Jubiläumsfonds der Oesterreichischen Nationalbank

Endbericht zu Projekt Nr. 7742:

Ein Framework für intelligente Agenten, basierend auf gegenseitigem Vertrauen

1. Aufgabenstellung

Bei Agentensystemen (Gruppe von Programmen, die autonom einzelne Aufgaben erfüllen und dabei zusammenarbeiten) wird primär davon ausgegangen, daß alle Teil-Agenten immer für das gemeinsame Wohl arbeiten und z. B. niemals absichtlich falsche Aussagen machen. Dies ist zwar bei geschlossenen Systemen sehr hilfreich, ist jedoch in bestimmten Bereichen nicht einsetzbar.

Demgegenüber müssen Agenten, die frei im Internet ihre Aufgaben erfüllen und dabei von Server zu Server umherwandern sollen, besonders geschützt sein und auch mit böswilligen Absichten umgehen können. Dazu ist es notwendig, daß, wie im normalen Leben, eine Vertrauensbeziehung zwischen den Partnern an einer Transaktion hergestellt wird. Dies kann einerseits durch eine zentrale Zertifizierung erfolgen (bisher lediglich zur Identitätsprüfung üblich), andererseits ist aber auch ein Modell denkbar, das auf der Weitergabe von Vertrauen basiert. Dieses "verteilte Vertrauen" besteht darin, daß Agenten sowohl gute als auch schlechte Erfahrungen, die sie mit anderen Agenten gemacht haben, an wiederum andere Agenten weitergeben. Dies schützt zwar nicht sicher vor Schaden, doch kann die notwendige Stärke des Vertrauens beliebig festgelegt werden (z. B. für die Wahl Geschäftspartner oder die Sicherheitsvorkehrungen bei der Abwicklung).

Es wurde untersucht, welche Voraussetzungen und Vorkehrungen zu implementieren sind, um die nötige Sicherheit zu gewährleisten, daß Agenten auch mit Geld (z. B. E-Cash) autonom und sicher durch das Internet wandern und dort Geschäfte abschließen können. Weiters wurde der Ansatz des "verteilten Vertrauens" untersucht.

2. Kurz-Zusammenfassung

Ein Framework für mobile Agenten wurde in Java erstellt und getestet. Agenten werden voreinander sowie der Server vor den Agenten geschützt. Für letzteres können Berechtigungen dynamisch vergeben werden, je nach den Informationen, welche der Agent preisgibt (So erhalten anonyme Agenten nur sehr geringe Berechtigungen. Identifizieren sie sich später so werden sie höher eingestuft und erhalten mehr Möglichkeiten). Über dieses System können auch zusätzliche „Qualifikationen“ wie Stammkundenkarten oder Gutscheine realisiert werden.

Agenten können gegenüber dem Server nur in der Hinsicht geschützt werden, daß ein Agent sich aussuchen kann, auf welchen Server er verlagert wird. Versucht das System (oder ein Benutzer) ihn ungewollt zu transportieren, so kann er dies verweigern.

Zwei prototypische Applikationen wurden erstellt, um die Praxistauglichkeit des Frameworks und seiner Elemente zu testen, sowie tatsächliche Anwendung zu finden:

1. Ein Agentensystem zur Nachrichten-Bearbeitung wurde implementiert. Von mehreren Quell-Agenten (z. B. POP Abfrage von E-Mail, Newsgroups oder Webseiten aus dem Internet) werden Nachrichten erstellt und an Filter-Agenten weitergeleitet. Diese entscheiden über die durchzuführenden Aktionen (etwa Antworten, Weiterleiten, Benutzer benachrichtigen) und führen diese teilweise selbst, teilweise über andere Agenten durch (Beispiel: Benachrichtigung des Benutzers auf einem anderen Rechner mittels einer Message-Box). Welche Agenten für welche Tätigkeiten verwendet werden (sowohl auf der Quell- wie auch auf der Aktions-Seite), wird über das Erfahrungs-System gesteuert: Jeder Agent speichert seine eigenen Erfahrungen mit einzelnen anderen Agenten und teilt diese den anderen Agenten mit. Dadurch fallen über längere Zeit Agenten mit schlechten Ergebnissen oder auch falschen Aussagen weg und werden nicht mehr beschäftigt.
2. Ein zweites System zur Suche nach Büchern im Internet ist zur Zeit noch in Entwicklung. Es soll bei verschiedenen Händlern im Internet nach einem bestimmten Buch suchen, die Preise vergleichen und schließlich beim günstigsten Anbieter dieses Buch kaufen. Hierzu werden ausschließlich die normalen Webseiten benutzt, indem WWW-Formulare vom Agenten ausgefüllt werden. Dies verhindert das Aussperren der Agenten zur Verhinderung der Vergleichs. Dieses System ist bis zur Anzeige des Buches mit den Daten der verschiedenen Anbieter entwickelt; der eigentliche Kauf ist noch nicht realisiert. Da das Projekt fortgesetzt wird, werden die fehlenden Teile in den nächsten Monaten entworfen und implementiert werden.

