulationsdurchlaufes, also nach 50 Perioden, eine Highscore-Liste und zeigt diese dem Benutzer an.

3.4 Das Modell im Mehrbenutzerbetrieb

Das hier besprochene Modell unterstützt einen Mehrbenutzerbetrieb. Es kann nun nicht mehr nur eine, sondern beliebig viele, über ein Netzwerk verbundene, Personen an der Simulation teilnehmen. Jede der Personen benötigt hierzu ein mit NetLogo ausgestattetes Endgerät.

Im Mehrbenutzerbetrieb nimmt das bisher erläuterte Modell und der Rechner auf dem es installiert ist die Rolle des Servers ein. Die über ein Netzwerk angebundenen Endgeräte dienen als Clients.

Um einen Mehrbenutzerbetrieb zu realisieren, müssen folgende Funktionalitäten implementiert werden. Zunächst muß ein angepasstes Interface für die Clients erzeugt werden. Darauf folgt die Anbindung der partizipierenden Endgeräte (Clients) an den bestehenden Server und die Implementierung der Kommunikation zwischen Server und Clients. Der Server muss zudem in der Lage sein, während der Kommunikation clientspezifische Daten zu speichern.

Diese clientspezifischen Daten werden in Listen gespeichert. Zusätzlich zu den Listen, die Informationen über das Vermögen der Clients enthalten, existiert eine Liste, die die Namen der Clients enthält, die der "informierten" Grup-

pe angehören und vier weitere Listen, die speichern, welche Werte die Spieler im Moment an ihren Slider und Choices angelegt haben.

Im Gegensatz zur beträchtlichen Mächtigkeit des Server hat der Client nur sehr eingeschränkte Möglichkeiten. Der gesamte Programmcode und die gespeicherten Daten der Clients liegen auf dem Server. Der Client hat keine Möglichkeit, dem Server aktiv Nachrichten zu senden, er kann lediglich durch Slider, Choices und Buttons Nachrichten erzeugen, die dann vom Server abgeholt werden. Das Empfangen von Nachrichten des Servers erfolgt über Plots und Monitore, wobei letztere Daten ausschliesslich als Strings interpretieren können.

3.4.1 Interface des Clients

Die Oberfläche der Clients wird im Prinzip analog zu der des Servers erzeugt und kann die gleichen Elemente enthalten. Mit Hilfe des Interface-Builders werden die Elemente in die Oberfläche eingefügt. Im Gegensatz zum Server kann einem Element im Client Interface allerdings keine Anweisung übergeben werden, sondern lediglich ein Name.

Im Mehrbenutzerbetrieb der Börsensimulation enthält die Oberfläche der Clients folgende Elemente (siehe Abbildung 7 auf Seite):

- Ein Plot zur Visualisierung der Kursverläufe.
- Monitore, die das Vermögen des Spielers, den aktuellen Preis der Aktien, einen Countdown der verbleibenden Zeit zur nächsten Kursveränderung, die verbleibenden Perioden bis zum Spielende und diverse Informationen (unter anderem über zukünftige Kursentwicklungen) anzeigen.
- Buttons zum Kauf bzw. Verkauf von Aktien.
- Slider, anhand derer der Client auswählen kann, wieviel er kaufen bzw. verkaufen möchte.
- Choices, anhand derer der Client auswählen kann, welche Aktie er kaufen bzw. verkaufen möchte.

Um dem Server Nachrichten zukommen zu lassen, stehen dem Client lediglich drei Moglichkeiten zur Verfügung: Er betätigt einen Button, einen Slider oder einen Choice. Wird im Client eines dieser Elemente betätigt, werden folgende Ereignisse ausgelöst: Wurde von Client X ein Button Y betätigt, erhält der Server auf Anfrage die Nachricht: Client X hat einen Button betätigt, der Y heißt. Verschiebt der Client einen Slider Y auf den Wert Z, oder wählt einen Wert Z aus dem Choice Y aus, erhält der Server die Nachricht: Client X hat ein Element namens Y auf den Wert Z gesetzt.

