Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projekt - Master Informatik

Ilia Revout

Envirement for distributed aplication (EDA)

Fakultät Technik und Informatik
Studiendepartment Informatik
Betreuer: Prof. Dr. von Luck
Datum: 15. Februar 2006

Ilia Revout

Projekt - Master Informatik

Envirement for distributed aplication (EDA)

Stichworte

Verteilte Anwendung, mobiler Code, mobile Agenten, Framework

Kurzzusammenfassung

Mit diesem Projekt war eine Untersuchung durchgeführt, inwieweit die Technik der mobilen Agenten für die Probleme bei verteilten Anwendungen in einem lokalem Netzwerk einsetzen lässt.

Inhaltsverzeichnis

1 Einführung

Die verteilten Anwendungen gewinnen rasch an ihrer Bedeutung. Bei der Entwicklung von solchen Anwendungen müssen Probleme gelöst werden, die für die Einzelnplatzanwendungen nicht in Frage kämen. Dabei ist die Komplexität solcher verteilten Probleme wesentlich höher im Vergleich zu normalen Problemen. Unter 'normalen' verstehe ich in diesem Kontext die Probleme, die bei der Entwicklung von Einzelnplatzanwendungen gelöst werden müssen.

Es existiert unzählige Menge von Strategien, wie die verteilten Anwendungen realisiert werden können. Abhängig von der Umgebung und der Art der Anwendung werden unterschiedliche Verfahren bevorzugt. In diesem Projekt wurde eine Architektur untersucht, die aus dem Bereich 'Mobile Agenten' kommt. Die Umgebung wurde dabei auf ein lokales Netzwerk abgegrenzt, damit die Untersuchung von Problemen des öffentlichen Netzes (z.B. Internet) nicht verzögert wird.

Im ersten Kapitel ist ein Szenario vorgestellt, anhand dessen ein Problem beschrieben wird. Dabei wird eine Idee vorgestellt, wie das Problem gelöst werden kann. Im Kapitel 'Zielsetzung' sind die Ziele des Projektes festgelegt. Da für die Untersuchung viele Bereiche irrelevant oder zu komplex sind, wurde die Umgebung eingeschränkt. Im Kapitel 'Rahmenbedingungen' ist beschrieben, wovon bei der Untersuchung ausgegangen wurde und welche Bereiche dabei nicht berücksichtigt sind. Im Kapitel 'Entwurf' ist die entstandene Architektur vorgestellt. Ohne auf Details einzugehen, werden die wichtigsten Ideen vorgestellt. Nachfolgendes Kapitel beschreibt die Realisierung. Abschließend wird im Kapitel 'Fazit' berichtet, ob die gesetzten Ziele erfüllt sind und welche Probleme dabei besondere Schwierigkeiten bereitet haben.

2 Vision

Wir stellen uns eine Firma vor, wo mehrere Rechner miteinander vernetzt sind. In dieser Firma laufen unterschiedlichste Anwendungen, die möglicherweise auch von unterschiedlichen Softwarehersteller stamen. Diese Szenario ist nichts ausgedachtes, sondern ist der Stand der heutigen Zeit. Oft tritt die Situation auf, dass bestimmte Ressourcen einer Anwendung von einer anderen benötigt werden. Es entsteht das Kommunikationsproblem zwischen zwei unterschiedlichen Anwendungen. Es existieren unterschiedlichste Verfahren (z.B. RMI, CORBA), die bei solchen Problemen zum Einsatz kommen. Allerdings zeigen solche Verfahren eine meiner Meinung nach große Schwäche. Beide Programme müssen entsprechend angepasst werden. In unserem Fall muss die betroffene Firma mehrere Schritte unternehmen:

- Beschreiben welche Ressourcen an welcher Stelle und von welcher Anwendung gebraucht werden
- Jeweils die betroffenen Softwarehersteller kontaktieren und Auftrag aufgeben

- Diese Softwarefirmen müssen eine gemeinsame Lösung erstellen
- Unsere Firma muss für beide Programme Update durchführen und beide Firmen bezahlen.

Es ist sicherlich klar, dass die dafür notwendige Zeit ziemlich lang und die Kosten ziemlich hoch werden können, besonders wenn mehr als zwei Produkte dabei betroffen sind.

Eine Idee, solche Probleme zu umgehen, ist eine universelle Schnittstelle zu schaffen, die es ermöglicht, einem Programm die Ressourcen eines anderen Programms beliebig zu nutzen.

