



UNIVERSITÄT HAMBURG

FACHBEREICH WIRTSCHAFTSWISSENSCHAFTEN

INSTITUT FÜR WIRTSCHAFTSINFORMATIK

## **Seminar zu Multiagentensystemen**

Prof. Dr. Stefan Voß

Thema: 9

### **Implementierung agentenbasierter Verhandlungsmechanismen in E-Learning-Szenarien**

Betreuer: Dr. A. Fink

Abgabe: 06.02.2004

Vorgelegt von:

Lars Hodum

Matr.-Nr. 5340733

Studiengang Wirtschaftsinformatik

Fachsemester 7

0Hodum@informatik.uni-hamburg.de

---

# Inhaltsverzeichnis

<b>ABBILDUNGSVERZEICHNIS</b>	<b>II</b>
<b>TABELLENVERZEICHNIS</b>	<b>III</b>
<b>ABKÜRZUNGSVERZEICHNIS</b>	<b>IV</b>
<b>1 EINLEITUNG</b>	<b>1</b>
1.1 PROBLEMSTELLUNG	1
1.2 ZIELSETZUNG	1
1.3 GANG DER UNTERSUCHUNG	1
<b>2 EINFÜHRUNG IN HOTFRAME</b>	<b>2</b>
<b>3 PLANUNG DER SOFTWARE</b>	<b>3</b>
3.1 PROGRAMMIERSPRACHEN UND ENTWICKLUNGSWERKZEUGE	3
3.2 KONFIGURATION DES ZU ERSTELLENDEN EXPERIMENTS	3
3.3 BENÖTIGTE KOMPONENTEN DER SOFTWARE	3
<b>4 ENTSTANDENE KOMPONENTEN</b>	<b>5</b>
4.1 VISUALISIERUNGSKOMPONENTEN	6
4.1.1 <i>Die Administrations-Homepage (admin.aspx)</i>	6
4.1.2 <i>Klienten-Homepage (client.aspx)</i>	7
4.2 INTERAKTIONSKOMPONENTEN	8
4.2.1 <i>Die C++ Engine (cppAgentenEngine)</i>	8
4.2.2 <i>Die C# Engine (CsAgentenEngine)</i>	8
<b>5 EINSATZ DER SOFTWARE</b>	<b>11</b>
5.1 INSTALLATION SERVERKOMPONENTEN	11
5.2 INSTALLATION DER CLIENTKOMPONENTEN	12
<b>6 BEKANNTE PROBLEME</b>	<b>12</b>
<b>ANHANG (OPTIONAL)</b>	<b>FEHLER! TEXTMARKE NICHT DEFINIERT.</b>
<b>LITERATURVERZEICHNIS</b>	<b>14</b>

## Abbildungsverzeichnis

Abbildung 1: Prinzip der Verhandlung in HotFrame entnommen aus A.Fink(2004)	3
Abbildung 2: Administrations-Homepage	6
Abbildung 3: Klienten-Homepage	7
Abbildung 4: Architektur der gesamten Software als UML Klassendiagramm	10

## **Tabellenverzeichnis**

Tabelle 1: Benutzte Datenbank	12
-------------------------------	----

---

## Abkürzungsverzeichnis

IT	<b>I</b> nformation <b>t</b> echnologie
GUI	<b>G</b> raphical <b>U</b> ser <b>I</b> nterface
C#	C-Sharp, Programmiersprache von Microsoft
HotFrame	<b>H</b> euristic <b>O</b> pTimization <b>F</b> RAMEwork
SQL	<b>S</b> tructured <b>Q</b> uery <b>L</b> anguage
DB	<b>D</b> aten <b>b</b> ank
TSP	<b>T</b> ravelling <b>S</b> alesman <b>P</b> roblem
WT	<b>W</b> aiting <b>T</b> ardiness
IP	<b>I</b> nternet <b>P</b> rotocol
DLL	<b>D</b> ynamic <b>L</b> ink <b>L</b> ibrary
IIS	<b>I</b> nternet <b>I</b> nformation <b>S</b> ervices
HTTP	<b>H</b> ypertext <b>T</b> ransfer <b>P</b> rotocol

# 1 Einleitung

Der nachfolgende Text behandelt eine Software, welche die Interaktion von Menschen mit dem Agentenframework Hotframe ermöglicht. Sie wurde für die Vorlesung Multiagentensysteme als Anschauungsmaterial er-, und im gleichnamigen Seminar vorgestellt. Dieses Dokument erstreckt sich nur über die Architektur und die Benutzung der Software, nicht deren genauere Erstellung.

## 1.1 Problemstellung

Für die Vorlesung Multiagentensysteme des Lehrstuhls Wirtschaftsinformatik der Universität Hamburg sollte Anschauungsmaterial zur Verhandlung von computergesteuerten Agenten, sowie deren Interaktion mit Menschen bereitgestellt werden. Zu diesem Zweck sollte eine Software erstellt werden, welche die Interaktion von Menschen mit Agenten welche bereits in Softwareform im Agentenframework HotFrame existierten erstellt werden. Diese sollte dann zusätzlich im Seminar Multiagentensysteme für empirische Tests eingesetzt und vorgestellt werden.

