Fachhochschule Bingen
Fachbereich Elektrotechnik und
Informatik

Fraunhofer Institut für
Graphische Datenverarbeitung

Diplomarbeit

Koordinierte, verteilte Informationssuche auf Basis mobiler Agenten und Peer-to-Peer Technologie

von

Andreas Reuter
Matrikelnummer 189460

Fachhochschule Bingen
Fachbereich Elektrotechnik und Informatik
Fachgebiet Angewandte Informatik
Fachrichtung Kommunikation und Medien
Berlinstraße 109
55411 Bingen

Betreuer: Dipl.-Ing.(FH) Ulrich Pinsdorf

Prüfer: Prof. Dr.-Ing. Peter Rausch

Aufgabenstellung für die Diplomarbeit von
Herr Andreas Reuter
Fachhochschule Bingen
Matrikel-Nr. 189460

Thema: “Koordinierte, verteilte Informationssuche auf Basis mobiler Agenten und Peer-to-Peer Technologie”

Das angestrebte Ziel der Arbeit ist es, Informationen in Computernetzen zu finden und diese dem suchenden Anwender zur Verfügung zu stellen. Das Netzwerk wird durch Computer realisiert, die als Middleware eine Plattform für mobile Agenten betreiben. In dem so gebildeten Netz besitzen die Teilnehmer die Charakteristiken von Peers in Peer-to-Peer Netzen.

Um nach Informationen zu suchen, migrieren in Peer-to-Peer Netzen mobile Agenten zu den einzelnen Peers. An einem Suchauftrag können sich koordiniert mehrere Agenten beteiligen. Des Weiteren sollen die Agenten während der Suche auftragsbezogene Informationen untereinander über eine Kommunikationsschnittstelle austauschen.

Der Fokus der Suche ist dahingehend variant, dass beliebige Informationsquellen auf beliebige Gesichtspunkte hin untersucht werden können. So ist es möglich nach Dokumenten, Bildern, Wasserzeichen, Tonfrequenzen u.a. zu suchen. Dies ermöglicht eine generische Realisierung der Applikation, die es erlaubt, verschiedene Suchalgorithmen zu nutzen. Der für einen Suchauftrag benötigte Algorithmus wird den Agenten mitgegeben und beim Peer lokal zur Informationssuche genutzt. Die Basis der Informationssuche sind Daten des Peers, die der Anwender bereitgestellt hat. Das Bereitstellen sollte für den Anwender sowohl einfach, wie auch kontrollierbar sein.

Die Ergebnisse einer Suche werden dem Anwender so präsentiert, dass dieser aus dem Gesamtergebnis die Daten entsprechend seinem Interesse auswählen kann. Die so angeforderten Daten werden dann von anderen Agenten aus dem Peer-to-Peer Netzwerk beschafft.

Der Anwender soll die Möglichkeit haben, mehrere Suchaufträge parallel zu initiieren.

Darmstadt, den 29.04.2002

Betreuer:

Dipl.-Ing.(FH) Ulrich Pinsdorf

Prof. Dr.-Ing. Peter Rausch

Abstract

Theme: “Koordinierte, verteilte Informationssuche auf Basis mobiler Agenten und Peer-To-Peer Technologie”

The aim of this diploma thesis is to find information in computer-networks and to make these available for the user. The network is based on computer, which use the same agent-plattform as middleware. The members of this network have properties and behaviour like peers in a peer-to-peer network.

In this peer-to-peer network mobile agents migrate to each peer to search specific information. The application is able to let one or more agents work on the same search job. A group of agents has to work together cooperatively, therefore they have to be coordinated by a supervisor. The cooperative work of agents is based on a communication interface, which they use to exchange messages.

The application has no focus on a specific kind of data, but it may deal with almost any datatype. Therefore the application has to be investigate information sources from different point of views, e.g. it has to be able to look for documents, images or watermarks. Therefore the application uses different search engines, which are carried along by the agent. He has to use his specific search engine on each peer, to find wanted information. The information sources on each peer are data-files. Each user can state which files he wants to provide for investigation. Furthermore agents shall not be allowed to perform somewhat harmful actions on a peer. These configurations of the application have to be handy for the user.

All files, which were found during a search job are displayed to the user in order to select certain files he is interested in. These files are being fetched by agents from the peer-to-peer network.

The application should allow the user to initiate parallel search jobs.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe.

Andreas Reuter Walluf, Oktober 2002

Danksagung

An dieser Stelle möchte ich mich bei den Dozenten der Fachhochschule Bingen für ihre fundierte Wissensvermittlung bedanken. Insbesondere gilt es hier Herrn Prof. Dr.-Ing. Peter Rausch zu nennen, der mir in diversen Praktika zu Software Engineering und objektorientierter Systementwicklung neben fachlichen Ratschlägen auch gute Empfehlungen zur beruflichen Bildung und Wahl des Diplomthemas gegeben hat.

Für seine fachliche Kompetenz und konstruktive Kritik, mit der mich Herr Dipl.-Ing.(FH) Ulrich Pinsdorf während der gesamten Diplomarbeit begleitet hat, möchte ich mich bei ihm besonders bedanken. Seine Betreuung, Erfahrung und sein Interesse an der Arbeit, haben viel zum Gelingen der Diplomarbeit beigetragen.

Auch bei seinem Kollegen, Herr Dipl.-Inf. Jan Peters, möchte ich mich für die gute Zusammenarbeit bedanken. Mit Interesse und Kompetenz war er auch immer ein guter Ansprechpartner, um Probleme zu erörtern und zu lösen.

Andreas Reuter Walluf, Oktober 2002

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einführung

1.1 Aktuelle Situation

„Man ertrinkt in Informationen, aber dürstet nach Wissen.“
— John Naisbitt

Dieses Zitat charakterisiert deutlich die Situation, in der sich jeder befindet, der heutzutage in Computernetzen nach Informationen sucht. Bei der Suche, selbst nach spezifischen Informationen zu einem bestimmten Thema, sieht sich der Suchende einer Informationsflut ausgesetzt, in der es zeitaufwendig und mühsam ist, die gewünschten Informationen zu finden. Denn das Ergebnis zu einer Suchanfrage setzt sich häufig aus vielen unrelevanten Informationen und einem geringen Teil relevanter Informationen zusammen. Daneben treten zwei Aspekte immer mehr in den Vordergrund. Zum einen befinden sich im gegenwärtigen Internet Informationen nicht mehr nur „im Zentrum“ des Internets, sondern zunehmend auch an dessen „Rändern“ - also bei den Anwendern. Des Weiteren haben die meisten bekannten Suchmaschinen, wie Excite, Google, Lycos oder Yahoo!, gegenwärtig nur einen Teil der bekannten Internetseiten indiziert und auch nur „geringen“ Zugriff auf das *Deep Web*[1] [23]. Vorhandene Informationen sind demach weit verstreut, nicht zwangsläufig allgemein zugänglich und nicht umfassend den Suchmaschinen bekannt.
Als Interessent von bestimmten Informationen ist es wünschenswert, dass die Menge der gefundenen Daten zu einer Suchanfrage möglichst umfassend ist und keine Daten beinhaltet, die thematisch außerhalb der Anfrage stehen.
Um diesem Anspruch gerecht zu werden, müssen zwei Bedingungen erfüllt sein. Zum einen muss der Informationspool möglichst groß sein und zum anderen muss mit geeigneten Verfahren in diesem nach Informationen gesucht werden, um ein möglichst genaues Suchergebnis zu erhalten.

1.2 Lösungsansatz

Ein potentieller Ansatz einen großen Informationspool zu schaffen, ist es ein System einzusetzen, dass auf die Informationsquellen der Anwender zurückgreift. Eine Technologie, die dies

[1] Als *DeepWeb* werden folgende Datenquellen bezeichnet: passwortgeschützte Datenbanken, unkonventionelle Datenformate, dynamische Internetseiten u.a. Einige Suchmaschinen haben sich auf die Erschließung von Teile des DeepWebs spezialisiert [77], [30], [31].

erlaubt, sind so genannte Peer-to-Peer Systeme. Dadurch sind als Datengrundlage für Suchanfragen die Datensammlungen der Anwender verfügbar. Diese Sammlungen stellen i.d.R. weniger eine breitgefächerte, als vielmehr eine umfassende Sammlung an Daten zu bestimmten Interessensgebieten dar. Diese thematischen Datensammlungen können gezielt auf die gewünschte Information hin untersucht werden. Mit spezialisierten Suchmaschinen kann erreicht werden, dass das Suchergebnis die geforderten Qualitäten besitzt.
Dieser Ansatz um zu einer besseren Informationsgewinnung zu gelangen, stellt das Ziel der Applikation dar, die im Rahmen dieser Diplomarbeit entwickelt wurde.
Die Grundlage der Applikation ist die Software *SeMoA* [73]. Diese wurde vom Fraunhofer Institut für Graphische Datenverarbeitung entwickelt und stellt ein agenten-basiertes Betriebssystem dar. Computer, die SeMoA nutzen, sind direkt miteinander verbunden und bilden ein Computernetzwerk aus. Dieses weist Charakteristiken eines Peer-to-Peer Netzwerks auf, was im Verlauf der Arbeit noch gezeigt wird. Die Applikation meiner Diplomarbeit soll ermöglichen, in SeMoA-Netzwerken bei den Netzwerkteilnehmern nach beliebigen Informationen zu suchen. Entsprechend der verwendeten Basistechnologie und der Ausprägung des Computernetzwerks, nutzt diese Applikation mobile Agenten zur Informationsbeschaffung und zur direkten Kommunikation mit anderen Peers. Somit wurde auf eine Client/Server Architektur bei der Realisierung der Applikation bewusst verzichtet. Der Einsatz von mobilen Agenten für den Lösungsansatz ist sinnvoll, da diese einen modularen Aufbau besitzen. Dies unterstützt den generischen Lösungsansatz der Appliketion damit diese verschiedene Suchmaschinen einsetzen kann. Daneben kann ein mobiler Agent, aufgrund seines adaptiven Verhaltens auf verschiedene Situationen in Peer-to-Peer Netzen entsprechend reagieren [10], wodurch dem dynamischen Verhalten von Peer-to-Peer Netzen Rechnung getragen werden kann.
Entsprechend dem skizzierten Lösungsansatz soll die Applikation folgende Leistungsmerkmale aufweisen, um eine Informationssuche mittels mobiler Agenten zu ermöglichen:

- Die Suche darf prinzipiell nicht auf bestimmte Medientypen beschränkt sein. Jedoch kann ein Anwender, der seine Dateien zur Verfügung stellt reglementieren, welche Medientypen explizit für Suchanfragen freigegeben sind.
- Die Suchanfragen werden von Agenten ausgeführt, die von Peer zu Peer migrieren.
- Zum Suchen besitzen die Agenten eigene Suchmaschinen, die sie mitbringen und am Peer vor Ort einsetzen, wenn sie vom Peer hierfür die entsprechende Erlaubnis erhalten.
- Um eine Suchanfrage während der Bearbeitung weiter zu präzisieren, sollen Agenten zusätzliche Informationen während ihrer Auftragsbearbeitung sammeln und auswerten.
- An einem Suchauftrag können sich mehrere Agenten kooperativ beteiligen.
- Während der Auftragsbearbeitung sollen beteiligte Agenten miteinander kommunizieren. Der Inhalt der Kommunikation stellen die gesammelten Informationen dar, die während der Suche ermittelt wurden. Dadurch lässt sich eine Suche zur Laufzeit zwischen verschiedenen beteiligten Agenten koordinieren.
- Gefundene Dateien werden dem Auftraggeber präsentiert, so dass dieser entscheiden kann von welchen Dateien er ein Duplikat haben möchte.
- Ausgewählte Dateien werden von speziellen Agenten im Peer-to-Peer Netz beschafft.
- Es sind Maßnahmen zu treffen, damit keine unbefugten Agenten die Suche und die gewonnenen Ergebnisse manipulieren können.

- Das Peer-to-Peer Netz soll fehlertolerant sein gegenüber dem Ausfall von Peers oder dem „sterben“ von Agenten.
- Zusätzlich zu diesen Eigenschaften soll die Applikation auch die Abwicklung paralleler Suchaufträge ermöglichen.

Um das Migrationsverhalten der Agenten sowohl bei Such- wie auch bei Dateibeschaffungsaufträgen besser verstehen zu können, soll folgendes Beispiel dienen, was mit den Abbildungen 1.1 und 1.2 verdeutlicht werden soll.

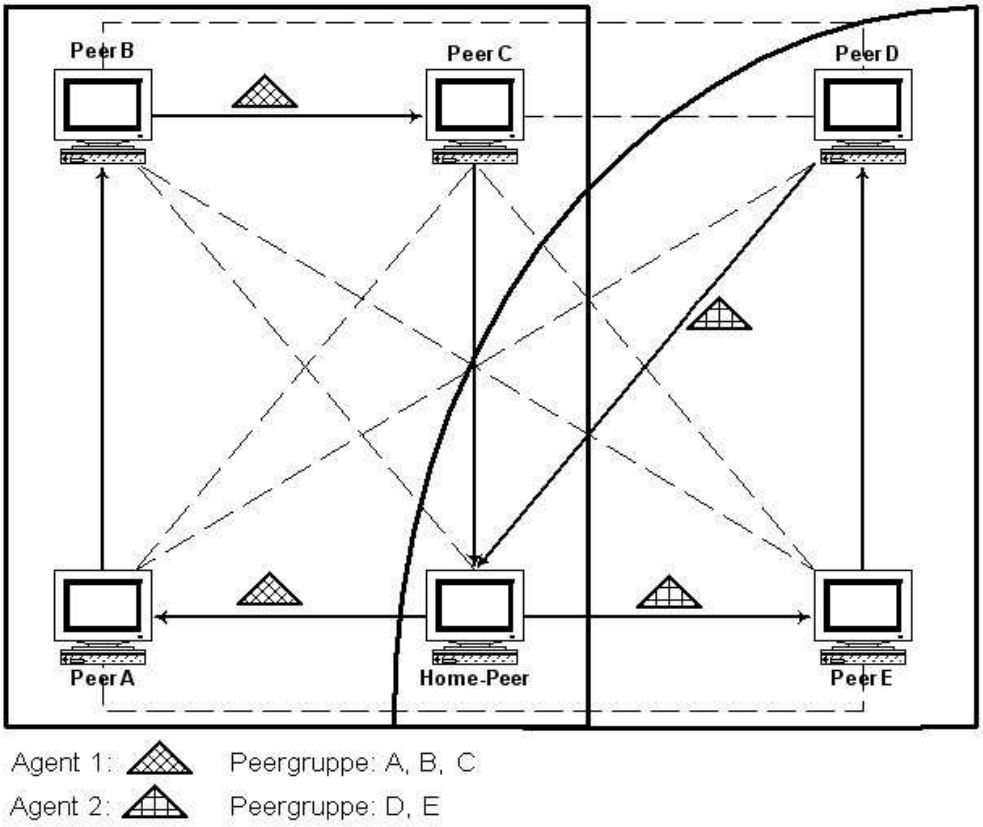

Abbildung 1.1: Suchen im Netzwerk

Es existiert ein Peer-to-Peer Netzwerk mit sechs Teilnehmern, die alle direkt miteinander verbunden sind. Vom Home-Peer aus wird ein Suchauftrag gestartet. Die Applikation erkennt, dass dass fünf weitere Peers im Netzwek existieren und teilt das Netzwerk in Bereiche ein. Dabei wird davon ausgegangen, dass ein Agent maximal zu drei fremden Peers migrieren soll, so dass zwei Peergruppen gebildet werden. Entsprechend werden insgesamt zwei Agenten mit der Suche beauftragt, die wie skizziert im Netz migrieren und anschließend wieder zum Home-Peer zurückkommen.
Aus dem Ergebnis hat der Anwender insgesamt drei Dateien von den Peers B, C und D ausgewählt und fordert von diesen ein Duplikat an. Die Applikation erhält diesen Auftrag und startet drei Agenten, welche die geforderten Dateien beschaffen sollen. Im Gegensatz zur Informationssuche wird jeder Agent nur einen Peer aufsuchen. Diese Lastverteilung hat mehrere Vorteile.

- Duplikate sind schneller am Home-Peer verfügbar.

- Die Netzbelastung ist ausgewogener.
- Große Dateien können transportiert werden. Da Agenten nur eine begrenzte Aufnahmekapazität für Dateien haben, wird diese Kapazität so nicht durch kleinerer Dateien verbraucht.
- Wenn einem Agenten auf einem Peer plötzlich alle Migrationspfade abgeschnitten sind, können wenigstens einige Duplikate dem Anwender beschafft werden. Würde nur ein Agent eingesetzt werden, bekäme der Anwender kein einziges Duplikat.

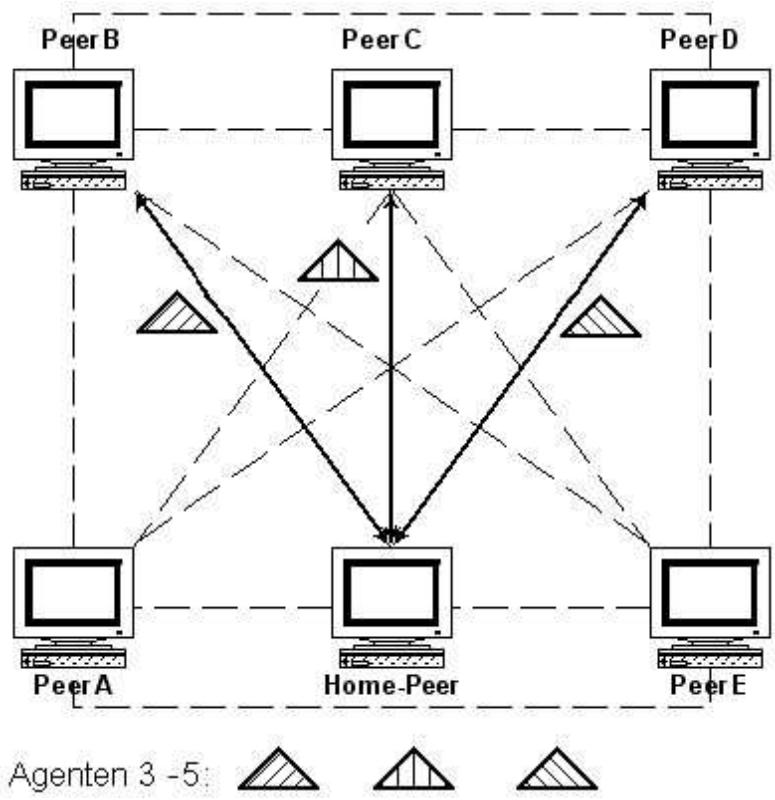

Abbildung 1.2: Dateibeschaffung im Netzwerk

Die Methoden, Konzepte und Werkzeuge des objektorientierten Software Engineering waren Grundlage für die Realisierung der Applikation. Dabei war ein generischer Lösungsansatz stets im Fokus der Entwicklung. Denn zum einen muss sicher gestellt werden, dass weitere Suchmaschinen implementiert und von den Agenten genutzt werden können. Zum anderen muss davon ausgegangen werden, dass das Programm noch weitere Änderungen im Rahmen einer funktionalen Erweiterung oder Anpassung erfahren wird.
Die vorliegende Arbeit beschreibt im Folgenden welche vergleichbaren Lösungen bereits existieren, um mittels mobiler Agenten gezielt nach Informationen zu suchen. Anschließend werden Grundlagen zur Agententechnologie und zur speziellen Suche in Textfiles vorgestellt. Ein Kapitel über die Entwicklung des Systems, das funktional den beschriebenen Lösungsansatz beinhaltet, folgt dem Grundlagen-Kapitel und wird mit einer Bewertung der Lösung abgeschlossen. Anschließend wird eine kurze Benutzereinführung in das Programm und Hinweise für die

Weiterentwicklung des Programms gegeben. Die Arbeit findet ihren Abschluss in einem Ausblick auf Verbesserungsmöglichkeiten und einem Fazit zur Diplomarbeit.

1.3 Verwandte Arbeiten

Um die Diplomarbeit einordnen zu können, sollen in diesem Abschnitt ähnliche Arbeiten vorgestellt werden. Die Betrachtung konzentriert sich dabei auf Peer-to-Peer Systeme zur Informationsbeschaffung und zum Dateienaustausch, die jeweils Agenten einsetzen. Als Basis für die Recherche dienten einige Themenseiten [3], [2], [7] zur Agententechnologie.

Bookmark-Agents. Bereits Mitte der 1990er Jahre wurden Applikationen entworfen, die einen Austausch von Bookmarks zwischen Anwendern eines Netzwerks ermöglichen sollten, um gezielt Informationen zu gewinnen; eine solche Applikationen ist bspw. *SiteSeer*. In ihrem Artikel [45] stellen Mori und Yamada ein Bookmark-Agent System vor, bei dem stationäre Agenten die Interessen ihrer Anwender anhand deren Bookmarks ermittelt. Diese Bookmarks stellen die Basis für Suchanfragen dar und werden entsprechend ausgetauscht. Damit ein Agent eine Suchanfrage korrekt interpretieren kann, bringt er eigenständig jede neue URL mit Schlüsselwörter in Verbindung, die er auf der entsprechenden Seite gefunden hat. Sind mehrere Bookmark-Agent Systeme durch ein Netzwerk miteinander verbunden, können die Agenten eigenständig ihre bekannten Bookmarks anhand der Interessen ihrer Anwender miteinander abgleichen.
Bei einer Suchanfrage eines Anwenders kann dessen Agent nun bei sich nach entsprechenden Bookmarks suchen und dazu auch die benachbarten Agenten konsultieren. Diese können die Anfrage dann weiterleiten oder beantworten. Auf diese Weise erhält der Anwender dann relativ genaue Ergebnisse zu seiner Anfrage, die in ihrer Anzahl deutlich unter denen der Suchmaschinen liegen können. Zu beachten ist, dass die Autoren den Einsatz in kleinen Teilnehmergruppen ($\approx$ 6 Teilnehmer) befürworten und in ihrer Applikation eine Ergänzung zu den Internet-Suchmaschinen sehen.
Dieses System nutzt zwar weder verschiedene Suchstrategien, noch mobile Agenten zum Informationsaustausch. Es beinhaltet aber einen Aspekt, den es auch bei der Applikationsentwicklung zu erörtern galt. Dies ist die Formulierung der Suchanfrage, die während der Auftragsabwicklung von Agenten eigenständig präzisiert werden sollte. Diese Präzisierung sollte auch auf Schlüsselwörter basieren, die den Inhalt eines Dokuments charakterisieren. Die Idee, Schlüsselwörter zu kategorisieren, wurde bei der Applikationentwicklung ebenfalls aufgegriffen, aber zu Gunsten anderer Mechanismen nicht umgesetzt. Im Kapitel 6 wird die Idee aber als Verbesserungsmöglichkeit erörtert.

CARROT. Das agenten-basierte System CARROT[2] [47] wurde in einer Projektgruppe um Nicholas und Cost an der Universität Maryland entwickelt und wird zum Dokumenten-Managemet und Informationsaustausch (Information Retrieval) genutzt. Dafür werden zwei Agententypen eingesetzt. Bei jedem Teilnehmer läuft ein Agent stationär, der die vorhandenen Daten indiziert und die Information als Metadaten an einen anderen Agenten weiterleitet, der als Broker agiert. Prinzipiell kann in diesem Netzwerk jeder Agent die Funktion des Brokers übernehmen. Deshalb kann der Agent eine Suchanfrage als Broker-Agent auch selbst beantworten oder diese im Netz als Backend-Agent weiter kommunizieren [49]. Auf diese Weise können dann mehrere Ergebnisse von mehreren Agenten zu einem Suchergebnis zusammen getragen werden. Die

[2] *CARROT* ist ein Akronym für *C*ollaborativ *A*gent-based *R*outing and *R*etrieval of *T*ext

Kommunikation zwischen den Agenten geschieht mittels KQML [3] [16], [9] und kann auch benutzt werden, um Agenten für die Konfiguration gezielt anzusprechen. In Abbildung 1.3 von Finin und Nicholas [17] wird aufgezeigt, welche Kommunikations- und Zugriffsstruktur ein Agent als Broker hat. Da ein Agent auch die Funktion eines Backend-Agents übernehmen kann, hat dieser die gleiche Struktur, wie der Broker-Agent.

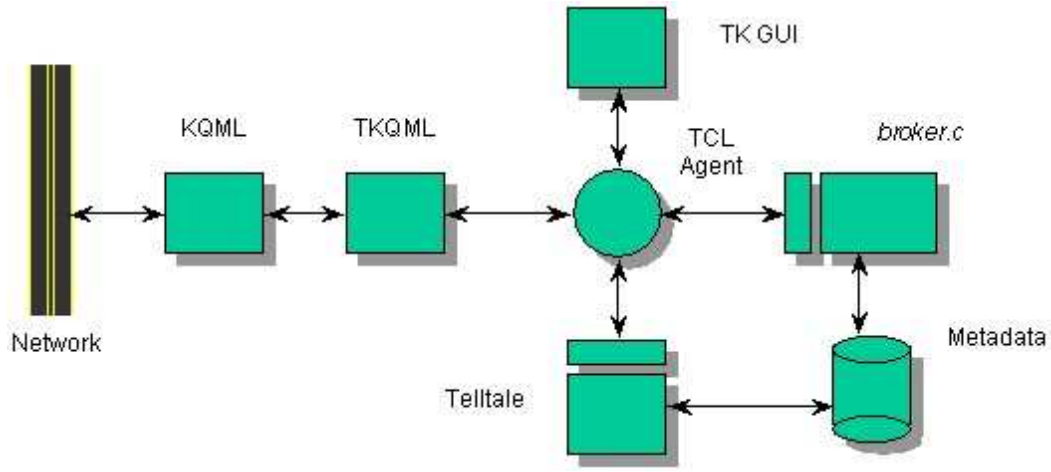

Abbildung 1.3: Struktur eines CARROT Broker-Agenten

Die Besonderheit von CARROT ist, dass es verschiedene Information Retrieval Werkzeuge für die Agenten unterstützt. Somit ist es auch möglich CARROT für andere Inhalte als textliche zu erweitern. Allerdings besitzen immer alle Agenten die gleichen Information Retrieval Werkzeuge, so dass keine Spezialisierung einzelner Agenten entstehen kann [48]. Da seitens der Entwickler erkannt wurde, dass es dem System nicht dienlich ist wenn jeder Agent als Broker auftritt und über alle verfügbaren Informationen Kenntnis hat, wurde die Möglichkeit vorgesehen um Agenten hierarchisch zu gliedern.

Im Gegensatz zur entwickelten Applikation der Diplomarbeit nutzt CARROT keine mobilen, sondern stationäre Agenten. Deshalb ist für dieses System die Kommunikation der Agenten zum Informationsaustausch von Bedeutung. Gemeinsam haben CARROT mit der entwickelten Applikation, dass Agenten, bezogen auf die Informationsverarbeitung, hierarchisch voneinander abhängig sind. So sind bei der entwickelten Applikation der Diplomarbeit die gewonnen Informationen der Agentengruppe *SearchAgent* die Grundlage für die Gruppe *FetchAgent*, damit diese ihre Arbeit aufnehmen können. Daneben können beide Applikationen mit verschiedenen Information Retrieval Systemen arbeiten. Ein neuer Aspekt den CARROT liefert, ist die Repräsentation der Metadaten durch *n-gramme* [4] [17]. Dieses ist für den Lösungsansatz der Applikation der Diplomarbeit interessant, um während der Suche einen Suchauftrag mit Informationen anzureichern.

[3] *KQML* ist ein Akronym für Knowlege Query Manipulation Language

[4] *n-gramme* sind neben Wörtern eine Alternative zur Repräsentation von Dokumenten. n ist hier die Anzahl an Buchstaben in einem Rahmen, welches über einen Text geschoben wird. Die so ermittelten Buchstabenkombinationen sind die Deskriptoren des Textes [39]. Der Einsatz von n-grammen wird bei OCR, multi-lingualen Texten oder non-lingualten Texten, wie bspw. Programmiersprachen, vorgeschlagen.

Knowbot. Knowbot wurde von CNRI [5] entwickelt und ist ein mobiles Agentensystem, um nach Informationen (Metadaten) im Internet suchen. Das Programm kann auch dazu genutzt werden, Änderungen auf Internetseiten festzustellen und diese dann als Information weiterzugeben. Sie nutzen hierfür Python [57] und Inter-Language Unification (ILU). Um eine Umgebung für die Agenten zu etablieren, müssen mehrere Hosts das Knowbot Operating System (KOS) und das Knowbot Programm betreiben; dadurch werden sie zu so genannten Knowbot Service Stations (KSS). Um die Agenten-Programme gegenseitig zu schützen, werden die Programme mit eingeschränkten Rechten auf dem jeweilgen Host betrieben. So besitzen die Programme bspw. keinen direkten Zugriff auf das Dateisystem. Dem Knowbot-Agenten werden anfangs alle seine Rechte auf seinem aktuellen Host entzogen, die er dann bei Bedarf zurück erhält. Um dieses zu kontrollieren wird ein Aufseher (Supervisor), zusammen mit Plugins der Service Stationen, eingesetzt. Abbildung 1.4 [28] zeigt, wie das Knowbot-System den eigenen Rechner vor Knowbot-Agenten abschirmt.

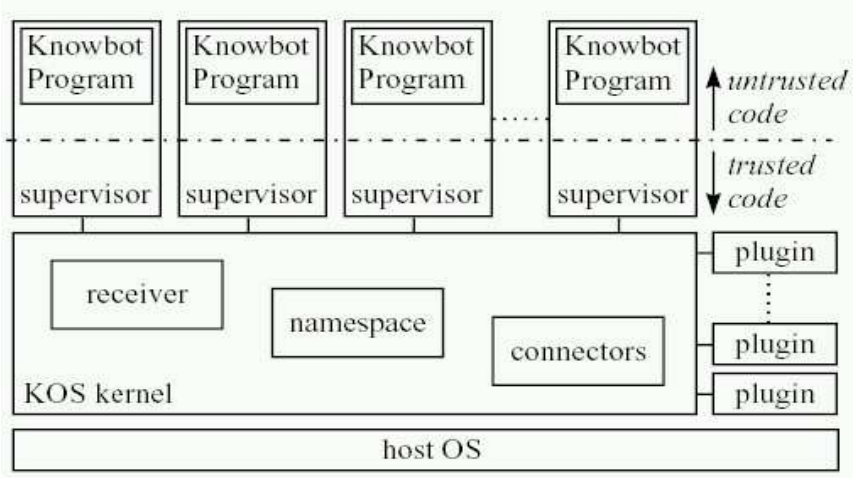

Abbildung 1.4: Struktur einer Knowbot Service Station

Der Aufseher ist auch dafür zuständig den Agenten die Erlaubnis zur Migration zu erteilen. Dafür wird der Agent vom Python Interpreter als Bytecode umgesetzt und kann anschließend serialisiert werden. Da der Python Interpreter über keine Möglichkeit verfügt den Bytecode zu verifizieren, ist dieser nicht vor Manipulationen geschützt.
Alle Knowbot-Agentenplattformen stehen zueinander in einem hierarchischen Verhältnis, das von einem „Worldroot“-Server ausgeht [79]. Diese Hierarchie wird dazu genutzt, um eindeutige Namen den Teilnehmern zu geben. Alle untergeordneten Server besitzen so einen eindeutigen Namen und einen zugewiesenen Bestand an freien Namen, die sie zur Identifikation nachfolgender Teilnehmer nutzen sollen.
Knowbot weist im direkten Vergleich gegenüber SeMoA einige unterschiedliche und gemeinsame Merkmale auf.
Gemeinsamkeiten zwischen SeMoA und Knowbot:

- Um eine Dienstleistung zu erbringen, werden mobile Agenten eingesetzt, die mit stationären Diensten zusammenarbeiten.
- Knowbot und SeMoA sind modulare Betriebssysteme.

[5] *CNRI* ist ein Akronym für *C*orporation for *N*ational *R*esearch *I*nitiatives

- Agenten haben erst nach Überwinden von Sicherheitsbarrieren Zugriff auf den lokalen Rechner.
- Beide Systeme arbeiten mit Rechten und Befugnissen für Agenten, die ihnen zugewiesen und entzogen werden können.

Unterschiede zwischen den Ansätzen sind:

- Knowbot nutzt eine Client/Server Architektur und kein Peer-to-Peer Netzwerk.
- Knowbot-Agenten haben die Möglichkeit sich zu klonen, anstatt zu migrieren.
- Der Python-Interpreter kann den Bytecode der Kowbot-Agenten nicht verifizieren; bei SeMoA ist dies gegeben.
- Während SeMoA in Java implementiert wurde, nutzt Knowbot Python.

Anthill. Die Univesität von Bologna hat ein Framework names *ANTHILL* [44] entwickelt. Dieses ist in Java codiert und dient zur Entwicklung agenten-basierter Peer-to-Peer Systeme. Dabei hat es nicht eine spezielle Applikation im Fokus, sondern soll gleichermaßen für Datei-, CPU- und Speicher-Sharing -Applikationen genutzt werden können. Peer-to-Peer Netze werden von Anthill nicht als technische Gebilde, sondern als biologische Organismen aufgefasst, in denen einzelne Peers (Ameisen-) Nester repräsentieren. Wie auch bei Peer-to-Peer Netzen, hat jedes Nest eine individuelle Identifikationsnummer. Des Weiteren besitzt jedes Nest einen *Ant Manager*, ein *Gateway* und einen Informaionspool, bestehend aus lokal vorhandenen Dokumenten. Ameisen wandern mittels der Gateways zwischen diesen Nestern und können dort ihre Aufträge abwickeln. Ameisen haben demnach die Eigenschaften Mobilität und Proaktivität. Ihr

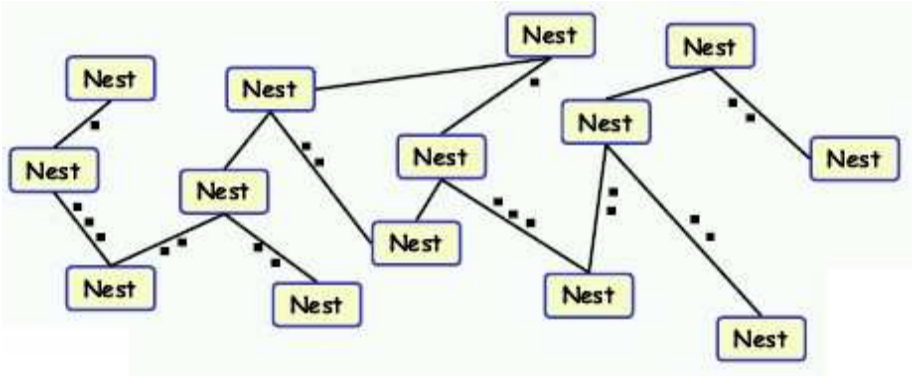

Abbildung 1.5: Modell von Anthill

eigentliches Verhalten wird aber durch den Algorithmus bestimmt, den jede Ameise mit sich führt. Wie aus dem Bild 1.5 [43] hervorgeht, ist das Netzwerk dezentral und nicht hierarchisch aufgebaut. Aus diesem Grund sind Ameisen in der Lage, sich während der Suche nach Informationen zu replizieren, was Abbildung 1.6 veranschaulicht. Auf diese Weise können mehrere Nester gleichzeitig untersucht werden, bis entsprechende Ergebnisse gefunden wurden.
Um dem Sicherheitsaspekt Rechnung zu tragen, existiert in jedem Nest ein Ant Manager. Dieser signiert den Code der Ameisen und verwehrt ihnen den Zugriff auf I/O-Kanäle, das Dateisystem und andere Netze. Daneben ist zusätzlich eingeschränkt, welche Operationen eine Ameise in

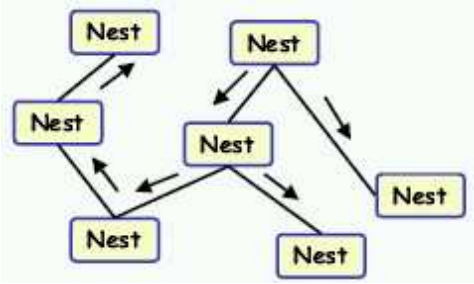

Abbildung 1.6: Suchpfad der Anthill-Ameisen

einem Nest ausführen darf. Verfügt eine Ameise über alle Rechte, ist sie in der Lage, neben der Migration das Nest auch über die Existenz anderer Nester zu informieren, Suchanfragen zu formulieren und eine Pheromonspur vom aktuellen Nest zu anderen Nestern legen. Der Informationsinhalt der Pheromone ist dabei nicht definiert, könnte aber bspw. den vorhandenen Umfang einer Datenquelle oder die Qualität bezüglich einer Suchanfrage beinhalten. Da diese Pheromone genau wie die natürlichen während einer Zeitspanne ihre Intensität verändern, wenn viele oder wenige Ameisen ihrer Spur folgen, kann so eine dynamische Pheromonkarte des Netzwerks erstellt werden. Auf dieser Basis kann dann eine Prognose über gute und schlechte Nester bezüglich einer Anfrage erstellt und entsprechend genutzt werden. Da Pheromonkarten periodisch aktualisiert werden müssen, ist mit ihnen ein hoher Verwaltungssaufwand notwendig, da sich mit steigender Anzahl von Suchanfragen auch die Anzahl der dazugehörigen Karten erhöht. Deshalb ist es sinnvoller Pheromonkarten nicht bezüglich einer Suchanfrage zu erstellen, sondern Pheromonkarten bezüglich einer Themengruppe anzulegen. Diese kann aus der Suchanfrage entsprechend ermittelt oder vom Anwender angegeben werden.
Neben den beschriebenen Eigenschaften von Anthill, besitzt dieses noch ein *Evaluation Environment*. Dieses überprüft die Qualität des Algorithmus, den eine Ameise besitzt. Um dies zu bewerten, werden im Evaluation Environment das Verhalten der Ameise simuliert und bewertet. Dies bezeichnet Montresor [41] als Szenario. Szenarien werden zufällig generiert und durch einen Satz von Parametern (*Chromosome*) beschrieben. Damit ist auch die Möglichkeit geschaffen, Ameisen mit Hilfe genetischer Algorithmen [35] evolutionär zu optimieren.
Als Ausführungsumgebung nutzt Anthill JXTA, wobei Anthill als JXTA-Service implementiert ist. Welche Eigenschaften die Nutzung von JXTA bietet, ist im folgenden Kapitel der Diplomarbeit nachzulesen.