3. Ergebnisse

Die Ergebnisse bestehen aus drei großen Teilen: Dem Basis-Framework und zwei Beispiels-Systemen, welche unter Verwendung dieses Frameworks implementiert wurden. Alle drei Elemente werden im Folgenden näher dargestellt.

3.1. Das Basis-Framework für mobile Agenten

3.1.1. Zielsetzung

Das Ziel war es, ein Framework für mobile Agenten in der Programmierpsrache Java zu entwerfen und implementieren, welches die folgenden Eigenschaften besitzt:

- „Heavy-weight“-Agenten: Die einzelnen Agenten besitzen eine größere Aufgabenfülle und sind in der Lage, alleine komplexe Aufgaben durchzuführen. Die „Intelligenz“ des Gesamtsystems ergibt sich daher nicht nur aus der Interaktion einer Vielzahl von unintelligenten Einzel-Agenten.
- Seltene Transfers: Bei dem Agenten-System wurde die Mobilität eher in den Hintergrund gerückt. Der Hauptaspekt liegt in der lokalen Verarbeitung und eine Verlagerung findet nur in (relativ) seltenen Fällen statt, wenn es unbedingt notwendig ist. Dies auch deshalb, da es sich bei den Beispiels-Systemen um komplexere Aufgaben handelt, bei welchen eine dauernder Wechsel nicht sinnvoll ist. Dies ist auch darin begründet, daß der Transfer eines Agenten eine aufwendige Aufgabe ist, welche auch ein paar Minuten dauern kann (mit Verschlüsselung oder auf langsamen Rechnern noch länger).
- Schwache Mobilität: Ein Agent besitzt einen festen Einsprungpunkt, an dem die Verarbeitung nach einer Verlagerung fortgesetzt wird. Es ist nicht möglich, inmitten eines Programmteils den Rechner zu wechseln und automatisch beim folgenden Befehl weiterzuarbeiten. Dies wurde ausgeschlossen, da hierzu ein viel höherer Aufwand nötig

wäre (Ein weiterer komplexer Eingriff in das Java Framework). Dies ist keine besondere Einschränkung, da eine Verlagerung relativ selten erfolgt, dies vom Agenten genau geplant werden kann, und diese Funktion leicht über die Programmierung des Agenten ersetzt werden kann.

- ✍ Sicherung der Agenten voreinander: Die Agenten werden voreinander geschützt, indem jeder in einem eigenen Speicherbereich und mit eigenen Berechtigungen ausgeführt wird. Ein direkter Zugriff auf Felder oder Methoden eines anderen Agenten ist daher nicht möglich. Kein Agent ist in der Lage, einem anderen zu schaden oder ihn irgendwie zu beeinträchtigen. Die einzige Möglichkeit dazu besteht über die öffentliche Schnittstelle, indem an ihn Nachrichten (in großer Anzahl oder mit speziellem Inhalt) gesendet werden, was aber durch den Programmierer eines Agenten verhindert werden soll.
- ✍ Sicherung des Host-Rechners vor Agenten: Um den Agenten-Server vor den einzelnen Agenten zu schützen (da diese ja von anderen Benutzern stammen können und dennoch lokal ausgeführt werden), werden deren Aktionen stark eingeschränkt. Der Rechner vergibt Berechtigungen an die Agenten je nach den Informationen, die er vom Agenten erhält, seiner Konfiguration und bisherigen Erfahrungen. Das Sicherheitssystem basiert auf dem statischen Systems von Java, erlaubt jedoch auf Grund einer aufwendigen Anpassung dynamische Veränderungen (sowohl Erweiterungen wie auch Einschränkungen) der Berechtigungen. Dadurch kann sichergestellt werden, daß unbekannten Agenten nur diejenigen Aktionsmöglichkeiten offenstehen, welche keine Gefahr für den Rechner darstellen.
- ✍ Erfahrungs-System: Um Agenten die Möglichkeit zu geben, auf einfache Weise ihre eigenen Erfahrungen zu speichern und gleichzeitig auch die anderer Agenten einfließen zu lassen, ist ein System zur Speicherung und Auswertung von Erfahrungen vorgesehen. Aus den Aussagen der anderen Agenten im Vergleich zu den eigenen Ansichten wird auch eine Rückbewertung von Aussagen vorgenommen. Dies ermöglicht es einem Agenten, über einen ihm bisher völlig unbekannten Agenten gewisse (wenn auch relativ unsichere) Aussagen zu machen. Dies ist besonders wichtig, da in einem offenen System ein Agent nicht alle Anderen kennen kann.