## 1.2 Zielsetzung

Die Erstellung der Software sollte mit Hilfe der Programmiersprache C# geschehen. Das fertige Produkt sollte die Interaktion vom Menschen und Softwareagenten ermöglichen. Außerdem sollten die Verhandlung persistent gespeichert werden, so dass eine Auswertung der Verhandlung zu einem späteren Zeitpunkt möglich wäre.

## 1.3 Gang der Untersuchung

Für die Erstellung der geforderten Software war eine umfangreiche Einarbeitung in die Programmiersprachen C++<sup>1</sup> und C# notwendig. Außerdem waren geringere Kenntnisse der Datenbankabfragesprache SQL nötig.

Als Einstiegspunkt wurde das vorliegende Agentenframework HotFrame analysiert, um Einstiegspunkte für eine Interaktion der Softwareagenten mit Menschen zu ermöglichen. Als diese Einstiegspunkte gefunden waren, wurde eine Basiskomponente geplant, welche sowohl HotFrame Elemente als auch Kommunikationselemente enthalten sollte. Anschließend wurde die Planung der Interaktionskomponente und der Visualisierungskomponente vorgenommen. Als letztes wurde die persistente Speicherung der Daten eingeplant.

---

<sup>1</sup> Dies war nötig, da Hotframe in C++ geschrieben wurde und ein direktes Eingreifen in Hotframe nötig war.

## 2 Einführung in HotFrame

Die nachfolgende Betrachtung von HotFrame erstreckt sich nur auf den Bereich den der Autor dieses Berichts kennen gelernt hat, darum besteht hier kein Anspruch auf eine vollständige Beschreibung dieses Frameworks<sup>2</sup>.

Bei HotFrame handelt es sich um ein in C++ geschriebenes Agentenframework, welches stark auf Templates<sup>3</sup> basiert. Dadurch wird es sehr gut an gegebene Experimente anpassbar. Die Basisarchitektur sieht einen Mediator vor, welcher die Verhandlung der einzelnen Agenten koordiniert. Dies kann man sehr gut mit Hilfe von Abb.1 nachvollziehen. Das Verhalten der Agenten wird durch Bewertungskriterien bestimmt, welche beim erstellen einer Agentenausprägung festgelegt werden. Zu Beginn der Verhandlung erzeugt der Mediator eine Basislösung welche alle Agenten vorgeschrieben bekommen. Anschließend erzeugt er in jeder Verhandlungsrunde eine von der Basislösung abweichende Lösung die er den Agenten vorschlägt. Die Art, in welcher diese Lösung von der Basislösung abweicht lässt sich durch Kriterien beeinflussen die dem Mediator beim Erschaffen mitgegeben werden. In der erstellten Software ist dies immer das RandomNeighbour Kriterium. Nun entscheiden die Agenten nach ihrem Bewertungskriterium ob sie die vorgeschlagene Lösung annehmen wollen oder nicht. Beim Erschaffen des Mediators wird bei diesem auch ein Abstimmungskriterium festgelegt, welches festlegt wie viele Agenten in einer Runde dem Vorschlag zustimmen müssen, damit der Vorschlag für alle Agenten übernommen wird. In der erstellten Software ist dieses Kriterium auf Paretooptimalität<sup>4</sup> festgelegt. Wenn alle Agenten ihre Entscheidung beim Mediator abgeliefert haben, wertet dieser die Runde aus und fängt eine neue an. Dies geschieht solange bis vordefinierte Zielkriterien erreicht sind. Dies können bestimmte Zielfunktionswerte oder eine feste Rundenzahl sein. In der erstellten Software ist eine statische Variable vorgesehen, mit der sich eine Rundenzahl vorgeben lässt.

---

<sup>2</sup> Eine ausführliche Beschreibung von HotFrame findet sich in A. Fink, S. Voß(2002).

<sup>3</sup> Bei Templates handelt es sich um generische Klassen für die noch zur Laufzeit konkrete Klassen eingesetzt werden können. Mehr zu diesem Thema z.B. in Stroustrup(2003).

<sup>4</sup> Dies bedeutet alle Agenten müssen dem Vorschlag zustimmen, damit er die neue Basislösung wird

In Abb.1 kann man den gesamten Vorgang nachvollziehen. Hier wird er zwischen zwei Agenten dargestellt. Grundsätzlich lässt sich aber auch die Anzahl der teilnehmenden Agenten frei variieren.