1.4 Fazit

Die hier vorgestellten verwandten Arbeiten zeigen ähnliche Ansätze auf, wie die entwickelnde Applikation dieser Diplomarbeit. Dabei ist aber kein Ansatz dabei, der die Funktionalität dieser Arbeit komplett abbildet. So arbeiten der Bookmark-Agent und CARROT nur mit stationären Agenten und tragen ihre Informationen ausschließlich über Kommunikation zusammen, während Knowbot und Anthill auch mit mobilen Agenten arbeiten. Gemeinsam ist bei allen, dass sie zur Informationsbeschaffung dezentral organisiert [6] sind. Daneben versuchen alle Systeme, die mobile Agenten nutzen, dem Sicherheitsaspekt durch geeignete Strategien Rechnung zu tragen. So wird bei Knowbot und Anthill der Zugriff auf das Dateisystem nur durch Zwischeninstanzen

[6] CARROT nutzt die Hierarchie zur Namensvergabe und welche Rolle ein Agent zu übernehmen hat.

ermöglicht. Doch keines der Systeme bietet zur Zeit Sicherheitsmerkmale wie Verifikation des Agenten-Bytecodes und sichere Kommunikations- und Migrationswege für Agenten an.
Weiterhin beinhalten die verwandten Arbeiten viele Ansätze, die geeignet sind die Leistungsfähigkeit der im Rahmen dieser Diplomarbeit entwickelnden Applikation zu steigern. So könnte der Einsatz von genetischen Algorithmen und Pheromonen beispielsweise zur Optimierung des reaktiven Migrationsverhaltens von mobilen Agenten genutzt werden.

Kapitel 2

Grundlagen

2.1 Grundlagen

Dieses Kapitel stellt die Grundlagen der Technologien dar, auf die sich der vorgestellte Ansatz stützt. Zu Beginn wird die Agententechnologie vorgestellt, welche das Fundament der Arbeit darstellt. Denn durch Nutzung dieser Technologie kann diese Applikation Eigenschaften wie parallele Suchanfragen, verteilte Suchanfragen und Anwendung beliebiger Suchmaschinen nutzen. Parallel zur Agententechnologie wurde in den vergangenen Jahren eine Technologie names JXTA [34] entwickelt. Beide Technologien scheinen für Applikationen in Peer-to-Peer Netzen gleichermaßen geeignet zu sein, so dass ein Vergleich beider dargestellt wird.
Neben der Betrachtung von Middleware sollen in diesem Kapitel auch Grundlagen zu Such- und Bewertungsstrategien in Textdateien vermittelt werden. Dies geschieht vor dem Hintergrund, dass die Applikation die Textsuchmaschine Lucene [32] des Jarkata Projekts nutzt. Abgeschlossen wird das Kapitel mit einer Einführung in die Technik, wie Schlüsselwörter extrahiert werden.

2.2 Agententechnologie

2.2.1 Mobile Agenten

Agenten wirken als autonome Softwareeinheiten in Agentensystemen mit der Aufgabe, eine Dienstleistung für den Anwender zu erbringen. Dieser formuliert einen eindeutigen Auftrag und deligiert ihn an den Agenten. Bei der Erbringung seiner Dienstleistung verfolgt der Agent diesen Auftrag und vertritt die Interessen seines Auftraggebers. Welche Merkmale ein Agent hierfür benötigt, wird durch seine Definition deutlich. Auch wenn das Gebiet der Agententechnologie noch relativ jung ist, existieren zahlreiche Definitionen eines Agenten [20], [22]. Die Eigenschaften, die Wooldridge und Jennings [83] für Agenten definiert haben, bilden eine allgemeine Grundlage für das Verständnis eines Software-Agenten. Demnach hat ein Agent folgende Merkmale:

Autonom Agenten operieren ohne direkten Einfluss von Außen durch einen Anwender und haben bestimmte Möglichkeiten ihr Verhalten selbstständig zu steuern.

Sozial Agenten kommunizieren mit anderen Agenten und Anwendern über eine Kommunikationsschnittstelle in einer bestimmten Agenten-Kommunikationssprache.

Reaktiv Agenten nehmen die Umwelt [1], in der sie sich befinden, bewusst wahr und reagieren auf diese, wenn sie sich verändert.

Proaktiv Agenten reagieren nicht nur auf Umwelteinflüsse, sondern sind in der Lage aus eigener Initiative zielorientiert zu handeln.

Mobil Mobile Agenten können zur Laufzeit ihren Ausführungsort ändern.

Bezogen auf das Mobilitätsmerkmal ist das soziale Verhalten für mobile Agenten nützlich. So können sie über die Kommunikation mit anderen Agenten zusätzlich Informationen austauschen, was besonders in einer dynamischen Umgebung wichtig ist. Des Weiteren sind sie in der Lage, komplexe Aufgaben zu lösen und entsprechende Informationen untereinander zu kommunizieren. Genauso benötigt ein mobiler Agent reaktive Eigenschaften, um zu gewährleisten, dass er während der Mirgation im Netz immer angemessen auf Änderungen in seiner Umwelt reagieren kann. Denn in einem dynamischen Netz können Migrationspfade blockiert werden, sich Hosts vom Netzwerk abmelden oder sich neue Migrationspfade zu einem Host ergeben.
Zur Veranschaulischung der Leistungsfähigkeit von mobilen Agenten dienen die folgenden beiden Beispiele:

Shopping-Agent: Wenn ein Anwender einen Preisvergleich im Internet durchführen möchte, kann er einen Agenten beauftragen nach Preisen für ein bestimmtes Produkt in einer gewissen Währung zu suchen, wobei der Preis innerhalb bestimmter Grenzwerte liegen muss. Nach der Auftragsvergabe an den Agenten, kann der Anwender andere Aufgaben erledigen, während der Agent selbstständig im Netz migriert und in Datenbanken nach dem Produkt unter den definierten Bedingungen des Anwenders sucht. Je nach Realisierung dieser Applikation könnten diese Aufgabe sowohl mehrere Agenten parallel, wie auch ein einzelner Agent abwicklen. Mit den gefundenen Preisen migriert der Agent dann zum Anwender zurück und präsentiert ihm das Ergebnis. Anschließend kann der Anwender die Bestellung bei dem Anbieter seiner Wahl vornehmen. Wenn entsprechende Rahmenbedingungen für Sicherheitsaspekte beim Datentransport, Schutz der privaten Informationen, Rechtssicherheit bei der Produktbestellung, etc. gegeben sind, ist auch vorstellbar, dass der Agent gar nicht mehr die Ergebnisse dem Anwender präsentiert, sondern die Bestellung eigenständig abwickelt. Dies setzt allerdings ein hohes Maß an Vertrauen voraus, dass dem Agenten von Seiten des Anwenders entgegen gebracht werden muss.

Raumreservierungs-Agent: Unter der Voraussetzung, dass die Mitarbeiter des Unternehmens ihre Terminplanung online pflegen und dass Informationen über Konferenzräume (Belegungsplan, max. Anzahl der Teilnehmer, Raumausstattung) online zur Verfügung stehen, könnte folgendes Szenario mit diesem Agenten realisiert werden. Ein Mitarbeiter, der eine Konferenz ansetzen möchte, teilt dem Agenten mit, an welchen Terminen diese stattfinden könnte, wie lange diese voraussichtlich dauern wird, welche Personen teilnehmen sollen und über welche Ausstattung der Raum verfügen muss. Ausgestattet mit diesen Informationen kann der Agent dann bei den entsprechenden Mitarbeitern das passende Zeitfenster zu den festgelegten Daten reservieren. Anschließend muss er prüfen, welche Räume den Erfordernissen entsprechen, und einen passenden Raum reservieren. Abschließend muss der Agent noch bei allen Konferenz-Mitgliedern einen entsprechenden Eintrag in ihren Terminplänen vornehmen. Wenn der Anwender, der die Konferenz einberufen möchte,

[1] Als *Umwelt* fassen Wooldridge und Jennings neben Computernetzen und Internet, auch die physikalische Umwelt auf.

selbst den Raum auswählen möchte, muss der Agent (als Zwischenschritt) wieder zum Mitarbeiter zurückkehren und ihm die Ergebnisse anzeigen, so dass dieser dann seine Entscheidung treffen kann.

Daneben gibt es bereits Applikationen mit Agenten in den verschiedensten Bereichen, wie industrielle Prozesskontrolle, Luftfahrtkontrolle, Informationsmanagement, medizinische Diagnostik oder Computerspiele [84].

2.2.2 Die Agentenplattform SeMoA

Die Agenten, die die Agentenplattform SeMoA einsetzen, basieren auf der obigen Agentenbeschreibung [54]. *SeMoA* [73] ist ein Akronym für *Secure Mobile Agents*, wodurch zum Ausdruck gebracht wird, dass der Fokus der Plattform auf dem Sicherheitsaspekt liegt. Diesem wird bei SeMoA in verschiedener Art und Weise Rechnung getragen. SeMoA und die dazugehörigen Agenten sind in Java geschrieben und nutzen auch dessen Sicherheitsfunktionalität. Darüberhinaus besitzt jeder Agent zur eindeutigen Identifikation einen individuellen Namen. Der Code des Agenten wird in einen statischen und einen dynamischen Teil aufgeteilt und getrennt signiert [64].
Die Menge an Daten und Code, die zur Entstehungszeit bekannt sind, bilden den statischen Teil des Agenten. Dieser wird mit dem digitalen Schlüssel des Besitzers signiert. So kann überprüft werden, ob dieser Agent manipuliert wurde. Daten, die der Agent zur Laufzeit sammelt, sowie der serialisierte Zustand des Agenten bilden den veränderten Teil. Dieser wird vor der Migration vom letzten ausführenden Server signiert. Abbildung 2.1 zeigt schematisch den Aufbau eines Agenten. Daneben werden mit der Abbildung noch weitere Sicherheitsmechanismen be-

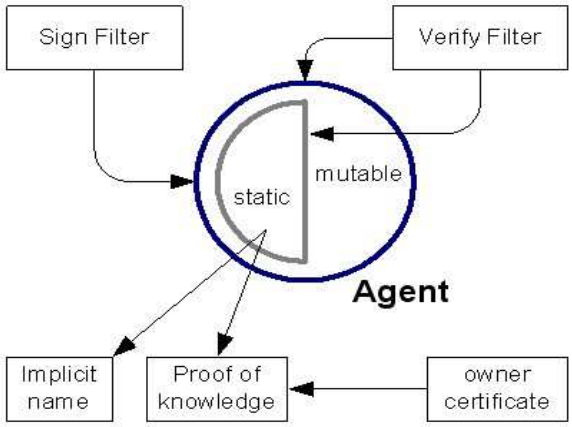

Abbildung 2.1: Aufbau eines SeMoA-Agenten

kannt. Der Inhalt des Agenten wird von den Agentenplattformen mittels zweier Filter überprüft, bevor er in einem Sicherheitsbereich seinem Auftrag nachgehen kann. Die Filter prüfen zum einen anhand der Zertifikate, ob der Code manipuliert wurde und zum anderen, ob der Code Sequenzen enthält, die Schaden auf dem lokalen Rechner verursachen oder ihn in seiner Leistungsfähigkeit blockieren könnten. Fällt eine Überprüfung des Agenten negativ aus, wird ihm der Zutritt verweigert und dieser aufgefordert die Plattform wieder zu verlassen. Ein anderes Sicherheitsmerkmal bei SeMoA ist, dass es Agenten nicht möglich ist Klassen untereinander zu tauschen oder bereitzustellen. Damit soll unterbunden werden das Agenten als Trojanische

Pferde missbraucht werden [64]. Ein SeMoA-Agentensystem besteht aus mehreren Agentenplattformen. Diese sind in ihrem Verhalten und in ihrem Serviceangebot voneinander unabhängig. Mobile Agenten migrieren zwischen einzelnen Plattformen und können sich so durch das Netzwerk bewegen. Dabei sind sie in der Lage von ihrem lokalen Rechner aus jeden anderen direkt zu erreichen. Abbildung 2.2 zeigt ein solches Agentennetzwerk.

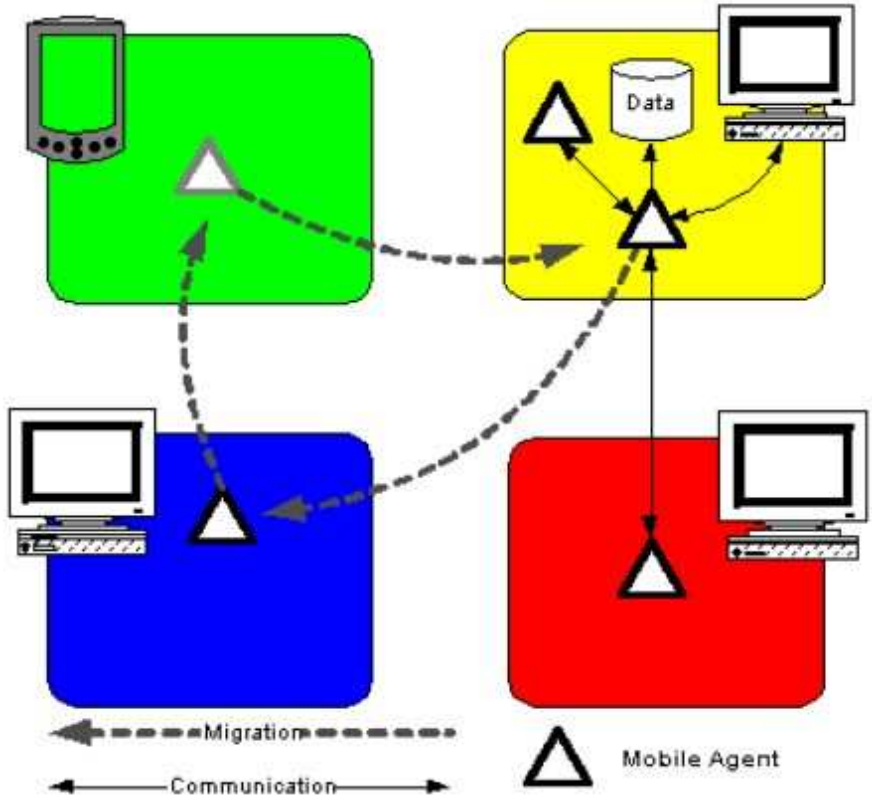

Abbildung 2.2: Beispiel eines SeMoA-Netzwerks

Dieses besitzt neben mehreren Rechnern auch mehrere Agenten. Auf diesen Plattformen können Agenten ihren Aufträgen nachgehen. Voraussetzung für ihre Handlungsfreiheit auf einem Rechner ist das passieren der Sicherheitsbarrieren. In Abbildung 2.3 wird die Sicherheitsarchitektur von SeMoA gezeigt. Dabei wird deutlich, dass neben Schutzmechanismen zwischen Agent und Plattform, auch für einen Schutz zwischen den Agenten gesorgt wird. Damit eine Agentenplattform Agenten von anderen Plattformen überprüfen kann, sind alle Agentenplattformen mit dem zertifizierten digitalen Schlüssel [68] ihrer Bereiber signiert. Dadurch sind Agentenplattformen in einer *PKI*-Hierarchie [2] eingeordnet. Zusätzlich zu diesen Sicherheitsmechanismen ist der Betreiber einer Agentenplattform in der Lage, einem Agenten, in Abhängigkeit seines Besitzers individuelle Rechte zuzuteilen oder zu verwehren. So kann bspw. gesteuert werden welche Agenten mit welchen Rechten auf das Dateisystem zugreifen dürfen und welche nicht. Damit Agenten auch die Möglichkeit haben auf den Agentenplattformen zu wirken, verfolgt SeMoA ein service-orientiertes Konzept [55]. In Abbildung 2.4 [66] wird dieses verdeutlicht.
Bei einem Server ist der Agent in der Lage Services zu registrieren, damit andere Agenten oder lokale Dienste darauf zugreifen können. Auf diese Weise nehmen sie direkten Kontakt mit

[2] *PKI* ist ein Akronym für *P*ublic *K*ey *I*nfrastructure

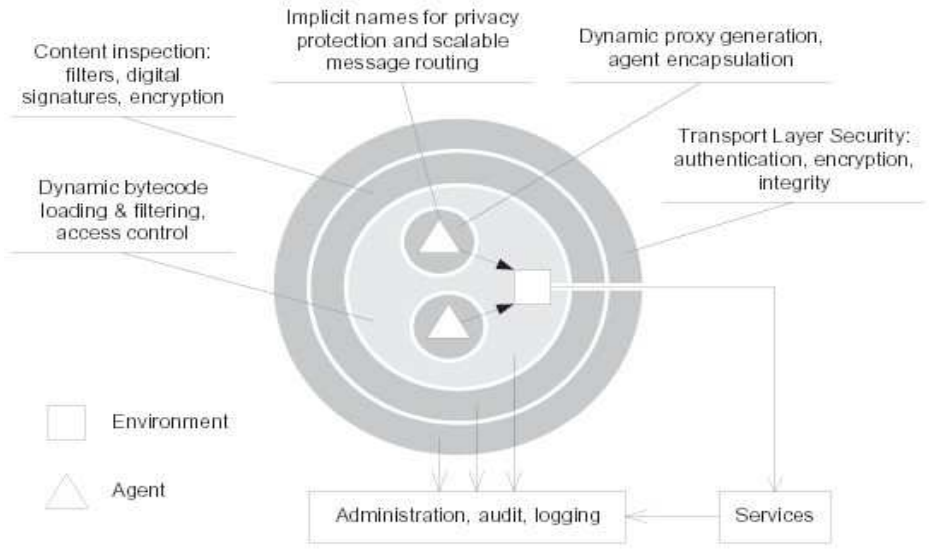

Abbildung 2.3: Sicherheitsarchitektur von SeMoA

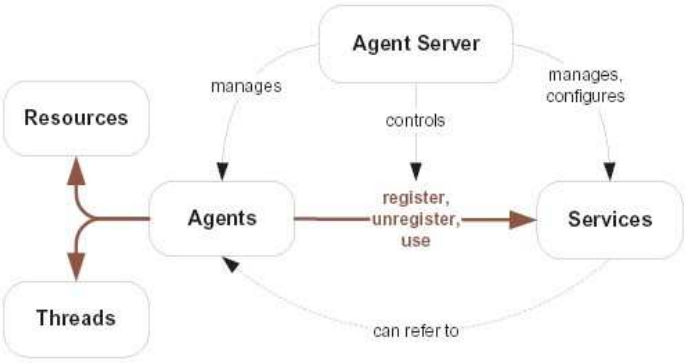

Abbildung 2.4: Abhängigkeitsmodell von SeMoA

dem Agenten auf. Weiter kann der Server auch Services registrieren und so Dienstleistungen dem Agenten anbieten. Auf dieser Basis verfolgt der Agent auf einem Rechner seinen Auftrag. Mit dem Beenden des Auftrags signalisiert der Agent dem System, dass er dieses wieder verlassen möchte. Die Plattform wird seinen Code zusammen mit seinen gesamten Daten in eine JAR-Datei packen (serialisieren), welche dann mit einer Signatur im PKCS#7 Format signiert wird [64]. Diese Datei wird anschließend zum gewünschten Zielsystem transferiert. In diesem Zustand ist der Agent in der Lage seinen Inhalt auch vertraulich durch unsichere Netze, wie das Internet, zu transportieren. Das Ziesystem des Agenten entpackt (deserialisiert) ihn, wenn er bei ihm angekommen ist und überführt ihn so von einem passiven in einen aktiven Zustand. Da bei dem Migrationsfortgang auch die Daten des Agenten berücksichtigt werden, sind diese nicht verloren gegangen, und der Agent kann auf der neuen Plattform mit diesen Daten weiter operieren.

2.3 Peer-to-Peer Netze

2.3.1 Einführung

In der klassischen Definition setzt sich ein Peer-to-Peer System zusammen aus einer verteilten Gruppe von *Peer-Knoten*. Diese verhalten sich gegenüber anderen Knoten autonom; d.h. sie können das Netzwerk jederzeit verlassen oder sich diesem wieder anschließen. Dabei tritt jeder Peer-Knoten sowohl als Client, als auch als Server auf. D.h. er bietet Dienste für andere an und nutzt selbst die Dienste der anderen Peer-Knoten. Dieses Konzept steht im ausgeprägten Kontrast zum Client-Server Modell, bei dem wenige spezialisierte Server ihre Dienste für eine große Menge Clients anbieten.
Möchte man ein Peer-to-Peer System beschreiben, lassen sich folgende Eigenschaften feststellen:

- Peers sind untereinander direkt verbunden.
- Ein Peer fungiert gleichermaßen als Dienstleistungsanbieter und als Nutzer der Dienstleistungen anderer Peers.
- Das System ist fehlertolerant.
- Die Anzahl der Teilnehmer kann sich spontan ändern, ohne die Funktionalität des Netzwerks zu beeinträchtigen.

Montresor definiert den Begriff Peer-To-Peer auch wie folgt [42]: „Peer-to-Peer is a class of applications, that put together ressources (storage, cycles, content, human persence) available at peer machines located at the edge of internet“.
Genutzt werden Peer-to-Peer Netzwerke in Applikationen, wie

- Datenaustausch-Programme
- Chat-Programme
- Informationspool-Programme für Projektgruppen
- Computerspiele

Der Ansatz des Peer-To-Peer Modells ist historisch betrachtet nicht neu. Die ursprüngliche Konzeption des Internets Ende des 1960er Jahre war bereits darauf ausgelegt, dass Computer das skizzierte Verhalten, sowohl als Konsument als auch als und Dienstleistungsanbieter aufzutreten, annehmen und ein fehlertolerantes und robustes Computernetz gegenüber Veränderungen ausbilden [53]. Doch schnell bildete sich eine andere Struktur heraus, als spezialisierte Server ihre Dienste einer großen Anzahl von unspezialisierten Clients zur Verfügung stellten — die Client/Server Architektur. Ab Mitte der 1990er Jahre wurde das Internet zunehmend zu einem Massenmedium und die Rechner, die auf der Clientseite eingesetzt wurden, verminderten mehr und mehr die Leistungsdiskrepanz zu den Servern, so dass sie auch die Funktion eines Servers wahrnehmen konnten. Deshalb besteht momentan die Chance, Computernetze aufzubauen, die dem damaligem Modell des Internets entsprechen.
Die Technologie von Peer-To-Peer Netzen wurde erst durch Projekte wie Seti@home [3] [74] 1995 und schließlich duch Napster [46] im Jahr 1999 einer Vielzahl von Anwendern bekannt. Dies wird durch die Anzahl der Anwender deutlich, die seit 1999 Napster und später auch ähnliche Dienste nutzten [42]:

[3] *Seti* ist ein Akronym für *S*earch for *e*xtra-*t*errestrial *i*ntelligence

Datum	Anwender von Napster
Januar 2000	$> 1.000.000$
November 2000	$> 23.000.000$
Februar 2001	$> 50.000.000$

Tabelle 2.1: Anzahl der Anwender von Napster

Gemeinsam haben Peer-to-Peer Systeme die Vorteile, dass die verteilten Teilnehmer keine feste IP-Adresse benötigen und im hohen Maße autonom gegenüber anderen Teilnehmern in bezug auf die Erbringung von Dienstleistungen und der Mitteilung von Ergebnissen sind. Darüber hinaus können Teilnehmer über einen längeren Zeitraum nicht am Netzwerk beteiligt sein und es dennoch nicht in dessen Funktionalität beeinträchtigen [76].
Um zu prüfen, ob es sich bei einem System um ein Peer-to-Peer System handelt, schlägt Shirky einen Lackmus-Test vor [76]:

- Unterstützt das System standardmäßig frequenzielle Verbindungen und zeitbegrenzte Netzwerkadressen?
- Unterstützt das System standardmäßig ein autonomes Verhalten der Peers?

Wenn beide Fragen für eingegebenes System positiv beantwortet werden, handelt es sich nach Shirky um ein Peer-to-Peer System.

2.3.2 Modelle von Peer-to-Peer Netzen

Die vorgestellte Definition von Peer-to-Peer Netzwerken ist unabhängig von der Netzwerktopologie und Organisationsmodell der Peers. Im Folgenden werden drei Modelle vorgestellt, die häufig in Peer-to-Peer Architekturen verwendet werden.

Broker-vermittelndes tauschen von Inhalten. Ein zentraler Broker verwaltet einen globalen Index über die angebotenen Dienstleistungen im Peer-to-Peer System. Ein Nutzer des Peer-to-Peer Systems registriert sein Angebot bei diesem Broker. Wenn der Anwender eine Ressource oder Dienstleistung im Netz suchen möchte, stellt er eine entsprechende Anfrage beim Broker. Dieser antwortet mit Informationen darüber, bei welchen anderen Nutzern das gesuchte Angebot zu finden ist. Auf diese Weise vermittelt der Broker den Anfragenden direkt zu dem Dienstleister, bei denen er das gewünschte Angebot erhalten kann. Dieses System hat sowohl zentrale Komponenten (Broker) wie auch dezentrale (Anwender), so dass das System auch als *hybrides Peer-to-Peer System* bezeichnet werden kann. Abbildung 2.5 zeigt ein solches System. Als Beispiel für ein solches System ist Napster [46] zu nennen.
Diese Variante des Peer-to-Peer Systems hat den Vorteil, dass ein Anwender immer ein optimales Ergebnis erhält, denn alle relevanten Angebote sind dem Broker bekannt und diese kommuniziert er auch vollständig an den Anwender. Allerdings wird dieser Vorteil damit erkauft, dass ein Ausfall des Brokers zum Ausfall des Gesamtsystems führt.

Reines Peer-to-Peer basiertes Inhaltetauschen. Im Gegensatz zum vorherigem Modell wird hier auf eine zentrale Einheit (Broker) mit einem globalem Index verzichtet. Stattdessen hat jeder Peer nur Kenntnis über seine eigenen Diensleistungen. Ein Anwender stellt seine Anfrage

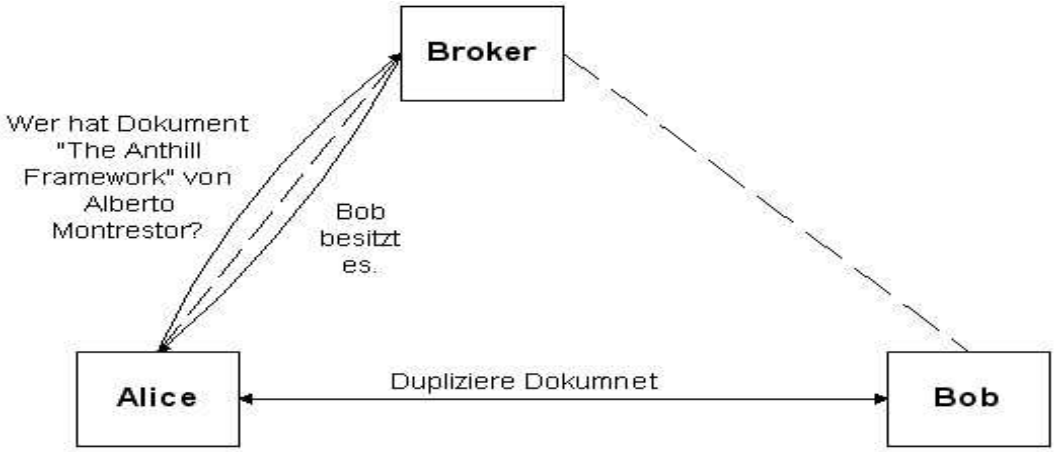

Abbildung 2.5: Broker-vermittelte Architektur

zu einer Ressource oder Dienstleistung nun an seine direkten Nachbarn. Diese können die Anfrage entweder positiv beantworten oder die Anfrage weiter zu ihren Nachbarn senden. Wie in Abbildung 2.6 gezeigt wird, bildet sich so eine Vermittlungskette von Netzteilnehmern, die dem anfragenden Anwender Zugang zu dem gesuchten Angebot ermöglichen. Ein Beispiel für ein solches System ist Gnutella [24].

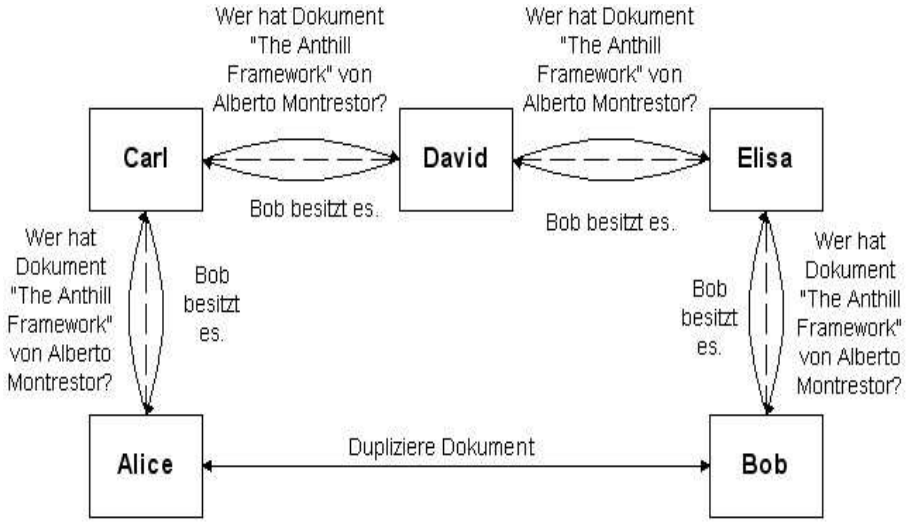

Abbildung 2.6: Reine Peer-to-Peer Architektur

Diese Variante des Peer-to-Peer Systems hat den Vorteil, dass sie ohne zentrale Instanzen auskommt und somit robust gegen den Ausfall einzelner Peers ist. Doch muss der Anwender dafür suboptimale Ergebnisse in Kauf nehmen, da nur ein Teil der Teilnehmer zu der Suchanfrage konsultiert wurde. Des Weiteren werden durch den Wegfall der zentralen Einheit die anstehenden Aufgaben auf die Teilnehmer-Knoten verschoben, was zum einen den Datenverkehr im Netz gleichmäßiger verteilt, aber auch Ressourcen auf den beteiligten Knoten einfordert.

Frequentielles teilen und bereitstellen zwischen Master und Slave. Bei diesem Modell handelt es sich um ein umgekehrtes Client/Server-Modell. Der Master verteilt zyklisch Daten an die beteiligten Peers, die diese Daten verarbeiten. Wenn diese dort verarbeitet wurden, werden die Ergebnisse zum Master gesandt, der dann die Teilergebnisse integrieren kann. Bei diesem Modell ist demnach der Fokus nicht auf die Daten und Dienstleistungen der einzelnen Teilnehmer gerichtet, sondern auf deren lokalen Systemressourcen. Ein prominentes Beispiel ist Seti@home [74], dessen Arbeitsweise in Abbildung 2.7 skizziert wird.

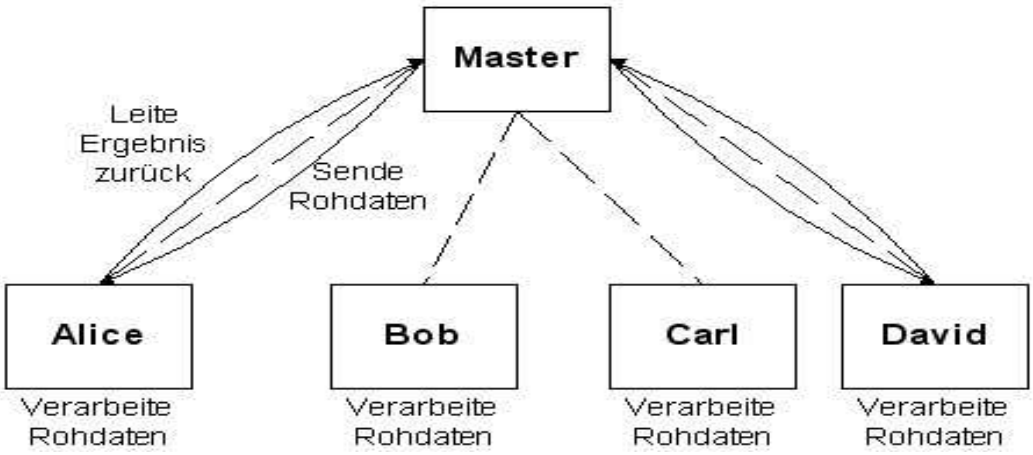

Abbildung 2.7: Frequentielles teilen und bereitstellen zwischen Master und Slave

Bezogen auf das Anwendungsgebiet dieses Modells ist der Vorteil des Masters, dass er auf eine hohe Gesamtperformance zurückgreifen kann, die sich aus den Einzelressourcen der beteiligten Slaves bildet. Trotz dieser Leistungsstärke kann der Master aber nur statistische Aussagen darüber treffen, wann Ergebnisse zu seinen verteilten Daten zu erwarten sind. So existiert für den Master keine Garantie, dass die Daten überhaupt verarbeitet werden und dass Ergebnisse kommuniziert werden. Auch kann er die Korrektheit der Ergebnisse nicht selbst prüfen.

2.3.3 Risiken

Der Einsatz eines Peer-to-Peer Systems ist mit einigen Risiken verbunden, denen adäquat begegnet werden muss. Ein Problem ist die unkontrollierte Verbreitung von Informationen in solchen Netzen. Dadurch können schützenswerte Daten (bspw. private Informationen, medizinische Daten oder unternehmensinterne Daten), die nicht verbreitet werden dürfen, diffundieren. Eine ähnliche Problematik ist die Nutzung von Peer-to-Peer Netzen in Projekt- oder Interessens-Gruppen. Ohne einen sorgfältigen Umgang mit dieser Technik können Daten, auch außerhalb dieser Gruppen eine Verbreitung erleben, die nicht erwünscht ist. Weiterhin besteht die Gefahr, dass Daten während des Transports manipuliert werden. Auch kann suggeriert werden, dass eine direkte Verbindung zwischen zwei Peers existiert, in der Realität aber ein weiterer Peer als Zwischeninstanz geschaltet ist (*Man-in-the-middle Attack*). Auf diese Weise können Ergebnisse mitgelesen, verändert oder abgefangen und nicht weitergeleitet werden. Prinzipiell besteht auch das Risiko, dass Daten als Trojanische Pferde missbraucht werden. Auf diese Weise können sich andere Programme zur Spionage oder Sabotage im Netz verbreiten. Neben dem Angriff von Peer-to-Peer Netzen über Daten, kann dies auch durch Aktionen von Anwender geschehen. Benutzer können bspw. mit *Denial-of-Service* Attacken Rechner, wie Informations-Broker,

lahm legen oder mittels *Storage flooding* Attacken eine Dateischwemme im Netz verursachen. Schließlich besteht noch das Risiko, das angebotene Daten oder Dienstleistungen nicht wahrheitsgemäß angegeben wurden. Dies bedeutet nicht eine Manipulation des Inhalts, sondern vielmehr falsche Angaben zu einer Dienstleistung zu machen. So könnte bspw. angeboten werden eine Berechnung mit einer bestimmten Genauigkeit und unter Nutzung bestimmter Programme durchzuführen, aber tatsächlich wird nichts berechnet sondern ein fikives Ergebnis weitergegeben. Dies ist gerade bei kostenpflichtigen Dienstleistungen ein Problem. In diesem Zusammenhang steht auch das Problem, dass die Teilnehmer prinzipiell ihre Identität ändern können um so Zugang zu Informationen zu erhalten.
Den meisten Risiken kann jedoch entgegen gewirkt werden. Zum einen sollten sich alle Teilnehmer eines Peer-to-Peer Netzes eindeutig identifiziert werden können, was z.B. mittels digitale Zertifikate für jeden Teilnehmer und einer PKI [61] möglich wird. Wenn das gewährleistet ist, kann der Zugriff und die Verbreitung von Daten von der Identität jedes Teilnehmers abhängig gemacht werden. Schließlich sollten die Daten, prinzipiell immer verschlüsselt übertragen werden, um deren Manipulation zu vermeiden [80]. Auch die Autoren Waldman, Cranor und Rubin des Kapitels „Trust“ in [53] befürworten den Einsatz einer (globalen) PKI, um das Vertrauen der Netzteilnehmer untereinander zu vergrößern.