3.4.2 Anbindung der Clients

Die Anbindung der Clients an den Server wird durch folgende Anweisungen realisiert: In der Prozedur *start* wird mit der Anweisung

```
hubnet-set-client-interface ''Clienttyp''
''Pfad''
```

dem System über die Parameter zum einen der Clienttyp (hier *Computer*), zum anderen der Pfad zur Datei in der das Client-Interface auf dem Server liegt, übergeben. Dieses Interface wird allen Clients, die sich in die Simulation einloggen, zugesendet. `hubnet-reset` startet HubNet, welches solange läuft, bis entweder NetLogo oder das Modell beendet wird.

3.4.3 Kommunikation und Benutzerverwaltung

Nachdem das Modell und HubNet gestartet wurde und die Clients angebunden sind, kann mit ihnen kommuniziert werden. Dies geschieht in der Prozedur *listenClients*.

Hier wird zunächst in einer Schleife mit der Anweisung `hubnet-message- waiting?` geprüft ob eine Nachricht vorliegt. Ist dies der Fall, wird sie mit `hubnet-fetch-message` abgeholt. Danach wird sequentiell in geschachtelten if-Anweisungen die Art der Nachricht abgefragt. Nachrichten können sein: Ein neuer Client hat sich eingeloggt, ein Client hat sich ausgeloggt oder ein Client hat einen Slider, Choice oder Button betätigt.

Hat sich ein neuer Client eingeloggt, wird ein neuer Aktionär angelegt, d.h. er wird in die *user-lst* eingefügt und alle weiteren Listen, die die Clients betreffen, werden aktualisiert. Unter anderem wird in die Liste *totalassets-lst* ein Element eingefügt und in einer *informed-lst* wird gespeichert, ob der neue Client ein informierter Spieler ist. Welche Spieler dieser Gruppe angehören, wird zufällig entschieden. Hat sich ein Client ausgeloggt, werden alle Listen, in denen Daten dieses Clients enthalten sind, aktualisiert.

Hat sich weder ein Spieler eingeloggt noch ausgeloggt wird mit

```
hubnet- message-tag = ''Interface-Element-Name''?
```

in weiteren if-Abfragen geprüft, von welchem Interface-Element die vorliegende Nachricht kam. Danach wird untersucht, welcher Client dieses Element bedient hat und die entsprechenden Daten dieses Clients aktualisiert. Hat er beispielsweise einen Kauf-Slider verschoben, wird in der Liste *buyingslider-list* das entsprechende Element aktualisiert.

Wurden von einem Client Aktien gekauft oder verkauft, werden die sein Vermögen betreffenden Daten neu berechnet, gespeichert und ihm auf die entsprechenden Monitore gesendet. Einzelnen Clients kann man mit der Anweisung

```
hubnet-send-message ''Client-Name'' ''Monitor-Name''
```

Daten zusenden. Statt einem Client kann man Nachrichten auch an eine Liste von Clients versenden. Dies geschieht in der Prozedur *setInformations* in der nur den Spielern Nachrichten gesendet werden, die in der Liste *informed-lst* enthalten sind.

Nachrichten, die alle Spieler betreffen (beispielsweise die Aktienpreise in der Prozedur *sendPrices*), werden mittels

```
hubnet-broadcast-message ''Monitor-Name''
```

an alle Clients versendet.

3.5 Fazit

In dieser Arbeit sollten am Beispiel einer Börsensimulation die Möglichkeiten von NetLogo und das darin eingebundene *HubNet* für multiagentenbasierte Simulationen getestet werden. Bei der Implementierung des Modells traten sowohl Stärken als auch Schwächen von NetLogo zum Vorschein.

Eine der Stärken ist der Interface-Builder, der ein unkompliziertes Erstellen der Benutzeroberfläche ermöglicht. Oberflächenelemente müssen nicht selbstständig programmiert werden, sondern können bequem per "drag and drop" eingefügt werden. Ein im besonderen Maße hilfreiches Oberflächenelement sind Plots, mit deren Hilfe die Darstellung

von Kurven und Histogrammen sehr einfach zu realisieren sind.