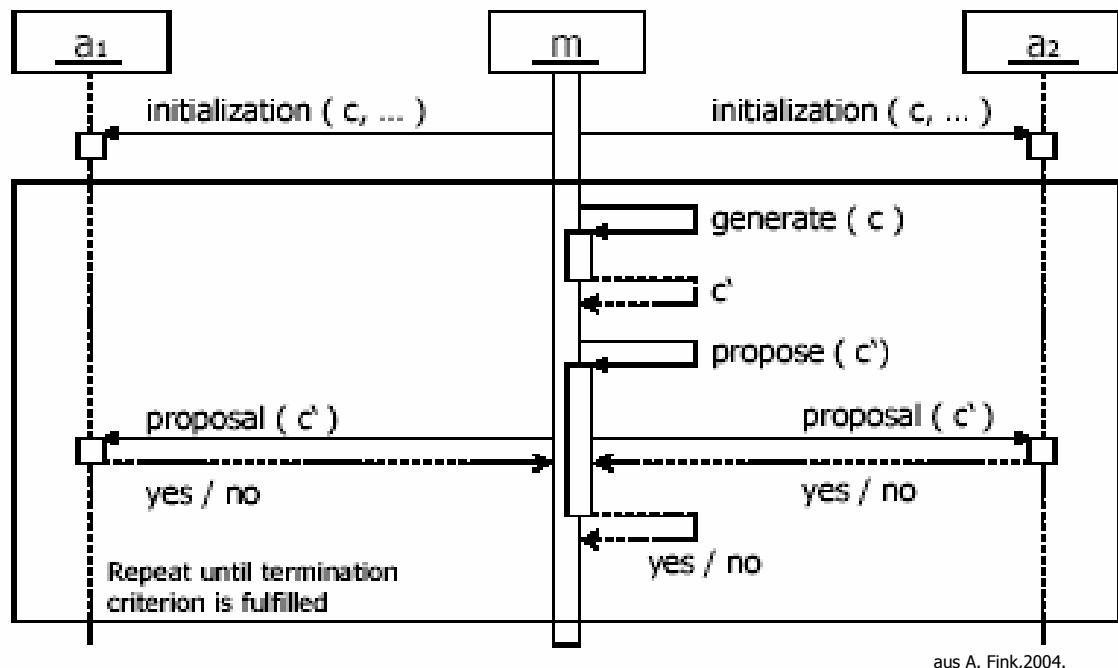


Abbildung 1: Prinzip der Verhandlung in HotFrame entnommen aus A.Fink(2004)

### 3 Planung der Software

#### 3.1 Programmiersprachen und Entwicklungswerkzeuge

Als Programmiersprache sollte C# zum Einsatz kommen. Dies ist eine Programmiersprache die von Microsoft .Net 1.1 unterstützt wird. Durch .Net 1.1 sollte auch ein einfacher Übergang von C# Komponenten zu C++ Komponenten möglich sein.

Als Programmierumgebung wurde Visual Studio .Net 2003 gewählt, da es als einzige Umgebung zu diesem Zeitpunkt C++ und C# in einem Projekt unterstützte.

#### 3.2 Konfiguration des zu erstellenden Experiments

Die Software beschränkt sich auf Experimente, in denen jeweils 2 Agenten miteinander Verhandeln. Der erste Agenten behandelt dabei ein TSPO Problem, der 2. ein WT Problem. Das zugrundeliegende Szenario beschränkt sich auf 5 Knoten, bzw. 5 Umrüstungen.

#### 3.3 Benötigte Komponenten der Software

Als Einstiegspunkt der Interaktionskomponente in das bestehende HotFrame für menschliche Eingriffe wurde ein neues Bewertungskriterium geschaffen: das AssessmentCriterionHuman.

Dies bedeutet, dass der Mensch keinen eigentlichen Agenten ersetzt, sondern ein computergesteuerter Agent ruft seine Entscheidungen beim Menschen ab, anstatt sie selbst zu treffen. Da das `AssessmentCriterionHuman` von `AssessmentCriterion`<sup>5</sup> erbt, ist es vollständig in `HotFrame` integrierbar. Zur Laufzeit ruft dieses Bewertungskriterium dann an der Interaktionskomponente eine Entscheidung des Menschen ab.

Im Code sieht das folgendermaßen aus:

Wenn der Mediator ein Proposal (siehe auch Abb.1) an den Agenten stellt, ruft dieser intern an seinem `AssessmentCriterion` die Methode `Assess` auf. Diese gibt im Normalfall eine Bewertung des Proposals zurück. Das `AssessmentCriterionHuman` leitet nun das aktuelle Proposal an die Interaktionskomponente weiter und wartet dann auf eine Entscheidung des zugeordneten menschlichen Spielers.

Die Interaktionskomponente musste auch in C++ gestaltet werden, damit eine Interaktion mit `HotFrame` überhaupt möglich war. Dies lag daran, dass das .Net 1.1 Framework von Microsoft noch keine Templates unterstützt und so eine totale Übersetzung des nativen<sup>6</sup> C++ Codes in managed<sup>7</sup> C++ Code nicht möglich war. Darum wurde die Interaktionskomponente als eine Art Übersetzerklasse gestaltet, die sowohl managed als auch nativen C++ Code enthielt.

Die Visualisierungskomponente sollte als Webseite dargestellt werden. Daher wurde der Einsatz von Webforms<sup>8</sup> geplant. Diese sollten eine leichte Interaktion zwischen dem Kernprogramm und den HTML Elementen ermöglichen. Dazu wurden HTML Seiten mit Code-Behind<sup>9</sup> Programmen erstellt.

Da es sich bei der Visualisierungskomponente um eine HTML basierte Variante handelte, konnte nur eine Pull-Driven<sup>10</sup> Architektur durch die Visualisierungskomponente umgesetzt werden. Dies bedeutet, die Klienten erfragen in bestimmten Zeitabständen ob im Moment

---

<sup>5</sup> Hierbei handelt es sich um eine Basisklasse für Bewertungskriterien die in `HotFrame` existiert

<sup>6</sup> Nativer Code ist nicht vom Microsoft angepasster C++ Code wie er auch in Stroustrup(2003) beschrieben wird.