3 Zielsetzung

In diesem Projekt soll die vorgestellte Idee umgesetzt und dabei getestet werden, ob sie tragfähig ist. Die folgenden Kriterien sollen dabei berücksichtigt werden:

- Die Zeit für die Durchführung des Projektes ist begrenzt
- Das Ergebnis soll sichtbar sein
- Es muss anwendbar sein

Dem entsprechend wurden folgende Ziele festgelegt:

- Erstellung einer Laufzeitumgebung, die eine universelle Schnittstelle anbietet und leicht zu installieren ist.
- Erstellung einer Beispielanwendung für das Testen der Umgebung und für die Demonstration des möglichen Einbinden der Umgebung in ein bestehendes System.

4 Rahmenbedingungen

In diesem Projekt wird davon ausgegangen, dass alle Rechner sich in einem lokalen Netz befinden, so dass alle Sicherheitsaspekte keine zentrale Rolle spielen. Entsprechend sind die Eigenschaften der Verbindung: schnell und stabil. Das ist eine typische Konfiguration in den Firmen, auf den Schiffen oder in den Ferienklubs.

Es wird eine Annahme getroffen, dass alle Anwendungen Java-Programme sind, und alle die gleiche Version der JVM (Java virtual machine) benutzen. Der Grund, warum dieses Projekt mit der Programmiersprache Java realisiert wurde, liegt darin, dass die J2SE5.0 sehr leistungsfähige Klassenbibliotheken besitzt, die für das Projekt notwendige Funktionalität bereit stellen. Damit wurde es möglich, an vielen Stellen viel Zeit zu sparen. Weiteres Aspekt ist die Entwicklungsumgebung. Das Projekt wurde mit Eclipse 3.1 implementiert und getestet. Für Eclipse

existiert unzählige Menge von Plug-Ins, die die Funktionalität der Entwicklungsumgebung erweitern. Die XML-Dateien wurden schon beim Schreiben validiert und das Projekt automatisch deployed. So eingerichtete Entwicklungsumgebung hat für weiteren Zeitersparnis beigetragen.

Wie es bereit erwähnt wurde, ist die gesamte Sicherheit (security und safety) in diesem Projekt nicht berücksichtigt wurde. An der Stelle verweise ich auf die Technologie der mobilen Agenten, da dort diese Aspekte sehr detailliert beschrieben und konkrete Realisierungsvorschläge gegeben werden.

5 Entwurf

Zuerst muss der Begriff 'Universelle Schnittstelle' geklärt werden. Es muss für Programm A möglich sein, beliebige Ressourcen von dem Programm B zu nutzen. Unter Ressourcen werden nicht nur die Hardwareressourcen des Rechners verstanden, sondern auch Klassen, die im Programm B zur Verfügung stehen. Die Technologie der mobilen Agenten bietet dafür eine passende Lösung. Dabei wird ein Objekt mit dem Code von einem Rechner zu einem anderem verschickt und auf dem entfernten Rechner ausgeführt. Sicherlich ist es nicht ohne weiteren Aufwand möglich. Um das Objekt empfangen und es ausführen zu können, muss eine spezielle Programmumgebung vorhanden sein. Bei mobilen Agenten übernimmt diese Arbeit den Agentenserver. Seine Aufgaben bestehen unter anderem darin, die Agenten zu empfangen, sie auszuführen und die Migration zu ermöglichen. Unter Migration wird im Kontext der mobilen Agenten die Übertragung des Agenten von einem Rechner zu einem anderem verstanden. Die Abbildung 1 zeigt eine grobe

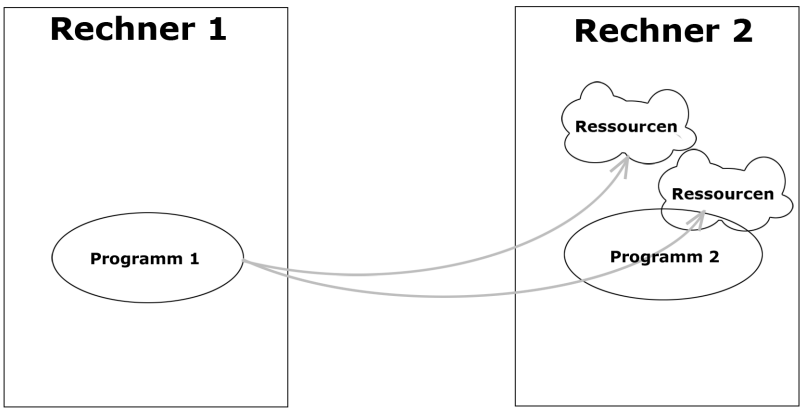

Abbildung 1: Architektur: grobe Skizze

Idee des Projektes. Ein Programm auf dem Rechner 1 braucht die Ressourcen die sich auf dem Rechner 2 befinden. Diese Ressourcen können auf unterschiedliche