Desweiteren stellt die relativ einfache Erzeugung und Handhabung der sogenannten Turtles im Graphics Window, die sich nach festgelegten Regeln verhalten, einen Vorzug von NetLogo dar.

Weiterhin ermöglicht *Hubnet* eine problemlose und einfache Anbindung der Clients an den Server. HubNet ist in NetLogo vorinstalliert, so dass eine einzige Anweisung im Programmcode eines Modells genügt um bestehende Clients anzubinden.

Im Zusammenhang mit dem komplexen Sachverhalt einer Börsensimulation traten jedoch auch Schwächen zum Vorschein. Zunächst wurde deutlich, daß die eingeschränkte Auswahl an Oberflächenelementen die Realisierung einer benutzerfreundlichen Oberfläche erschweren. So ist es beispielsweise nicht möglich, dem Benutzer Tabellen anzuzeigen, da nur Monitore zum Anzeigen von Daten zur Verfügung stehen, die keinen Zeilenumbruch erlauben.

Noch eingeschränkter als die Benutzeroberfläche des Servers ist die der Clients. Hier gibt es beispielsweise keine Möglichkeit, Textfelder zu realisieren, über die der Benutzer Eingaben tätigen kann. So ist in diesem Modell das Kaufen bzw. Verkaufen über Slider, Choices und Buttons recht unübersichtlich. Zudem kann die Oberfläche nicht dynamisch gestaltet werden. Die Oberflächenelemente der Clients sind identisch und jedes Element besitzt einen statischen Namen.

Eine weitere Schwierigkeit besteht in der Tatsache, daß der Client weder Daten speichern, noch aktiv Nachrichten versenden kann. So muß der Server in einer Schleife ununterbrochen Nachrichten nachfragen und umständlich ermitteln und speichern, welcher Art die Nachrichten sind und von welchem Sender sie stammen.

Die umständliche Handhabung von Listen stellt eine weitere Schwierigkeit dar. Das Erstellen, Aktualisieren und Zugreifen auf Listen und ihre Elemente ist nicht nur kompliziert und unübersichtlich sondern gestaltete sich auch durch die lückenhafte Dokumentation der NetLogo Syntax schwierig.

Zudem treten in der Anwendung der Modelle Unregelmäßigkeiten auf, die nicht zu beheben sind: Zum Beispiel erscheint zuweilen beim Einloggen eines Clients die IP des Servers zur Auswahl, manchmal aber auch nicht. Die Farben der Kurse im Client werden zeitweise korrekt dargestellt, ab und zu aber erscheint der ein oder andere Kurs in Schwarz.

Das Fehlen einer internen Fehlerbehandlung stellte ein weiteres Problem vor allem deshalb dar, weil die Funktionalität der NetLogo Syntax einige Defizite aufweist. Beispielsweise kann nach einer Eingabe des Administrators in ein Benutzereingabefeld nicht geprüft werden, ob es sich bei der Eingabe um einen String oder um eine Zahl handelt. Gibt der Benutzer also fälschlicherweise einen Buchstaben oder ein anderes Zeichen ein, so erfolgt beim Umwandeln der Eingabe in einen numerischen Ausdruck ein Fehler, der die Simulation abbricht. Die umständliche Fehlerbehandlung dieses Problems wäre mit Hilfe einer internen strukturierten Fehlerbehandlung weitaus leichter zu realisieren.

Aufgrund der genannten Stärken und Schwächen läßt sich sagen, daß NetLogo sehr gut geeignet ist für die Simulation einfacher Phänomene, in denen die Clients keine komplizierten Einflußmöglichkeiten haben. Ein Beispiel wäre ein Modell, in dem jeder Client das Verhalten eines Turtles kontrolliert, der sich lediglich im Graphics Window bewegt, seine Farbe ändert oder dergleichen.