<sup>7</sup> Managed Code ist eine von Microsoft angepasste und veränderte Variante von C++. Dadurch soll dieser Code vollständig in .Net integrierbar sein. C++ wurde in dieser Variante z.B. um einen Garbage Collector erweitert.

<sup>8</sup> Webforms sind Elemente wie Buttons, Textboxen, Lists etc. die sich im namespace `System.Web.UI.WebControls` des .Net 1.1 Frameworks befinden. Durch sie wird ein einfacheres Ansprechen, und Modifizieren von entsprechenden HTML Objekten ermöglicht. Mehr Informationen findet man z.B. unter [www.MSDN.com](http://www.MSDN.com) wenn man den Suchbegriff „web forms“ mit der entsprechenden Programmiersprache eingibt.

<sup>9</sup> Dies bedeutet dass im Header der HTML Seite auf ein Stück Code in einer extra Datei hingewiesen wird, diese kann vorkompiliert sein, aber auch erst zur Laufzeit kompiliert werden. Außerdem wird bei den einzelnen HTML Elementen festgelegt ob der zugehörige Code auf dem Webserver oder dem Klienten ausgeführt werden soll.

<sup>10</sup> Es ist nicht möglich über das HTTP Klienten direkt Daten zu übersenden (Push-Driven). Diese müssen immer explizit selbst danach Fragen (Pull-Driven).

eine Entscheidung benötigt wird. Wenn dies der Fall ist holen sie sich die entsprechenden Daten für die Entscheidung.

Die Speicherung der Rundenverläufe wurde direkt in die AssessmentCriteria hart encodiert. Diese bauen nach einer Entscheidung eine Datenbankverbindung zu einer SQL Datenbank auf und speichern alle ihnen bekannten Informationen der Runde in der DB.

## **4 Entstandene Komponenten**

Im nachfolgenden werden die entstandenen Komponenten deren Architektur und deren Funktion erklärt. Es empfiehlt sich zum Verständnis die Abb. 4 zu rate zu ziehen. Dabei handelt es sich um eine vereinfachtes UML-Klassendiagramm bei denen aus Gründen der Übersichtlichkeit, nicht alle Attribute und Methoden eingetragen wurden.

## 4.1 Visualisierungskomponenten

### 4.1.1 Die Administrations-Homepage (admin.aspx)

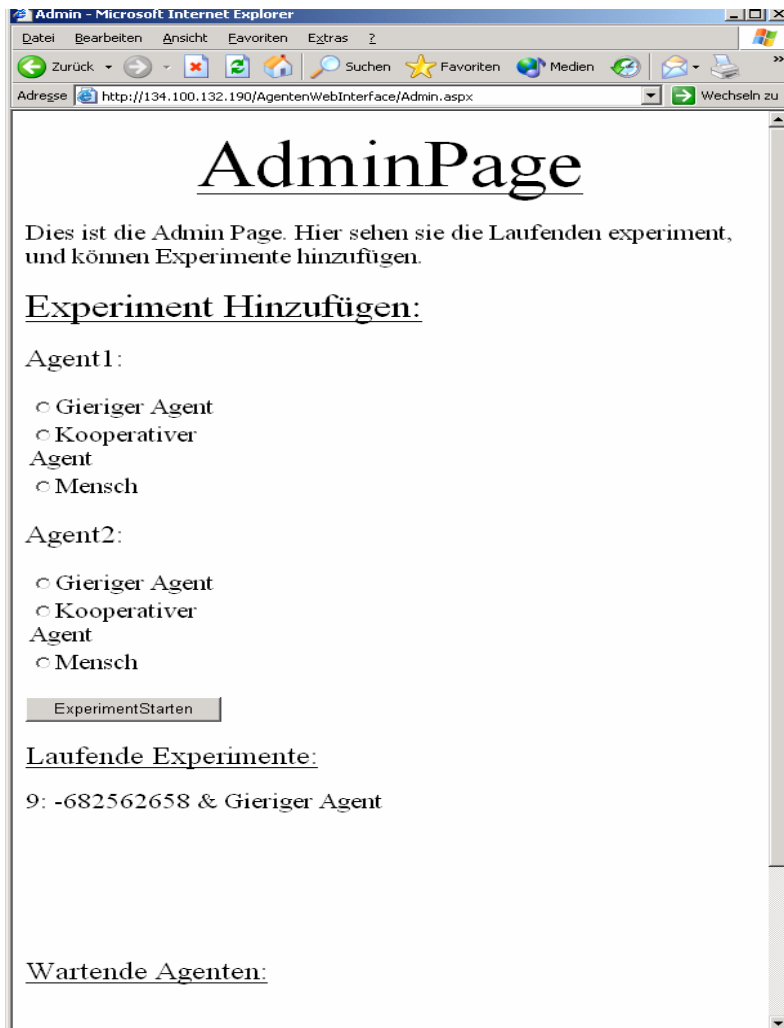


Abbildung 2: Administrations-Homepage

Hier ist das Erstellen von Experimenten möglich. Dazu wählt man aus zwei Checkboxes die Bewertungskriterien aus mit denen die Agenten erstellt werden. Die erste Checkbox stellt den TSP-Agenten dar, die 2. den WT-Agenten.