2.3.4 SeMoA als Peer-to-Peer Netzwerk?

Die Applikation der Diplomarbeit verwendet SeMoA-Plattformen für ein Peer-to-Peer Netzwerk. Anhand der Definitionen in Abschnitt 2.2.2 wird deutlich, dass die Charakterisierung von Peer-to-Peer Netzen auch auf die SeMoA-Plattformen zutrifft. Die Teilnehmer sind direkt miteinander verbunden. Sie können sowohl als Nutzer wie auch als Dienstleister im Netzwerk in Erscheinung treten. Das System ist fehlertolerant, da es den Betrieb nicht blockiert, wenn ein Teilnehmer in einen undefinierten Zustand versetzt wird. Genauso bleibt die Stabilität des Systems erhalten, wenn sich im laufenden Betrieb Teilnehmer im Netzwerk hinzukommen oder wegfallen. Der Lackmus-Test von Shirky auf SeMoA angewandt bringt folgende Aussagen:

1. SeMoA unterstützt eine frequentielle Teilnahme einer Plattform im Netzwerk und assoziiert mit dieser Plattform keine bestimmt Netzwerk-Adresse, so dass sie variabel ist.

2. Die Teilnehmer können sich unabhängig vom Gesamtsystem ab- oder anmelden, ohne dessen Betrieb zu unterbrechen.

Damit genügt SeMoA den Anforderungen, die Shirky in seinem Lackmustest fordert.
Ordnet man SeMoA den vorgestellten Netzwerk-Modellen zu, wie sie für Peer-to-Peer Netzwerke eingesetzt werden, so ist SeMoA vergleichbar mit dem reinen Peer-to-Peer Modell, denn das SeMoA-Netzwerk ist denzentral organisiert und benötigt keine zentralen Komponenten, damit ein Informationsaustausch stattfinden kann. Daneben bietet SeMoA auch Lösungen für die angesprochenen Risiken an, wie sie schon bei der Vorstellung des Systems im Kapitel „Die Agentenplattform SeMoA“ dargestellt wurden. Des Weiteren ist die Sicherheit der Plattform in zentralen Teilen an digitalen Signaturen gebunden. Zudem ist bereits die Möglichkeit gegeben, Rechte für Agenten jedes Teilnehmers individuell und feingranular zu spezifizieren.
Der Einsatz von SeMoA als Peer-to-Peer System ist nicht nur möglich, sondern auch sinnvoll. Denn als Multi-Agenten System hat es sowohl für Peer-to-Peer Systeme, als auch für die Applikation speziell nutzbare Vorteile:

Reduzierung der Bandbreite: Der Agent migriert mit dem Suchalgorithmus zu den Daten und nicht – wie üblicherweise – umgekehrt. Dadurch wird je nach Anwendungsfall eine Kompression von über 99% erreicht [65].

Verteilte Suche: Eine Gruppe von Agenten kann eine Anzahl von Peers parallel durchsuchen.

Dezentrale Suche: Zur Suche werden Ressourcen des Peers verwendet und so für eine gleichmäßige Lastverteilung gesorgt.

Koordinierte Suche: Agenten können während der Suche miteinander kommunizieren um die Suche zu optimieren. Neue Erkenntnisse werden geteilt. Daneben kann mittels Kommunikation auch das Migrationsverhalten der Agenten verbessert werden. Denn durch Prüfung der verfügbaren Bandbreite auf einem Migrationspfad zu einem Peer, kann der beste Pfad ausgewählt und den anderen Agenten mitgeteilt werden. Genauso kann ein Agent versuchen einen Pfad zu einem Peer zu finden, wenn er von seinem aktuellen Peer aus keinen vorfinden sollte.

Kooperative Suche: Kooperierende Agenten können Aufträge effizienter (Zeit- und Ressourcen-Nutzung) bearbeiten und haben die Chance, selbst in dezentralen Peer-to-Peer Netzwerken Auftragsergebnisse zu erzielen, die sonst nur in zetralen Peer-to-Peer Netzwerken zu erreichen sind.

Freie Wahl des Algorithmus: Agenten bringen den Such-/Evaluationsalgorithums zur Datenquelle mit. Dadurch kann man digitale Daten nach beliebigen Kriterien (Stichwörter, Frequenzspektren, Wasserzeichen) und mit beliebigen Algorithmen durchsuchen. Dadurch kann sich jeder Anwender eine eigene und spezialisierte Sammlung von Suchmaschinen zusammenzustellen, die einzigartig im Peer-to-Peer Netzwerk ist [4]. Darüber hinaus hat der Anwender den Vorteil, dass die Daten nach selbstgewählten Richtlinien untersucht werden. Dies nimmt dem Datenbesitzer die Möglichkeit, den Berwertungmaßstab zu seinen Gunsten zu ändern.

Umwelt-Anpassung: Mobile Agenten sind für den Einsatz in Peer-to-Peer Netzwerken optimal ausgestattet. Denn sie benötigen keine permanente Verbindung zum ihrem Ausgangspeer und können durch ihr Migrationsverhalten optimal auf die Dynamik von Peer-to-Peer Netzwerken reagieren.

Vertrauensbildung: Agenten, die vom Anwender signiert wurden, sind eine Grundlage für die Vertrauensbildung im Netzwerk. Dadurch lassen sich Agenten in Abhängigkeit ihrer Besitzer differenziert Rechte zu- und absprechen.

Daten-Integrität: Agenten, die in der Lage sind ihre Daten und Informationen zu kapseln und zu verschlüsseln, können in Peer-to-Peer Netzwerken für eine Daten-Integrität sorgen, so dass diese Daten für Dritte nicht einsehbar sind.

Rechtsverbindlichkeit: Signierte Agenten sind notwendig für die rechtsverbindliche Ausführung von Suchaufträgen. Damit lassen sich Dienstleistungen abrechnen und nachweisen.

Unabhängigkeit vom Netzwerk: Mobile Agenten können selbst dann weiterarbeiten, wenn der aktuelle Knoten keine Netzverbindung aufweist, z.B. weil es sich um ein mobiles Endgerät handelt! Sobald die Netzwerkverbindung wieder verfügbar ist, kann der Agent wieder kommunizieren bzw. die Plattform verlassen.

Neben diesen positiven Aspekten gibt es auch einige, die sich nachteilig für Anwender auswirken:

[4] Es wäre interessant, diese Sammlung wiederum als Dienstleistung anderen Peers zur Verfügung zu stellen.

Anonymität: Peer-to-Peer Netze mit den vorgeschlagenen Eigenschaften verhindern eine anonyme Teilnahme ihrer Benutzer am Netzwerk. Es gibt jedoch die Möglichkeit pseudonym aufzutreten.

Tracking: Mobile Agenten hinterlassen Spuren im Netzwerk. Somit kann durch sie unter Umständen ein Migrationspfad zu ihrem Besitzer aufgezeigt werden.

Profiling: Wenn mehrere Peers zusammenarbeiten und Informationen über den Suchauftrag Unbefugten bekannt machen, ist es möglich Interessensprofile der Netzwerkteilnehmer zu erstellen.

Nachrichtenaufkommen: Damit der Nachrichtenverkehr, den die Agenten bei ihrer Aktivität verursachen, nicht zu einer unnötigen Belastung der Netzwerkkapazitäten führt, sollten Nachrichten nur bei Relevanz verschickt werden und dies auch zielgenau zum betreffenden Agenten.

Zeitaufwand: Bei der Applikation ist es schwer abzuschätzen, wie lange die Informationsbeschaffung andauert. So beeinflussen die Wahl des Medientypes und die verwendete Suchmaschine die Dauer des Auftrags genauso, wie die Leistungsfähigkeit der einzelnen Peers und das dort jeweils vorhandene Datenangebot.

Verlust von Agenten: Es ist nicht garantiert, dass alle mobilen Agenten wieder zum Ausgangspeer zurückkehren, da zwischenzeitlich andere Peers ihre Migrationspfade gesperrt haben könnten.

Alterung der Informationen: Agenten, die in der Applikation eingesetzt werden, prüfen erst das Vorkommen an relevanten Dateien, ohne diese selbst schon zu bevorraten. Erst in einem weiteren Schritt aquirieren Agenten die angeforderten Dateien. Allerdings kann zwischenzeitlich ein Netzwerkteilnehmer sich nicht mehr am Netzwerk beteiligen, die Rechte für den Besitzer des Agenten eingeschränkt haben oder die betreffende Datei gesperrt haben.

Kostenverschiebung, statt -reduktion: Der Informationsanbieter erkauft die Einsparungen der Netzwerkbandbreite durch eine höhere CPU- und Speicherlast.

Risikopotential Netzwerk-Teilnehmer: Der Informationsanbieter muss eine Agentenplattform betreiben. Dies stellt, wie z.B. Webserver auch, ein Risiko für die IT-Infrastruktur dar. Der Kunde muss ebenfalls ein Agentensystem betreiben, mit den genannten Risiken für sein System.

2.3.5 JXTA

Der Name JXTA kommt von „JuXTApose“ und stammt von Bill Joy, dem Chef-Wissenschaftler bei Sun Microsystems. Dabei deutet er JXTA als eine Möglichkeit Dinge aneinander zu setzen, wie es der Bedeutung von Peer-to-Peer entspricht [42]. Entwickelt wurde es von Sun Microsystems Inc. und ist inzwischen als Open Source Projekt [34] weiter in der Entwicklung. Die Idee von JXTA ist, die Applikationsentwicklung im Bereich Peer-to-Peer effizienter zu gestalten. Es wurde nämlich erkannt, dass viele Applikationen, wie Napster, Gnutella, Seti@home oder auch Instant Messenger, wie der AOL Instant Messenger oder ICQ, konzeptionell die gleichen Eigenschaften besitzen. Diese Eigenschaften sind das Untersuchen von Peers und das Suchen bzw. Versenden von Dateien und Daten. Dennoch müssen diese Applikationen immer wieder

für eine Plattform angepasst werden und sind auch nicht in der Lage Informationen mit anderen Applikationen zu teilen, was einen redundanten Aufwand in der Informationsgewinnung und -haltung bedeutet.
JXTA ist eine Sammlung von allgemein definierten Peer-to-Peer Protokollen, die es ermöglichen dezentral organisierte Netzwerke zwischen verschiedenartigen Komponenten, wie Server, Rechnern, PDA oder Mobilfunktelefone aufzubauen, so dass diese sich wie Peers verhalten. JXTA-Protokolle sind unabhängig von Programmiersprachen und somit auch nicht an eine spezielle Plattform gebunden. Insgesamt umfasst JXTA sechs Protokolle, die es Peers ermöglichen, eigenverantwortlich *Peer-Gruppen* aufzubauen oder zu verwalten. Daneben gibt es in JXTA zwei Möglichkeiten für Peers miteinander zu kommunizieren. Dies ist neben dem Versenden von *Nachrichten* die Nutzung sog. *Pipes*. Damit werden unidirektionale Kommunikationskanäle zwischen Peers beschrieben. Ein solcher Kanal kann neben der direkten Verbindung zweier Peers auch als ein Kanal mit meinem Eingang und mehereren Ausgängen definiert werden, womit sich eine Broadcast-Kommunikation aufbauen lässt.

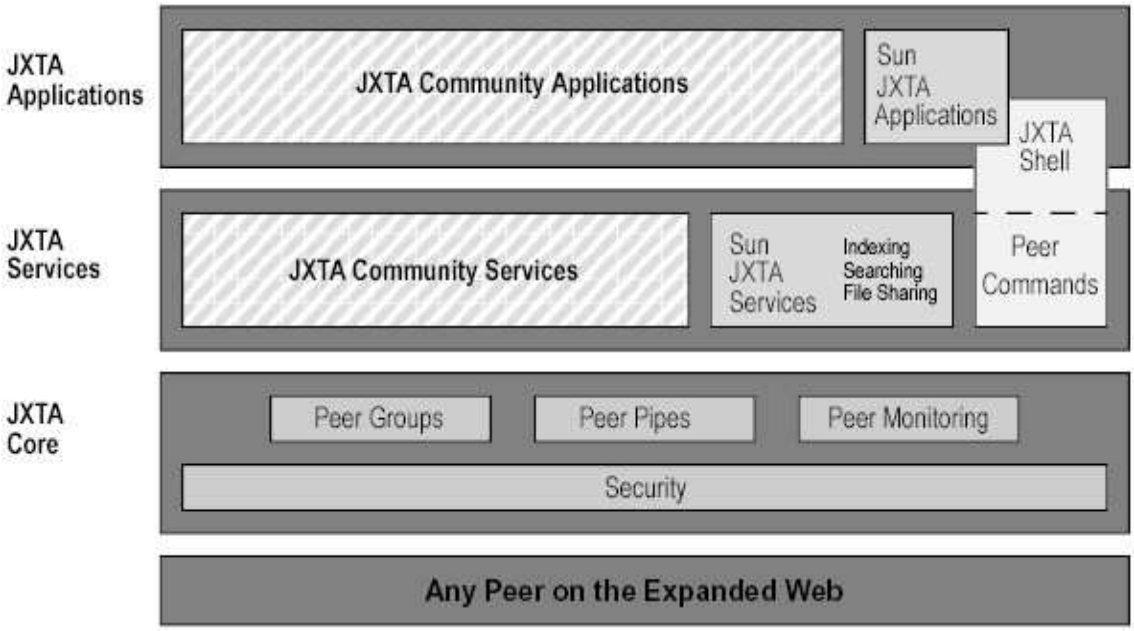

Abbildung 2.8: JXTA Architektur

Die Architektur von JXTA wird in Abbildung 2.8 [81] deutlich. Diese ist in drei Ebenen untergliedert. Der *JXTA Core* bildet die unterste Schicht des Modells. In ihr sind die angesprochenen Protokolle und Konzepte realisiert, auf denen JXTA aufbaut. Diese sind für Peer-to-Peer Applikationen in JXTA essentiell. Die mittlere Schicht sind die *JXTA Services*. Hier werden, komplexere Dienste zur Verfügung gestellt, wie bspw. ein Authentifizierungsmechanismus oder eine Suchfunktion, die auf die Möglichkeiten, die der JXTA Kern bietet, zurückgreift. Diese Schicht beinhaltet also Netzwerkdienste, die für Peer-to-Peer Applikationen nicht zwingend notwendig sind. In der obersten Schicht sind die *JXTA Applications* lokalisiert. Um ihre Funktion zu realisieren, können Applikationen auf Dienste als auch auf die Kernschicht zurückgreifen.
JXTA will folgende Ziele erreichen:

Interoperabilität: JXTA soll eine Plattform sein, welche die notwendigen Basisfunktionen für

ein Peer-to-Peer Netzwerk bereitstellt. So sollen Peers in der Lage sein, sich gegenseitig zu lokalisieren und miteinander zu kommunizieren.

Plattform-Unabhängigkeit: JXTA ist nicht an eine bestimmte Programmiersprache (Java, C/C++, Perl etc.) gebunden und ist unabhängig von Transportprotokollen (TCP/IP, Irda, Bluetooth, HomePNA etc.) und Entwicklungsplattformen.

Allgegenwärtigkeit: JXTA ist nicht an ein bestimmtes Endgerät gebunden und kann sowohl auf Servern und Clients, wie auch auf PDAs und Mobilfunktelefonen eingesetzt werden.

Mittels JXTA soll jedem Peer unabhängig von dessen genutzer Soft- und Hardware ermöglicht werden, den Kontakt zu anderen Peers herzustellen und so beliebige Peer-to-Peer Netz aufzubauen [81], [26].
Mit den Protokollen von JXTA können Applikationen entwickelt werden, die folgende Eigenschaften aufweisen [25], [86]:

1. Auffinden anderer Peers.
2. Netzwerkweites Verteilen und Tauschen von Daten.
3. Bilden von Peer-Gruppen, die zusammen eine Dienstleistung für die anderen Teilnehmer des Netzwerks anbieten.
4. Überwachen eines Peers über remote-Verfahren.
5. Sicheres Kommunizieren über das Netzwerk mit anderen Peers.
6. Verteilte Speicherung von Daten.
7. Peer-to-Peer eMail-Service.
8. Peer-to-Peer DNS.

Im Hinblick auf die Allgegenwärtigkeit ist anzumerken, dass sich Peers in der Regel an den Rändern eines Netzwerks befinden und nicht zwangsläufig mittels DNS addressiert werden können. Deshalb vergibt JXTA eigene IDs, die es den Peers ermöglichen, sich im Netzwerk zu bewegen, Transport- und Netzwerk-Adresse zu verändern. JXTA-Peers sind immer mindestens einer Peer-Gruppe zugeordnet, in der sie aber trotz der Gruppierung die beschriebenen Freiheiten besitzt. Die JXTA-ID ermöglicht auch noch addressierbar für andere Peers zu sein, wenn der Peer kurzzeitig keine Verbindung zum Netz hat. Ein JXTA-Peer publiziert Netzwerk-Endpunkte, die zur direkten Verbindung zwischen zwei Peers dienen. Dabei müssen Peers nicht direkt miteinander verbunden sein, sondern können über „Zwischen-Peers“ in Verbindung treten und somit NAT, Firewalls, Proxies etc überwinden.
Damit JXTA erfolgreich in Peer-to-Peer Netzwerken eingesetzt werden kann, muss es für verschiedene Sicherheitsbereiche Lösungen vorsehen. JXTA ermöglicht einen differenzierten Zugriff auf Ressourcen. Dies wird in einem rollen-basierten Vertrauensmodell geregelt, in das die JXTA-Peers eingebunden sind. Dabei bilden diese Peers aber keine PKI aus, sondern, ähnlich wie bei PGP [5], ein *Web of Trust*.

Authentität: Es muss garantiert werden können, dass sich ein Anwender nur mit seiner eigenen Identität im Netzwerk ausweist.

[5] *PGP* ist ein Akronym für *P*retty *G*ood *P*rivacy

Vertraulichkeit: Es muss garantiert werden, dass Nachrichten nur für den Empfänger einsehbar sind.

Autorisierte Aktionen: Es muss sichergestellt sein, dass ein Absender von Nachrichten auch die entsprechende Befugnis hat, diese Nachricht zu versenden.

Daten-Integrität: Daten dürfen auf ihrem Transport oder zu einem anderen Zeitpunkt nicht von Unbefugten manipuliert werden können.

Widerlegbarkeit: Es muss gerantiert werden können, dass versendete Nachrichten direkt vom Absender kommen.

Eine Nachricht in JXTA wird im XML-Format formuliert und besteht im wensentlichen aus zwei Komponenten. Zum einen ist dies die eigentliche Nachricht, von der Hashwerte berechnet werden und die mit dem öffentlichen Schlüssel des Empfängers verschlüsselt wird. Damit ist für eine Datenintegrität und -vertraulichkeit gesorgt. Die zweite Komponete der Nachricht stellt der öffentlich zugängliche Teil dar. Dieser umfasst Identifikations-Informationen des Absenders für dem Empfänger. Während des Transports der Nachricht wird auf diesen Bereich der Nachricht zugegriffen um zu prüfen, ob der Absender auch das Recht hat, dem Empfänger eine Nachricht zu senden. Dafür können diesem Bereich der Nachricht noch zusätzlich Referenzen, Zertifikate, Kennwerte angehängt werden. Daneben wird auch die Absendeadresse der JXTA-Nachricht mit der Adresse dieser Referenz geprüft, um diese zu beglaubigen. Um den Nachrichtentransport zuküftig sicher zu gestalten, soll JXTA auch die Protokolle SSL [6] und IPSec [7] nutzen [81].
Der Lackmustest von Shirky bringt das Ergebnis, dass JXTA die notwendigen Charakterisiken von Peer-to-Peer Netz besitzt. Folgende Eigenschaften sind dafür bei JXTA von Bedeutung:

- Direkte Verbindung zwischen zwei Peers.
- Peer sind gleichermaßen Dienstleistungsnutzer und -anbieter.
- Ein JXTA-System ist fehlertolerant.
- Peers können dynamisch in eine Peergruppe eingeordnet werden oder diese verlassen.

Da JXTA aber nur Protokolle für Peer-to-Peer Netzwerke zur Verfügung stellt und keine eigentliche Applikation ist, bleibt zu klären, welche Modelle von Peer-to-Peer Netzwerken mit JXTA realisisert werden können.
JXTA benötigt für die eigene Funktionalität keine zentralen Komponenten, somit kann mit JXTA ein reines Peer-to-Peer System entwickelt werden. Allerdings befürworten die Autoren Brookshier, Govoni und Kirshnan den Aufbau von hybriden Peer-to-Peer Systemen, um so Charakteristiken einiger Dienste, wie bspw. eMail-Dienste, gerecht zu werden [8].

2.3.6 Gegenüberstellung: JXTA und SeMoA

Das Ziel dieser Diplomarbeit ist, allgemein formuliert, eine Applikation zu entwickeln, die Peer-to-Peer Netze nutzt, um in diesen mit verschiedenen Suchmaschinen nach Informationen zu suchen. Hierfür benötigt die Applikation eine Middleware, die die Bildung von Peer-to-Peer Netzen ermöglicht. Wie in der Diplomarbeit bereits beschrieben, leisten dies sowohl JXTA, wie auch die Agentenplattform SeMoA. Im Folgenden werden die Unterschiede zwischen beiden

[6] *SSL* ist ein Akronym für *S*ecure *S*ockets *L*ayer
[7] *IPSec* ist ein Akronym für *I*nternet *P*rotocol *Sec*urity

Plattformen herausgearbeitet und erläutert, wieso letztlich SeMoA und nicht JXTA bei der Diplomarbeit eingesetzt wird.

Bei einem Vergleich von JXTA mit mobilen Agenten können folgende Aussagen festgehalten werden: Die allgemeine Definition eines Agenten besagt, dass es dessen Aufgabe ist, eine Dienstleistung für einen Anwender zu erbringen. Dafür kann er alleine oder im Verbund mit anderen Agenten agieren und sowohl nur stationär operieren oder migrieren. Im direkten Vergleich ist aber eine JXTA-Applikation weniger ein mobiler Agent als vielmehr ein stationärer. Denn JXTA-Applikationen bewegen sich nicht durch das Netz, sondern tauschen Nachrichten aus oder kommunizieren um ihre Dienstleistung zu erbringen. Somit ist ein JXTA-Peer als stationärer Agent anzusehen, obwohl er sich dynamisch im Netz verhalten kann und dieses, wie bereits erwähnt, auch kurzzeitig verlassen kann [8]. Trotzdem bietet JXTA prinzipiell die Möglichkeit beide Agententypen zu nutzen. So ist bspw. Anthill als JXTA-Service realisiert. Dabei nutzt Anthill mobile Agenten zum Informationsaustausch und JXTA als Ausführungsumgebung.

JXTA ist, bezogen auf seinen Umfang, ein relativ simples Framework, da es nur Protokolle zur Verfügung stellt. Aber es bietet neben den geforderten Eigeschaften von Peer-to-Peer Netzwerken noch weitere Charakteristiken, wie das integrieren beliebiger netzwerkfähiger Komponenten, ein Sicherheitskonzept für den Datenverkehr und den Netzteilnehmer, das Bilden von Peergruppen, sowie die Kommunikation mittels Pipes.

Allerdings liegen nur Teile dieser Eigenschaften im Fokus der Applikation. Vorteilhaft von JXTA ist dessen Unabhängigkeit gegenüber Programmiersprachen. Dies ist nützlich in Bezug auf die Integration von verschiedenen Suchmaschinen. Im Vergleich dazu ist SeMoA vollständig in Java implementiert und ist funktional auf diese Sprache fixiert. Aber gerade die Sicherheitsfunktionen, wie bspw. Zugriffskontrolle, Privilegien oder die Bytecode-Verifikation sind wichtige Eigenschaften von SeMoA. Da mit Java auch Programme anderer Programmiersprachen genutzt werden können, ist die Festlegung auf Java nicht nachteilig für SeMoA.

Daneben muss in JXTA Netzwerken den Endgeräten Beachtung geschenkt werden, denn diese können neben herkömmlichen Computern auch Mobilfunktelefone und PDA sein. In Bezug auf die Applikation der Diplomarbeit ist der Einsatz einer Suchmaschine auf Mobilfunktelefonen, zumindest wie sie heute verbreitet sind, auszuschließen. Des Weiteren würden PDAs und Computer als mögliche Zielumgebungen verbleiben. Aufgrund der beschränkten Energie-Ressourcen eines PDAs ist der Einsatz der Applikation hier nur eingeschränkt sinnvoll. So sollte vom PDA aus nur die Erstellung eines Suchauftrags initiiert werden können, jedoch sollte das Gerät nicht als Informationspool für andere Agenten zur Verfügung stehen.

Resümierend kann festgestellt werden, dass JXTA als alleinige Middelware nicht die Funkionen von SeMoA bietet, wie sie für die Applikation benötigt werden. Zwar kann mittels JXTA auch ein Peer-to-Peer System aufgebaut werden, doch verfügt dieses per se über keine Mobilität. So müsste bspw. noch eine Applikation wie Anthill hinzugenommen werden, um mobile Agenten zur Verfügung zu stellen. Dabei müßte dann auch das Sicherheitskonzept von JXTA und Anthill aufeinander abgestimmt werden, um Agenten und Plattformen den notwendigen Schutz zu bieten. Das bedeutet einen zu großen zeitlichen Aufwand, um während der Bearbeitungszeit der Diplomarbeit JXTA und Anthill den Bedürfnissen der Applikation anzuspassen. SeMoA erfordert diesen Aufwand nicht. SeMoA ist ein homogenes System, das bezogen auf die Applikation, alle benötigten Funktionen besitzt und auch über notwendige Sicherheitsmechanismen verfügt. Zusätzlich bietet SeMoA bereits die Möglichkeit an, fremde mobile Agenten mit SeMoA zu nutzen [56].

2.4 Inhaltsbasierte Suche in Textdateien

Die in der Diplomarbeit zu entwickelnde Applikation soll in der Lage sein, verschiedene Suchmaschinen einzusetzen, um nach Informationen zu suchen. Welcher Medientyp verwendet werden soll, ist hierbei nur von der Suchmaschine abhängig. Exemplarisch wurde eine Open Source Suchmaschine *Lucene* [32] integiriert. Diese ist eine Textsuchmaschine, d.h. sie durchsucht Dateien nach einer vorher festgelegten Zeichenfolge und gibt die Dateien als Treffer an, wenn ihr Inhalt diese Zeichenfolge beinhaltet. Um ein Verständis für die Funktionsweise solcher Suchmaschinen zu schaffen und weil Komponenten der Applikation Methoden des Information Retrievals in Textdokumenten beinhalten, wird die textbasierte Suche näher vorgestellt.

2.4.1 Suchstrategie in Textdateien

Die Suche nach Informationen in Textdateien ist einer der wichtigsten Aufgaben des Information Retrievals (IR). Graphisch kann ein solches System, nach Fuhr [21] dabei wie in Abbildung 2.9 modelliert werden.
Dabei ist für IR-Systeme eines der größten Probleme eine geeignete Interpretation des textuellen Inhalts. Beipielsweise ist dabei eine Analyse von mehrdeutigen Wörtern in Bezug auf den Kontext besonders wichtig. Es gibt zwei verschiedene Ansätze, wie textbasierte Informationssuche realisiert werden kann. So existiert neben einer Freitextsuche auch die Möglichkeit, jede Textdatei durch einen Diskriptor eindeutig zu charakterisieren. Auf Basis dieser abstrahierten Repräsentation des Textes kann dann die Suche erfolgen.
Im Folgenden wird beschrieben, wie die Freitextsuche in Textdateien funktioniert. Bevor ein Text untersucht werden kann, muss er in drei Stufen vorbereitet werden. Dadurch sollen Komponenten des Textes, die für eine Informationsgewinnung hinderlich sind, beseitigt werden.

1. Erkennung eines Satzendes. Auf diese Information wird in einigen Freitextsuch-Funktionen Bezug genommen, so dass diese Information gewonnen werden muss. Die Detektion eines Satzendes ist schwierig, weil ein Satzendepunkt auch als Abkürzungspunkt interpretiert werden kann. Somit sind diese Ergebnisse nur näherungsweise gültig und hängen neben der Textvorlage auch von der Qualität der Detektion ab [21].

2. Aus dem vorliegenden Text werden einzelne Wörter anhand von Leer- und Interpunktionszeichen detektiert. Als Zwischenergebnis erhält man eine Wortliste.

3. Trennen der informationstragenen Wörter von den informationsarmen Wörtern aus der vorliegenden Wortliste. Der Informationsgehalt eines Wortes kann mittels der relative Häufigkeit bestimmt werden. Demnach haben Wörter, die selten im Text auftreten einen höhreren Informationsgehalt als Wörter, die häufig im Text verwendet wurden. Besonders häufige Wörter sind bspw. Artikel oder Pronomen, diese werden in einer sog. Stoppwort-Liste zusammengefaßt.

Nach dieser Vorbereitung des Textes kann die Freitextsuche nach zwei Ansätzen durchgeführt werden. Der *informatische Ansatz* interpretiert eine Suchvorgabe als Zeichenfolge, die in einem Text gefunden werden soll. Der *computerlinguistische Ansatz* prüft den Text auf das Vorkommen von Wörtern. Hierzu müssen die Worte des Textes in ihre Normalform transformiert werden, z.B. „unschlüssig“ (Zweigwort) → „Schluss“ (Wortstamm) → „schließ(en)“ (Wortwurzel). Dafür werden spezielle morphologische und syntaktische Verfahren eingesetzt.
Die Vorverarbeitung eines vorliegendes Textes ist in den letzten beiden Schritten relativ einfach. Allerdings führt Fuhr einige Probleme an, die sich bei dieser Informationssuche, bezogen auf die extrahierten Wörter, ergeben [21]:

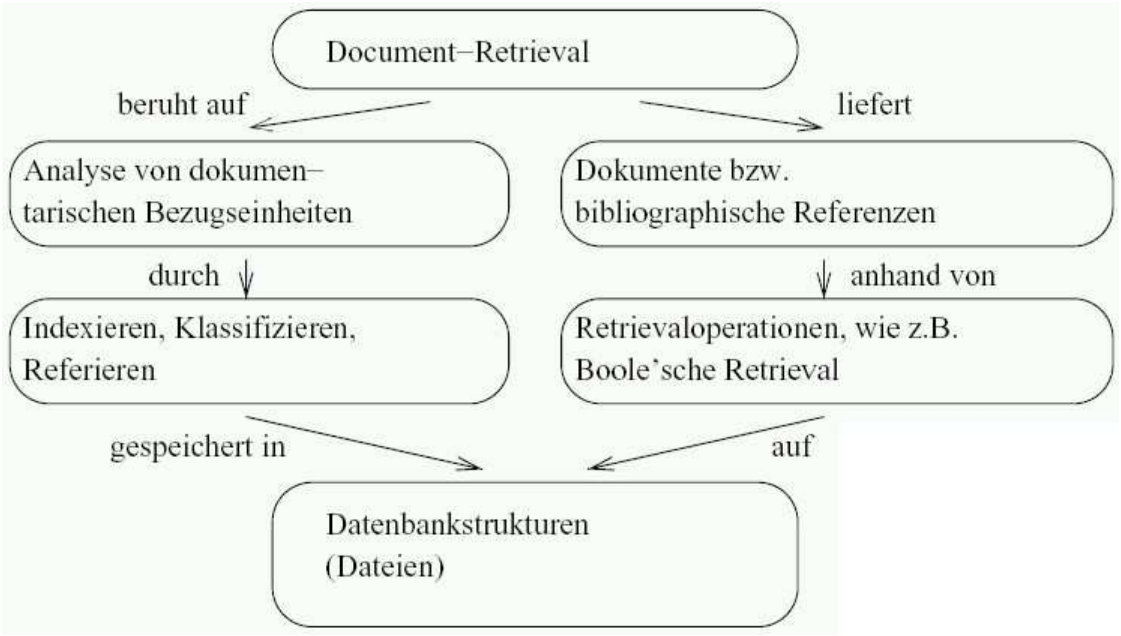

Abbildung 2.9: Inhaltsbasierte Suche

Homographen: verschieden gesprochene Wörter mit gleicher Schreibweise (Tenor: Sänger / Ausdrucksweise).

Polyseme: Wörter mit mehreren Bedeutungen (Bank: Sitzgelegenheit / Geldinstitut)

Flexionsformen: Formen, die durch Konjugation und Deklination eines Wortes entstehen (Haus - (des) Hauses - Häuser oder schreiben - schreibt - schrieb - geschrieben)

Derivationsformen: verschiedene Wortformen zu einem Wortstamm (Formatierung - Format - formatieren)

Komposita: mehrgliedrige Ausdrücke (Bundeskanzlerwahl - Wahl des Bundeskanzlers oder information retrieval - retrieval of information - information was retrieved)

Diesen Problemen kann nur durch intelligentes Parsen der Informationen entgegengewirkt werden. Intelligent meint hier, dass basierend auf gegebenen Regeln und/oder Erfahrungswerten erkannt werden kann, wo in einem gegebenen Text die beschriebenen Probleme auftreten. Die Komplexität des Problems wird allein dadurch schon deutlich, dass der Inhalt des Textes genau erkannt werden muss, um die aufgeführten Fehler zu umgehen.

2.4.2 Bewertung von Textdateien

Um detektierte Dateien miteinander vergleichen zu können, ist es notwendig, an diese einen Bewertungmaßstab (Index) anzulegen. Im Information Retrieval gibt es verschiedene Verfahren der Indizierung von Dokumenten. Nohr zählt er hierfür folgende Verfahren auf [51] [50]:

- Einfache Stichwortextraktion / Volltextinvertierung
- Statistische Verfahren

- Informations- bzw. Computerlinguistische Verfahren
- Regelbasierte Verfahren
- Wörterbuchbasierte Verfahren
- Pattern-Matching-Verfahren
- Begriffsorientierte Verfahren

Die im Rahmen dieser Diplomarbeit verwendete Suchmaschine nimmt eine Indizierung mittels statistischer Verfahren vor. Auf dieses Verfahren richtet sich im Besonderen der Fokus dieses Kapitels. Statistische Verfahren sind reine deterministische Verfahren, die mit geeigneten Kennzahlen versuchen ein gegebenes Dokument zu bewerten. Aus diesem Grund brauchen linguistische Methoden, um bspw. die Bedeutung eines Wortes in einem Text zu ermitteln, nicht weiter berücksichtigt werden. Die Notation der folgenden Gleichungen ist in Tabelle 2.2 zusammengefasst.

Notation	Bedeutung
$DOKFREQ_{tk}$	Anzahl der Dokumente mit betreffendem Term in der Kollektion
$FRRQ_{td}$	Frequenz eines Terms in einem Dokument
$FRRQ_{tk}$	Frequenz eines Terms in der Kollektion
$GESAMT_{td}$	Gesamtzahl der Terme im Dokument
$GESAMT_{tk}$	Gesamtzahl der Terme in der Kollektion
IDF_t	Inverse Dokumentfrequenz
N	Anzahl der Dokumente in der Kollektion
n	Anzahl der Dokumente, in denen der Term auftritt
S	Signifikanz
TF_{td}	Termfrequenz in einem Dokumant
TF_{tk}	Termfrequenz in der Kollektion
$TFIDF_t$	Termfrequenz-Inverse-Dokumentfrequenz

Tabelle 2.2: Notations-Tabelle

Eine erste Aussage über die Relevanz eines Textes wird mit der Termfrequenz getroffen. Diese sagt aus, mit welcher relativer Häufigkeit ein Term in einem Dokument vorkommt:

$$TF_{td} = \frac{FRRQ_{td}}{GESAMT_{td}} \tag{2.1}$$

Um den Wertebereich der ermittelten Gewichte gering zu halten, schlägt Nohr alternativ folgende Formel vor:

$$TF_{td} = \frac{ld(FRRQ_{td} + 1)}{ld(GESAMT_{td})} \tag{2.2}$$

Um das Gewicht der Termfrequenz weiter zu verarbeiten wird noch eine weitere Kennzahl benötigt, die eine Aussage darüber trifft, wie häufig der Term in dem vorliegendem Textbestand (Kollektion) im Verhältnis zur Summe aller Terme im Textbestand vorkommt. Hierfür gilt die Formel:

$$TF_{tk} = \frac{FRRQ_{tk}}{GESAMT_{tk}} \tag{2.3}$$

Mit Hilfe dieser beiden Kenngrößen lässt sich nun die Signifikanz S ermitteln. Dieser ergibt sich aus der Differenz der beiden Werte:

$$S = TF_{td} - TF_{tk} \tag{2.4}$$

Mittels Signifikanz lassen sich nun zwei Aussagen über Terme treffen:

1. Tritt ein Wort in einem Textdokument häufig auf, hat dieses eine höhere Signifikanz, als Wörter mit geringem Vorkommen.

2. Wörter, die in der Dokumenten-Kollektion seltener vorkommen, haben eine geringere Signifikanz, tragen aber einen höheren Beitrag zur Charakterisierung eines Textes bei.