Die Implementierung eines Modells, das viele Oberflächenelemente benötigt, oder in dem der Client häufig Daten mit dem Server austauschen muss, gestaltet sich mit NetLogo schwierig.

4. BENUTZEROBERFLÄCHE

4.1 Administrator

Der Administrator ist die Person, die die Steuerung der Simulation übernimmt. Er kontrolliert den Start der Simulation, kann sie unterbrechen und beenden. Er hat Überblick über die teilnehmenden Spieler, kann sie aus dem Spiel entfernen oder ihnen Informationen zukommen lassen. Zusätzlich hat er die Möglichkeit, sich selbst am Handel der Aktien beteiligen.

Nach dem Start von NetLogo gelangt der Administrator über den Hauptmenüpunkt *File* zur *Models Library*. Die Models Library stellt alle momentan verfügbaren Netlogo Modelle zur Auswahl bereit (siehe Abb. 3). Hier findet man im Ordner *Hubnet Computer Activities* das Modell *Börsensimulation*.

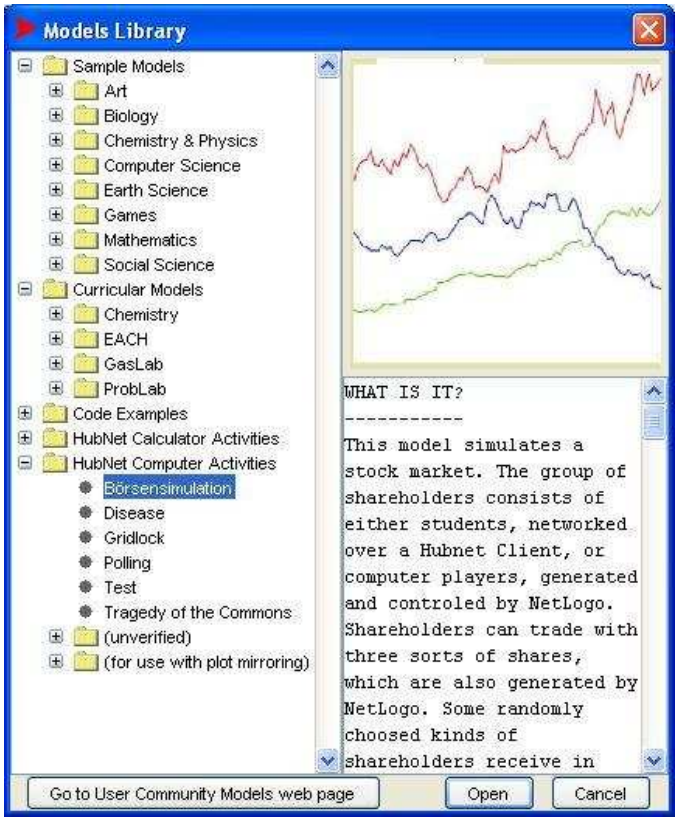


Figure 3: Models Library

Hat der Benutzer dieses Modell ausgewählt und auf die Anfrage *Enter a unique name for your computer, such as your name* einen beliebigen Namen angegeben, wird die Börsensimulation gestartet. Die Oberfläche, die dem Admi-