Weise erreicht werden. Man kann die Ressourcen in zwei Kategorien unterteilen:

- Direkt ansprechbar
- Indirekt ansprechbar

Die Ressourcen, die man direkt nutzen kann, sind meistens die Hardwarekomponenten. CPU, Festplatte, ISDN-Karte sind Beispiele dafür. Anders geht es bei Ressourcen, die nur über einen Softwareschicht erreichbar sind. Z.B. wenn die Datenstream über eine ISDN-Karte übertragen werden soll, dann ist es sinnvoll es dafür geschriebenes Programm zu nutzen, als direkt die ISDN-Karte mit den Low-Level Befehlen anzusprechen.

Um dieses Szenario zu ermöglichen, wird in diesem Projekt die Migrationstechnik angewendet. Das Programm, welches die Ressourcen nutzen will, soll zu dem Rechner mit diesen Ressourcen gehen. Dabei ergeben sich weitere Vorteile, die im Zusammenhang mit einem asynchronen Aufruf eine Rolle spielen. So ist der Homerechner komplett entlastest, während das Programm auf dem entfernten Rechner seine Arbeit durchführt.

Aus diesen Überlegungen herauskristallisieren sich folgende Komponenten des Projektes.

- Framework - enthält die notwendigen Interfaces für das Schreiben von mobilen Programmen
- Laufzeitumgebung - ist in der Lage, mobile Programme zu empfangen, sie auszuführen und sie weiter zu versenden

Obwohl die mobilen Programme viele Eigenschaften der mobilen Agenten aufweisen, sind sie prinzipiell unterschiedlich. Die mobilen Programme in diesem Projekt sind am Anfang von der Laufzeitumgebung unabhängig. Wenn die Migration nicht notwendig ist, unterscheiden sie sich kaum von einem normalen Programm. Erst wenn das Programm zu einem anderen Rechner migrieren soll, wird die Laufzeitumgebung angesprochen.

6 Realisierung

Das Herz des Projektes ist die Laufzeitumgebung. Die Abbildung 2 zeigt die wesentlichen Klassen. Eine zentrale Stelle besitzt die Klasse 'Transmitter'. Diese Klasse ist als Singelton-Pattern implementiert. Sie hält unter anderem die Instanzen der Klassen 'Sender' und 'Receiver'. Receiver-Klasse ist ein Thread und stellt ein Server dar. Dieser Server wird sofort nach der Erstellung einer Instanz von der Transmitter-Klasse gestartet und überwacht einen bestimmten Port. Der Receiver hat die Möglichkeit, viele 'ReceiverObjectLisener' zu registrieren. ReceiverObjectLisener-Inerface schreibt die Methode vor, die aufgerufen wird, sobald ein mobiles Programm empfangen wurde. Als Parameter in dieser Methode wird das mobile Objekt übergeben, der aus dem Programm-Objekt und dem Programm-Code besteht.

In diesem Projekt übernimmt die Rolle der ReceiverObjectLiseners die Klasse 'Trasmitter'.