Oft werden Termfrequenz und das Vorkommen des Termes in einer Dokumenten-Kollektion mit der *inversen Dokumentfrequenz* in ein Verhältnis gesetzt:

$$IDF_t = \frac{FRRQ_{td}}{DOKFREQ_t} \tag{2.5}$$

Analog zur Berechnung der Termfrequenz kann auch hier der Wertebereich der inversen Termfrequemz unter Verwendung der nachfolgenden Formel beschränkt werden.

$$IDF_t = 1 + \left(ld \left(\frac{N}{n} \right) \right) \tag{2.6}$$

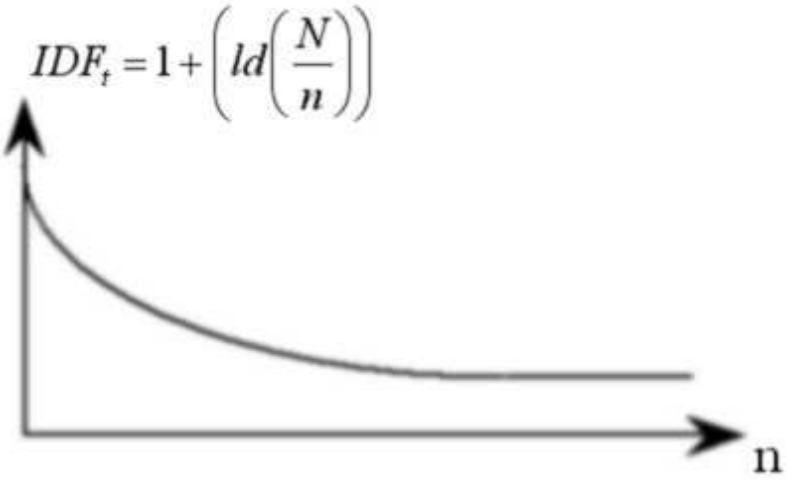

Abbildung 2.10: Inverse Dokumentfrequenz

Abbildung 2.10 macht deutlich, dass ein Term, der nur in wenigen Dokumenten vorkommt, einen höheren Frequenzwert bekommt. Die Verwendung des Logarithmus hat den Vorteil, dass der Bereich, in welchem sich ein Term auf wenige Dokumente in der Kollektion verteilt, stärker differenziert werden kann, da die Steigung der Kurve stärker ausgeprägt ist. Es ist anzumerken, dass in der Litratur verschiedene Definitionen zur inversen Dokumentfrequenz existieren. So wird sowohl mit dem Logarithmus dualis, wie auch dem Logarithmus zur Basis 10 gerechnet. Des Weiteren besteht die Gleichung bei einigen Autoren [21], [50], [72], [38], [33] aus einem

einzelnen oder mehreren Termen. Mit der Ermittlung der beiden Werte Termfrequenz und inversen Dokumentfrequenz liegen nun alle Werte vor, um die Gewichtung eines Dokuments mit der TFIDF vorzunehmen. Dafür gilt die Formel:

$$TFIDF_t = TF_t \cdot IDF_t \tag{2.7}$$

Die Formel der Termfrequenz-Inverse-Dokumentfrequenz ist ein Gewichtungsverfahren, welches eine weite Verbreitung bei Information Retrieval Anwendungen findet. So ist es bspw. Bestandteil der Suchmaschine *Lucene* [32] und wird von Yaung et al. [85] für den Einsatz mit einem mobilen Agentensystem preferiert.
Es ist auffällig, dass wenige Suchmaschinen Angaben dazu machen, wie sie gefundene Informationen gewichten. Selbst beim Open Source Produkt *Lucene* wird nicht offiziell auf die Nutzung der Formel der Termfrequenz-Inverse-Dokumentfrequenz hingewiesen. Vielmehr wurde es in [36] diskutiert und konnte im Quellcode von Lucene [82] belegt werden.

2.4.3 Darstellung der Ergebnisse

Um die gewonnenen Ergebnisse aus einem textbasierten Information Retrieval graphisch zu repräsentieren, wird heute meist das *Vektorraummodell* eingesetzt. In diesem Modell werden die Dokumente und die Suchanfrage als Vektoren in einem mathematischen Raum aufgefasst, welcher durch die Terme definiert wird, die sich aus der Datenquelle ergeben. In einer Vorverarbei-

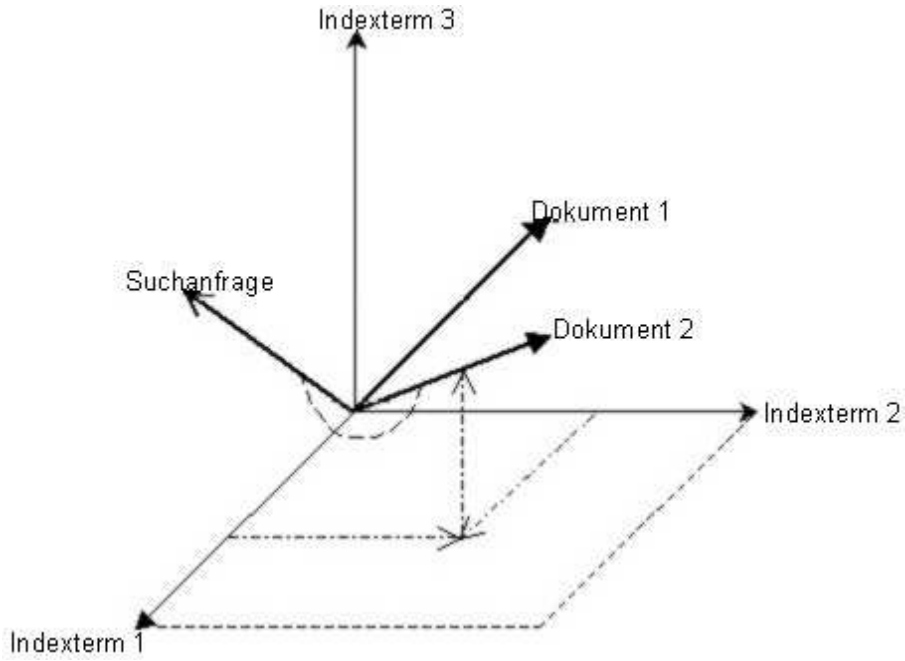

Abbildung 2.11: Vektorraummodell

tung muss dafür von jedem Dokument eine Gruppe an Termen extrahiert sein, für die dann eine Termfrquenz bestimmt wird. Diese Terme ergeben sich aus dem Verfahren, wie oben beschrieben und werden dann auf Basis von informationslinguistischen Auswahl- und Bearbeitungsregeln ausgewählt. Für die graphsiche Repräsentation werden diese dann orthonormalisiert, so dass die n-dimensionalen Vektoren dann Werte 0 oder 1 annehmen können. Dabei entspricht der

Wert Null dem Fehlen des betreffenden Terms in einem Dokument und eine Eins, dass der Term in dem Dokument vorkommt [1], [50], [21]. Dadruch kann ein Vektorraum wie in Abbildung 2.11 gebildet werden. Dieses zeigt ein Beispiel, in dem aus der vorliegenden Datenquelle drei Terme auf Grundlage zweier Dokumente ermittelt wurden. Dabei deckt Dokument 2 alle Terme ab, während Dokument 1 den Indexterm 1 nicht beinhaltet. Die Suchanfrage, die an diese Datenquelle gerichtet wurde, enthält nicht Indexterm 2. So kann nun im Folgenden die Anfrage mit den Dokumenten graphisch verglichen werden.
Im Folgenden ist zu prüfen, ob das Vektorraummodell geeignet ist, Ergebnisse innerhalb der Applikation zu repräsentieren. Dafür spricht, dass das Vektorraummodell ein graphisches und keine textuelles Modell, wie bspw. eine Tabelle ist. Tabellen haben den Nachteil, dass sich ihre Aussage langsamer erschließt, als bei graphischen Modellen. Allerdigns kann ein graphisches Modell mit der Anzahl der darzustellenden Ergebnisse nicht mehr davon profitieren, dass sich einem das Ergebnis schnell erschließt. Vielmehr können anfangs nur Tendenzen festgestellt werden. Einer hohen Trefferquote zu einer Suchanfrage, steht die Alternative gegenüber, die Suchanfrage präziser zu formulieren. Doch je detaillierter eine Suchanfrage wird, desto mehr Dimensionen werden zur Darstellung benötigt. Damit wäre dann auch eine Darstellung mittels 3-dimensionaler Körper oder nach Quine-McCluskey sinnvoller, allerdings ist die Komplexität hoch und damit ein schneller Informationsgewinn nicht möglich. Neben diesen Aspekten ist anzumerken, dass Dokumente vorverarbeitet werden müssen, damit eine entsprechende Arbeit mit dem Vektorraummodell möglich wird.
Letztlich spricht aber gegen das Vektorraummodell, dass es nicht auf andere Medien angewendet werden kann und nur ein Einsatz bei einer textbasierten Suche sinnvoll ist.

Kapitel 3

Systementwicklung

3.1 Einführung

Im folgenden Kapitel wird die Entwicklung der Applikation vorgestellt. Dabei wird ausgehend von der Aufgabenbeschreibung in Kapitel 1.2 die objektorientierte Systementwicklung angewandt, wie sie von Prof. Rausch gelehrt wird. Entsprechend werden Verfahren eingesetzt, wie sie in den entsprechenden Vorlesungen für die objektorientierte Software-Entwicklung vorgestellt wurden [60], [59].
Den einzelnen Phasen des Software-Engineering ist gemein, dass diese hochgradig iterativ sind. Darüber hinaus kann es das Ergebnis einer Phase erfordern, die bisherigen Ergebnisse zu verwerfen und im Lebenszyklus der System-Entwicklung bei vorangestellten Phasen mit der Entwicklung fortzufahren. Aus diesem Grund wird hier nicht immer der komplette Entwicklungsschritt, sondern lediglich das Ergebnis präsentiert. Zur allgemeinen Veranschaulichung der Ergebnisse wird die UML [69], [70], [52] mit deren Symbolik genutzt. Am Anfang des Kapitels wird eine theoretische Einführung in die jeweilige Phase gegeben und allgemein die dort benutzten graphischen Modelle beschrieben. Auf dieser Grundlage werden dann im weiteren Verlauf des Kapitels die entsprechenden Ergebnisse vorgestellt, wie sie bei der Entwicklung der Applikation bestimmt wurden.
Schon zu Beginn des Prozesses stand fest, dass sich die Problemlösung auf die Technik der mobilen Agenten stützt und in der Sprache Java implementiert wird. Diese Entscheidung fiel vor dem Hintergrund, dass die Applikation für die Agentenplattform SeMoA vorgesehen war. Trotzdem wurde während der gesamten Systementwicklung eine Lösung angestrebt, die allgemeine Gültigkeit besitzt und nicht von einer bestimmten Programmiersprache oder einer Agentenplattform abhängig ist.
Die Analysephase ist der Beginn der System-Entwicklung. In ihr wird ausgehend von der Aufgabenbeschreibung zuerst definiert, welche Aufgaben die Applikation zu bewältigen hat. Diese Aufgaben muss die Applikation unabhängig davon lösen, in welcher Sprache sie entwickelt wird bzw. in welcher Zielumgebung sie zum Einsatz kommt. Auf diese Weise kristallisiert sich die Essenz des Systems heraus. Da zu diesem Zeitpunkt der Systementwicklung nur ein abstraktes Modell des Systems besteht, das Design- und Implementierungsaspekte unberücksichtigt lässt, erhält man auf diese Weise die Logik des Systems und somit die Aussage „was“ das System leisten soll und nicht „wie“ diese Leistungsfähigkeit hergestellt werden kann.
Um ein abstraktes Modell des Systems zu erhalten, wird das System durch verschiedene Modelle beschrieben und dabei weiterentwickelt. Diese Modelle werden im Folgenden beschrieben.

3.1.1 Use Cases

Use Cases kapseln das System als *Black Box* ab und beschreiben nur dessen Reaktion auf je eine bestimmte Aktion des Anwenders. Es wird somit aus Sicht des Benutzers das Verhalten des Systems analysiert. Der Benutzer tritt als externer Aktor mit dem System in Kontakt, wodurch beschrieben wird, auf welche Eingabe das System in welcher Art und Weise zu reagieren hat. Daneben werden aber auch die Grenzen des Systems gegenüber seiner Umwelt aufgezeigt. Das Systemverhalten ist mit der Beschreibung einer Ausgabe, bezogen auf die Eingabe, umfassend beschrieben. Wenn bei einer Aufgabe, die das System zu leisten hat, erkannt wird, dass dazu mehrere Teilaufgaben zu bearbeiten sind, die für sich eigenständige Systeme darstellen, so ist das Use Case dahingehend zu verfeinern und zu konkretisieren. Da zwischen diesen Use Cases eine Abhängigkeit besteht, bildet sich so eine Hierarchie heraus. Auf diese Weise wird die grobe Architektur des Systems ersichtlich. Man erhält eine Sammlung von Use Cases, die Systemkomponenten beschreiben und in ihrer Summe das Systemverhalten komplett darstellen.

3.1.2 Szenarien

Szenarien dienen dazu, den Ablauf genauer zu beschreiben, die in einem Use Case aufgeführt werden. Dabei wird festgehalten, welche Teilschritte notwendig sind, um die festgelegte Reaktion und/oder Ausgabe zu einer Eingabe zu erzielen. Inhaltlich besteht somit ein Szenario aus einer Sequenz von Aktionen. Um das Verhalten genauer zu definieren, wird mit der Erstellung eines Szenarios begonnen, der einer *Best Case* Situation zugrunde liegt. In diesem Szenario konzentriert man sich nicht auf mögliche Fehler, die während der Aktion auftreten können, sondern auf den fehlerfreien Ablauf. Auf mögliches Fehlverhalten des Systems bei dieser Aktion wird dann in anderen Szenarien eingegangen. Auf diese Weise soll später vermieden werden, dass die Applikation bei einer *Worst Case* Situation in ein undefiniertes Verhalten übergeht, sondern einem festgelegten Verhaltensmuster folgt. Neben diesen Aspekten eines Szenarios erfolgt noch eine weitere wichtige Konkretisierung der Applikation. Denn neben der externen Sicht des Anwenders auf das System, wird in einem weiteren Schritt die *Black Box*-Sichtweise auf das System zugunsten einer *White Box*-Sicht aufgegeben. So können interne Systemkomponenten ermittelt werden, die dann nicht nur die externe Interaktion mit dem Anwender bestimmen, sondern auch die interne Interaktion zwischen den einzelnen Komponenten.

3.1.3 Sequenzdiagramme

Eine graphische Repräsentation der Szenarien sind die Sequenzdiagramme. In diesen werden die beschriebenen Aktionen und Interaktionen der Systemkomponenten durch Nachrichten bildlich dargestellt und weiter konkretisiert. Diese Nachrichten beinhalten neben dem Aufruf einer gewünschten Methode eines Objekts auch die Parameter, mit denen die Methode operieren soll. Diese gibt dann einen definierten Rückgabewert zurück. Es wird somit die Kommunikation zwischen den einzelnen Objekten deutlich. Der Informationsgewinn der Sequenzdiagramme gegenüber den Szenarien resultiert vor allem in der Information, wie lange ein Objekt eine Methode anfordert und diese somit für andere blockieren könnte. Daneben ist es mit einem Sequenzdiagramm auch möglich parallele Aktionen zu definieren. Allerdings sollte vor der Definition paralleler Prozesse genau geprüft werden, ob die bisherige Systementwicklung korrekt ist und den Qualitätsansprüchen genügt. Denn Parallelität ist ein Indiz dafür, dass zusammengehörige Aktionen voneinander getrennt und verschiedenen Objekten zugeordnet wurden. Dies kann negative Auswirkungen auf den gesamten Software-Lebenszyklus der Applikation zur Folge haben [60], [59].

3.1.4 Klassendiagramme

Grundlage für das Klassendiagramm der Applikation sind die entworfenen Sequenzdiagramme. Diese beinhalten die Objekte der Systems in Form von Aktoren und die notwendigen Methoden in Form von Nachrichten. Um das Klassendiagramm zu entwerfen, werden die einzelnen Objekte als Klassen aufgefasst, wobei deren Methoden aus den jeweiligen Nachrichten entstehen, die von einem Objekt einer anderen Klasse angefordert wurden. Zusätzlich zu den Methoden besitzt eine Klasse auch Attribute, welche im Sequenzdiagramm durch die Übergabeparameter der Nachrichten gewonnen werden. Das so entstandene Klassendiagramm beinhaltet alle notwendigen Funktionen der Applikation, so dass dies die Grundlage für die Implementierung der Applikation in einer Zielsprache erfolgen kann. Um die Applikation problemlos in bestehende Systeme integrieren zu können, müssen die Modifizierer der Methoden und Attribute der Klassen definiert werden, um die Sichtbarkeit dieser zu regulieren. Dabei gilt, dass alle Komponenten, die einen Zugriff von externen Methoden zulassen müssen als *public* zu definieren sind und alle anderen als *private*. Somit kann verhindert werden, dass über nicht-öffentliche Klassenkomponenten der Zustand eines Klassenobjektes manipuliert werden kann. In den Abbildungen A.15 und A.16 werden neben einem datailiertem Klassendiagramm auch ein vereinfachtes dargestellt, um den Zusammenhang einzelner Klassen aufzuzeigen.

3.2 Anforderungen

Die Beschreibung zur Applikation, welches die Grundlage für die gesamte Software-Entwicklung darstellt, fordert folgende Leistungsmerkmale der Anwendung:

- Agenten suchen auf Peers nach Informationen in deren Dateien.
- Agenten bringen selbst die Suchmaschinen mit, die sie für ihre Informationssuche benötigen.
- Suchmaschinen können die Dateien nach unterschiedlichsten Gesichtspunkten untersuchen (Wort-Erkennung, Bild-Recherche, Wasserzeichen-Suche, Frequenz-Erkennung,...). Die Applikation hat demnach den Charakter eines Data Miners.
- Die Bewertung der Ergebnisse erfolgt nach Methoden, die der Agent selbst für die Suche zur Verfügung stellt.
- Die Suche soll von mehreren Agenten kooperativ erledigt werden.
- Agenten sollen mittels Kommunikation die Suche optimieren.
- Die Applikation soll mehrere Suchaufträge parallel bearbeiten können.
- Die Teilnehmer des Netzwerks können selbst festlegen, wo im Dateiverzeichnis nach Informationen gesucht werden darf und welche Dateiformate diese Dateien haben dürfen.
- Basierend auf dem Suchergebnis kann der Auftraggeber Dateien auswählen, von denen er ein Duplikat haben möchte.
- Es sind Maßnahmen zu treffen, dass keine unbefugten Agenten die Suche und die gewonnenen Ergebnisse manipulieren können.

Zusätzlich zu diesen Leistungsmerkmalen wurde die Möglichkeit geschaffen, mittels Agenten direkt mit den Netzwerkteilnehmern zu kommunizieren. Dies soll die Anonymität verringern und als Mittel genutzt werden Vertrauen zwischen den Netzwerkteilnehmern zu schaffen. Der gewählte Name der Applikation richtet sich nach dessen Funktionalität. Diese besteht im Kern daraus, dass mit verschiedenen Suchmaschinen auf einen Datensatz zugegriffen werden kann, um so unterschiedliche Informationen zu bekommen. So könnten bspw. Bilder als solche gesucht werden; die Bilddatenbank kann aber auch genutzt werden, um nach einem Gesicht, einem Handelsmarke oder einem anderen Referenzmuster zu suchen. Entsprechend dieser möglichen Leistungsfähigkeit wird die Applikation als *DataMiner* bezeichnet.

3.3 Use Cases

Entsprechend der vorangestellten Vorstellung von Use Cases wird im Folgenden die Entwicklung dieser für die Applikation vorgestellt. Um die Abhängigkeit der einzelnen Use Cases übersichtlich darzustellen, wird anfangs eine Hierarchie dieser in Abbildung 3.1 und A.1 dargestellt.

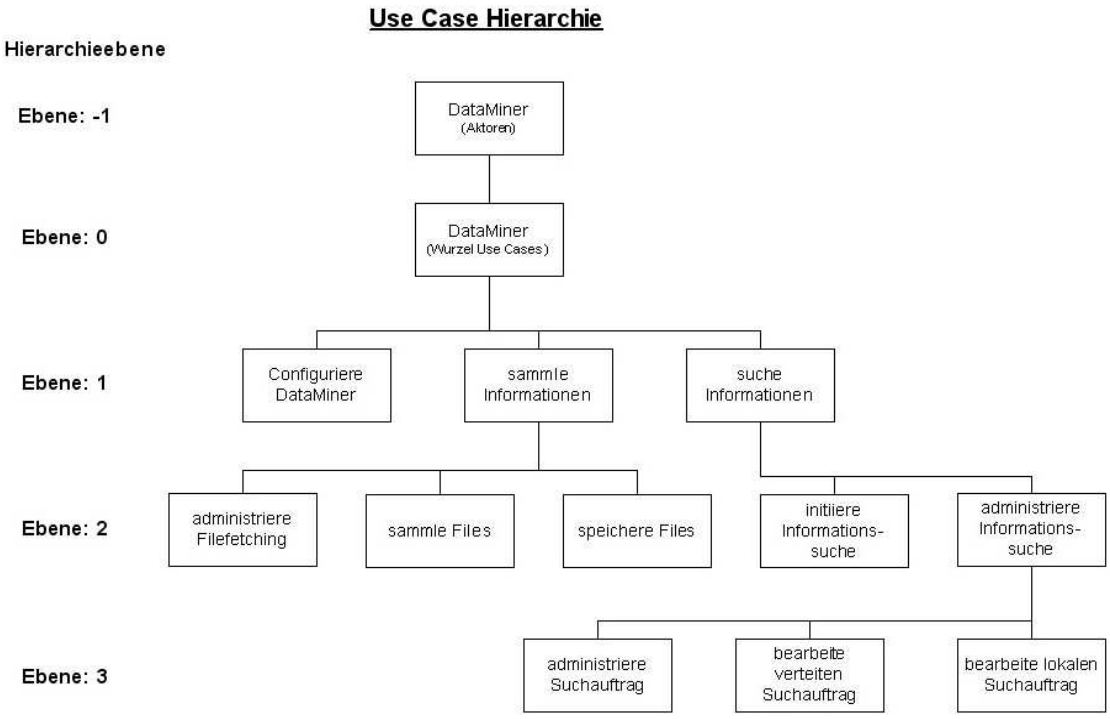

Abbildung 3.1: Use Case-Übersicht

Wie aus Abbildung A.2 ersichtlich ist, wurden drei Aktoren ermittelt, die im direkten Kontakt mit dem System stehen. Dies sind neben dem Anwender zwei mobile Agenten, die im Folgenden beschrieben werden.

Anwender: Der Anwender als Informationssuchender möchte die Applikationen allgemein nutzen. Er tritt mit ihr in Kontakt, wenn er sie parametrisieren möchte oder wenn er nach Informationen suchen lassen möchte. Die Parametrisierung umfasst hierbei den Aspekt,

ab welcher Datei-Hierarchie nach Informationen gesucht werden kann und welches Dateiformat die Dateien haben dürfen. Dafür soll dem Anwender eine Datenbank zur Verfügung gestellt werden, die Dateiformate in Ordnung der MIME-Types [40] präsentiert, so dass der Anwender gezielt das Format finden kann, was er freigeben möchte. Neben diesen Parametern, die ein Suchen in seiner Verzeichnisstruktur betreffen, kann der Anwender auch festlegen, wie viele Agenten für die Suche eingesetzt werden sollen. Zur Beschaffung der Dateien wird noch der Parameter benötigt, wo diese Dateien in seiner Verzeichnisstruktur abzulegen sind.

SearchAgent: Der SearchAgent sucht im Netzwerk nach den Dateien auf Grundlage der Informationen, die er vom Anwender erhalten hat. Damit er zu einem Ergebnis kommen kann, benötigt er noch einen Suchalgorithmus, der die Daten sowohl vom Anwender wie auch von den Dateien, klassifizieren kann. Der SearchAgent wird eine bestimmte Anzahl von Netzwerkteilnehmern besuchen und dort nach relevanten Dateien fahnden. Als Ergebnis nimmt er aber nur Informationen über die Datei und nicht sie als solche mit, denn es ist nicht davon auszugehen, dass immer alle gefundenen Dateien im Interesse des Anwenders sind, so dass es sinnvoller ist, erst dessen Auswahl zu erfragen und dann die entsprechenden Dateien zu beschaffen.

FetchAgent: Der FetchAgent übernimmt den Part der Informationsbeschaffung. Seine Aufgabe besteht darin, dem Anwender die Dateien zu beschaffen, die dieser auf Grundlage der Ergebnisse des SearchAgents ausgewählt hat. Dafür soll der Agent anhand einer Liste zu den entsprechenden Netzwerkteilnehmern migrieren, dort ein Duplikat der Datei erstellen und dieses, wieder auf dem Heimatrechner angekommen, dem Anwender im Verzeichnis hinterlegen.

Entsprechend der drei unterschiedlichen Aufgabengebiete des Programms *Konfiguration, Suche* und *Beschaffung* kann die Applikation nun verfeinert werden. Im Folgenden muss nun, soweit notwendig, mit der Verfeinerung der einzelnen Aufgaben fortgefahren werden.

PeerService: Der PeerService wurde eingeführt, um Dienstleistungen auf dem Peer anzubieten. Diese sind im wesentlichen Zugriffe auf die lokale Dateistruktur, damit die Applikation Dateien editieren kann. Daneben ist der PeerService auch für Agenten (fremde, wie eigene) zuständig. Denn der PeerService wird im Netzwerk bekannt gemacht, so dass fremde Agenten über diesen ihre Suchmaschinen am Peer nutzen können. Dies wird später allerdings voraussetzen, dass genau überprüft werden muss, ob ein entsprechender Agent über die notwendigen Rechte verfügt, seine Suchmaschine vor Ort nutzen zu dürfen oder nicht.

Die Abbildungen A.4, A.5 und A.6 zeigen jeweils die Verfeinerung der vorher genannten Use Cases. Die Komponenten werden im Folgenden besprochen.

ManagementService: Der ManagementService stellt die Schnittstelle zwischen der Benutzeroberfläche und der Funktionalität des Programms dar. Seine Aufgaben sind die Koordination aller Suchaufträge, also das Initiieren, Ergebnisse überprüfen und Beenden dieser, sowie die Beschaffung angeforderter Dateien. Die Ergebnisse der Suchaufträge sollen an ihn übergeben werden, wenn die Suchaufträge abgeschlossen sind.

SearchService: Der ManagementService delegiert die Suchaufträge an SearchServices. Ein SearchService ist für einen speziellen Suchauftrag verantwortlich. Er soll alle relevanten Daten eines Suchauftrags verwalten und eine gewisse Anzahl von SearchAgents mit der Informationsbeschaffung beauftragen.

SearchAgent: Sie werden vom SearchService beauftragt im Netzwerk bestimmte Teilnehmer aufzusuchen. Dort sollen sie mittels Suchmaschinen nach Informationen suchen. Erkenntnisse, die während der Suche gewonnen wurden und einen Beitrag zu einer effizienteren Auftragsabwicklung leisten, sollen sie direkt während der Suche den anderen SearchAgents kommunizieren. Ihre Resultate, die die Suchmaschine bei den Netzwerkteilnehmern ermittelt hat, sollen die SearchAgents abschließend an den SearchService kommunizieren, der sie mit der Suche beauftragt hat.

db_MediaTypes: Der Inhalt dieser Datenhaltung sind Informationen über allgemein bekannte Medientypen, die für eine Suche in Dateien der lokalen Verzeichnisse explizit freigegeben wurden.

db_KeywordSet: Der Inhalt dieses Datensatzes soll die Formulierung einer Suchanfrage durch den Anwender unterstützen. Sie enthält Vorschläge zu bestimmten Medientypen; dies können z.B. Wörter, Bilder oder Audioinformationen sein.

db_PeerSet: Diese Datenhaltung beinhaltet Informationen über die Teilnehmer im Netzwerk. Diese können dann von anderen Diensten der Applikation abgerufen werden. Aktuell umfaßt dies nur Informationen über die RAW-URL. Diese wird von Agenten benötigt, um zu dem entsprechenden Peer zu migrieren. Mit dem Einsatz des *SHIP* [1]-Protokolls werden aber umfangreichere Informationen über die Peers verfügbar sein. Die für die Applikation wichtigen Informationen über die RAW-URL zum Migrieren und der POD-URL zu Kommunikation können dann gezielt gewonnen werden.

initiiere Informationssuche: Dieses Use Case enthält alle Funktionen um einen Suchauftrag zu erstellen. Eine Mediendatenbank stellt in Abhängigkeit des Medientypes verschiedene Suchmaschinen zur Verfügung, die dem System und damit dem Anwender zur Verfügung stehen. Die Interaktion des Anwenders mit dem System erfolgt in der Konfiguration dieser Suchmaschine. Um diese Parametrisierung weiter zu konkretisieren, wird auf eine schlüsselwort-Datenhaltung zurück gegriffen. Dadurch soll der Anwender die Möglichkeit bekommen, seinen Suchauftrag mit weiteren Daten zu ergänzen. Der ManagementService ist als neuer Aktor hinzugekommen. Er verarbeitet die Eingaben, die der Anwender in einer Benutzeroberfläche gemacht hat, weiter.

administriere Informationssuche: Der ManagementService erstellt aus den bisherigen Informationen einen Suchauftrag, der die Administration speziell für diese Suche übernimmt. Die Aufgabe des ManagementService besteht nun darin, die verschiedenen Suchaufträge zu verwalten und zu überwachen. Des Weiteren ist dieser Service die Schnittstelle zwischen der Benutzeroberfläche und der Applikation. Der SearchService konkretisiert nun den Suchauftrag, indem Informationen über die beteiligten Peers vom PeerSet erfragt und den eigentlichen Suchauftrag SearchAgents zuweist.

Somit ist der eigentliche Suchprozess in dem Use Case integriert. Um nun zu definieren, welche Komponenten dieser Prozess nutzt, muss eine weitere Verfeinerung dieses Use Cases erfolgen.

Gemäß den beschriebenen Aufgaben des Use Case „administriere Informationssuche“ ist dieses nun in drei Teile zu gliedern.

[1] *SHIP* ist ein Akronym für *S*imple *H*ost *I*nformation *P*rotokoll

administriere Suchauftrag: Dieser Use Case beinhaltet alle administrativen Aufgaben zu einem Suchauftrag, wie bereits erläutert wurde. Er umfasst somit die lokalen Aufgaben eines Suchauftrags, während die anderen beiden für die Aufgaben mit den mobilen Agenten in Zusammenhang stehen.

bearbeite verteilten Suchauftrag: Die Aufgaben und Funktionalitäten des SearchAgents werden in diesem Use Case beschrieben. Gemäß des Anforderungsprofiles dieses Agententyps sind hier alle Aktionen zu beschreiben, die der Agent vor einer Migration zu einem anderen Netzwerkteilnehmer machen muss. Es muss aber auch beschrieben werden, wie ein Agent mit einem Netzwerkteilnehmer in Kontakt treten kann und was er nach einer Ankunft bei diesem zu erledigen hat. Wie aus der Use Case-Graphik zu entnehmen ist, ist der PeerService der Kontakt zwischen Agent und Peer. Da der PeerService somit den Zugang auf das eigenen Computersystem von Netzwerk aus bereit stellt, muss dem PeerService in Hinblick auf Sicherheitsaspekte besondere Beachtung geschenkt werden.

bearbeite lokalen Suchauftrag: Wenn ein Agent bei einem Peer ankommt, kommt es zur lokalen Bearbeitung des Suchauftrags bei dem entsprechenden Peer. Dies machen auch die beiden beteiligten Aktoren SearchAgent und PeerService deutlich.

Damit sind für den Suchauftrag die Komponenten und Aufgaben hinreichend beschrieben, so dass nun mit der Konkretisierung fortgefahren wird, die das Use Case „sammle Informationen“ noch erfahren muss. Dieser Use Case beschreibt das Verfahren, welches anzuwenden ist, wenn der Anwender Duplikate von Dateien anfordert.

administriere FileFetching: Dieses Use Case beschreibt die Administration, die der ManagementService und der FetchService zu erledigen haben. Nachdem der Anwender die Dateien ausgesucht hat, werden diese Informationen an den ManagementService weiter gegeben. Dieser wird dann einen FetchService, ähnlich dem SearchService, beauftragen, die notwendigen Aktionen zu tätigen, dass die entsprechenden Dateien gefunden werden. Somit wird der FetchService entsprechend FetchAgents mit der Akquisition beauftragen. Die Dateien, von denen die Agenten Duplikate besorgen konnten, werden abschließend an der FetchService weiter gegeben.

sammle Files: Vergleichbar mit dem SearchAgent ist hier das Verhalten des FetchAgents zu beschreiben, wie er Duplikate zu besorgen hat. Dabei ist sein Verhalten allerdings weniger komplex, als das des SearchAgents. Der FetchAgent soll nur zu einem einzigen Peer migrieren, um dort die Datei zu akquirieren und benötigt auch keine Kommunikationsschnittstelle zu andern Agenten. Allerdings benötigt er auch den PeerService als Kontakt zum Peer, um einen Zugriff auf dessen Dateisystem zu erhalten.

speichere Files: Das Use Case legt fest, wie die gefundenen Dateien im Verzeichnis des Anwenders abgelegt werden sollen. Dies geschieht durch den FetchService, der abschließend dem ManagementService noch mitzuteilen hat, welche Dateien erfolgreich beschafft werden konnten.

3.4 Szenarien

Wie bereits erläutert, müssen die definierten Use Cases im Folgenden detailliert beschrieben werden. Allerdings wird hier nur auf die *Best-Case* Situation eingegangen. Da dieses genügt um

die Funktionsweise der Applikation zu verstehen. Daneben beinhalten Szenarien nur essentielle Schritte, die zur beschriebenen Funktionalität in den einzelnen Use Cases notwendig sind.

Szenario zum Use Case „Konfiguriere DataMiner“

1. Gebe Verzeichnis-Pfad an, in welchem nach Informationen gesucht werden kann.
2. Gebe Verzeichnis-Pfad an, in welchem angeforderte Dateien gespeichert werden sollen.
3. Gebe die Anzahl der Peers an, die je ein Agent bei der Informationsbeschaffung aufzusuchen hat.
4. Gebe die Dateiformate an, die explizit für eine Informationssuche freigegeben wurden, nach MIME-Types sortiert.

Szenario zum Use Case „initiiere Informationssuche“

1. Wähle Medientyp aus, in welchem gesucht werden soll.
2. Wähle Suchmaschine aus, die diesen Medientyp unterstützt.
3. Formuliere Suchauftrag für Suchmaschine.
4. Füge der Suchanfrage ggf. noch weitere Suchspezifika hinzu (Wörter, Bilder, ...).
5. Starte Suchauftrag.

Szenario zum Use Case „administriere Informationssuche“

1. Stelle Auftragsinformationen zusammen.
2. Verwalte Suchauftrag.
3. Führe Suchauftrag aus.

Diese Sequenz muss in Folgenden weiter verfeinert werden, was in den anderen Use Cases geschieht.

Szenario zum Use Case „administriere Suchauftrag“

1. Leite einem auftragsbezogenem Administrationsdienst die Informationen weiter.
2. Stimme Suchauftrag mit gegebene Netzwerk ab.
3. Gebe Suchauftrag an Agenten weiter.
4. Gebe Agenten zur Suche frei.
5. Warte auf Resultate der Agenten.

Szenario zum Use Case „bearbeite verteilten Suchauftrag“

1. Warte auf Suchfreigabe durch administrativen Dienst.
2. Migriere zu Netzwerkteilnehmer.
3. Gebe gesammelte Ergebnisse an administrativen Dienst weiter.

Szenario zum Use Case „bearbeite lokalen Suchauftrag“

1. Prüfe Postfach auf eingegangene Post.
2. Starte Suchmaschine.
3. Verarbeite erhaltenes Ergebnisse.
4. Sende Nachricht zu anderen Agenten.
5. Migriere vom jetzigen Peer weg.