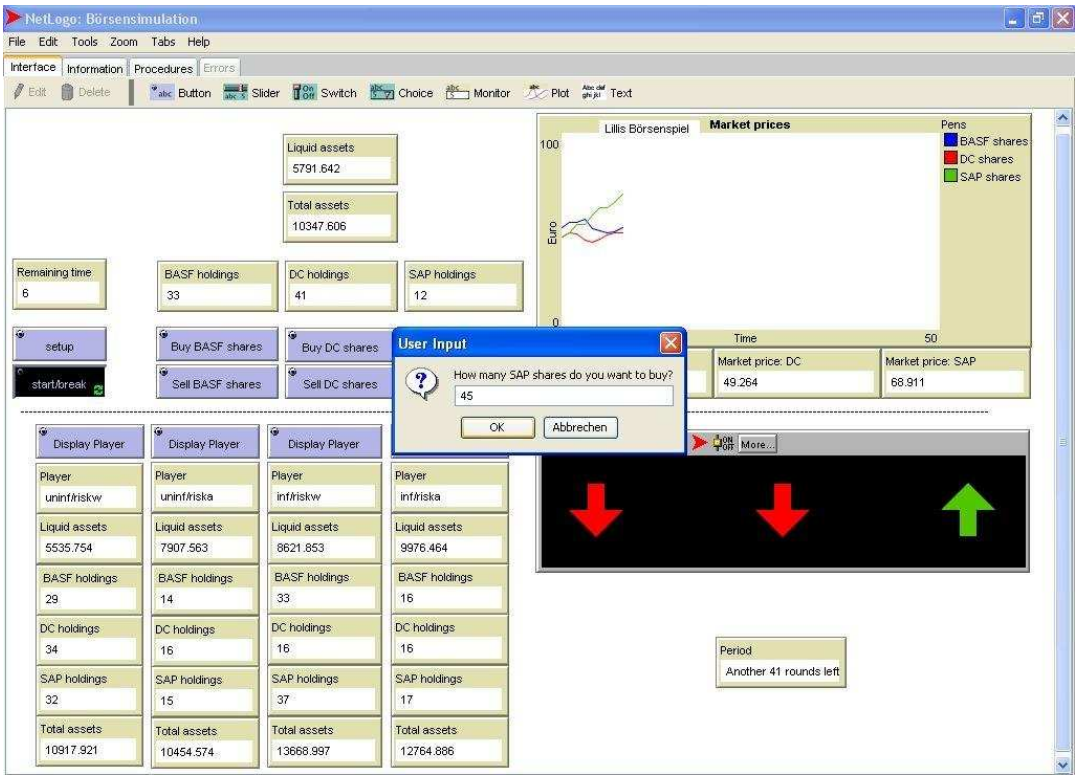


Figure 4: Benutzeroberfläche des Administrators

nistrator zur Verfügung steht, wird in Abbildung 4 dargestellt.

Ein zentraler Bereich der Oberfläche besteht aus den Button *setup* und *start/stop*. Mit *setup* wird das Modell initialisiert, mit *start/stop* startet der Administrator die Simulation, bzw. unterbricht sie. Der *start/stop* Button ist ein sogenannter "forever" Button der, nicht wie bei einem "once" Button ein einmaliges, sondern ein durchlaufendes Ereignis auslöst. So wird automatisch alle zehn Sekunden das Ereignis "nächste Periode" ausgelöst. Durch abermaliges Klicken auf den Button wird die automatische Wiederholung der Ereignisse unterbrochen. Im Monitor *Remaining time* ist der zehnsekündige Countdown zu beobachten.

Der zweite Bereich dient der Visualisierung der Kursverläufe, der hier Simulierten Aktien "BASF", "Daimler Chrysler" und "SAP". Der sogenannte "Plot" *Kurse* zeigt den Verlauf graphisch, während die sich darunter befindenden "Monitore" den jeweils aktuellen Kurswert anzeigen.

Zwischen diesen beiden Bereichen liegen alle Komponenten, die der Administrator benötigt, um selbst Handel betreiben zu können. Hier befinden sich sowohl diverse Monitore, die sein Gesamtvermögen, sein Bargeldvermögen und die Anzahl der in seinem Besitz befindlichen Aktien anzeigen, als auch je Aktienunternehmen einen *buy* und einen *sell* Button.

Betätigt der Administrator diese, öffnet sich ein Eingabefenster, in welches er die Zahl der gewünschten Aktien eingeben kann, die er kaufen bzw. verkaufen möchte. Der Versuch, mehr Aktien zu verkaufen als sich in seinem Besitz befinden, Mengen zu kaufen, die seine liquiden Mittel übersteigen oder die falsch getätigte Eingabe von Buchstaben statt Zahlen, löst eine Fehlermeldung aus.