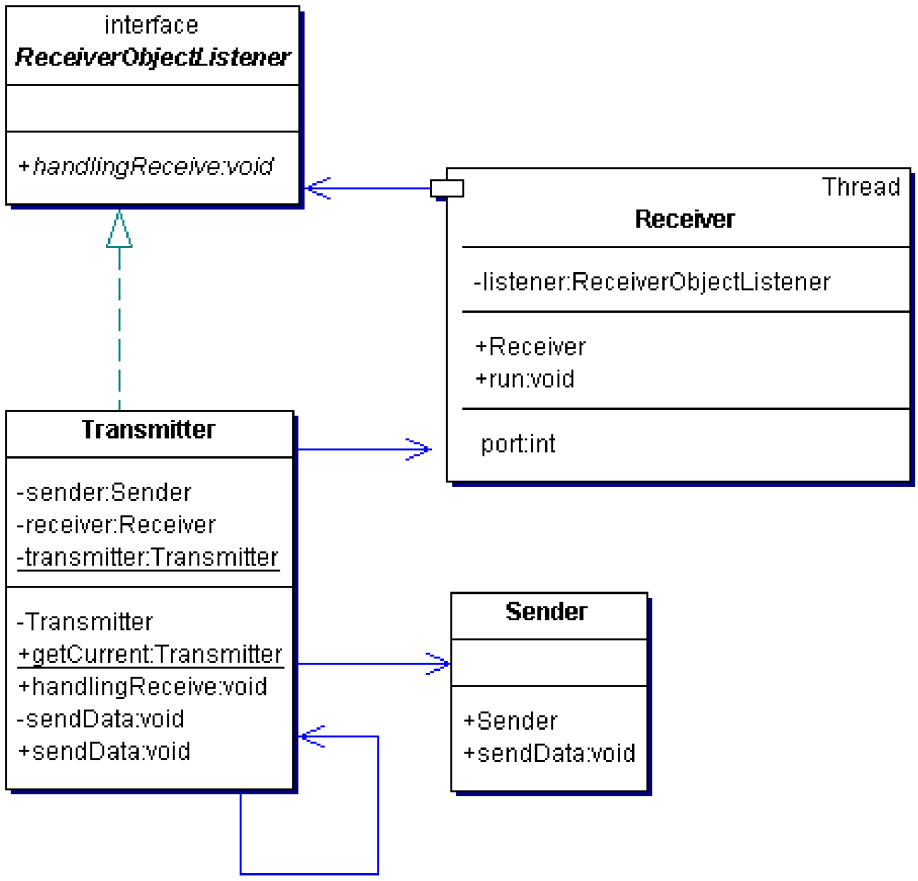

Abbildung 2: Laufzeitumgebung

Sobald die Trasmitter-Klasse das mobile Objekt bekommen hat, werden folgende Schritte durchgeführt:

- Aus dem mobilem Objekt werden alle Klassen geholt und mit Hilfe von einem ClassLoader geladen.
- Das Programm-Objekt wird aus dem mobilen Objekt geholt und daraus eine Instanz erstellt.
- Das Programm wird gestartet

Die Aufgabe der Sender-Klasse besteht darin, das Programm (Objekt und dazu notwendigen Code) zu einem anderen Rechner zu senden. Dabei wird das Programm-

Objekt zusammen mit dem Code zu einem mobilen serialisierbaren Objekt (TransferObject) zusammengefasst. Die Abbildung 3 zeigt die Aufbau der Klasse TransferObject. Die interne Struktur für die notwendigen Klassen ist versteckt und nicht

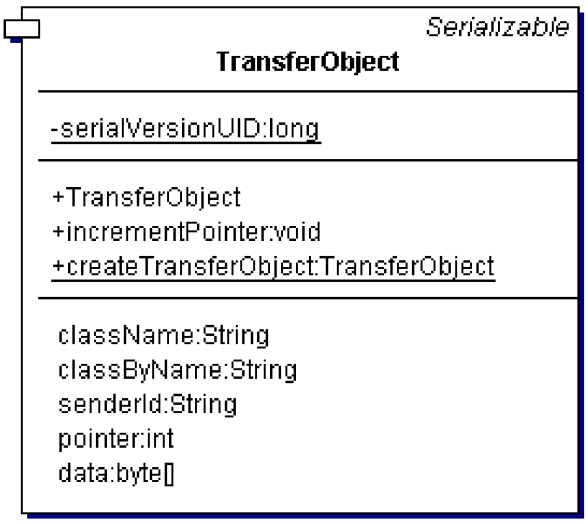

Abbildung 3: TransferObject

in der Abbildung 3 dargestellt. An dieser Stelle ist nur zu erwähnen, dass mit dem Programm-Objekt alle dafür notwendigen Klassen mitübertragen werden. Das entspricht etwa dem Push-Migration aus der Technologie der mobilen Agenten.