Szenarien zum Use Case „sammle Informationen " Dieses Use Case ist in seiner Aufgabe dem Use Case „administriere Informationssuche" ähnlich, so dass auch dieses in mehrere Use Cases unterteilt werden kann.
Szenario zum Use Case „administriere Filefetching"

1. Wähle Dateien aus.
2. Stelle Informationen zum Fetchauftrag zusammen.
3. Gebe Fetchauftrag an Agenten weiter.

Szenario zum Use Case „sammle Files"

1. Migriere zu Peer.
2. Fordere Duplikat eine Datei an.
3. Kehre zum Auftraggeber zurück.

Szenario zum Use Case „speichere Files"

1. Speichere Duplikate im Dateiverzeichnis.
2. Informiere über Auftragsergebnis.

3.5 Sequenzdiagramme

Die im Folgenden vorgestellten Sequenzdiagramme leiten sich aus den vorangehenden Use Cases und Szenarien ab. Allerdings wurde eine abschnittsweise Vorstellung der Sequenzdiagramme zu Gunsten einer besseren Übersichtlichkeit verzichtet. Die Diagramme zeigen eine *White Box*-Sichtweise auf das System, so dass weitere Informationen in den Diagrammen die Sicht auf das System erschweren würden. Des Weiteren wird in den Fällen, in denen ein Sequenzdiagramm mehrere Szenarien beinhaltet explizit darauf hingewiesen.
Innerhalb der einzelnen Sequenzdiagramme kommt es zu Wechselwirkungen mit dem Anwender, so dass dieser als *Aktor* mit einem eigenen Symbol in die Diagramme aufgenommen wurde.
Suche Informationen
Entsprechend den Use Cases und Szenarien, wird im Folgenden „suche Informationen" durch mehrere Sequenzdiagramme beschrieben. Das erste Sequenzdiagramm in Abbildung A.8 zeigt die ersten drei Schritte des Szenarios. Diese umfassen die Aktionen, die der Nutzer auf jeden Fall tätigen muss, um eine Suche zu initiieren. Die Suchmaschinen sind MIME-Types zugeordnet, so dass der Anwender in einem ersten Schritt die MIME-Type Gruppe, wie Text, Bild oder Audio, aussuchen muss, in der er nach Informationen suchen möchte. Daraufhin werden die Suchmaschinen angezeigt, die mit dem ausgewählten MIME-Type in Verbindung stehen. Die

GUI der ausgewählten Suchmaschine wird dann angezeigt. Im Sequenzdiagramm wird der generische Ansatz, der hierfür vorgesehen wurde, sichtbar. Denn die Methode *loadDynamicPanel* wird anhand der Variablen *searchType_* die Suchmaschine identifizieren können und die entsprechende GUI dem Fenster der Applikation zuordnen, in dem Suchaufträge formuliert werden. Konkret wird diese GUI durch eine Namenskonvention gefunden, da sich der Namen der dazugehörigen Klasse aus dem Namen der Suchmaschine und der Endung „GUI“ zusammen setzt. Entsprechend wird im Verlauf der Anwendung auch die Klasse zur Kontrolle der Suchmaschine gefunden werden.
Zur vollständigen Initiierung der Informationssuche werden in einem weiteren Sequenzdiagramm von Abbildung A.9 die restlichen Schritte aufgezeigt. Dabei wird im wesentlichen auf Grundlage des Suchreferenz nach weiteren bereits bekannten Referenzen gesucht und dem Anwender angeboten. Dieser kann aus diesen weitere Referenzen bestimmen, die den Suchauftrag zugefügt werden und diesen so präzisieren sollen. Weiter wird im Sequenzdiagramm deutlich, wie der ManagementService seine Kompetenz zu dem Auftrag an den SearchService abgibt. Dieser nimmt mit dem Aufruf der Methode *startSearchService* die weiteren administrativen Aufgaben wahr.
In dem Sequenzdiagramm *administriere Suchauftrag* in Abbildung A.10 kommen zwei Besonderheiten der Applikation vor. Zum einen ist dies der Einsatz der *FetchPeerPropertyAgents*. Dieser Agententyp wird eingesetzt, um Informationen von Peers zu beschaffen, die nur lokal beim Peer bekannt sind. Relevant ist dies für Kommunikationsports, die nicht über das *VICINITY*-Netzwerk zu erfragen sind. Da die Offenlegung der Kommunikationsports bei SeMoA-Peers auf freiwilliger Basis geschieht, besitzt der SearchService eine Funktion, die periodisch prüft, ob der *FetchPeerPropertyAgents* schon alle Kommunikationsports in Erfahrung gebracht hat. Konnten diese bis zu einem definierten Zeitpunkt nicht in Erfahrung gebracht werden, wird das von der Applikation registriert und auf eine Kommunikation der Agenten während der Suche verzichtet. Bei dem Einsatz des *FetchPeerPropertyAgents* handelt es sich um eine Zwischenlösung, die genutzt werden muss, bis das SeMoA das SHIP-Protokoll verwendet. Wenn der Agent mit dieser Portinformation von allen Peers zurückkommt, sind die Suchagenten in der Lage, während der Suche miteinander zu kommunizieren. Dafür müssen sie sich auch untereinander kennen lernen, was in der Methode *introduceAgents* geschieht.
Die zweite Besonderheit, die in diesem Sequenzdiagramm zum Ausdruck kommt, betrifft den Schutz der Applikation vor fremden Agenten. Denn die eigenen Agenten müssen sich nach ihrer Entstehung gegenüber dem SearchService erst positiv identifizieren und dann registrieren lassen. Dies wird in der Methode *register* durchgeführt.
Des Weiteren wird im Verlauf des Sequenzdiagramms jeder Agent mit den notwendigen Auftragsinformationen ausgestattet, die durch die Realisierung der einzelnen *set...*-Methoden nicht mehr von Dritten modifiziert werden können.
Das folgende Sequenzdiagramm „bearbeite Suchauftrag“ in Abbildung A.11 beschreibt die beiden Szenarien „bearbeite verteilten Suchauftrag“ und „bearbeite lokalen Suchauftrag“. Hierbei werden wieder Kommunikations- und Sicherheitsaspekte deutlich. Kommt ein Agent beim Peer an, prüft er mit der Methode *getGroupMessages*, ob andere Agenten Nachrichten an ihn gesendet haben. Er selbst kann nach der Ausführung der Suchmaschine ebenfalls Nachrichten mit der Methode *communicateLocalResults* an andere Agenten versenden.
Wie aus dem Diagramm hervor geht, nimmt die Nutzung der Suchmaschine nicht der Agent, sondern der PeerService vor. Dieser Dienst stellt die Verbindung zwischen dem Peer und dem Agenten da, so dass dieser in seinen Methoden auch immer zu prüfen hat, ob ein Agent über die notwendigen Rechte verfügt, eine Aktion am Peer ausführen zu lassen. Sollte dies nicht der Fall sein, verweigert der PeerService dem Agenten die angeforderte Dienstleistung. Das

Migrationsverhalten des SearchAgentss, während der Informationssuche wird in Abbildung 1.1 skizziert.
Abschließend wird das Use Case „sammle Informationen“ in drei Sequenzdiagrammen der Abbildungen A.12 bis A.14 beschrieben. Wie bei dem Zusammenwirken zwischen Service und Agenten beim Suchauftrag, so wird auch beim Duplizieren von Dateien verhindert, dass fremde Agenten Dateien an den *FetchService* liefern können. In Abbildung A.13 wird hierzu das gleiche Verfahren gezeigt, wie in Abbildung A.10 um Agenten gegenüber dem Service zu registrieren. Das Migrationsverhalten der FetchAgenten, welches sich vom Verhalten der SearchAgents unterscheidet, wird in Abbildung 1.2 beschrieben.

3.6 Klassendiagramme

Aus den vorgestellten Sequenzdiagrammen kann nun ein Klassendiagramm gebildet werden, welches die Grundlage der Implementierung darstellt. Dabei lässt sich das Klassendiagramm in verschiedene Bereiche untergliedern. Die darin aufgeführten Klassen erfüllen einen bestimmten Teil der Gesamtleistung des Systems, in dem sie als Einheit eine bestimmte Funktion des Systems erfüllen. In Abbildung A.15 wird die Gesamtdarstellung des Klassendiagramms gezeigt. Abweichend von anderen UML-Klassendiagrammen wird auf eine Darstellung der Klassenvariablen verzichtet, sondern nur deren Methoden aufgezeigt. Um die Orientierung im Klassendiagramm zu vereinfachen, zeigt die nachfolgende Abbildung A.16 ein abstrahiertes Klassendiagramm, dass nur die Funktionsbereiche und die hierarchischen Verbindungen zwischen den Klassen beinhaltet. Dieses entspricht ebenfalls nicht der UML, erfüllt aber den Zweck einer übersichtlichen Darstellung. Die Zuordnung einzelner Klassen des Diagramms zu den Funktionsbereichen zeigen die Abbildungen im Abschluß des Anhangs A.

Kapitel 4

Besprechung der Lösung

4.1 Einführung

Im Kapitel 3 wurde die Systementwicklung der Applikation, die im Rahmen der Diplomarbeit entwickelt wurde, umfassend vorgestellt. Ziel dieses Kapitel ist es, diese Realisierung zu bewerten. Dabei ist zu klären, wie gut einzelne Aspekte berücksichtigt wurden und welche Nachteile mit der entsprechenden Lösung auftreten. Neben diesen Bereich der Applikationsevaluierung, werden in diesem Kapitel auch Informationen zur Performance und in diesem Zusammenhang auch mit der Effizienz der Applikation gegeben. Allerdings können diese nur theoretisch vorgestellt werden und nicht durch entsprechende Versuche gestützt werden. Dafür konnte während der Diplomarbeit keine entsprechende Zeit zur Verfügung gestellt werden.

4.2 Technologischer Ansatz

Die Realisierung des Lösungsansatzes orientiert sich technologisch an einer Kombination aus Peer-to-Peer Netzen und mobilen Agenten der Agentenplattform SeMoA. Dies vor dem Hintergrund direkt auf die Informationsquellen der Anwender zugreifen zu können und ein möglichst effizientes Suchverfahren zu ermöglichen. Durch die Wahl von SeMoA als Agentenplattform, wurde es möglich Agenten einzusetzen, die migrieren können und über eine Kommunikationsschnittstelle verfügen, so dass diese sich untereinander während der Bearbeitung eines Suchauftrags koordinieren können. Zusätzlich haben mobile Agenten den Vorteil, dass sie für ihre Migration nur kurzzeitig eine Verbindung zwischen den beteiligten Peers benötigen und auf keine permanente Verbindung angewiesen sind. Daneben sind Agenten durch ihr pro- und reaktives Verhalten in der Lage ihre Migrationstrategie zu ändern, wenn dies die Situation erfordert. Ist ein Peer nicht erreichbar, kann der Agent entscheiden, ob er seinen Auftrag bei anderen Peers fortsetzt und zwischenzeitlich immer wieder versucht den blockierten Peer zu erreichen. Er kann aber auch versuchen über andere Peers zum Ziel-Peer zu gelangen und schließlich kann er auf seinem jetzigen Peer verbleiben und warten, bis die Netzwerk-Situation für ihn vorteilhafter ist. Agenten passen mit ihrem Verhalten somit gut zur Dynamik in Peer-to-Peer Netzen. Allerdings benötigen Agenten für die letzte Migration eine Verbindung zu ihrem Heimat-Peer. Existiert diese nicht, kann der Agent nicht zurückkehren. Das hat zur Folge, dass nur schwer prognostiziert werden kann, ob und wann ein Agent seinen Auftrag beenden kann. Neben dem Verhalten im Netzwerk, bieten Agenten noch andere Merkmale, die nützlich für die Applikation sind. Der Aufbau eines Agenten, wie er im Grundlagen-Kapitel beschrieben wurde, ist modular. Somit

kann ein spezifischer Agenten durch austauschen eines Moduls seine Fähigkeiten ändern. Dies ist für den geforderten generischen Aspekt der Applikation wichtig, da so verschiedene Suchmaschinen und verschiedene Datenformate dem Agenten zugänglich gemacht werden können.
Neben den Merkmalen der Agenten besitzt SeMoA noch Möglichkeit einzelne Peer zu einem Netzwerk zusammenzufassen. Insgesamt präsentiert sich SeMoA als homogenes System, das die beide Technologien mobile Agenten und Peer-to-Peer aus eine Hand anbietet, und so eine geeignete Grundlage für die entwickelte Applikation darstellt.
Wie in Kapitel 2 beschrieben kann mit SeMoA ein reines Peer-to-Peer Netz aufgebaut werden. Allerdings ist diese Netzwerkstruktur variabel, denn das Migrationsverhalten der Agenten kennzeichnet letztlich die Netzwerkstruktur. Wenn sich durch die Migration exponierte Peers herausbilden, die spezielle Aufgaben, wie die eines Brokers, übernehmen, kann das dezentrale Netz als zentrales Netz aufgefasst werden.
Doch sind mit dem Funktionsumfang von SeMoA auch einige Nachteile verbunden, die trotz einer guten Dokumentation der Software bestehen. So ist die Parametrisierung und Personalisierung des Systems nur mit entsprechendem Fachwissen möglich. Sonst können bspw. die Sicherheitsmechanismen, wie Rechtevergabe der Agenten und Kontrolle dieser nicht korrekt angewandt werden. Dadurch kann es in einem anscheinend sicherem System zu Schwachstellen kommen. Um aber Agenten mit den notwendigen Befugnissen auf dem eigenen Peer auszustatten, damit diese ihren Auftrag erledigen können, ist administrativer Aufwand notwendig.
Nachteilig für die Bereitstellung der Kommunikationsschnittstelle der Agenten hat sich ausgewirkt, dass bei den Peer die notwendigen Ports zur Kommunikation sind nicht genauso problemlos von den einzelnen Netzwerkteilnehmern zu erfragen sind, wie die Ports zur Migration. Momentan muss noch ein Agent zuerst jeden Teilnehmer aufsuchen und dort die veröffentlichte Kommunikationsport-Nummer in Erfahrung bringen. Die so ermittelten Portnummern werden dann anderen Agenten mitgeteilt, damit sie in der Lage sind, sich zu unterhalten.
Für die Applikation ist von Bedeutung, dass die Vorteile, die mit dem Einsatz eines Peer-to-Peer Netzes verbunden sind, durch den Einsatz von SeMoA erhalten bleiben. Dies sind neben einer Robustheit des Netzes durch fehlertolerantes Verhalten, auch eine gleichmäßige Lastverteilung im reinen Peer-to-Peer Netz. Dieses kann SeMoA genauso einhalten, wie die Tatsache, dass Teilnehmer ihre administrativen Rechte eigenverantwortlich wahrnehmen können. So kann der Teilnehmer ohne Einschränkung Dateiformate und Verzeichnisse für die Informationssuche freigeben oder sperren, was auch als Funktionsmerkmal von der Applikation der Diplomarbeit gefordert wird. Daneben kann der Anwender auch Agenten, in Abhängigkeit ihrer Besitzer, individuell Privilegien oder Rechte auf seinem Rechner zuteilen oder verwehren.
Doch ist weder durch die dezentrale Topologie des SeMoA-Netzwerkes, noch durch den Einsatz der mobilen Agenten das konzeptionelle Problem behoben, dass in der Regel nur ein Teil der Resultate ermittelt wird, die im Netz tatsächlich zur Verfügung stehen. Ein umfassendes Ergebnis kann nur zufällig erzielt werden oder wenn die zu untersuchende Datenmenge und/oder die Anzahl der Netzwerkteilnehmer hinreichend klein ist.
Bei der Bewertung des technologischen Ansatzes muss auch die Implementierungssprache berücksichtigt werden. Die Agentenplattform SeMoA ist, wie deren mobile Agenten, in Java implementiert. Somit ist von Seiten der Sprache für eine Systemunabhängigkeit gesorgt, die den Einsatz der Applikation in verschiedene Betriebssysteme ermöglicht. Dabei muss beachtet werden, dass die Programmierung der GUI-Komponenten nicht nur aufwendig ist, sondern auch nicht 1:1-Kompatibel zwischen verschiedenen Betriebssystemen, wie Windows oder UNIX ist. Daneben ist Java aber eine verbreitete Programmiersprache, so dass vielen Interessenten der Zugang zu SeMoA leichter ist. Daran ist auch die Dokumentationsmöglichkeit gebunden, die auch SeMoA nutzt, was neben einer schnelleren Orientierung in der Funktionsweise des Programms

auch die Weiterentwicklung dessen unterstützt.
Für die Anwendung der Diplomarbeit sind aber zwei Aspekte von Java von besonderer Bedeutung. Java kann Programme, die in anderen Sprachen, wie bspw. C oder C++, codiert wurden, über *Native Interfaces* einzubinden. Dies stellt eine Erweitung des Spektrums an Suchmaschinen dar, auf die die Applikation zurückgreifen kann. Die Anforderung an das Programm, verschiedene spezialisierte Suchmaschinen einzusetzen, kann u.a. aus diesem Grund berücksichtigt werden. Der zweite Aspekt umfasst die Sicherheit in Peer-to-Peer Netzen für die Verbindungen, den einzelnen Peers und den mobilen Agenten. Dieser Sicherheitsaspekt ist für die Akzeptanz der Applikation von großer Bedeutung, da sichergestellt werden muss, dass vertrauliche und persönliche Daten permanent davor geschützt werden, anderen Personen, als den beteiligten, zugänglich zu sein. Die Sicherheitsmerkmale von SeMoA, die auch auf Java basieren, bieten diesen notwendigen Schutz und ermöglichen so, dass die Applikation ein sicheres Peer-to-Peer Netz im Internet aufbauen könnte.

4.3 Systemarchitektur

Durch die eingesetzte Technologie, wie sie im vorherigen Abschnitt besprochen wurde, konnten die Anforderungen an das System, wie:

- die Organisation des System als Peer-to-Peer Netz
- die explizite Medienfreigabe zur Informationssuche durch den Anwender
- der Einsatz von Agenten, die über Migrations- und Kommunikationsmerkmale verfügen

erfüllt bzw. die Grundlage dafür gebildet werden. Die Berücksichtigung der anderen Anforderungen erfolgt durch die Systemarchitektur, worauf im Folgenden eingegangen wird.
Eine wichtige Eigenschaft für die Applikation ist, dass verschiedene Suchmaschinen eingesetzt werden können, um unterschiedliche Datenformate nach Informationen zu untersuchen. Dies wird in der Systemarchitektur durch den Einsatz von Schnittstellenklassen in Verbindung mit Kontrollklassen ermöglicht. Damit eine Suchmaschine richtig arbeiten kann, benötigt sie eine korrekte Parametrisierung, eine Datenaufbereitung, eine Ablaufsteuerung und eine Ergebnisauswertung, die jeweils speziell auf eine Suchmaschine angepasst werden müssen. Diese Spezialisierung wird durch zwei Kontrollklassen ermöglicht, die jede Suchmaschine benötigt. Während eine Klasse die GUI für die Suchmaschine bereit stellt, wird von der anderen die Ausführung der Suchmaschine zusammen mit der Vor- und Nachbereitung der Daten bereitgestellt. Innerhalb der Applikation wird auf diese Kontrollklassen über Schnittstellenklassen zugegriffen. Diese Klassen besitzen eine allgemeine Beschreibung der notwendigen Methoden, welche durch die Kontrollklassen konkretisiert werden. Der Vorteil dieser Verfahrensweise liegt in der Trennung zwischen dem Programmcode der Applikation und dem der Suchmaschine. Demnach muss der Programmcode der Applikation nicht manipuliert werden, um eine weitere Suchmaschine für die Applikation nutzbar zu machen. Es genügt die erwähnte Konkretisierung der Schnittstellenklassen in einer Kontrollklasse. Durch die Nutzung der Schnittstellenklassen können Agenten beliebige Suchmaschinen nutzen. Da Agenten diese auch selbst mitbringen, ist es nicht notwendig, die Suchmaschinen im Peer-to-Peer Netz zu verbreiten. Es ist lediglich notwendig, dass die Zuordnung der Dateiformate zu den MIME-Types im Netz identisch ist. Allerdings birgt dieser Ansatz auch Nachteile. So kann durch den generischen Ansatz keine allgemeine Vorverarbeitung der Daten am Peer vorgenommen werden, da dies Teil der Merkmale der Suchmaschine ist. Dadurch muss immer eine spezifische Datenaufbereitung *on demand* am Peer durchgeführt

werden, wodurch die Informationssuche im Netz wesentlich verzögert wird. Daneben ist es nachteilig für den Agenten, dass er immer seine Suchmaschine mitbringen muss und so sein Volumen vergrößert wird. Dies wirkt sich nachteilig auf die Migration aus.
Um dem Sicherheitsanspruch von SeMoA gerecht zu werden, wurden auch Sicherheitsmerkmale bei der Systemarchitektur berücksichtigt. Jeder Anwender kann in Abhängigkeit des Besitzers eines Agenten festlegen, welche Rechte dessen Agenten auf seinem Rechner besitzen. Dies kann variieren zwischen dem Entziehen einzelner oder sämtlicher Rechte, die für die Ausführung des Suchauftrags notwendig sind; beispielsweise kann Agenten nicht gestattet werden ihre Suchmaschine zu nutzen oder Dateien zu kopieren. Die so definierten Rechte werden durch die Applikation abgefragt und entsprechend berücksichtigt. Mit den bisherigen Erläuterungen zur Systemarchitektur wurde aufgezeigt, dass das System die geforderten Anforderungen wie

- prinzipiell uneingeschränkte Nutzung verschiedener Dateiformate, die nur durch den Anwender eingeschränkt werden kann
- Rechtevergabe an Agenten durch den Anwender
- die Nutzung verschiedener Suchmaschinen durch Agenten
- die Bewertung der Suchergebnisse nach eigenen Methoden, die der Agent mitbringt
- den Einsatz der Suchmaschinen bei jedem Peer vor Ort

erfüllt. Maßnahmen, die getroffen wurden um zu verhindern, dass unbefugte Agenten die Suche oder Ergebnisse manipulieren können werden im Folgenden beschrieben. Verhindert wird dies durch eine Registrierung der eigenen Agenten in der Applikation. Agenten bekommen bei ihrer Erschaffung eine individuelle Zeichenfolge, die nur ihnen und der Applikation bekannt ist. Nach ihrer Erschaffung wird diese Zeichnfolge zwischen beiden verglichen und bei positiver Übereinstimmung wird der Name des Agenten registriert. Der Name des Agenten ist ein individuelles Merkmal, durch das dieser sich im Folgenden gegenüber der Applikation identifizieren muss. Die Zeichenfolge wird danach bedeutungslos. Daneben verhindern Realisierungen von Methoden, dass Agenten während ihrer Auftragsabwicklung andere Auftragsparameter oder -daten bekommen. Damit ein Anwender keine Dateien bekommt, die er nicht angefordert hat, werden zu den gefundenen Dateien jeweils so genannte Hashwerte, basierend auf den SHA-Algorithmus, errechnet. Wenn später eine Datei angefordert wird, wird dieser Wert dem entsprechenden Agenten mitgegeben. Wenn er, ausgehend von der bekannten Pfadangabe der Datei, die gesuchte Datei gefunden hat, wird von dieser erneut der Hashwert bestimmt. Nur wenn dieser Wert dann mit dem vorherigen übereinstimmt, wird die Datei dupliziert und vom Agenten zur Verfügung gestellt. Neben diesen Sicherheiten bestehen weiter die Sicherheitsmerkmale von SeMoA, wie bspw. eine Verifikation des Bytecodes der Agenten.
Für die Abwicklung eines Auftrags ist gefordert, dass dieser von mehreren Agenten erledigt wird, die kooperativ miteinander arbeiten. In der Applikation wird dies durch zwei Merkmale berücksichtigt. Zum einen wird jedem Agenten nur einen Teil der verfügbaren Peers zugewiesen, so dass jeder nur ein Teil zum Gesamtergebnis beiträgt. Daneben soll die Kommunikationsschnittstelle genutzt werden, um Informationen untereinander auszutauschen. Dies wurde zwar prinzipiell ermöglicht. Allerdings ist die implementierte Lösung nicht effizient, wie in Abschnitt 4.4. noch gezeigt wird, so dass die Kommunikation nur bedingt dazu genutzt werden kann auftrags-unterstützende Informationen unter den beteiligten Agenten zu verbreiten. Damit der Anwender Suchaufträge präziser formulieren kann, wurden den Agenten Methoden zur Verfügung gestellt, die aus gefundenen Dateien entsprechende Informationen ermitteln. Realisiert wurde dies für Text-Dateien, in denen nach Schlüsselwörtern gesucht wird. Mittels der

angesprochenen Methode sind Agenten in der Lage in den gefundenen Dateien weitere Schlüsselwörter zu finden. Da diese aber nicht immer in einem gewünschten Zusammenhang mit dem Schlüsselwort stehen, können die gefundenen Wörter bspw. nicht während der Suche unter den Agenten kommuniziert werden, um direkt die Suche zu verbessern. In der aktuellen Realisierung der Applikation werden die extrahierten Wörter gesammelt und dem Anwender am Ende präsentiert. Dieser kann dann festlegen, welche Wörter in direkter Verbindung mit dem Schlüsselwort stehen. Wenn nachfolgend ein Suchauftrag mit dem gleichen Schlüsselwort formuliert wird, werden die vorher ausgewählten Wörter dem Anwender zur Aufnahme in den Auftrag angeboten. Gemäß den Anforderungen der Applikation muss es dem Anwender möglich sein, mehrere Suchaufträge parallel zu initiieren. In der Systemarchitektur wird dies durch zwei Managementklassen berücksichtigt, die hierarchisch voneinander abhängen. Die untergeordnete Klasse übernimmt hierbei das Management für einen konkreten Suchauftrag, während der übergeordnete die untergeordneten Klassen managet und die Ergebnisse der einzelnen Suchaufträge sammelt. Im Programmablauf kümmert sich die übergeordnete Klasse mit um einen Suchauftrag bis dieser vollständig formuliert ist und mit dem Anwender nicht mehr in Kontakt getreten werden muss. Liegt der vollständige Suchauftrag vor, übernimmt die weitere Initiierung des Auftrags die untergeordnete Klasse. Die übergeordnete prüft nur noch frequentiell, ob zu einem Auftrag schon Ergebnisse vorliegen. Diese Managementklasse beendet die untergeordnete nach einer definierten Zeitspanne oder sobald alle Agenten zu einem Auftrag wieder beim Peer eingetroffen sind. Damit eine Applikation auch korrekt genutzt werden kann, ist es notwendig, diese mit einer geeigneten GUI auszustatten. Die GUI der Applikation setzt sich aus mehreren Fenstern zusammen, die nur sichtbar bzw. zur Verfügung stehen, wenn dies im Programmablauf notwendig ist. Auf diese Weise soll eine übersichtliche Informationseingabe und -präsentation ermöglicht werden. Unterstützt wird diese Aspekt auch durch das Layout der Fenster, das bei allen möglichst gleich ist. So soll der Anwender gleiche Funktionen immer an der selben Position in verschiedenen Fenstern wiederfinden. Um fehlerhafte Eingaben zu verhindern, werden Felder im Parametrisierungsfenster mit Default-Daten angezeigt und Bedien- sowie Parametrisierungselemente bei der Auftragserstellung sukzessiv freigeschaltet, anschließend gesperrt und dynamisch aufgebaut. Nachteilig für die GUI wirkt sich allerdings aus, dass diese nicht systemunabhängig ist. So kann bspw. die Platzierung von Fenstern unter UNIX nicht korrekt wiedergegeben werden, wie dies unter Windows möglich ist. Somit kann die GUI unübersichtlich und unkomfortabel für Anwender werden. Obwohl auch auf eine funktionale Platzierung einzelner Bedienelemente geachtet wurde, kann die Bedienung dem Anwender teilweise nicht intuitiv zugänglich sein. Bezüglich dem Komfort weist auch die Konfiguration der Applikation Mängel auf. So entspricht die Festlegung von Pfadangaben nicht dem heutigen Standard und ist unnötig aufwendig.

4.4 Effizienz

Wie eingangs erwähnt, sind die hier vorgeschlagenen Performance-Tests nicht durch eigene Versuche fundiert. Dennoch können die vorgestellten Verfahren und Überlegungen als Indikatoren für die Leistungsfähigkeit der Anwendung dienen. Dabei wird jeweils zu Grunde gelegt, dass sich alle Teilnehmer im Netzwerk mit Daten einbringen und es zu keinem Ungleichgewicht zwischen Informationsanbieter und -sucher gibt. Denn nach Analysen von Adar und Huberman [53] von Xerox PARC aus dem Jahr 2000 stellen über 70% der Teilnehmer von Gnutella keine Informationen zur Verfügung, sondern suchen nur nach diesen. Dies hat neben einem eingeschränkten Informationsangebot zur Folge, dass es zu einer ungleichmäßigen Lastverteilung

im Netzdatenverkehr zu Ungunsten der Informationsanbieter kommt. Was letztlich die tatsächliche Performance des Netzwerks verzerrt und falsch wiedergibt.
Wie schon angesprochen, ist die Kommunikation zwischen mobilen Agenten ein nicht-triviales Problem. Dabei exisieren momentan zwei Probleme. Zum einen existiert ein Lokalisierungsproblem der Agenten im Netzwerk, d.h. es muss ermittelt werden, auf welchem Peer sich ein bestimmter Agent momentan befindet. Zum anderen wird der Kommunikationsport des Peers nicht publiziert, so dass er erst mittels eines speziellen Agenten abgefragt werden muss.
Eine mögliche Lösung, um einem Agenten trotzdem Nachrichten zukommen zu lassen, ist das Verteilen der Nachricht für den Agenten an alle Peers, die dieser Agent besucht. Einzig die Peers, die der sendende Agent selbst besucht, erhalten keine Nachricht von ihm. Denn jede Peergruppe, die ein Agent zu besuchen hat, ist von den Gruppen anderer Agenten disjunkt. Die folgende Tabelle zeigt beispielhaft das damit verbundene hohe Nachrichtenaufkommen.

$$Nachrichtenaufkommen = Agenten_{Anz} \cdot ((Agenten_{Anz} - 1) \cdot Peergruppe) \quad (4.1)$$

Anzahl Peers	Anzahl Agenten	Peergruppe pro Agent	Nachrichtenaufkommen
20	5	4	80
40	5	8	160
60	5	12	240

Tabelle 4.1: Nachrichtenaufkommen (Agentenanzahl konstant)

In der Tabelle 4.1 wurde die Anzahl der Agenten konstant gehalten, so dass mit steigender Anzahl der Peers jeder Agent auch mehr Peers besuchen muss. Die konstante Anzahl der Agenten sorgt dafür, dass sich die Anzahl der Nachrichten verdoppelt, obwohl zu einer Peergruppe eines Agenten nur vier Peers additiv hinzukommen. Anhand dieses Beispiels wird die Ineffizienz dieses Lösungsansatzes deutlich.
In SeMoA kann der Aufenthaltsort der sich dynamisch bewegenden Agenten mittels ATLAS [67] effizient bestimmt werden. Doch momentan erlaubt SeMoA keine Abfrage des Kommunikationsports der entsprechenden Plattform. Abhilfe soll hier das SHIP-Protokoll schaffen, welches aktuell in SeMoA implementiert wird. SHIP erlaubt es, über einen Peer beliebige Informationen, wie Kommunikations- und Migrationsports, unterstützte Protokolle, physischer Standort des Peers, etc. zu erfragen. Daneben wird in SeMoA möglich sein, SHIP transparent zu nutzen, und eine Nachricht direkt an einen Agenten zu senden. Eine zukünftige Version dieser Applikation wird diesen Mechanismus nutzen. Damit kann die Effizienz beim Nachrichtentransport gegenüber der skizzierten Lösung signifikant gesteigert werden, denn das „Gießkannenprinzip“ und die damit verbundenen Nachrichtenschwemme, wie sie Tabelle 4.1 zeigt, wird unterbunden.
Im Zusammenhang mit dem Nachrichtenaufkommen muss auch die Erreichbarkeit einer Nachricht untersucht werden. Die Nachricht kann einen Agenten erreichen, wenn sich dieser bei einem Peer aufhält. Nur während der Migration erreicht die Nachricht den Agenten nicht. Nach Angabe von Roth [65] beträgt die Zeit, die ein Agent für die Migration benötigt, weniger als eine Sekunde. Diese Zeitspanne beginnt mit dem Stoppen der Aktivität des Agenten am aktuellen Peer und endet mit dem Starten der Aktivität des Agenten am neuen Aufenthaltsort. Wenn nun eine Aufenthaltsdauer der Suchagenten bei einem Peer von 20 Sekunden angenommen wird, um dort nach Informationen zu suchen, ergibt sich folgende Erreichbarkeit des Agenten:

$$Erreichbarkeit_{allgemein} = \frac{Aufenthaltsdauer}{Aufenthaltsdauer + Migrationsdauer} \quad (4.2)$$

$$Erreichbarkeit = \frac{20sec}{20sec + 1sec} = 95,24\% \tag{4.3}$$

Da für diese Berechnung von einer *worst case* Situation bei der Migration ausgegangen wird, ist die Erreichbarkeit von 95,24% als schlechteste Wahrscheinlichkeit anzusehen.
Konträr zur Performancebelastung durch das Nachrichtenaufkommen und vielen Agenten steht die Dauer der Informationsbeschaffung. Diese verringert sich durch eine größere Anzahl an Agenten, die parallel einen Auftrag bearbeiten. Im aktuellem Programm kann definiert werden, wie viele Peers ein Agent maximal zu untersuchen hat. Somit richtet sich die Anzahl der Agenten nach den Teilnehmern im Netzwerk. Immer wenn der definierte Schwellwert (Anzahl der Peers) überschritten wird, erhöht sich inkrementell die Anzahl der Agenten. Dieses Verhalten entspricht einer Treppenfunktion. Alternativ hierzu könnte auch eine exponentielle Zuordnung erfolgen, um in kleinen Netzwerk mit wenigen Agenten zu operieren und mit steigender Anzahl der Netzwerkteilnehmer mehr Agenten einzusetzen. Demnach haben in einem wachsenden Netz die Agenten immer weniger Peers zu untersuchen. Dies hat zur Folge, dass sich der Aufwand eines Agenten verringert, allerdings die Belastung der Netzinfrastruktur zunimmt. Abschließend ist noch eine logarithmische Zuordnung zwischen der Anzahl der Agenten und den Peers, die ein Agent zu durchsuchen hat, möglich. Damit kann die Anzahl der Agenten in großen Netzen beschränkt werden, die dann eine immer größere Anzahl an Peers zu untersuchen haben. Somit steigt der Aufwand der Agenten und sinkt die Belastung der Netzwerkinfrastruktur. Bei den Tests mit diesen drei Varianten wird wohl letztlich eine hybride Lösung das Optimum darstellen, die ab einer gewissen Anzahl an Peers einen Strategiewechsel in der Zuordnung zwischen Peers und Agenten vornimmt. Es darf allerdings nicht außer Acht gelassen werden, dass die Dauer der Informationsbeschaffung auch wesentlich von dem Datenvolumen beim jeweiligen Peer und der eingesetzten Suchmaschine abhängt. Gerade letzter Punkt sollte zu einer Justierung der eingesetzten Strategie mit berücksichtigt werden. Neben der Anzahl der beteiligten Agenten ist die Aufbereitung der Daten am Peer zur Informationssuche zeitaufwendig und verlängert die Dauer der Informationsbeschaffung. In Abhängigkeit einer permanenten Indexeditierung, eine periodischen Indexeditierung und eine Indexeditierung *on demand*. Die Erstellung eines Index ist notwendig für die Funktion einer Suchmaschine. Allerdings fehlt in der aktuellen Realisierung der Applikation eine Kontrolle, ob es notwendig ist ein Index erstellt. Eine Indexerstellung ist nicht notwendig, wenn sich der Datenbestand seit der letzten Suchanfrage nicht verändert hat. So kann es passieren, dass eine noch gültiger Index verworfen wird, statt mit diesem zu operieren. Diese Ineffizienz kann durch einen lokalen Dienst beseitigt werden, der auf jedem Peer operiert und periodisch den Index aktualisiert.

Kapitel 5

Benutzerführung

In diesem Kapitel wird das entwickelte Programm *Data Miner* aus Sicht von Anwendern vorgestellt.
Bevor das Programm erstmalig gestartet werden kann, muss der Benutzer in der Datei `semoa.policy` Rechte eintragen, damit Agenten auch die notwendigen Rechte zugewiesen bzw. entzogen werden können. Die zu ändernden Rechte sind in dem Block `p2pSEARCH` zusammengefasst. Das Programm DataMiner benötigt zusätzliche Dateien, die Konfigurationseinstellungen beinhalten. Diese sind im Ordner `DataMinerConfigFiles` im Verzeichnis `semoa/etc` hinterlegt. Eintragungen in der Datei `semoa.etc` sollten nicht verändert werden, wenn sie sich auf diesen Ordner beziehen. Andernfalls wäre eine Änderung dieses Ordners mit weiteren Korrekturen im Quellcode nötig. Die anderen Eintragungen der `Peer2PeerPermission` für `copyFile` und `useSearchEngine` können dagegen beliebig verändert werden. Dabei ist für die Permission `copyFile` der Pfad einzutragen, in dem Dateien von anderen Peers gespeichert werden können, während für die Permission `useSearchEngine` das Verzeichnis einzutragen ist, in dem andere Agenten nach Informationen suchen können.
Nach dem Start der SeMoA-Plattform kann das Programm mit dem Kommando `javaDataMiner.User` gestartet werden und das Startfenster, wie es in Abbildung 5.1 gezeigt wird, erscheint.

Abbildung 5.1: Startfenster

Wenn das Programm erstmalig aufgerufen wird, muss der Anwender zuerst das Programm konfigurieren. Nach Anwahl des entsprechenden Menüs erscheint die in Abbildung 5.2 gezeigte Eingabemaske.