- Gieriger Agent bedeutet, dass ein vollständig computergesteuerter Agent erschaffen wird der sich gierig verhält.
- Kooperativer Agent ein bedeutet, dass ein vollständig computergesteuerter Agent erschaffen wird der sich kooperativ verhält.
- Mensch bedeutet, dass ein computergesteuerter Agent erschaffen wird der den Mensch die Entscheidungen treffen lässt.

In der Liste „Laufende Experimente“ sieht man alle zurzeit laufenden Experimente. Bereits abgeschlossene Experimente werden wieder aus der Liste gelöscht und nicht mehr angezeigt.

In der Liste „Wartende Agenten“ sieht man alle zurzeit eingeloggt Menschen, die an keinem laufenden Experiment teilnehmen und auf eine Einteilung warten. Zu sehen ist eine Nummer mit evtl. negativem Vorzeichen. Dies ist ein Hash-Wert über die IP-Adresse des Rechners an dem der Mensch sitzt.

Wenn man eine Auswahl getroffen hat und auf den Knopf „ExperimentStarten“ drückt, wird ein Experiment mit den Ausgewählten Agenten als einzelner Thread<sup>11</sup> gestartet. Dabei werden evtl. gewählte Menschen von oben aus der Liste entfernt und eingeteilt. Auch evtl. Fehler, wie das Nichtanwählen aller Checkboxes, werden schon an dieser Stelle intern abgefangen.

#### 4.1.2 Klienten-Homepage (client.aspx)

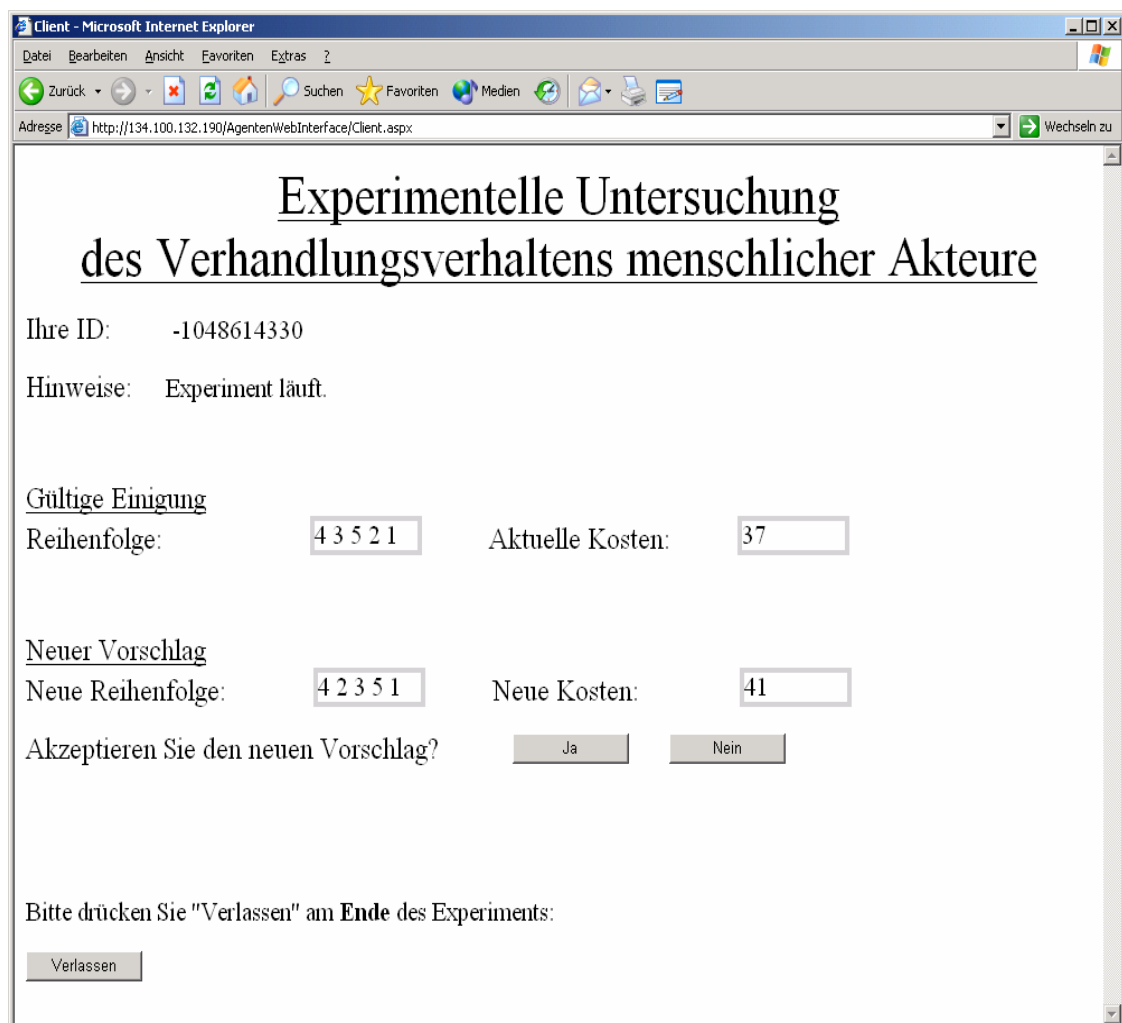


Abbildung 3: Klienten-Homepage

Öffnet ein Mensch die Klienten-Homepage, trägt sich diese beim Starten automatisch in die Liste der wartenden Agenten ein.