Die Sirealisierung und der Classloader wurde bereits kurz erwähnt. Obwohl Java sehr mächtige Bibliotheken in diesem Bereich zur Verfügung stellt, mussten doch einige Änderungen vorgenommen werden. So ist der standard Classloader in diesem Kontext nicht nutzbar. Das Problem liegt in der Tatsache, dass der standard Classloader die Klassen nur ein einziges Mal in den Speicher holt. Die Änderungen der Klasse zur Laufzeit haben keine Auswirkungen und werden nicht berücksichtigt. In diesem Projekt ist es aber sehr wichtig, dass die Klasse auf jedem Fall neu geladen wird, da es durchaus möglich ist, dass die Klasse in einem veränderten Zustand wieder ausgeführt werden soll. Die Abbildung 4 zeigt die Aufbau der Klassen EDAClassLoader und ObjectSerialise. Die ObjectSerialise-Klasse ist für die gesamte Serialisierung verantwortlich. Die Objekte werden auf ein Array von Bytes abgebildet und entsprechend aus einem Array von Bytes wird ein Objekt erstellt. Auch an dieser Stelle mussten ein Paar Änderungen vorgenommen werden. Die Deserialisierung in Java geschiet mit Hilfe der ObjectInputStream-Klasse. Leider geht die Klasse davon aus, dass alle für die Deserialisierung notwendigen Klassen mit dem standard Classloader erreichbar sind. Mobile Klassen werden mit einem speziellen EDAClassLoader geladen und sind deswegen nicht in dem standard Classloader sichtbar. Deswegen war es notwendig, die ObjectInputStream-Klasse

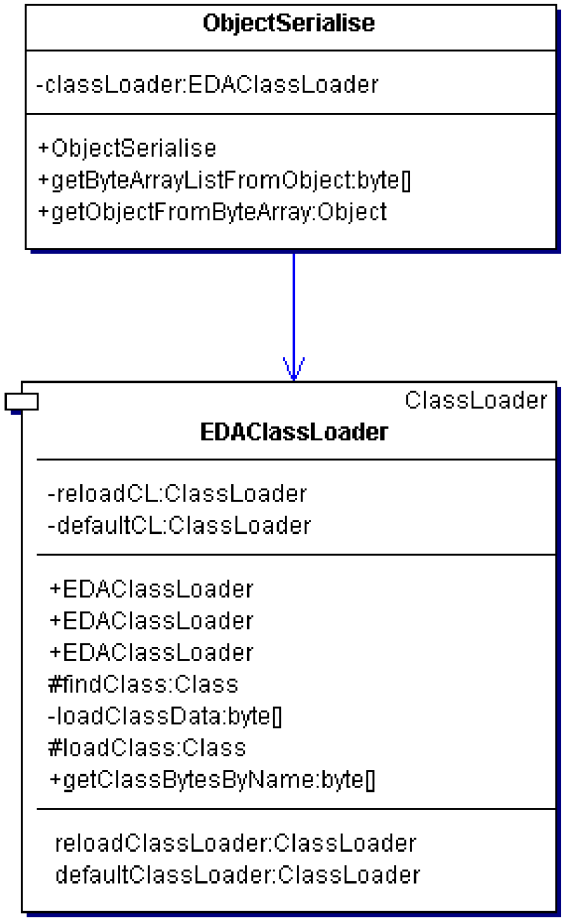

Abbildung 4: ClassLoader

entsprechend zu erweitern. Mit der Klasse EDAObjectInputStream ist die notwendige Erweiterung realisiert. Die genauere Aufbau der Klasse ist im Sourcecode zu finden und wird hier nicht weiter erläutert.

Die oben beschriebenen Klassen bilden den Kern der Laufzeitumgebung des Projektes. Weitere Klassen sind zum Framework einzuordnen. Sie haben die Aufgabe, bestimmte Richtlinien vorzuschreiben und nützliche Funktionalität zur Verfügung zu stellen. Die Abbildung 5 zeigt ein Interface, das jedes mobile Programm in diesem Projekt implementieren muss. Dieses Interface schreibt eine einzige Methode 'enterPoint' vor. Diese Methode bekommt als Parameter ein int-Wert, der besagt, welcher Teil des Programms ausgeführt werden soll. Diese Technik ist vergleichbar mit der 'schwachen Migration mit beliebigem Einstiegspunkt' bei mobilen Agenten. Der Ausschnitt aus dem Beispielprogramm, in der Abbildung 6,

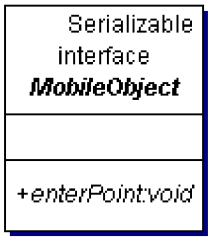

Abbildung 5: MobileObject

soll diese Technik verdeutlichen. Interessant ist auch die Zeile, wo die Transmitter-

```
/* (non-Javadoc)
 * @see de.revout.eda.transfer.MobileObject#doFoo(int)
 */
public void enterPoint(int paramPointer) throws PointerNotExistException{
    final Wurmchen wurmchen = this;
    try {
        switch (paramPointer) {
        case 0:
            showWindow(new WindowAdapter(){
                @Override
                public void windowClosed(WindowEvent paramE) {
                    try {
                        Transmitter.getCurrent().sendData("PC2",TransferObject.createTransferObject(wurmchen,0));
                    } catch (ClassNotFoundException e) {
                        e.printStackTrace();
                    }
                }
            });
            break;
        case 1:
            showWindow(new WindowAdapter(){
                @Override
                public void windowClosed(WindowEvent paramE) {
```