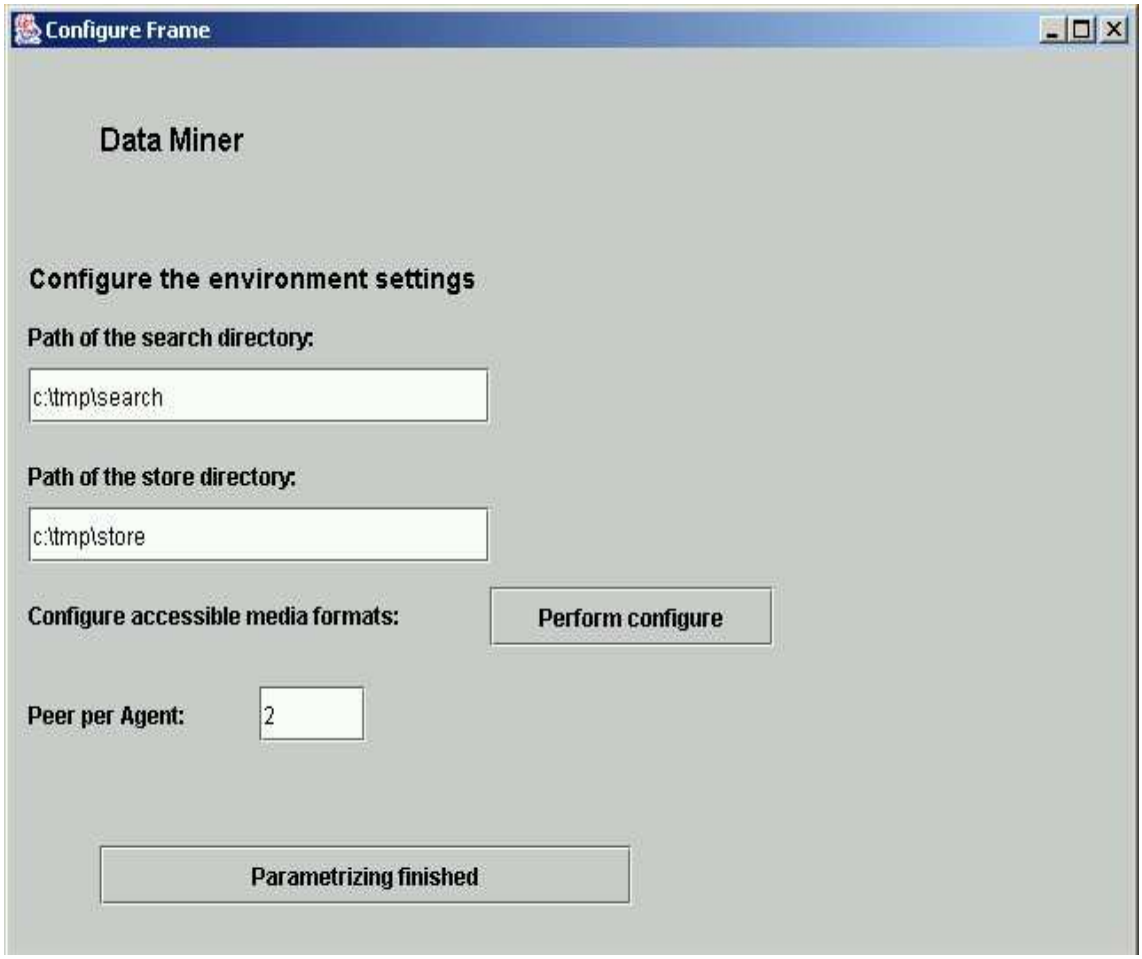

Abbildung 5.2: Parametrisierungsfenster

Der *Store-Pfad* kennzeichnet das Verzeichnis, in das Dateien abgelegt werden sollen, die Agenten dem Anwender aus dem Peer-to-Peer Netzwerk besorgt haben. Mit dem *Search-Pfad* macht der Anwender kenntlich, ab welchem Verzeichnis Ordner rekursiv nach Informationen durchsucht werden können. Im dritten Eingabefeld kann der Anwender festlegen, wie viele Peers ein Agent zu besuchen hat, um dort nach Informationen zu suchen. Standardmäßig besucht ein Agent zwei Peers.

Des Weiteren hat der Anwender die Möglichkeit im Konfigurationsfenster, wie in Abbildung 5.3 gezeigt, Medienformate auszuwählen, die explizit von Agenten zur Informationssuche genutzt werden können. Alle anderen Medientypen, die nicht ausgewählt wurden, werden dann ausgeschlossen. Nach dem Konfigurieren des Programms kann mit der Informationssuche begonnen werden.

Das Programm unterstützt verschiedene Suchmaschinen, die je einem Medientypen zugeordnet sind. Der Anwender kann nun anhand der Medientypen eine Vorauswahl an Suchmaschinen treffen und die gewünschte dann aus der Gruppe der Suchmaschinen aussuchen, die dem Medientype zugeordnet sind.

Abbildung 5.4 zeigt eine mögliche Auswahl einer Suchmaschine und die Integration der spezifischen Eingabemaske der Suchmaschine in dem Suchfenster. In diesem Fenster wird der An-

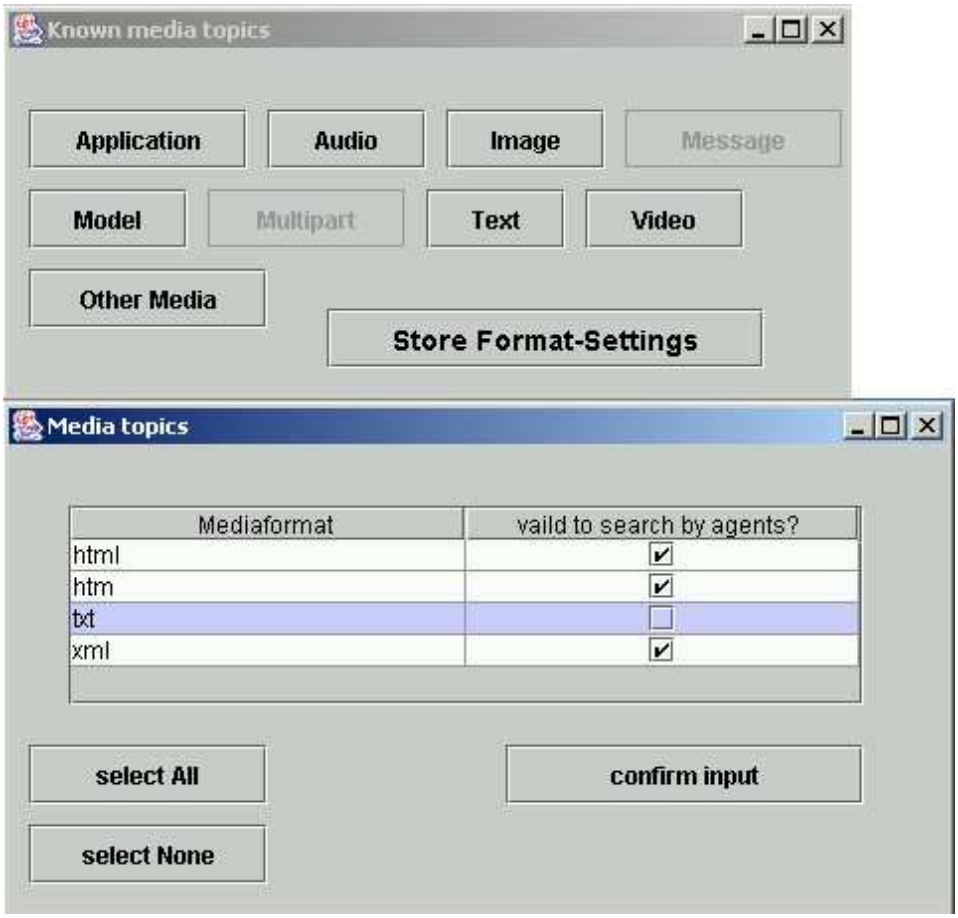

Abbildung 5.3: MIME-Typen Freigabe

wender die ausgewählte Suchmaschine parametrisieren. Je nach Suchmaschinentype können anschließend noch zusätzliche Schlüsselwörter, vergleichende Bilder oder andere Informationsträger dem Suchauftrag zugefügt werden, um so eine genauere Suche zu ermöglichen. Anschließend werden die benötigten Agenten erzeugt und mit der Suche beauftragt.

Über den Suchverlauf kann sich der Anwender über den Menüknopf `CheckResults` informieren. Dieser zeigt ein Fenster an, das Angaben über die laufenden Suchaufträge, die Anzahl der beteiligten Agenten und schon wiedergekehrten Agenten beinhaltet. In Abbildung 5.5 wird ein entsprechendes Fenster dargestellt.

Der Anwender kann nun entweder einen weiteren Suchauftrag initiieren, eine Chat-Session zu einem anderen Anwender starten oder auf das Resultat des erteilten Suchauftrags warten. Dieses wird in Form eine Liste angezeigt.

Da die Agenten bisher nur Informationen über die gefundenen Dateien, nicht aber selbige gesammelt haben, muss der Anwender nun die Dateien auswählen, von denen er ein Duplikat besitzen möchte. Nach einer entsprechenden Auswahl, werden weiter Agenten versuchen diese Dateien zu beschaffen. Des Weiteren kann sich der Anwender, über das Fenster mit dem Ergebnis des Suchauftrags, Schlüsselwörter anzeigen lassen, die die Suchmaschine in den Files gefunden hat. Diese werden ihm in einem separaten Fenster angezeigt. Abbildung 5.7 zeigt Schlüsselwörter an, die bei der aus vorhandenen Dateien zur Suchanfrage „Ant Colony“ gefun-

Abbildung 5.4: Suchauftragskonfiguration

Abbildung 5.5: Suchauftragsübersicht

den wurden. Mittels Button in der Spalte „Choose“ können gezielt Schlüsselwörter aus der Liste ausgewählt und übernommen werden. Weiterhin gibt es aber auch die Möglichkeit Schlüsselwörter aus mehreren zusammenzusetzen und zu editieren. Dafür müssen die Schlüsselwörter aus Menüknöpfe der Spalte „Combine“ ausgewählt werden. Alle editierten und übernommenen Schlüsselwörter werden schließlich mit „Copy selected Keywords“ übernommen bzw. mit „Cancle“ verworfen.

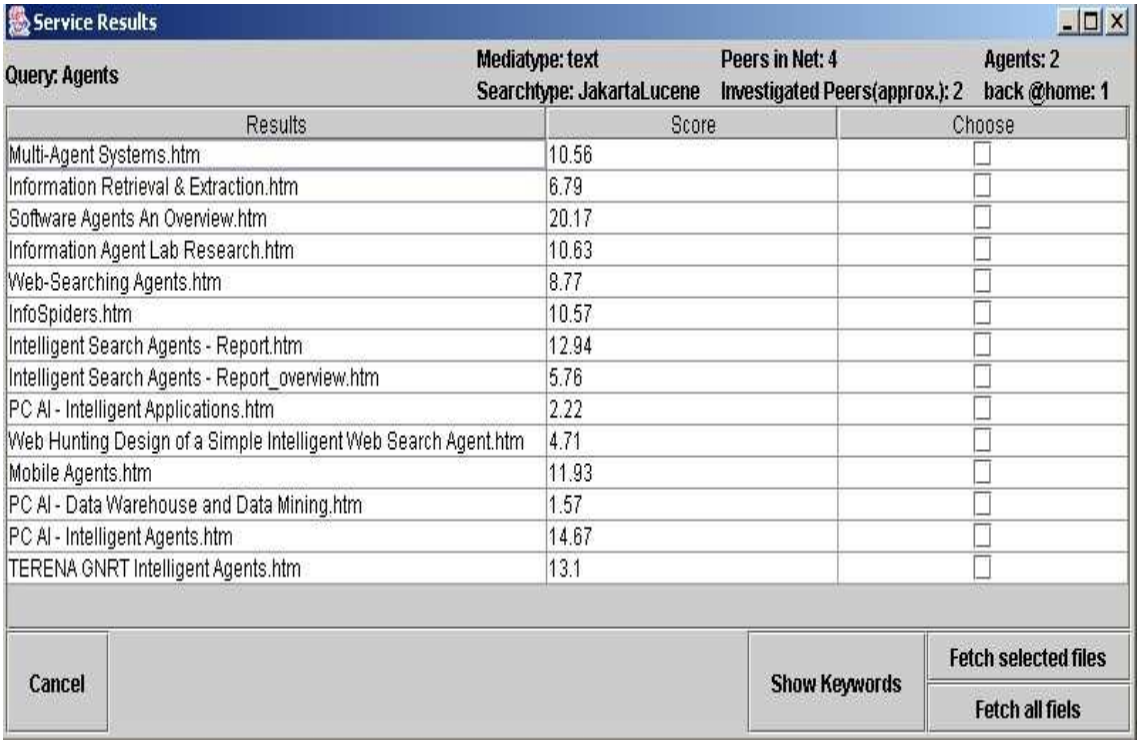

Abbildung 5.6: Suchergebnisübersicht

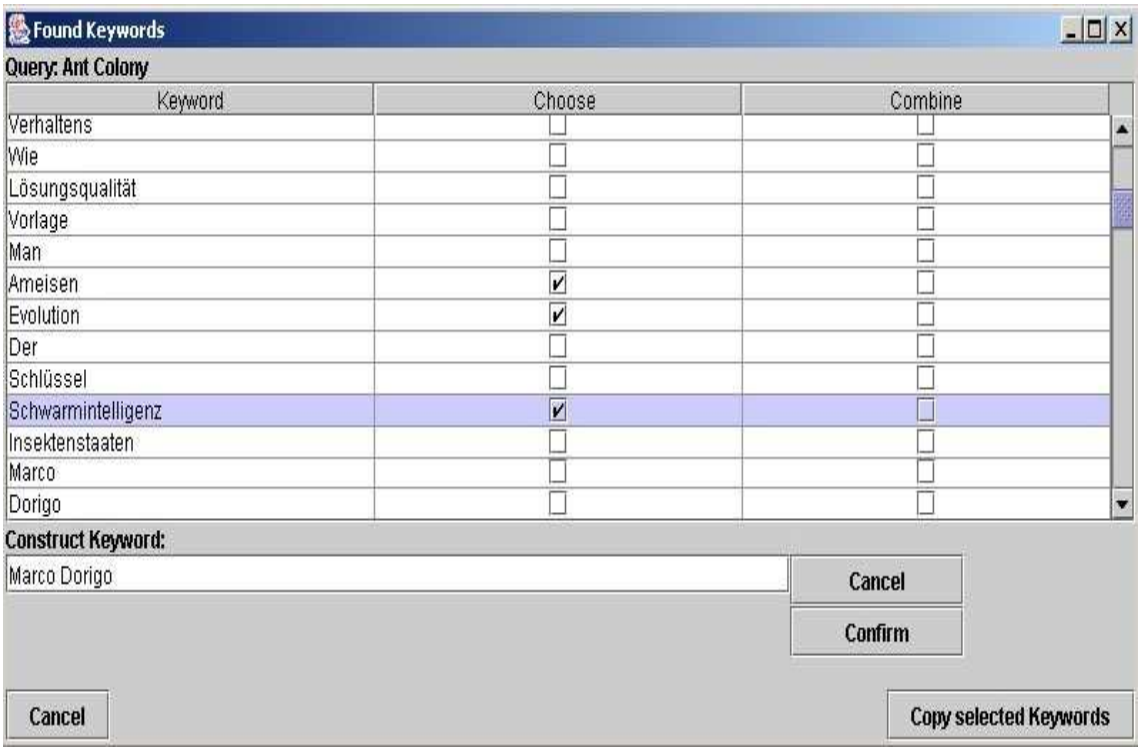

Abbildung 5.7: Schlüsselwörterübersicht

Kapitel 6

Ausblick und Verbesserungsmöglichkeiten

6.1 Erweiterungsmöglichkeiten

Neben der vorgestellten Nutzung der Applikation, kann der Anwender auch die Leistungsfähigkeit der Applikation erweitern. Diese stehen nicht in Verbindung mit den Verbesserungsvorschlägen, die im nachfolgenden Unterkapitel vorgestellt werden, sondern sind Bestandteil der persönlichen Anpassung der Applikation an die eigenen Bedürfnisse.
Konkret ist dies die Anbindung weiterer Suchmaschinen an die Applikation, so dass diese von dieser genutzt werden können. Folgende Verfahrensweise ist dafür notwendig.
Anfangs muss die Suchmaschine in der Applikation publiziert werden. Hiezu sind Eintragungen in der Datei `IFsearch.types` notwendig. Diese beinhaltet eine Zuordnung der Suchmaschine zu den MIME-Types, die mit ihr untersucht werden können. Eine Übersicht über MIME-Types kann bei der entsprechenden Internetseite [40] unter *Registered Media-types* abgerufen werden. Um eine Suchanfrage zu formulieren, bietet DataMiner die Möglichkeit an, individuelle graphische Eingabemasken für jede Suchmaschine einzubinden. Diese müssen lediglich das Interface `EvaluationGUI` und den `ActionListener` implementieren. Neben der Eingabemaske für den Suchbegriff, können auch weitere Parameter der Suchmaschine festgelegt werden, die dann in einer LinkedList abgespeichert werden. Damit die Agenten nun mit der Suchmaschine arbeiten können, muss noch ein Kontrollklasse für die Suchmaschine definiert werden, die das Interface `EvaluationControl` implementiert. Weitere Ergänzungen des Programmcodes sind nicht notwendig, um die gewünschte Funktionalität zu erreichen.
Mit dem Hinzufügen weiterer Suchmaschinen wird auch eine Anpassung der Applikation notwendig zu einer Suchanfrage weitere Referenzen anderer Medientypen hinzuzufügen. In der aktuellen Realisierung können nur Schlüsselwörter zum Medientyp „text“ einer Suchanfrage hinzugefügt werden. Dafür sind folgende Ergänzungen im Programmcode notwendig:
Die Methode `getSpecificSet` der Klasse `KeywordSet` ist mit einen Block zu erweitern, der sich auf den gewünschten MIME-Type bezieht. Das Ergebnis dieses Blocks ist eine LinkedList mit den Objekten, die zur Suchanfrage hinzugefügt werden können. Dabei ist es wichtig, dass der Wert der Variabel `counter` der Anzahl der Objekte der LinkedList entspricht. In der Methode `keywordSetGUI` der Klasse `User` werden die gefundenen Schlüsselwörter zur Anzeige aufbereitet. In der Methode muss der `setValueAt`-Befehl eine geeignete Repräsentation des KeywordSet-Inhalts übergeben bekommen. Wenn also ein Bild eingefügt werden soll, muss

die Information so aufbereitet werden, dass dieses Bild in der Tabelle dargestellt werden kann. Um die ursprünglichen Informationen aus dem `KeywordSet` wieder zu erhalten, müssen die u.U. modifizierten Daten aus der graphischen Darstellung bearbeitet werden. In der Methode `setJobKeywords` des `ManagementService` werden die ausgesuchten Schlüsselwörter in der Klasse `SearchJob` hinterlegt. Da in dieser Methode die Information verfügbar ist, in welchem MIME-Type gerade gesucht wird, muss hier eine entsprechende Unterscheidung nach MIME-Types erfolgen. Weitere Ergänzungen des Programmcodes sind nicht notwendig, um die gewünschte Funktionalität zu erreichen.

6.2 Verbesserungsmöglichkeiten

Die Applikation, die im Rahmen der Diplomarbeit entstanden ist, kann nur als die Basis für ein umfassendes Datenaustausch- und Suchmaschinen-Programm verstanden werden. Dafür sind, nicht zuletzt durch die generische Lösung der Applikation, schon einige Grundlagen geschaffen worden, die es im Folgenden auszubauen gilt. Nachfolgend sind hierzu einige Verbesserungsmöglichkeiten genannt.

GUI. Die komplette Eingabemaske entspricht nicht dem Stand der heutigen Technik. Es wurde zwar versucht ein durchgehendes Schema einzuhalten (gleiches Aussehen der Fenster und Platzierung einzelner Buttons an den selben Stellen), aber das Multi-Frame-Konzept kann durch übersichtlichere Gestaltungen abgelöst werden und so dem Anwender einen Umgang mit dem Programm vereinfachen. Hierzu bietet sich bspw. folgende Variante an: Darstellung der Informationen mit mehreren Registern. Hierbei werden acht Register (Konfiguriere Programm, Bearbeite Schlüsselwörter, erstelle Suchanfrage, Auftragsübersicht, Auftragsergebnisse, gefundene Schlüsselwörter, History (new stored files), Chat) für die einzelnen Aufgaben benötigt. Der Vorteil bei dieser Darstellung ist die kompakte Darstellung und Übersichtlichkeit der Informationen. Diese kann allerdings durch die Chat-Funktion wieder beeinträchtigt werden. Denn wenn man sich mit mehreren Personen gleichzeitig unterhalten möchte, ist es nicht praktikabel die verschiedenen Gespräche in einem Fenster anzuzeigen, sondern mehrere anzubieten.
Neben dieser graphischen Anordnung kann auch die gesamte Konfiguration des Applikation verbessert werden. Denn die Angabe von Pfaden zu Ordnern geschieht momentan noch rein textlich und könnte durch Nutzung von Verzeichnisbäumen vereinfacht werden.
Obwohl der Anwender seine relevanten Dateien in verschiedenen Verzeichnissen gleicher Hierarchieebene hinterlegen kann, kann in der Applikation nur ein Suchpfad angegeben werden. Dadurch kann nur auf einen Teil der Daten zugegriffen werden, wenn der Anwender seine Dateiablage nicht anders organisiert. Es ist für die Applikation und alle Beteiligten des Peer-to-Peer Netzes von Vorteil, wenn sich die Applikation den Bedürfnissen der Anwender anpasst und nicht umgekehrt. Deshalb müsste die Eingabe mehrer Suchpfade genauso ermöglicht werden, wie die dynamische Angabe von Speicherpfaden von beschafften Dateien. Im Zuge eine besseren individuellen Steuerung der Rechte für Agenten kann zukünftig neben deren Besitzer auch die Suchpfadangabe berücksichtigt werden. Dies ist vor allem für Interessensgruppen und Projektgruppen ein interessantes Merkmal.
Damit der Anwender die Applikation besser seinen Bedürfnissen anpassen kann, sollte dieser zukünftig auf mehr Parameter Zugriff haben, wie Periodenzeiten von zeitlich gesteuerten Aktionen.
In der aktuellen Darstellung der Ergebnisse werde mehrfache Treffer einer Datei auch mehrfach angezeigt. Die Darstellung muss dahin gehend noch angepasst werden, dass immer nur eine

einfache Darstellung einer Datei erscheint, auch wenn sie mehrfach vorhanden ist.

Agenten-Kommunikation. Neben den graphischen Aspekten kann auch die Kommunikation der Agenten besser genutzt werden, um den Suchauftrag effizienter zu gestalten. Wie bei der Untersuchung der Effizienz des Programms bereits besprochen, ist es zwingend notwendig für die Applikation, dass sie das SHIP-Protokoll nutzt. Inhaltlich kann die Kommunikation den Austausch neuer Schlüsselwörter umfassen, wenn diese mit einer entsprechend guten Qualität gefunden werden. Daneben könnten Suchagenten über Peers informieren, die zwar aufgesucht werden könnten, auf denen sie aber keine Rechte verfügen. Diese Information ist für den SearchService interessant, denn dann kann er diese Peers aus einem Suchauftrag ausschließen. FetchAgenten könnten die Kommunikation nutzen, wenn sie erkennen, dass sie für den Erwerb eines Duplikates noch eine große Zeitspanne benötigen sei es wegen der Größe der Datei und/oder der schlechten Performance des Peers. Diese Information könnten sie entweder nur als Information an den FetchService zurück senden oder es aber als Indikator dafür nehmen, dass eine andere Fetchstrategie angewandt werden sollte. Diese kann wie folgt aussehen:

Performance beim Peer zu langsam: Suche speziell nach alternativen Anbietern der gewünschten Datei. Wenn entsprechende gefunden wurden, versuche von dort ein Duplikat zu erwerben und informiere den initiierenden FetchAgenten, damit er seinen Erwerbsprozess abbricht.

Datei ist zu voluminös: Kommuniziere an FetchService, welche entsprechende Datei zu voluminös ist und somit eine große Zeitspanne benötigt, bis sie komplett dupliziert wurde. Unter der Voraussetzung, dass die betreffende Datei partitioniert werden kann, sollte der FetchService dem Agenten darauf hin eine Nachricht schicken, wie groß sein Fragment der betreffenden Datei ist und dass er mit diesem dann zurück kehren soll. Andere FetchAgenten sollen dann entweder beim selben Peer portionsweise sich andere Fragmente der Datei besorgen, oder diese parallel von anderen Quellen, die zur Verfügung stehen. Auf diese Weise kann die Dateiduplizierung beschleunigt werden.

Eine weitere Nachricht, die allerdings nicht verschickt wird, sondern vom Agenten signiert beim Peer verbleibt, kann eine Aussage darüber treffen, wann der Index, der für die Suchanfrage notwendig ist, erstellt wurde. Ein Agent, der die selbe Suchmaschine nutzt, kann entscheiden, ob ihm der vorhandene Index als Datengrundlage ausreicht oder ob er eine neue erstellen muss.

Agenten-Migrationsverhalten. Bezogen auf die Stabilität der Netzwerkverbindungen und der Erreichbarkeit einzelner Peers ist zu prüfen, ob die Migrationstrategie nicht wie folgt zu modifizieren ist. Wenn nach mehrmaligen Versuchen ein Peer nicht zu erreichen ist, versucht der Agent einen anderen Peer zu erreichen und den anderen zu einem späteren Zeitpunkt versuchen aufzusuchen. Kann der Agent keinen weiteren Peer mehr erreichen, soll er wieder zum Ausgangspeer zurück kehren. Bei diesem berichtet der Agent dann, neben den gewonnen Resultaten, welche Peers nicht zu erreichen waren. Dies ist aus Sicht der Informationsgewinnung eine bessere Lösung, als bspw. auf dem Peer zu verbleiben, von dem aus der nächste nicht zu erreichen war. Denn bevor auch der Rückweg abgeschnitten wird, ist es besser nur Teilergebnisse zu erzielen, als gar keine.

Agenten-Suchverhalten. Zusätzlich zu den kommunikationsspezifischen Verbesserungen, sollte auch die Suche der Agenten modifiziert werden. Möchte man darauf vertrauen, dass ein Anwender eines Peers eine Suchmaschine nicht zu seinen Gunsten manipuliert, könnte ein Service

sich speziell um die Indizes oder anderer Vorbereitungen für einen Einsatz der Suchmaschinen kümmern. Dieser könnte dann periodisch diese Vorbereitungen den evtl. geänderten Bedingungen (Dateien wurden in der Verzeichnisstruktur editiert) auf dem Peer angleichen. Der Service kann aber auch aktiv die Verzeichnisstruktur überwachen und Änderungen vornehmen, wenn sich die Verzeichnisstruktur ändert.

Peergruppen-Informationsverteilung. Mit der Möglichkeit Peer-Gruppen, in Abhängigkeit der Nutzerkennung, zu definieren könnten spezielle Informationsnetze oder Projektnetze aufgebaut werden. D.h. wenn Agenten ein spezielles Verzeichnis zugewiesen werden kann, kann dieser beauftragt werden den Inhalt des eigenen Verzeichnisses mit denen der anderen Teilnehmer abzugleichen, so dass alle immer über den gleichen Informationsstand verfügen.

Suchkoordination. Das System namens „Anthill“, welches im Kapitel 1 vorgestellt wurde, bedient sich Pheromonen als Informationsträger. Unter der Voraussetzung, dass die Teilnehmer des Netzwerks immer eine feste IP-Adresse besitzen, könnten Informationen über ihr Verhalten bei verschiedenen Suchanfragen ermittelt werden. Dies könnte in einem ersten Schritt beispielsweise sein, wie viele Treffer auf einem Peer zu einem bestimmten Medientype erzielt wurden. Diese Information könnte der Agent dann seinem Ausgangspeer zukommen lassen, so dass dieser sich ein Bild darüber machen kann, welche Peers zu einem Medientyp am wahrscheinlichsten gute oder gar keine Ergebnisse erzielen werden. Ähnlich wie bei Pheromonen, könnte so eine Gewichtung von Peers vorgenommen werden, wobei auch periodisch Peers mit einer schlechten Gewichtung aufzusuchen wären, um die vorgenommene Einstufung zu validieren.

Suchmaschinen. Speziell für die Suchmaschine lassen sich folgende Verbesserungen vorschlagen:

- Extraktion eines Abschnitts des Dokuments um mehr über dessen Inhalt zu erfahren.
- Hervorheben von Schlüsselwörtern in diesen Textauszügen. Ein entsprechendes Programm gibt es auf der Homepage von Jakarta Lucene.
- Ergebnisse, insbesondere die Schlüsselwörter nicht nur Listen, sondern auch seitenweise anzeigen.
- Angabe darüber, bei welchem Peer welche Files gefunden wurden. Dann kann der Chat-Agent auch dazu genutzt werden mit diesem gezielt in Kontakt zu treten.

Bei der Suche nach relevanten Dateien, sollten auch Mehrfachfunde einer Datei auf verschiedenen Peers berücksichtigt werden. Allerdings sollte der Eintrag dann nur einmal dem Anwender präsentiert und die Namen der Besitzer dazu aufgezählt werden. Der Vorteil in dieser Strategie ist darin begründet, dass die Wahrscheinlichkeit, dass der Anwender seine angeforderte Datei erhält, dadurch wesentlich vergrößert wird. Wenn ein FetchAgent registriert, dass die Datei bei einem Peer, aus diversen Gründen, nicht mehr zur Verfügung steht, so kann er versuchen, diese von anderen Netzwerkteilnehmern zu erhalten. Daneben ist diese Information wichtig, um gleichzeitig eine Duplikat einer Datei von verschiedenen Peers aus zu erstellen, wie dies bereits oben erläutert wurde.

FetchAgent. Der FetchAgent sollte Informationen sammeln, wenn Dateien nicht beschafft werden konnten. Diese Informationen könnten folgenden Inhalt haben:

- Peer nicht erreichbar
- Rechte am Peer verweigert
- Datei nicht mehr vorhanden

Wie bei der Agenten-Kommunikation bereits angesprochen, können diese dann Informationen dazu genutzt werden, dass die Service der Applikation ihre Strategien anpassen.

Kategorisieren von Schlüsselwörter. In der aktuellen Realisierung der Applikation werden Schlüsselwörter mit einer Suchanfrage verknüpft, ohne die inhaltliche Bedeutung der Suchanfrage zu berücksichtigen. Da Suchanfragen auch Polyseme sein können, tritt das Problem auf, dass Schlüsselwörter miteinander vermischt werden, die in keinem inhaltlichen Zusammenhang stehen. Abhilfe schafft eine entsprechende Kategorisierung der Schlüsselwörter. Wenn diese entsprechend gegliedert sind, kann beim Erstellen eines Suchauftrags das Angebot an Schlüsselwörtern für den Anwender besser differenziert werden. Als Grundlage einer Kategorisierung kann auf die *Universelle Dezimalklassifikation* [21], [4], [12], [29] zurückgegriffen werden.

Kapitel 7

Fazit

Die im Rahmen dieser Diplomarbeit entwickelte Applikation ermöglicht es, mit mehreren kooperierenden Agenten nach beliebigen Informationen zu suchen. Sie nutzt erfolgreich SeMoA als Basis für Peer-to-Peer Netzwerke und setzt deren Agenten zur Informationsrecherche und Dateibeschaffung ein. Dabei hat SeMoA die Vorzüge, dass es die beiden Merkmale Nutzung mobiler Agenten und Aufbau von Peer-to-Peer Netzwerken bereits besitzt und dahingehend nicht angepasst werden muss. Daneben ist auch der Sicherheitsanspruch von SeMoA für die Applikation von Bedeutung, da dieser dazu beiträgt, dass in den Netzwerk eine Vertrauensbasis zwischen den einzelnen Anwendern gebildet werden kann.

Auf Grundlage der eingesetzten Technologie konnte die Applikation so entwickelt werden, dass sie nach beliebigen Informationen in Peer-to-Peer Netzen suchen kann. Diese Beliebigkeit wird durch den Einsatz mehrerer Suchmaschinen definiert, die je nach Ausprägung bspw. nach Bildern, Dokumenten, Stimmen oder Wasserzeichen suchen können. Um dies zu verdeutlichen, wurde die Suchmaschine Jakarta Lucene [32] für Text Retrieval implementiert. Diese wurde als Open Source-Projekt bei Apache entwickelt. Es war vorteilhaft für die Applikation diese Suchmaschine einzusetzen, da sich die Suchmaschine bereits erfolgreich bei verschiedenen Projekten [27], [14], [6] bewährt hat und somit eine Leistungsüberprüfung der Suchmaschine nicht notwendig war. Daneben ist sie in der gleichen Programmiersprache implementiert wie die Applikation, was die Integration in die Applikation beschleunigte.

Mit der Suchmaschine wurde gezeigt, dass eine Informationssuche mit den in der Aufgabenstellung genannten Randbedingungen mit SeMoA möglich ist. Allerdings machte der Einsatz dieser Suchmaschine auch deutlich, dass die Performance je nach Datenbestand gering werden kann. Denn die Strategie der Applikation ist, dass Agenten ihre eigenen Suchmaschinen mitbringen und den Datenbestand immer erneut für eine Informationssuche aufbereiten. Diese Aufbereitung kann einen zeitlich hohen Aufwand darstellen, so dass im Rahmen einer Weiterentwiclung der Applikation versucht werden sollte, diesen Aufwand zu verringern.

Insgesamt ist die Applikation ein gutes Framework um Informationen in Peer-to-Peer Netzen zu suchen und somit eine gute Beispielapplikation für sichere, mobile Agenten. Die Eigenschaften der Agenten passen gut zu den sich ständig ändernden Umweltbedingungen in Peer-to-Peer Netzen und können so die Leistungsfähigkeit von Agenten zeigen. Dabei ist es aber von Bedeutung, dass die Applikation ihre Kommunnikation auf das SHIP-Protokoll stützt, um performanter zu werden. Erstmalig wird *DataMiner* in einem Projekt integiert werden, welches die ontologie-basierte Suchtechnologie Melvil^{Tm} [58] einsetzt. *DataMiner* soll dieses Projekt mit seinen Eigenschaften unterstützen und zum Gesamterfolg beitragen.

Interessant für die Zukunft ist, welche Strategieänderungen in der Arbeitsweise des Programms vorgenommen werden. So ist vorstellbar, dass eine verstärkte inhaltliche Nutzung der Kommunikationsschnittstelle und der Einsatz genetischer Algorithmen zur Optimierung von Parametern die Effizienz der Applikation noch steigern können.