Der letzte Bereich dient dem Administrator lediglich zur Information. Zum einen sind die Daten von vier Spielern zu sehen. Welche Spieler sich der Administrator ansehen möchte kann er beliebig auswählen. Sowohl die Daten von virtuellen als auch die von menschlichen Spielern können hier angezeigt werden. Betätigt der Administrator einen *Display Player* Button, öffnet sich ein Auswahlfenster indem er den gewünschten Spieler auswählen kann:

Zum anderen werden, im unteren rechten Bereich der Benutzeroberfläche, in einem "Graphics Window" Informationen über den zukünftigen Kursverlauf der Aktien mittels farbiger Pfeile angezeigt. Grüne nach oben gerichtete Pfeile kündigen das Ansteigen des entsprecheneden Kurses an, rote nach unten gerichtete Pfeile das Fallen.

Am Ende des Simulationsdurchlaufes wird dem Administrator ein Highscore der beteiligten Spieler präsentiert. Auch dieser umfasst sowohl die virtuellen als auch die menschlichen Spieler.

Die Teilnahme weiterer "Spieler" wird mit Hilfe des Participatory Simulation System *Hubnet* (siehe Kapitel 3.1) verwirklicht. Die Benutzeroberfläche der "Spieler" unterscheidet sich von der des Administrators, beinhaltet aber auch einen "Plot" zur Visualisierung der Kurse. Damit der graphische Verlauf der Kurse auch für die Mitspieler sichtbar wird, ist es notwendig, dass der Administrator im "Hubnet Control Center" (siehe Abb. 5) die Einstellung *Mirror Plots on Clients* vornimmt.

Das "Hubnet Control Center" erreicht man über das Hauptmenü (*Tools* → *Hubnet Control Center*). Hier bekommt der Administrator auch die Information, welche Spie-

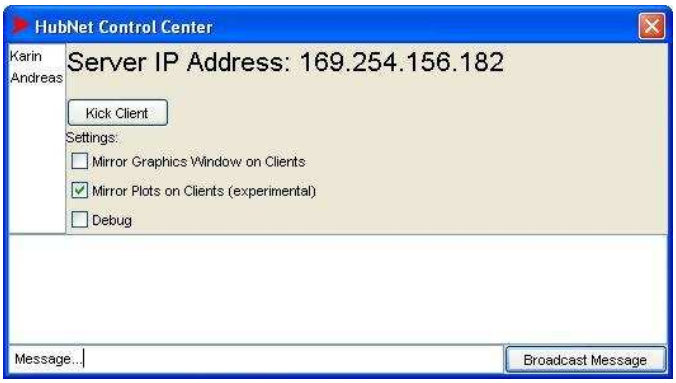


Figure 5: Hubnet Control Center

ler eingeloggt sind. Er hat die Möglichkeit, sie über den *Kick Client* Button aus dem Spiel zu entfernen oder im unteren Eingabefenster Nachrichten zu verfassen und sie ihnen mit *Broadcast Message* zu senden. Beispielsweise könnte er den Teilnehmern Informationen über den zukünftigen Verlauf der Kurse zukommen lassen.

Die Simulation wird entweder automatisch nach der fünfzigsten Periode oder durch ein "Setup" des Administrators beendet.

4.2 Spieler

Wie schon erwähnt unterscheiden sich die Oberfläche und die Möglichkeiten der Spieler von denen des Administrators. Zum einen ist das Starten, Unterbrechen und Beenden des Spieles dem Administrator vorbehalten. Zum anderen stehen den Spielern nicht alle Information über zukünftige Kursentwicklungen zur Verfügung.

Die Spieler sind in zwei Kategorien unterteilt. Die erste Kategorie, die "uninformierten" Spieler, erhalten keinerlei Informationen über die zukünftige Entwicklung der Kurse, die zweite Kategorie, die "informierten" Spieler, erhalten hin und wieder Informationen über einzelne zu erwartende Kursentwicklungen. Die Zuordnung der Spieler zu diesen Kategorien findet intern automatisch und zufällig statt, weder der Spieler selbst noch der Administrator haben Einfluss darauf.