<sup>11</sup> Ein Thread ist ein programmiersprachliches Werkzeug um eine parallele Abarbeitung verschiedener Vorgänge zu ermöglichen.

Auf dieser Seite wartet ein eingeloggtter Mensch auf ein Experiment. Wenn er in ein solches eingeteilt wurde, wird ihm zum gegebenen Zeitpunkt einer Runde ein Vorschlag gemacht. Dieser enthält die aktuelle Basislösung und den neuen Vorschlag, sowie die jeweils errechneten Funktionswerte der Lösungen. Nun darf er entscheiden, ob er die neue Lösung annehmen darf oder nicht. Auf der Client-Homepage wird keine Historie des Verlaufs des Experiments dargestellt. Zum gegenwärtigen Zeitpunkt wird auch nicht angezeigt, in welche Runde sich der Client befindet.

Außerdem ist es dem Klienten möglich sich aus der Liste der wartenden Klienten über den „Verlassen“ Button zu entfernen<sup>12</sup>. Dies ist nur im wartenden Zustand möglich, da das System keine Austritte während eines Experimentes verarbeiten kann.

Wenn ein Experiment durchgelaufen ist, tritt man wieder in den wartenden Status ein und wird erneut in die Liste der wartenden Agenten eingetragen.

## **4.2 Interaktionskomponenten**

### **4.2.1 Die C++ Engine (cppAgentenEngine)**

Diese Komponente enthält den bereits erwähnten gemischten Code aus nativem und managed C++. Sie stellt die eigentliche Schnittstelle zu HotFrame da, und stellt eine Reihe von Methoden, die zur Interaktion mit den Agenten nötig sind für andere Objekte bereit. Außerdem hält sie Sammlungen mit Informationen über die gestarteten Experimente und deren Zustand vor. Durch diese wird es auch für die Klienten möglich festzustellen ob im Moment eine Entscheidung benötigt wird oder nicht. Diese Komponente wird zur weiteren Verarbeitung in eine .DLL kompiliert

### **4.2.2 Die C# Engine (CsAgentenEngine)**

Dies ist eine reine Kommunikationsklasse zwischen den Homepages und der cppAgentenEngine. Sie ist als Singleton<sup>13</sup> implementiert. Das CsAgentenEngine-Objekt ist das Objekt auf welches sowohl die Klienten- als auch die Administration-Homepage direkt zugreifen. Das CsAgentenEngine-Objekt importiert die cppAgentenEngine DLL mit Hilfe einer internen Wrapperklasse (AgentWrapper). Es hält außerdem eigene Sammlungen über die laufenden Experimente und die Anwesenden Klienten sowie deren Status. Mithilfe der bereitgestellten Methoden der cppAgentenEngine repliziert das Objekt sich Daten über gestartete Experimen-

---

<sup>12</sup> Wenn dieser Button angeklickt wird, wird die Verbindung zum Server beendet, dadurch erscheint eine Fehlerseite deren Betitelung vom Browser abhängt. Dies ist gewollt und völlig normal.

<sup>13</sup> Vgl. E.Gamma et al.(1995).

te und deren Runden in seine Sammlungen. Außerdem startet es bei einer Anfrage der Administrations-Homepage neue Experimente. Das Objekt stellt weiterhin Methoden zum abfragen der Runden durch die Klienten bereit.