Anhang A

Abbildungen

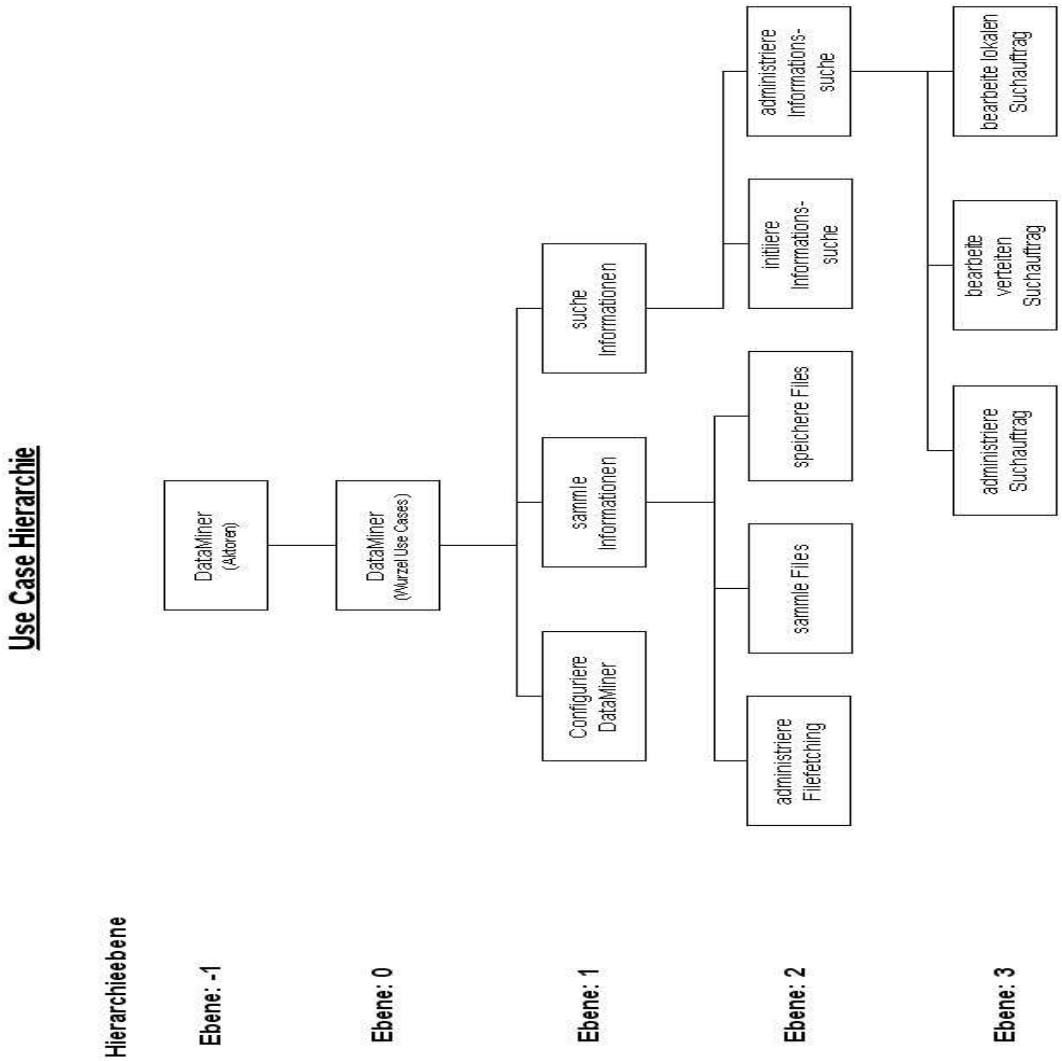

Abbildung A.1: Use Case-Übersicht

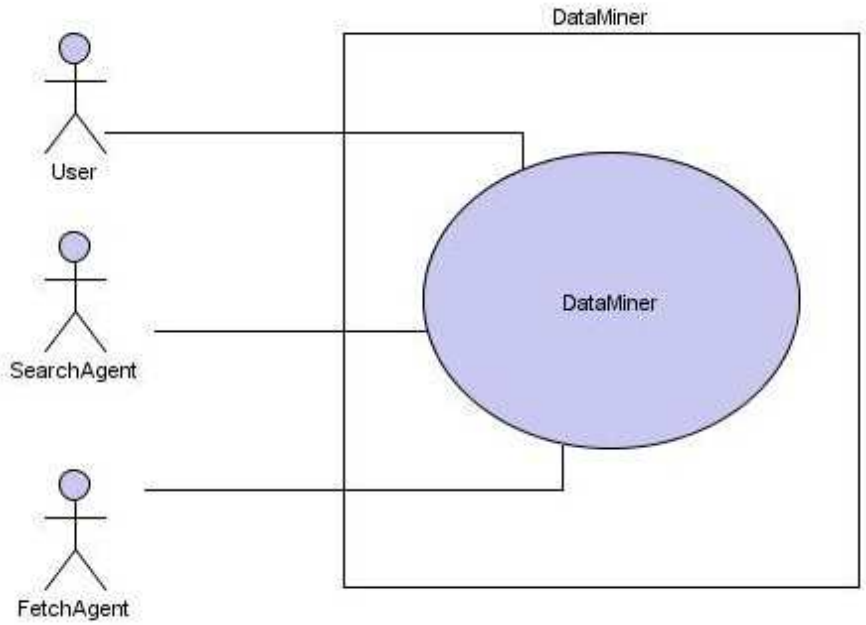

Abbildung A.2: Kontextdiagramm

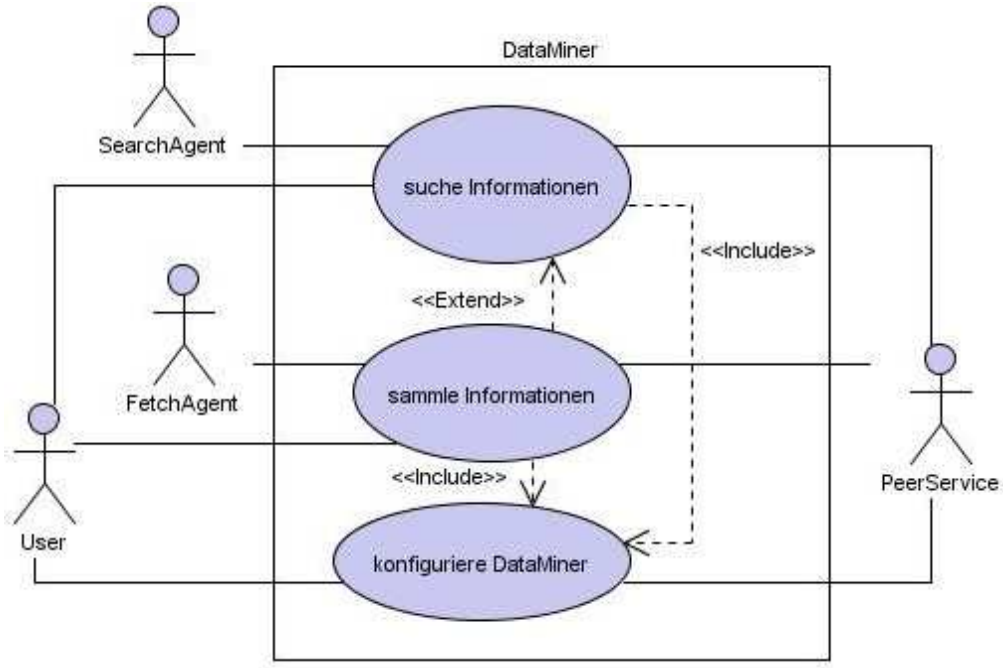

Abbildung A.3: Use Case Ebene 0

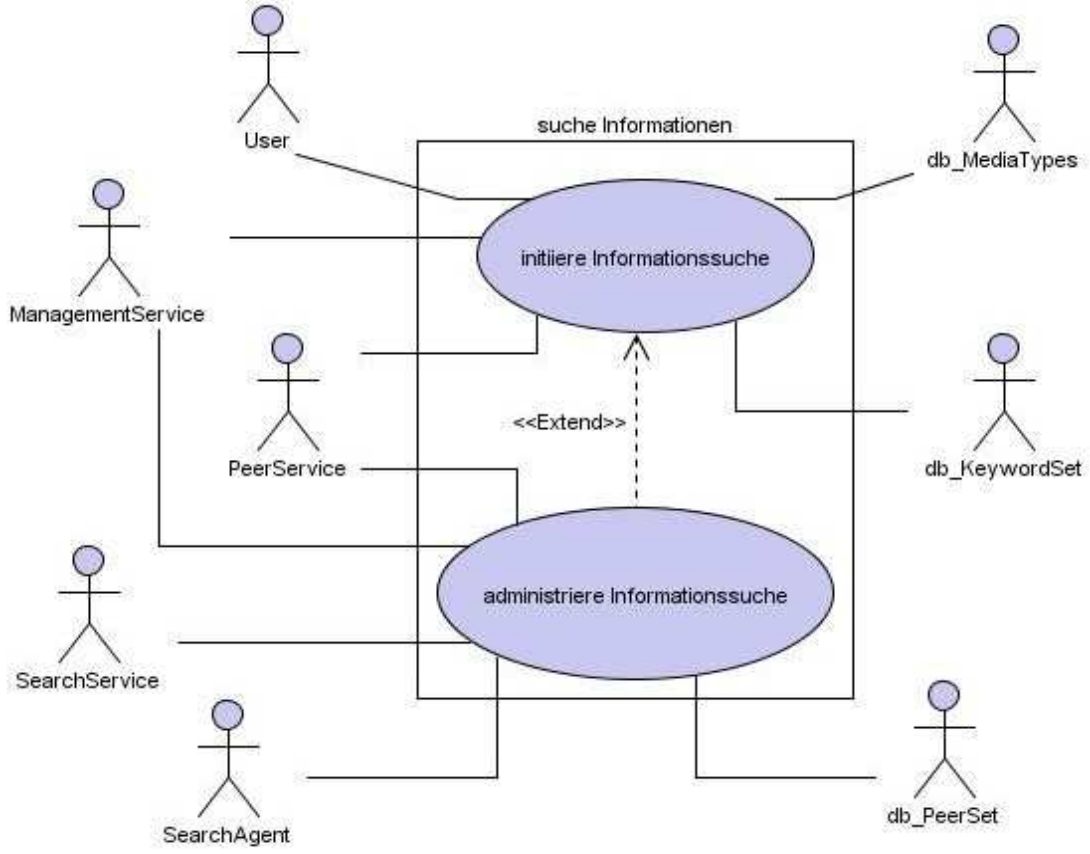

Abbildung A.4: Use Case Ebene 1.1

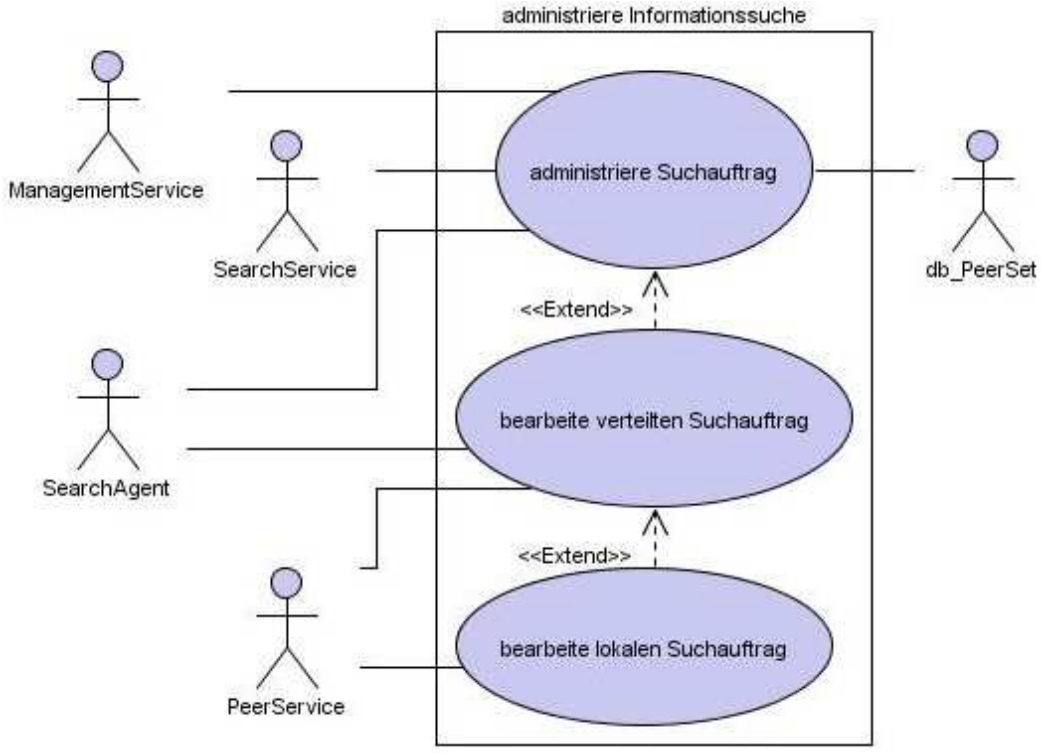

Abbildung A.5: Use Case Ebene 1.1.2

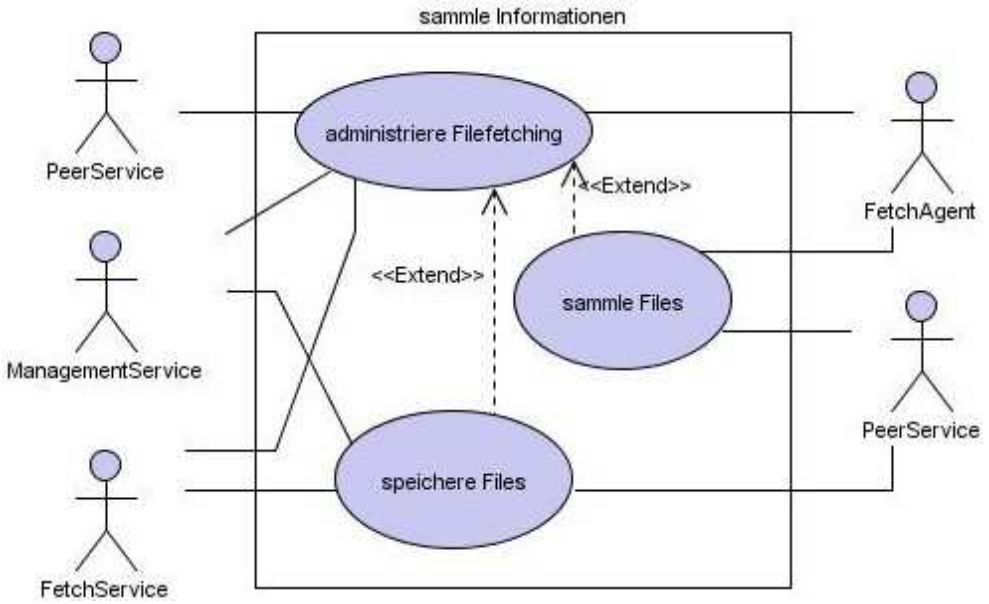

Abbildung A.6: Use Case Ebene 2.2

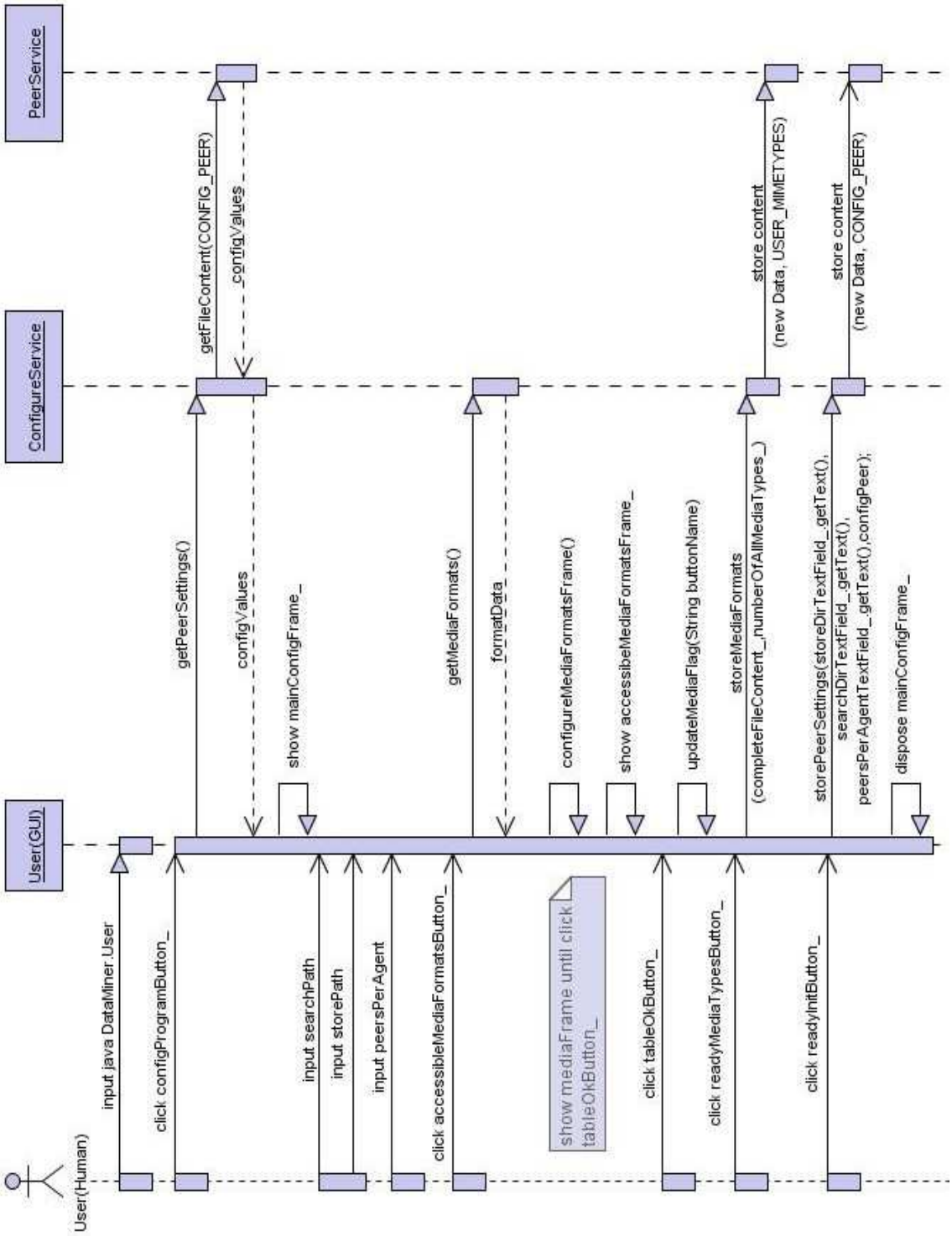

Abbildung A.7: Sequenzdiagramm „Konfiguriere DataMiner“

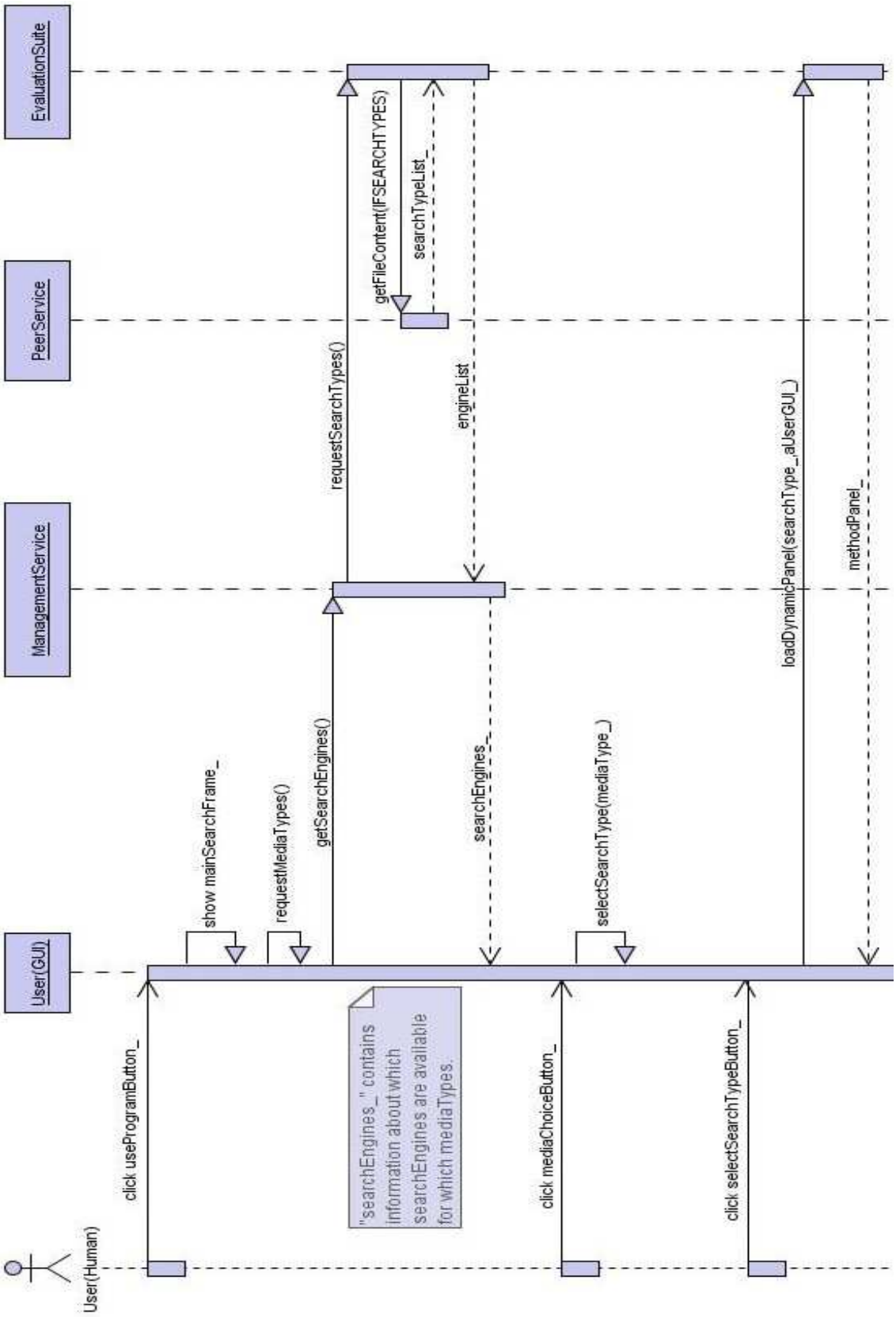

Abbildung A.8: Sequenzdiagramm „Initiiere Informationssuche“, Abschnitt 1

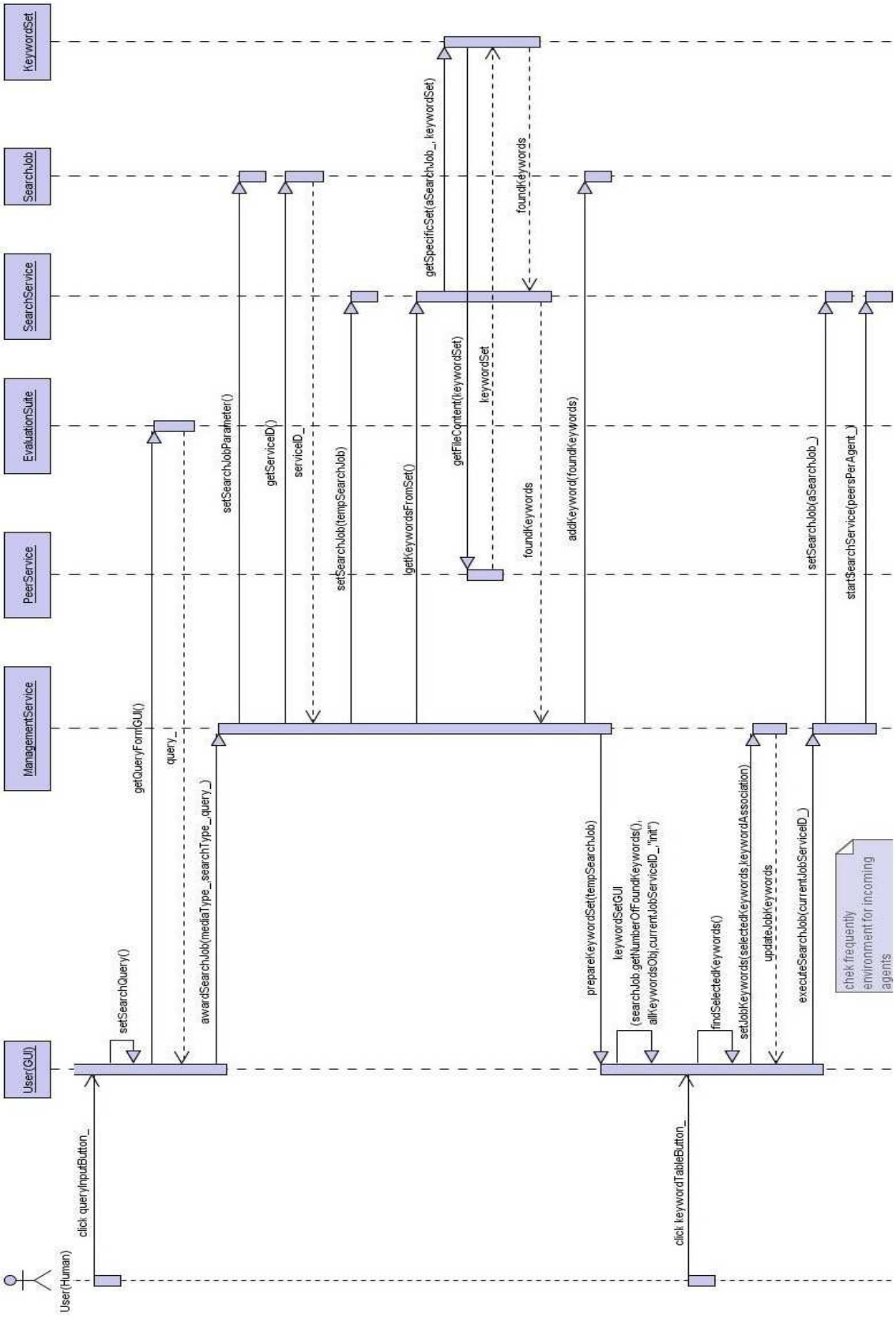

Abbildung A.9: Sequenzdiagramm „Initiiere Informationssuche“, Abschnitt 2

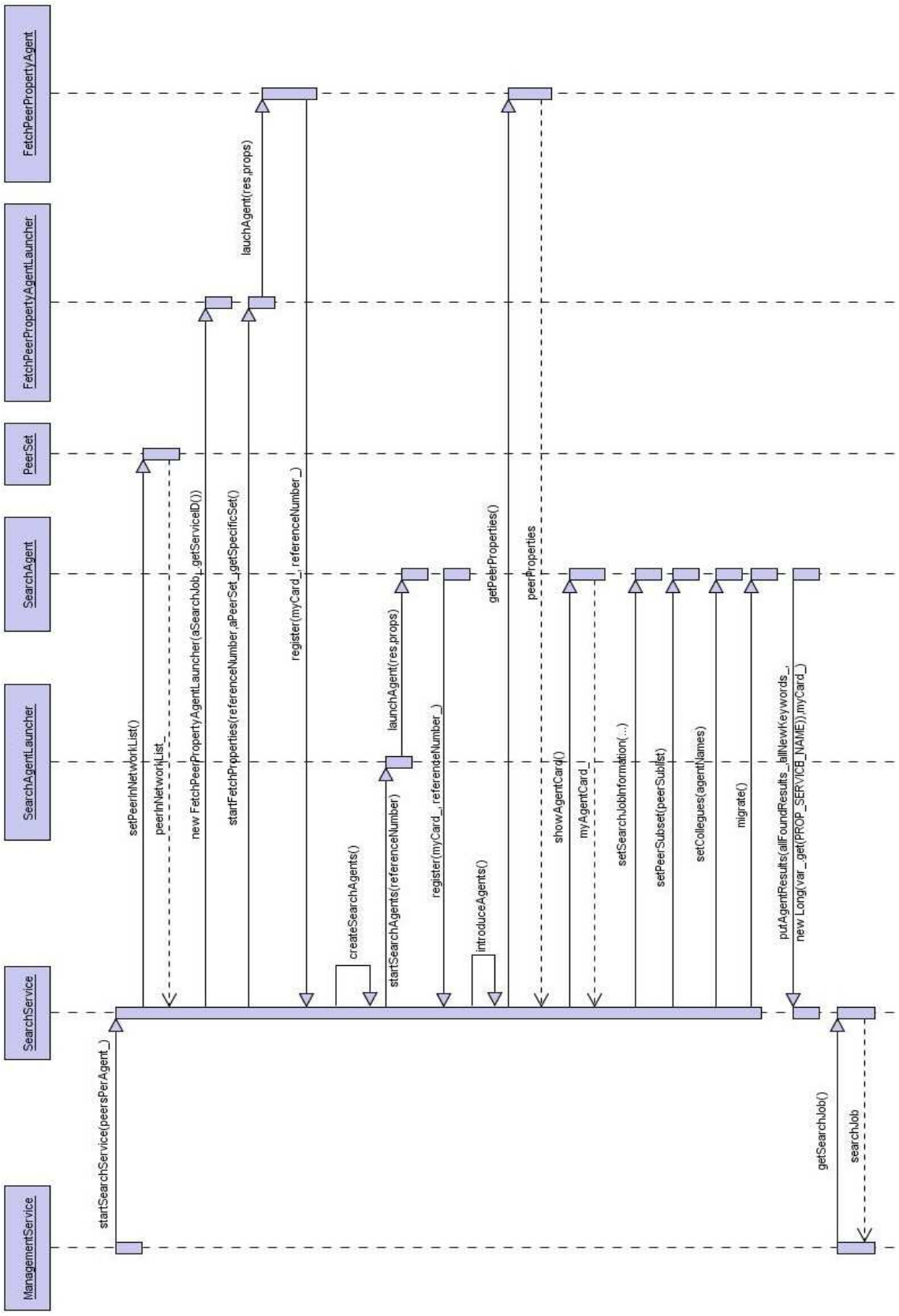

Abbildung A.10: Sequenzdiagramm „Administriere Suchauftrag“

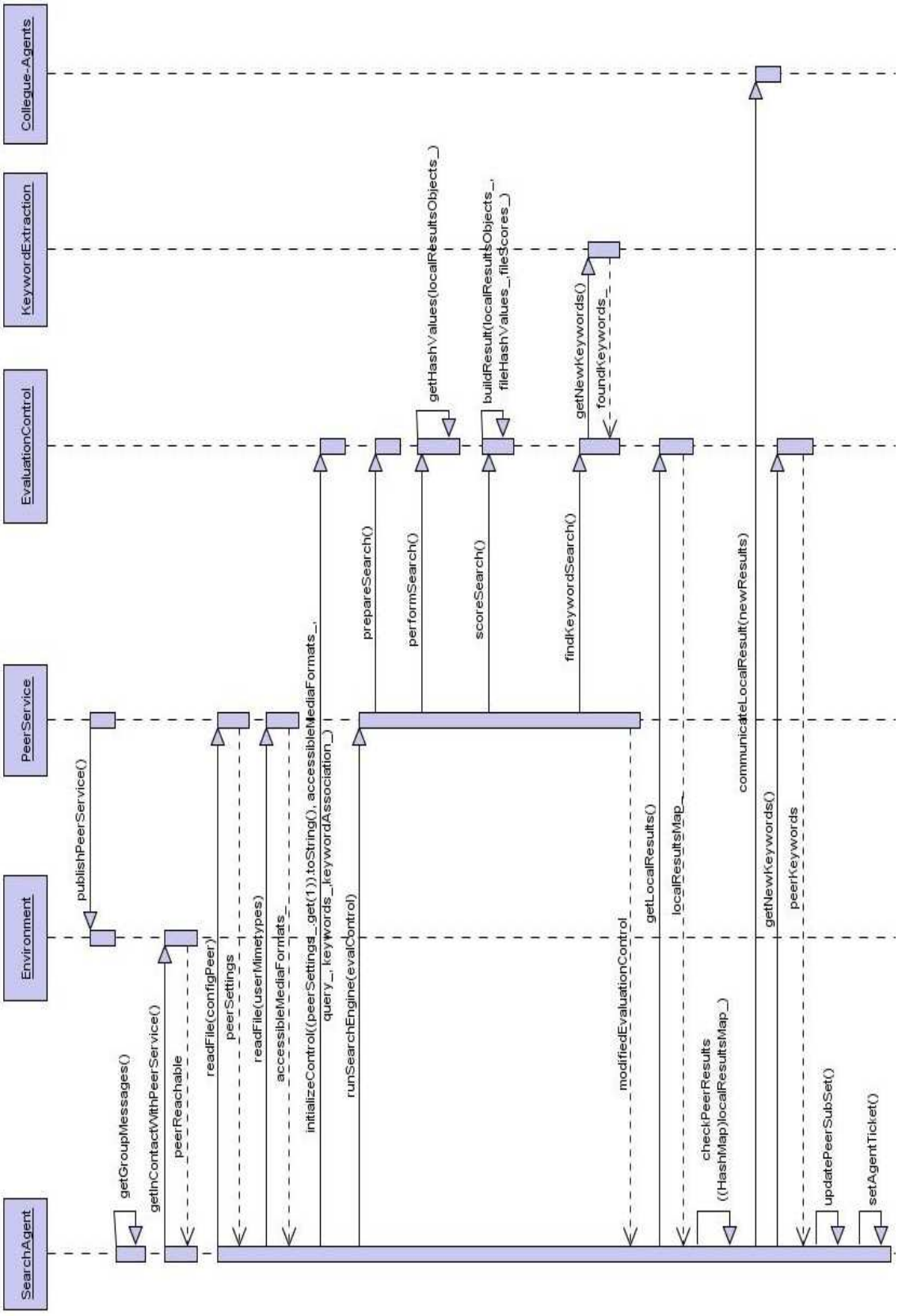

Abbildung A.11: Sequenzdiagramm „Bearbeite Suchauftrag“

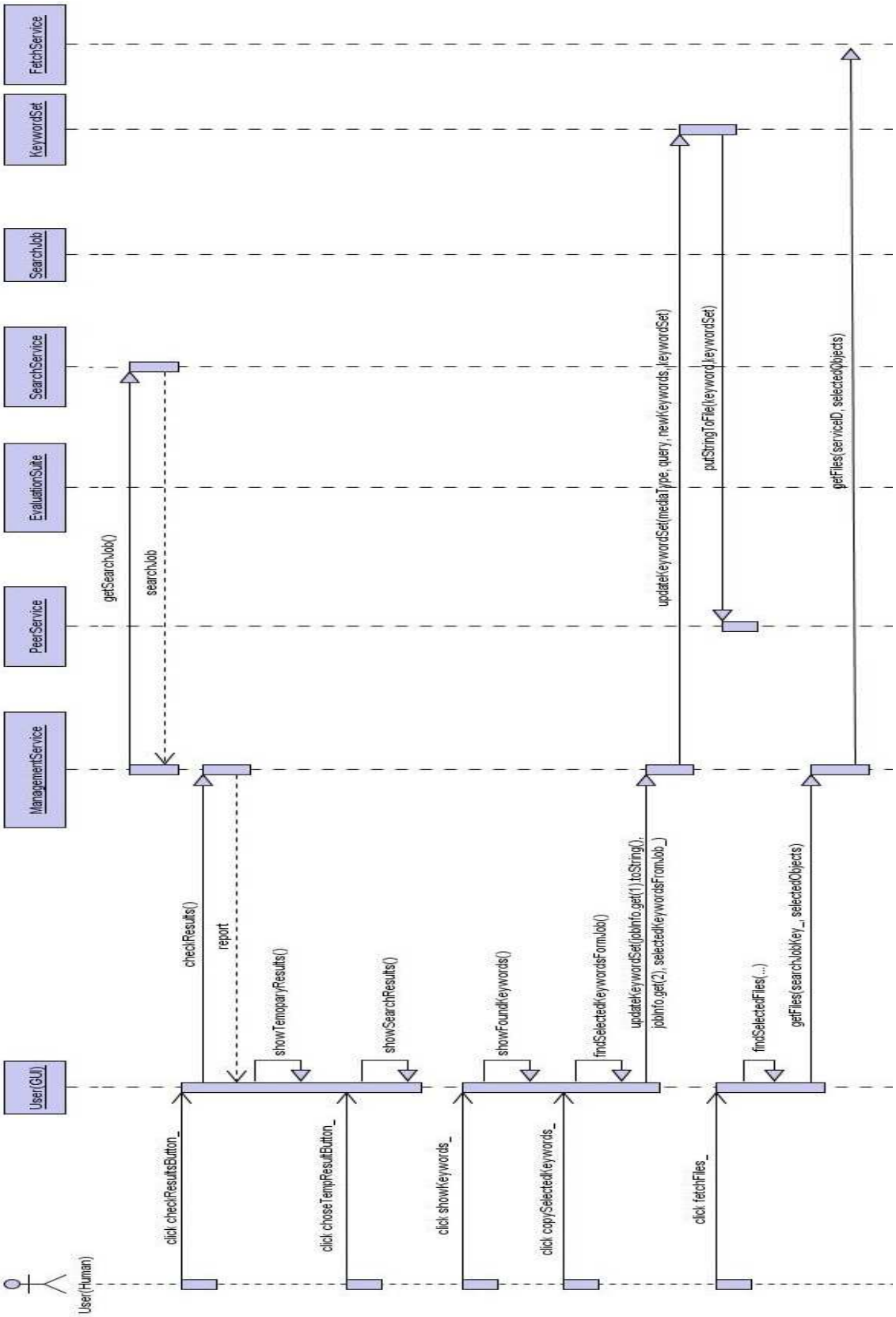

Abbildung A.12: Sequenzdiagramm „Administriere Dateibeschaffung“

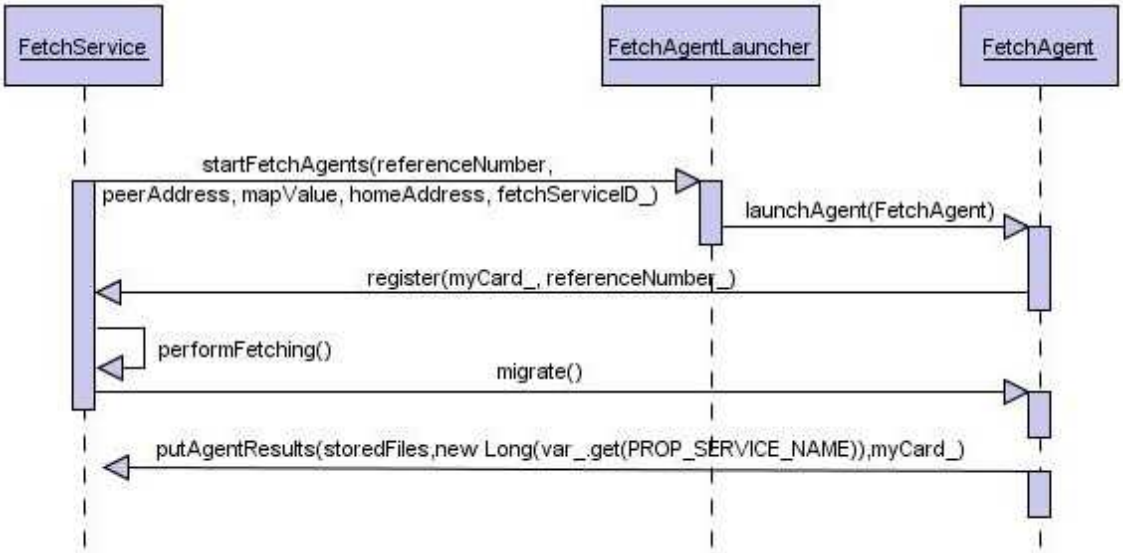

Abbildung A.13: Sequenzdiagramm „Sammle Dateien“

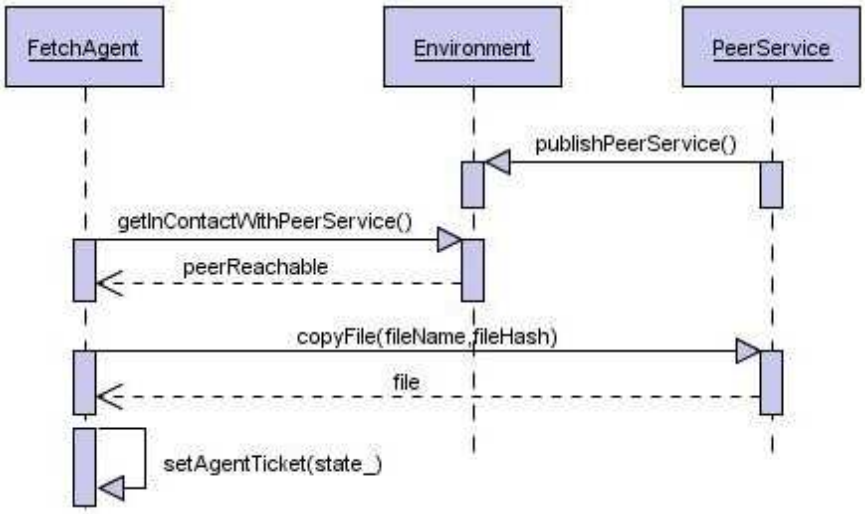

Abbildung A.14: Sequenzdiagramm „Speichere Dateien“

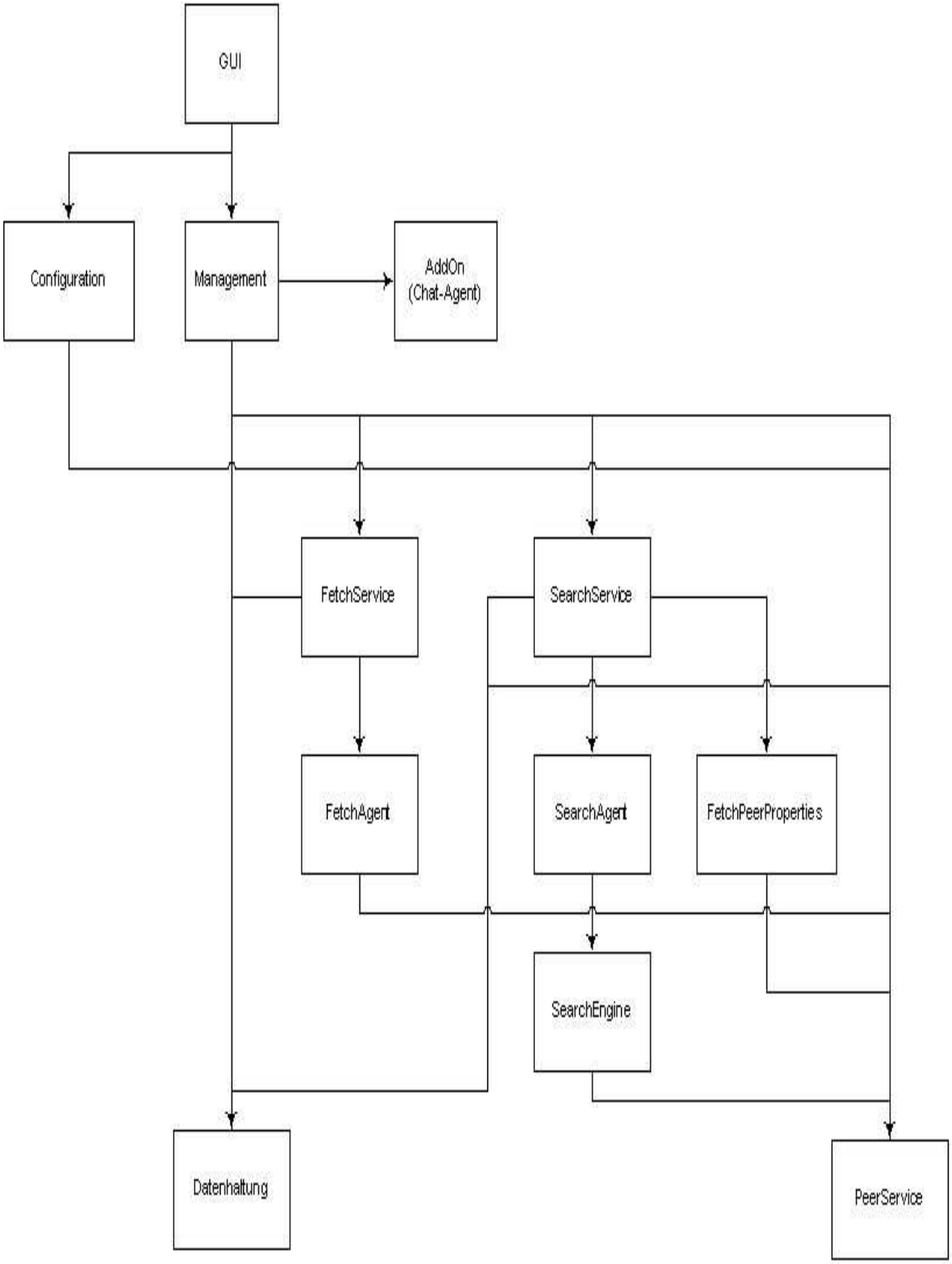

Abbildung A.15: Abstrahiertes Klassendiagramm

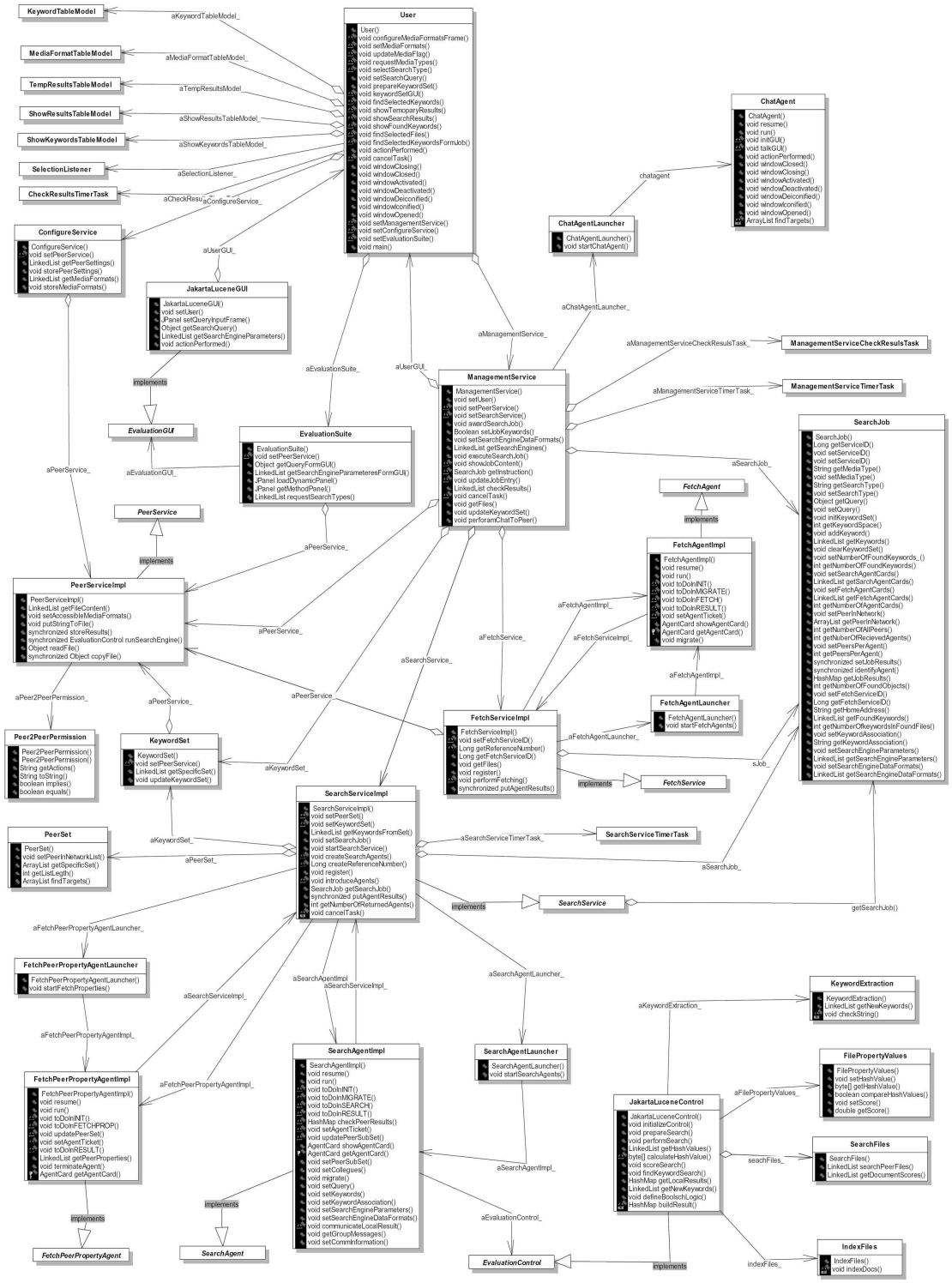

Abbildung A.16: Klassendiagramm

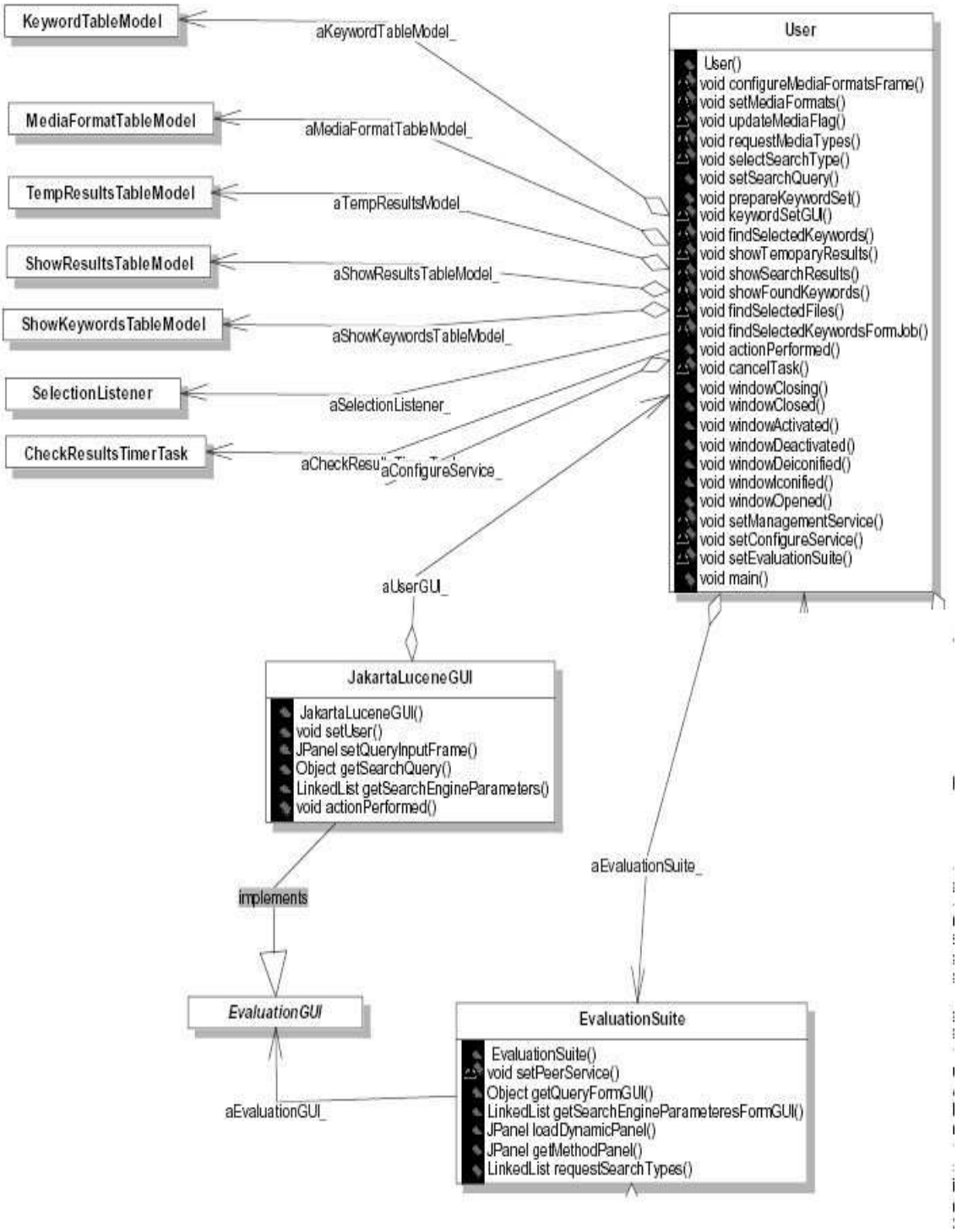

Abbildung A.17: Graphische Bedienoberfläche

Abbildung A.18: Konfiguriere Applikation

Abbildung A.19: Manage Applikation

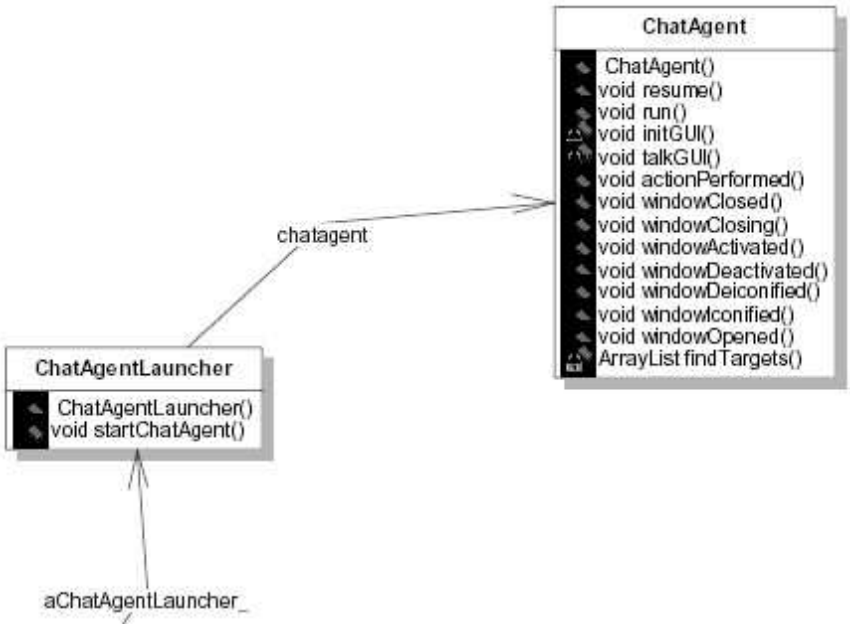

Abbildung A.20: Chat-Agent

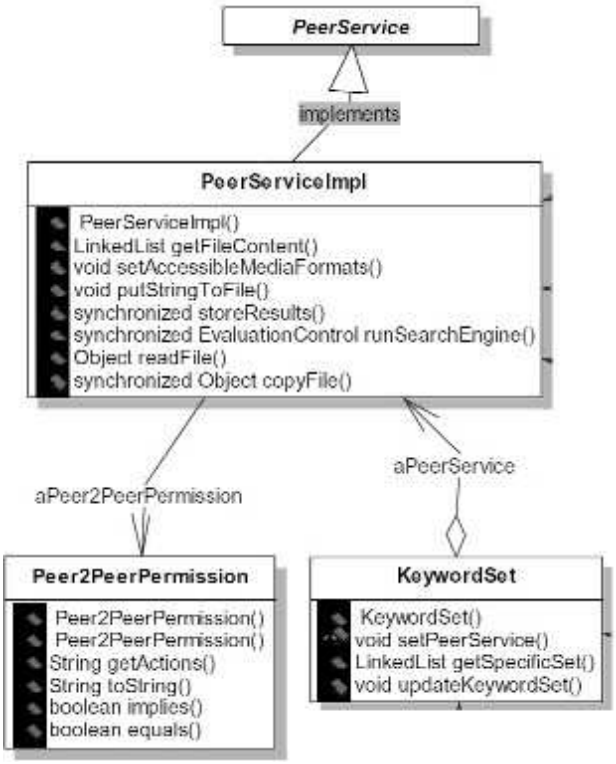

Abbildung A.21: Peerdienste

Abbildung A.22: Datenhaltung

Abbildung A.23: Search-Service

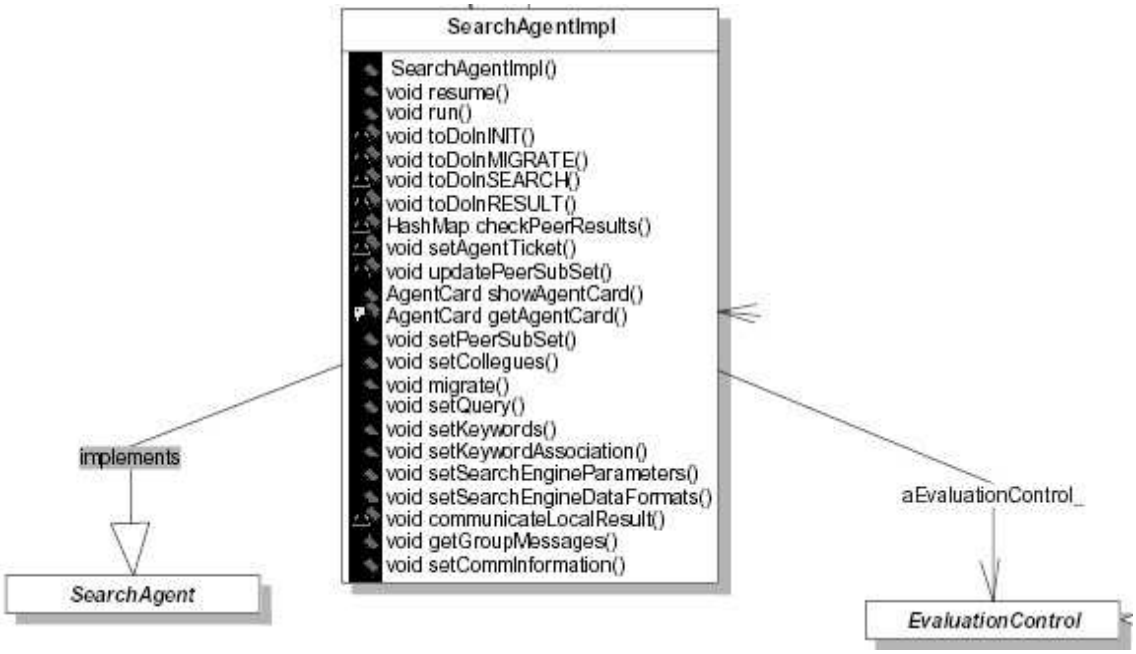

Abbildung A.24: Search-Agent

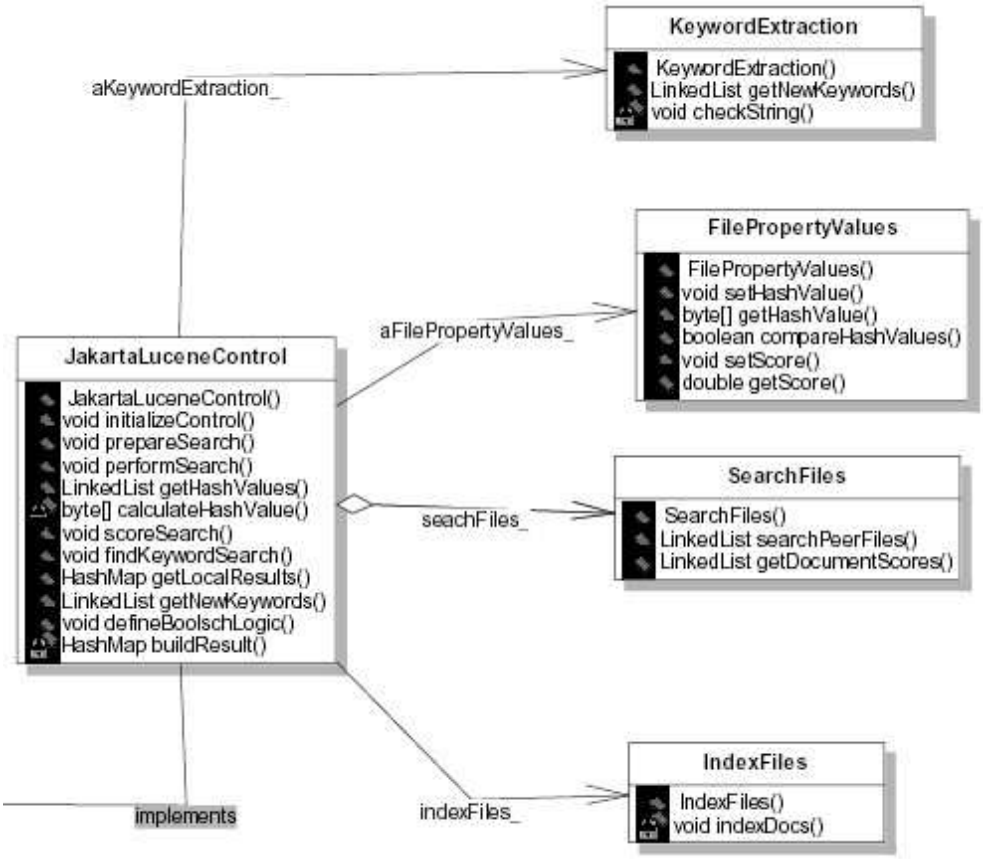

Abbildung A.25: Suchmaschine (Bsp. Jarkata Lucene)

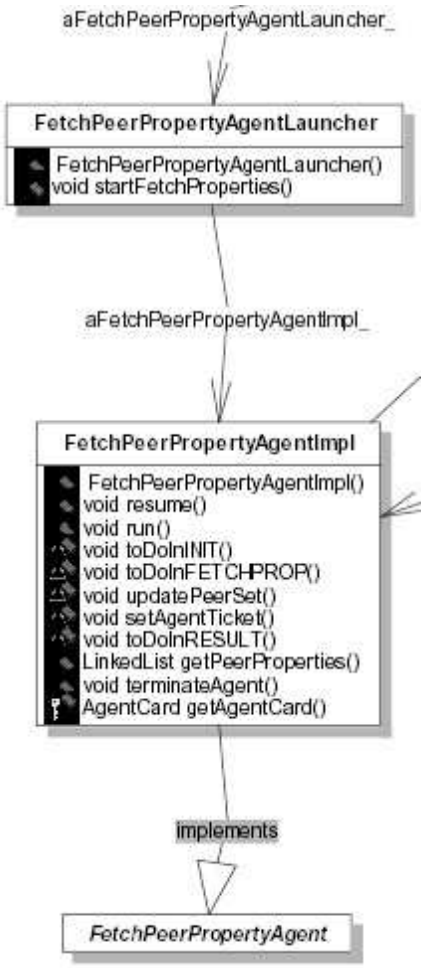

Abbildung A.26: Fetch Peerproperties

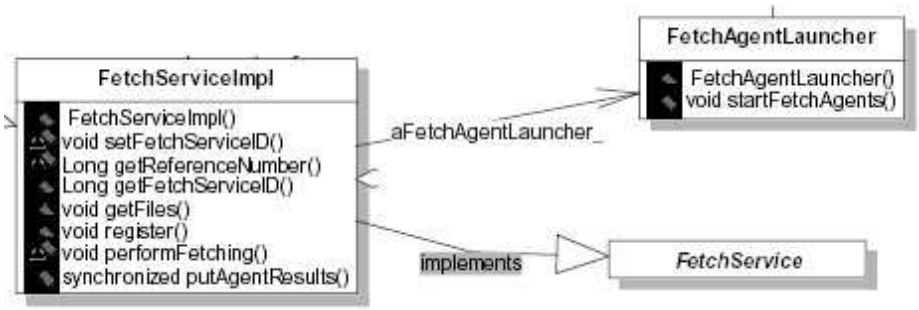

Abbildung A.27: Fetch-Service

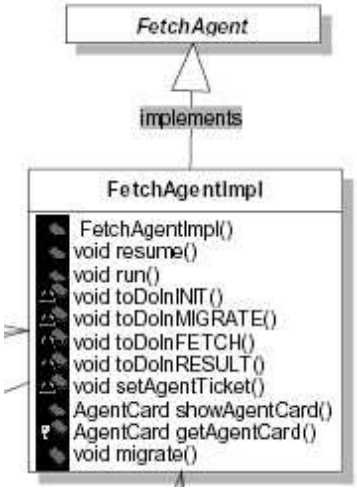

Abbildung A.28: Fetch-Agent

Literaturverzeichnis

[1] Karl Aberer. Chaper 3: Information Retrieval Part 1: Vector Space Retrieval. Technical report, Laboratoire de systèmes d'information répartis, 2002.

[2] Agentland. (`http://www.agentland.com`).

[3] At Web. (`http://www.at-web.de`).

[4] Christian Baumann, Reiner Diedrichs, Karl Werner Finger, et al. Einführung und Nutzung der Dewey Decimal Classification (DDC) im deutschen Sprachraum. Technical report, Arbeitsgruppe Klassifikatorische Erschließung im Auftrag der Konferenz für Regelwerksfragen, 2000.

[5] Teresa Blaxton and Christopher Westphal. *Data Mining Solutions (Methods and Tools for Solving Real-World Problems)*. Wiley, 1998. ISBN 0-471-25384-7.

[6] Bob Dylan Homepage. (`http://www.bobdylan.com`).

[7] Botspot. (`http://www.botspot.com`).

[8] Daniel Brookshier, Darren Govoni, and Neveneeth Kirshnan. *JXTA: Java P2P Programming*. SAMS, 2002. ISBN 0-672-32366-4.

[9] R. Scott Cost, Ian Soboroff, Jeegar Lekhani, Tim Finfin, Ethan Miller, and Charles Nicholas. TKQML: A Script Tool for Building Agents. Technical report, Knowledge System Laboratory, Stanford University, 1997.

[10] Jonathan Dale and David C. DeRourde. A Mobile Agent Architecture for Distributed Information Management. Technical report, Multimedia Research Group, Department of Electronics and Computer Science, University Southampton, UK, March 1997.

[11] Ian Darwin. *Java Cookbook*. O'Reilly, 2001. ISBN 0-59400-170-3.

[12] Dewey Decimal Classification System . (`http://www.oclc.org/dewey`).

[13] Tina Eliassi-Rad. Building intelligent agents that learn to retrieve and extract information. Technical report, University of Wisconsin, Madison, 2001.

[14] Eyebrowse Homepage. (`http://eyebrowse.tigris.org`).

[15] Richard Fikes, Adam Farquhar, and Wanda Pratt. Information Brokers: Gathering Information from Heterogeneous Information Sources. Technical report, Knowledge System Laboratory, Stanford University, 1996.

[16] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. KQML as an Agent Communication Language. Technical report, Computer Science Department, University of Maryland Baltimore County, 1998. (McKay and McEntire: Valley Forge Laboratory, Unisys Corporation).

[17] Timothy Finin and Charles Nicholas. Software Agents for Information Retrieval. Technical report, University of Maryland Baltimore County, 1999.

[18] David Flanagan. *Java Foundation Classes*. O'Reilly, 2000. ISBN 3-89721-191-2.

[19] David Flanagan. *Java in a Nutshell*. O'Reilly, 2000. ISBN 3-89721-190-4.

[20] Stan Franklin and Art Graesser. What is an agent? Technical report, (Institute for Intelligent Systems, University of Memphis), unknown.

[21] Norbert Fuhr. *Information Retrieval Skript*. Universität Dortmund, NRW, Germany, 2002.

[22] Norbert Fuhr and Norbert Gövert. Ein Agentensystem für digitale Bibliotheken im WWW. Technical report, Universität Dortmund, NRW, Germany, 1999. (`http://ls6-www.sc.uni-dortmund.de/ir`).

[23] Wolf Garbe. BINGOOO - Die Transformation des World Wide Web zur vitruellen Datenbank. Technical report, BINGOOO, 2001. (`http://www.bingooo.com`).

[24] Gnutella Homepage. (`http://www.gnutella.com`).

[25] Li Gong. Project JXTA: A Peer-to-Peer Network Programming Environment. Technical report, Sun Microsystems, Inc., 2001.

[26] Li Gong. Project JXTA: A Technology Overview. Technical report, Sun Microsystems, Inc., 2001.

[27] Homepage des Finnischen Militärs. (`http://www.mil.fi`).

[28] Jeremy Hylton and Guido van Rossum. Using the Knowbot Operating Environment in a Wide-Area Network. Technical report, Corporation for National Research Initiatives, 1997.

[29] Informationsforum Dewey-Dezimalklassifikation Deutsch. (`http://www.ddb.de/professionell/ddc_info.htm`). Die Deutsche Bibliothek.

[30] Internets. (`http://www.internets.com`).

[31] Invisible Web. (`http://www.invisibleweb.com`).

[32] Jakarta Lucene Homepage. (`http://jakarta.apache.org/lucene`).

[33] Thorsten Joachims. A probabilistic analysis of the Roccio Algorithm with TFIDF for text categorization. Technical report, School of computer science, Carnegie Mellon University, Pittsburgh, 1996.

[34] JXTA Homepage. (`http://www.jxta.org`).

[35] Werner Kinnebrock. *Optimierung mit genetischen und selektiven Algorithmen*. Oldenburg Verlag, 1994. ISBN 3-48622-697-5.

[36] Dave Kor and Kian Wei. Thread: [Lucene-users] request for detailed score computation explanation. GeoCrawler - The knowlege archive (`http://www.geocrawler.com`), 2001. Message Number: 6302110.

[37] Doug Lea. *Concurrent Programming in Java - Design Principles and Patterns*. Addison-Wesley Publishing Company, 1997. ISBN 3-8273-1243-4.

[38] Dieter Merkl. Information Retrieval - Repräsentation von Texten. Technical report, Institut für Softwaretechnik, Technische Universität Wien, 1999.

[39] Klaus Meyer-Wegener. Auffinden von elektronisch gespeicherten Dokumenten in Datenbanken und im World-wide Web. Technical report, Institut für Betriebssysteme, Datenbanken und Rechnernetze, Fakultät Informatik, Technische Universität Dresden, 1998. Vortrag am Fraunhofer-Institut für Naturwissenschaftlich-Technische Trendanalysen (INT, Euskirchen).

[40] MIME Types RFC 2045, 2046. (`http://www.nacs.uci.edu/indiv/ehood/MIME/MIME.html`).

[41] Alberto Montresor. Anthill: A Framework for Design and Analysis of Peer-to-Peer Systems. Technical report, Computer Science Department, University of Bologna, 2001.