Nach dem Starten der Anwendung *Hubnet Client.exe* öffnet sich ein Fenster, in dem der Spieler die Möglichkeit hat, sich in die Simulation einzuloggen (siehe Abb. 6). Um

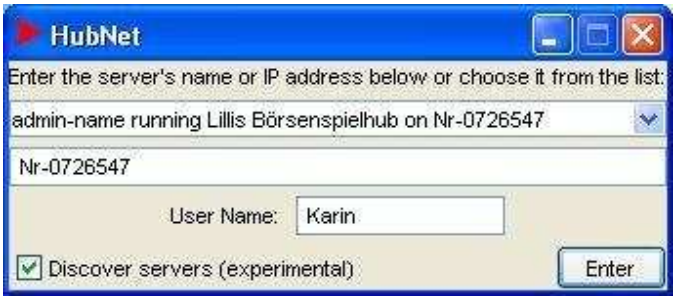


Figure 6: Login Client

zur Simulationsoberfläche zu gelangen, sind in das Eingabefenster ein beliebiger Spielername und entweder der Computername oder die IP-Adresse des Servers einzugeben. Falls diese nicht bekannt ist, findet der Administrator diese im *Hubnet Control Center* des Servers (siehe Abb. 5). Durch das Markieren der Checkbox *Discover Servers* erscheint die Adresse des erwünschten Servers zur Auswahl in der Listbox[7].

Auch die Simulationsoberfläche der Spieler ist in verschiedene Bereiche unterteilt (siehe Abb. 7). Der erste Bereich dient dem Handel und beinhaltet unter anderem einige Monitore, die über das eigene Gesamtvermögen, das verfügbare Bargeldvermögen und den Aktienbesitz Auskunft geben.

In diesem Bereich befinden sich zusätzlich Slider zur Auswahl der Anzahl der erwünschten Käufe bzw. Verkäufe und Choices, mittels derer man die Aktie auswählen kann. Der tatsächliche Kauf bzw. Verkauf wird erst durch das Bedienen des *Buy*- bzw. *Sell*-Buttons vollzogen. Ein Monitor *Remainig time* zeigt dem Spieler die verbleibende Zeit bis zum Übergang in die nächste Periode an, d.h. wie lange er noch zum aktuellen Preis kaufen bzw. verkaufen kann.

Der zweite Bereich stimmt vollständig mit dem entsprechenden Bereich in der Oberfläche des Administrators überein. Auch hier werden die Kursverläufe der diversen Aktien graphisch in einem Plot dargestellt und die aktuellen Kurswerte in jeweils einem Monitor.

Der dritte Bereich stellt den "informierten" Spielern Informationen über die zu erwartende Kursentwicklung der Aktien bereit. Diese Spieler erhalten gelegentlich über einen Monitor Informationen über den zukünftigen Kursverlauf einer Aktie. Der uninformierte Spieler erhält keine Informationen.

5. TEST

Das in den vorherigen Kapiteln dargestellte Modell wurde schließlich Tests unterzogen, die dessen Lauffähigkeit sicherstellen sollen. Zu diesem Zweck wurde das Modell zunächst im Einbenutzerbetrieb getestet. Es wurden 139 Simulationen durchgeführt um die Fehleranfälligkeit zu prüfen und das Investitionsverhalten der computergenerierten Spieler zu untersuchen.

Während der Durchläufe traten keine technischen Fehler auf. Die Untersuchung des Investitionsverhaltens ergab die in Tabelle 1 dargestellten Werte für den Mittelwert und die Standardabweichung des Gesamtvermögens der Spieler am Ende eines Simulationsdurchlaufes:

Zunächst fällt auf, daß die informierten Spieler im Mittel besser abgeschnitten haben als die uninformierten. Da diese Spieler in einigen Perioden der Simulationen Informationen über Kurse zum Zeitpunkt `t+1` erhalten, überrascht dieses Ergebnis nicht.