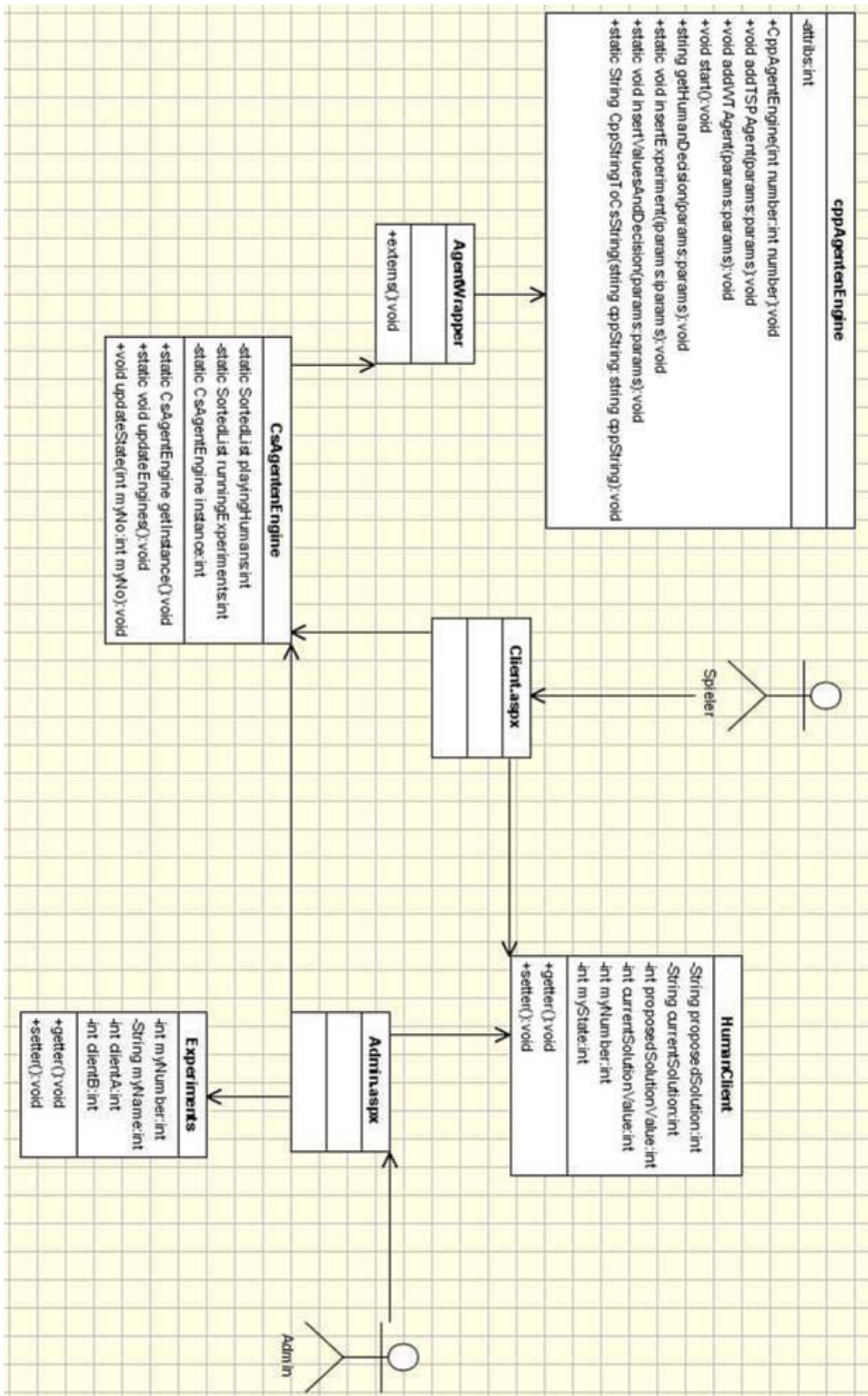


Abbildung 4: Architektur der gesamten Software als UML Klassendiagramm

## 5 Einsatz der Software

### 5.1 Installation Serverkomponenten

Als Server wird ein ASP 1.1 fähiger Webserver benötigt, der auch in der Lage sein muss die Kompilierung des serverseitigen Codes anzustoßen. Im Testsystem kam ein IIS 5.1<sup>14</sup> zum Einsatz. Außerdem muss eine .Net 1.1 SDK Redistribution installiert sein. Die ASP 1.1 Filter der Redistribution müssen möglicherweise von Hand in den Webserver eingepflegt werden<sup>15</sup>. Außerdem ist die Installation eines SQL Servers erforderlich. Es wird empfohlen ein MySQL<sup>16</sup> Server zu benutzen bei dem ein ODBC Treiber der Version 3.51 installiert ist, da dieser bereits in den Code der Software eingepflegt ist. Ein abweichender ODBC Treiber macht eine Änderung des Quellcodes notwendig. Auf dem SQL Server sollte der „localhost“<sup>17</sup> Schreibrechte auf die benutzte Datenbank besitzen. Diese kann über ein File<sup>18</sup> oder von Hand erstellt werden. Die Struktur ist folgende:

---

<sup>14</sup> Hierzu sollte gesagt sein, dass die Versionsnummer vom Betriebssystem abhängt. Man muss also nur den IIS des jeweiligen Betriebssystems installieren, da es meist gar nicht möglich ist andere Version auf das Betriebssystem zu portieren.

<sup>15</sup> Dies funktioniert in der Regel Halbautomatisch, indem man sich in das Verzeichnis der gewünschten .Net Installation begibt und dort das Programme „aspnet\_regiis.exe“ mit dem Parameter „-i“ ausführt. Als Beispiel: „C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322\aspnet\_regiis -i“.

<sup>16</sup> Sowohl der Server als auch der Treiber sind unter [www.mysql.com](http://www.mysql.com) für private Zwecke frei verfügbar.

<sup>17</sup> Damit ist ein Recht für den lokalen Rechner gemeint, was alle (auch anonyme Benutzer) erhalten, welche vom lokalen Rechner aus operieren. Dies ist hier notwendig, da sich die erstellte Software nur anonym anmeldet.