[42] Alberto Montresor. Introduction to Peer-to-Peer. Technical report, Computer Science Department, University of Bologna, 2001.

[43] Alberto Montresor. The Anthill Framework. Technical report, Computer Science Department, University of Bologna, 2001.

[44] Alberto Montresor, Özalp Babaoğlu, and Hein Meling. Anthill: a Framework for the Development of Agent-Based Peer-to-Peer Systems. Technical report, Computer Science Department, University of Bologna, 2001. (Meling: Department of Telematics, Norwegian University of Science and Technology).

[45] Mikihiko Mori and Seiji Yamada. Sharing bookmarks among same interest persons. Technical report, CISS, IGSSE, Tokyo Institute of Technology, 2002.

[46] Napster Homepage. (`http://www.napster.com`).

[47] Charles Nicholas and R. Scott Cost. CARROT 2 Project Page. (`http://www.csee.umbc.edu/~cost/carrot2/`), 2002. Center for Architectures for Data-driven Information Processing, University of Maryland Baltimore County.

[48] Charles Nicholas, R. Scott Cost, Srikanth Kallurkar, Hemali Majithia, and Youngmei Shi. Integrating Distributed Information Sources with CARROT 2. Technical report, Center for Architectures for Data-driven Information Processing, University of Maryland Baltimore County, 2002.

[49] Charles Nicholas and Grace Crowder. Meta-Data for Distributed Text Retrieval. Technical report, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, 1997.

[50] Holger Nohr. Automatische Dokumentenindexierung - Eine Basistechnologie für das Wissensmanagement. Technical report, Lehrgebiet Informationswirtschaft, Fachhochschule Stuttgart, 2000.

[51] Holger Nohr. Automatische Verfahren der Dokumentanalyse. Technical report, Lehrgebiet Informationswirtschaft, Fachhochschule Stuttgart, 2001.

[52] Open Management Group (OMG). (`http://www.omg.org`).

[53] Andy Oram et al. *Peer to Peer: Harnessing the Benefits of a Distriputive Technology*. O'Reilly, 2001. ISBN 0-596-00110-x.

[54] Ulrich Pinsdorf. Mobile Agenten. Fraunhofer Institut für Graphische Datenverarbeitung, Darmstadt, 2001. Vortrag an der Fachhochschule Bingen.

[55] Ulrich Pinsdorf, Jan Peters, Mario Hoffmann, and Piklu Gupta. Context-Aware Services based on Secure Mobile Agents. In Nikola Rozic and Dinko Begusic, editors, *10th International Conference on Software, Telecommunications & Computer Networks (SoftCOM 2002)*, pages 366–370, University of Split, R. Boskovica, HR-21000 Split, Croatia, October 2002. IEEE Communication Society, Ministry of Science and Technology Republic of Croatia and University of Split, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture. ISBN 953-6114-52-6.

[56] Ulrich Pinsdorf and Volker Roth. Mobile Agent Interoperability Patterns and Practice. In *Proceedings of Ninth IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS 2002)*, Computer Graphics Edition, pages 238–244, University of Lund, Lund, Sweden, April 2002. Institute of Electrical and Electronics Engineers, IEEE Computer Society Press. ISBN 0-7695-1549-5.

[57] Python Homepage. (`http://www.python.org`).

[58] Peter Rantasa and Christian Doegl. The European Music Navigator: To Build Bridges Between Local, National and Global Cultures Using The Ontology Based Search Technology Melvil^{Tm}. In *1st International Conference on WEB Delivering of Music Scores IEEE ISBN: 0-7695-1284-4*, November 2001. (Rantasa: IAMIC - International Association Of Music Information Centres), (Doegl: uma Information Technologie AG).

[59] Peter Rausch. *Objektorientierte Systementwicklung mit der Unified Modeling Language (UML)*. Fachhochschule Bingen, 2000.

[60] Peter Rausch. *Konzepte der Objektorientierung*. Fachhochschule Bingen, 2001.

[61] Andreas Reuter. Studienarbeit - Publik Key Infrastrukturen. Technical report, Fachhochschule Bingen, 2001.

[62] Bradley J. Rhodes. Just-In-Time Information Retrieval. Technical report, MIT Media Lab, Cambridge, MA, 2000.

[63] Bradley J. Rhodes and Pattie Maes. Just-In-Time Information Retrieval Agents. Technical report, MIT Media Lab, Cambridge, MA, 2000. published in IBM SYSTEMS Journal.

[64] Volker Roth. Security in SeMoA Version 2: An overview. Technical report, Fraunhofer Institut für Graphische Datenverarbeitung, Darmstadt, 2000.

[65] Volker Roth. *Sichere verteilte Indexierung und Suche von digitalen Bildern*. Ph.D. thesis, Technische Universität Darmstadt, Wilhelminenstraße 7, 64283 Darmstadt, Germany, June 2001.

[66] Volker Roth, Mehrdad Jalali, Roger Hartman, and Christophe Roland. An Application of Mobile Agents as Personal Assistents in Electronic Commerce. Technical report, Fraunhofer Institut für Graphische Datenverarbeitung, Darmstadt, 2001. (Roland: Thomson-CSF Communications, Gennevilliers Cedex, France).

[67] Volker Roth and Jan Peters. A Scalable and Secure Global Tracking Service for Mobile Agents. In *Proc. Mobile Agents 2001*, Lecture Notes in Computer Science. Springer Verlag, December 2001.

[68] Volker Roth and Ulrich Pinsdorf. SeMoA Installation and Configuration Guide. Technical report, Fraunhofer Institut für Graphische Datenverarbeitung, Darmstadt, 2001.

[69] James Rumbough, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual (Addison-Wesley Object Technologie Series)*. Addison-Wesley Publishing Company, 1998. ISBN 0-20130-998-x.

[70] James Rumbough, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language User Guide*. Addison-Wesley Publishing Company, 1998. ISBN 0-20157-168-4.

[71] Daniela Rus, Robert Gray, and David Kotz. Transportable Information Agents. Technical report, Department of Computer Science, Dartmouth, Hanover NH, 1997.

[72] Kai-Uwe Sattler. Software-Agenten zur Informationssuche und -filterung. Technical report, Institut für Technische und Betriebliche Informationssysteme, Otto-von-Guericke-Universität Magdeburg, 1998.

[73] SeMoA Homepage. (`http://www.semoa.org`).

[74] Seti@home Homepage. (`http://setiathome.ssl.berkeley.edu`).

[75] Ayse Yasemin Seydim. Intelligent agents: A data minig perspective. Technical report, Department of Computer Science and Engineering, Southern Methodist University,Dallas, TX, 1999.

[76] Clay Shirky. What is P2P and what not. Technical report, The Accelerator Group, 2000. (`http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html`).

[77] The Library's Index to the Internet. (`http://www.lii.org`).

[78] Christian U. Ullenboom. *Java ist auch eine Insel*. Galileo Press, 2002. ISBN 3-89842-174-0.

[79] unknown. Overview of the Knowbot Operating Environment Software. Technical report, Corporation for National Research Initiatives, 1998.

[80] unknown. Peer-to-Peer: A Security Nightmare or a Secure Opportunity? Technical report, Endeavors Technology, 2001. (`http://www.endeavors.com`).

[81] unknown. Project JXTA: Java Programmer's Guide. Technical report, Sun Microsystems, Inc., 2001.

[82] unknown. Similarity. (`http://jakarta.apache.org/lucene`), 2002. Package: org.apache.lucene.search.

[83] Michael Wooldridge and Nicholas R. Jennings. Intelligent Agents: Theory and Practice. Technical report, (Wooldridge: Department of Computing, Manchester Metropolitan University), 1995. (Jennings: Department of Electronic Engineering, Queen Mary & Westfield College.

[84] Michael Wooldridge and Nicholas R. Jennings. Applications of intelligent Agents. Technical report, Queen Mary & Westfield College, University of London, 1998.

[85] Jihoon Yang, Prashant Pai, Vasant Honavar, and Les Miller. Mobile intelligence Agents for Document Classification and Retrieval: A Machine Learning Approach. Technical report, AI Research Group, Department of Computer Science, Iowa State University, 1998.

[86] Anne Zieger. The impact of Peer-to-Peer on data management issues for developers. Technical report, PeerToPeerCentral.com, 2001.