Abbildung 6: Beispiel

Klasse angesprochen wird. Diese Zeile sorgt nun dafür, dass das Objekt zusammen mit dem Code zu dem Rechner PC2 weiter geschickt wird. Wichtig an dieser Stelle ist die Tatsache, dass die Ausführung des Programm damit nicht automatisch beendet wird. Es ist dem Entwickler überlassen, ob die Ausführung nach der Migration sofort endet, wie im Beispiel, oder nicht.

Welcher Rechner unter der Bezeichnung 'PC2' verborgen ist, wird zur Zeit in einer XML-Datei festgelegt. Diese Lösung kann eingesetzt werden, da alle Rechner sich in einem lokalen Netzwerk befinden und bekannt sind. Um dem Entwickler möglichst viel Arbeit abzunehmen, bietet das Framework eine Hilfsklasse an, die das Auslesen der XML-Datei versteckt. Die Klasse ist als Singelton-Pattern realisiert und steht damit überall in der Anwendung zur Verfügung. Die XML-Datei wird bei der Initialisierung der Klasse ausgelesen. Damit werden weitere Zugriffe auf die XML-Datei gespart. Weitere Details zur Klasse sind im Sourcecode des Projektes zu finden.

7 Fazit

Dieses Projekt hat gezeigt, dass die vorgeschlagene Idee tragfähig ist. Bei der Entwicklung wurden viele Aspekte weggelassen, so die Fragen der Sicherheit, oder auch die Redundanz der übertragenen Daten. Entscheidend war es zu prüfen, ob die Technik mobilen Agenten auch bei der Lösung von Problemen solcher Art einsetzbar ist.

Die Schwerpunkte lagen dabei bei der Reimplementierung der Standardklassen der Java. Nachdem diese Schwierigkeiten gelöst waren, war es kein Problem mehr, eine Testanwendung zu schreiben und damit das Projekt ausreichend zu testen. Die Einbindung der Laufzeitumgebung zu einem vorhandenen Projekt hat ohne Probleme funktioniert und es war sofort möglich, die Ressourcen der Anwendung zu nutzen.

Dieses Projekt hat deutlich gezeigt, dass noch nicht alle Möglichkeiten im Bereich der verteilten Anwendungen erprobt sind und es durchaus die Technologien gibt, die den schon erprobten Techniken Paroli bieten können.

Literatur

[1] http://www11.informatik.tu-muenchen.de/lehre/lectures/ss1994/va/chap_1/verte.html

[2] http://de.wikipedia.org/wiki/Verteilte_Systeme

[3] NET.ObjectDAYS2000 Tutorial 2 :Von verteilten Objekten zu mobilen Agenten

[4] http://www.cs.berkeley.edu/projects/sprite/sprite.html

[5] http://lampwww.epfl.ch/~zenger/papers/partyd.pdf

[6] http://www.omg.org

[7] http://www.fipa.org

[8] http://www.omg.org/docs/orbos/98-03-09.pdf

[9] http://www.fipa.org/specs/fipa00001

[10] http://www.informatik.hu-berlin.de/Institut/struktur/systemanalyse/diplom/dorn04.pdf

[11] J. Bradshaw (Ed.); Software Agents; The MIT Press, 1997

[12] W. Brenner, R. Zarnekow, H. Wittig; Intelligente Softwareagenten - Grundlagen und Anwendungen; Springer-Verlag

[13] http://www.cordis.lu/infowin/acts/analysys/products/thematic/agents/ch4/ch4.htm

[14] http://www.trl.ibm.com/aglets/

[15] http://www.recursionsw.com/

[16] http://ki.informatik.uni-wuerzburg.de/teach/ss-004/vki/Agentenkommunikation_4up.pdf

[17] http://www.rn.inf.tu-dresden.de/scripts_lsrn/lehre/komplex/RMI-Versuch/print/EinfV-V10-03.pdf

[18] http://www.zdnet.de/news/tkomm/0,39023151,39134180,00.htm

[19] http://agent.cs.dartmouth.edu

[20] http://www.fh-wedel.de/~si/seminare/ss04/Ausarbeitung/4.Rohr/MobileAgentenMatthiasRohr.pdf