<sup>18</sup> (ergänzen)

Datenbankname: agentendb		
Tabellenname:	Parameter:	DefaultWert:
experiments	int(11) expNo,	0
	varchar(39) Participants	NoParticipantsSet
round	int(11) expNo,	0
	int(11) Round,	0
	int(11) AgentNo,	0
	int(11) AgentType,	0
	varchar(20) ProposedSolution,	0-0-0-0-0
	int(11) ProposedSolutionValue,	0
	varchar(20) CurrentSolution,	0-0-0-0-0
	int(1) Decision,	0
	int(11) CurrentSolutionValue	0

Tabelle 1: Benutzte Datenbank

Die Software wird in einem Ordner namens „AgentenWebInterace“ bereitgestellt. In diesem sind bereits vorkompilierte Komponenten enthalten. Dieser Ordner muss vom Webserver zugänglich gemacht werden<sup>19</sup>. Die Startseite ist die Datei „start.aspx“ im entsprechenden Ordner. Wenn man nun den Webserver und den SQL-Server startet ist das System betriebsbereit.

## 5.2 Installation der Clientkomponenten

Da die gesamte Software auf dem Server ausgeführt wird ist auf dem Client nur ein aktueller Webbrowser erforderlich. Im Testsystem wurde ein Internet Explorer der Version 6.0 benutzt. Das System ist auf Kompatibilität mit anderen oder älteren Browsern getestet worden.

## 6 Bekannte Probleme

- Die Administrations- und Clientpage laden sich alle paar Sekunden selbst neu ein um aktuelle Ergebnisse anzuzeigen. Dadurch kann eine bereits getroffene Anwahl der Checkboxen auf der Administrationsseite wieder gelöscht werden. Hier ist es erforderlich die Anwahl einfach schnell vor dem nächsten Laden zu vollziehen.
- Zu schnelles erstellen von Experimenten auf der Administrationsseite kann die Engine zum Absturz bringen. Diese ist vermutlich auf eine Verwicklung der Threads mit den

<sup>19</sup> Im IIS sollte dieser Ordner dafür am besten in das Verzeichnis Inetpub – welches vom IIS angelegt wurde – kopiert werden. Dann kann man dieses Verzeichnis z.B. bei der Standardwebseite unter der Kategorie Basisverzeichnis angeben und unter Dokumente die Datei „start.aspx“ eintragen.

eingehenden Buttonevents zurückzuführen. Dem Problem wurde nicht weiter nachgegangen, da es nur auftritt, wenn man Experimente unter einer halben Sekunde oder kürzer hintereinander startet.

- Es wird dringend angeraten die Administrationswebseite direkt auf dem Server auszuführen. Remote-Administration ist zwar möglich, es sind jedoch unerklärliche Fehler bei Remotezugriffen aufgetreten.<sup>20</sup>
- Die Clients sollten die Homepage nur über „Verlassen“ Button verlassen. Ein verlassen durch normales schließen des Fenster, führt aufgrund der Architektur dazu dass der Server nicht mitbekommt, dass ein Client nicht mehr anwesend ist. Es ist dem Client allerdings durch einfaches Neuöffnen des Fensters möglich, wieder in das Experiment einzusteigen. Von dort aus kann er dann ganz normal weiterspielen oder das Experiment Ordnungsgemäß verlassen.
- Sollte man einen Fehler bei den Experimenten feststellen, ist es erforderlich den Webserver neu zu starten damit die Engine auch komplett neu gestartet wird. Danach sollten auch die Tabellen des SQL-Servers geleert werden<sup>21</sup>, da die Engine dann wieder von vorne zählt, und sonst Experimente doppelt eingetragen werden
- Es sollte immer darauf geachtet werden, das beim starten der Engine die Tabellen des SQL-Servers leer sind, da sonst bereits oben genanntes Problem eintreten kann

---

<sup>20</sup> Z.B. wurden nicht existente „leere“ Agenten in Experimente eingeteilt. Dieser Fehler konnte nicht aufgeklärt werden, da der Code genau solch einen Fehler im Normalfall abfangen würde, da jeder Agent vorher geprüft wird. Die Vermutung liegt nah, dass bei der Übertragung der Buttonevents Informationen auf dem Weg zum Server verloren gehen.

<sup>21</sup> Dies sollte möglicherweise das abspeichern der Tabellen, für spätere Wiederverwendung beinhalten.

## Literaturverzeichnis

- [A.Fink(2004)]** A. Fink  
*Supply Chain Coordination by Means of Automated Negotiation*, at 37th Hawaii International Conference on System Sciences (HICSS-37)  
2004
- [A. Fink, S. Voß(2002)]** A. Fink, S. Voß.  
HotFrame: A Heuristic Optimization Framework. In: S. Voß, D.L. Woodruff (Eds.): *Optimization Software Class Libraries*. Boston : Kluwer, 2002, S. 73
- [E.Gamma et al.(1995)]** E. Gamma; R. Helm;R. Johnson;J.Vlissides; G.Booch.  
*Design Patterns*. not known : Addison-Wesley, 1995
- [MSDN(2004)]** Microsoft.  
*MSDN*. [www.msdn.com](http://www.msdn.com)
- [S. Voß, D.L. Woodruff (2002)]** S. Voß, D.L. Woodruff (Eds.).  
*Optimization Software Class Libraries*. Boston : Kluwer, 2002
- [Stroustrup(2003)]** Bjarne Stroustrup.  
*Die C++-Programmiersprache*. 4. Aufl., München : Addison-Wesley, 2003.  
– ISBN 3–8273–1660–X