Weiterhin ist zu sehen, dass die Ergebnisse der risikowilligen Spieler im Mittel höher sind als die der risikoaversen und zugleich eine höhere Standardabweichung aufweisen. Auch dies entspricht der Intuition.

Auch im Mehrbenutzerbetrieb wurde das Modell getestet. An diesem Test waren fünf gleichzeitig eingeloggte Spieler beteiligt. Die Spieler bekamen als Endgerät je einen PC zur Verfügung gestellt, die über ein LAN verbunden waren. Auch in diesem Test schnitten die informierten Spieler

[7]Diese Option von *HubNet* funktioniert nicht immer problemlos

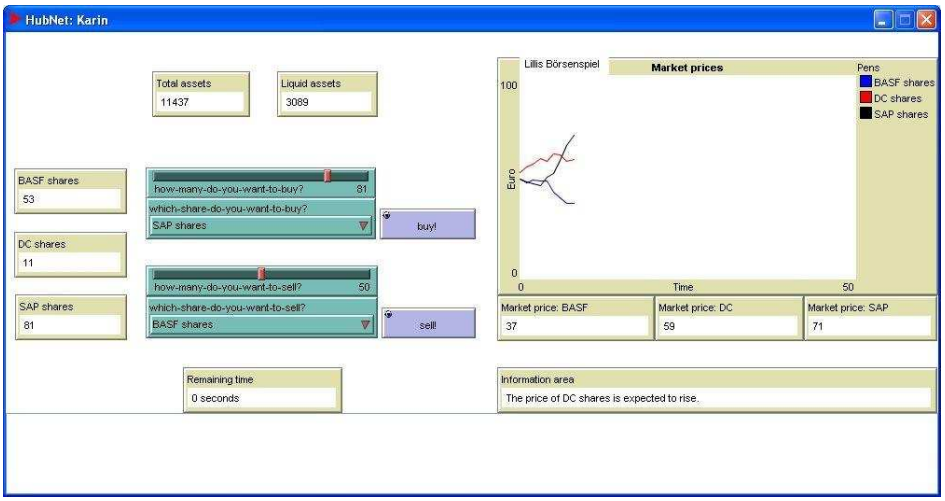


Figure 7: Benutzeroberfläche des Clients

Table 1: Testergebnisse (Gesamtvermögen)

	Spieler 1	Spieler 2	Spieler 3	Spieler 4
Risikoaversion	niedrig	hoch	niedrig	hoch
Informationsstand	niedrig	niedrig	hoch	hoch
Simulationsergebnisse:				
Mittelwert	13550	11650	23996	20843
Standardabweichung	1662	725	4144	3333

im Schnitt weitaus besser ab, als die nicht informierten. Die Spieler wurden gebeten, nach jedem Durchlauf Anmerkungen zur Bedienbarkeit des Modells zu notieren. Ferner erhielten sie die Anweisung, technische Probleme zu vermerken, falls solche auftreten.
Betreffend der Bedienbarkeit wurden folgende Probleme geäußert:

- Viermal: Kaufen/Verkaufen über Slider und Choices unhandlich.
- Viermal: Wenig Zeit, Informationen zu lesen und rechtzeitig zu handeln.
- Einmal: Zwei Kursverläufe wurden im Plot schwarz angezeigt, was die Übersichtlichkeit stark einschränkt.

Die Periodenlänge, die vor dem Test fünf Sekunden betrug, wurde aufgrund dieser Anmerkungen auf zehn Sekunden verlängert. Technische Probleme wurden von keinem der Teinehmer vermerkt.

6. LITERATURVERZEICHNIS

Ingersoll, Jonathan E. Jr. (1987): *Theory of Financial Decision Making.* Savage: Rowman

Roth, Martin T. und Beck, Hanno (2002): *So funktioniert: Die Börse.* Societäts-Verlag

Spann, Martin (2002): *Virtuelle Börsen als Instrument zur Marktforschung.* Deutscher Universitätsverlag GmbH

Wilensky, U. (1999): *NetLogo 2.0.1 User Manual.* Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. `http://ccl.northwestern.edu/netlogo/`.