3.1.2. Einschränkungen

Aus Zeit- und Personalgründen wurden einige Einschränkungen vorgenommen, die teilweise eigene Forschungs-Gebiete wären und daher im Rahmen dieses Projektes nicht im Detail zu untersuchen waren. Dies sind im Einzelnen:

- ? Keine Public-Key-Infrastruktur: Zur Identifikation der Besitzer von Agenten werden Zertifikate verwendet, ebenso als Identität von Agenten. Besitzer-Zertifikate können jedoch nicht erzeugt werden, sondern nur importiert. Wiederrufs-Listen sind nicht integriert, werden also (sofern überhaupt vorhanden) nicht überprüft. Identitäts-Zertifikate für Agenten werden jedoch im System erzeugt und verwaltet, doch sind auch hier keine Wiederrufs-Listen vorgesehen.
- ? Kein Expertensystem: Das Agenten-System beinhaltet kein Expertensystem, sondern Agenten besitzen grundsätzlich nur diejenige Intelligenz, die sie durch ihre Programmierung vom Hersteller erlangen. Für viele praktische Aufgaben ist dies auch ausreichend. Sollte eine Erweiterung in diese Richtung notwendig werden, so müßte sie in das Basis-System eingebaut werden, da ansonsten wegen der Programm- und Datengröße der Transfer eines Agenten auf ein anderes System nicht mehr praktikabel wäre (Siehe dazu auch Abschnitt 5).
- ? Trennung auch zusammengehöriger Agenten: Erzeugt ein Agent einen oder mehrere Sub-Agenten, so sind auch diese vollkommen voneinander getrennt und jede Kommunikation

muß über die normalen Schnittstellen erfolgen. Dies ist ein Nebeneffekt des Sicherheitssystems. Dadurch wird sichergestellt, daß jeder Agent eigene Berechtigungen besitzt und sich auch eigenständig bewegen kann. Werden die besonderen Möglichkeiten (Mobilität, Kommunikation, etc.) nicht benötigt, so kann ein Agent auch als normales Objekt erzeugt werden. In diesem Fall ist ein ungehinderter Zugriff darauf möglich.

3.1.3. Das Sicherheitssystem

Das Sicherheitssystem basiert auf dem Gedanken, daß jede Berechtigung und jede Ressource mit Geld bewertet werden kann. Bei Ressourcen kommt es auf die vorhandene Quantität und die derzeitige Belegung an, während Berechtigungen danach bewertet werden, welchen Schaden ein Mißbrauch verursacht, bzw. wie groß die Gefahr dafür ist. Manche Berechtigungen werden für einzelne Agenten gratis sein (z. B. Agenten des eigenen Betriebs), für andere kostenpflichtig (z. B. Stammkunden), und für wieder andere gar nicht erhältlich sein (z. B. Agenten unbekannter Personen oder anonyme Agenten). Berechtigungen werden auf Grund von zwei Komponenten vergeben: Der Besitzer des Agenten und der Hersteller des Programmcodes des Agenten. Beides wird durch Zertifikate und digitale Signaturen überprüft.

Gegenüber der binären Entscheidung, eine Berechtigung zu erteilen oder zu verweigern, besitzt dieses Modell einige Vorteile:

? Detaillierte Sicherheitseinstellungen: Jede Berechtigung kann nach der damit verbundenen Gefahr beurteilt und ihr ein entsprechender Wert zugewiesen werden, je nachdem von welcher Personengruppe er verwendet wird oder wer den Code herstellte. Beispielsweise werden Provider einem Agenten oft die Möglichkeit einräumen, Festplatten-Kapazität zu nutzen, wenn der dafür zu entrichtende Preis höher ist als die Erweiterung des zur Verfügung stehenden Platzes oder anderen Agenten temporär dies zu verweigern. Dies ermöglicht auch eine komplexere Preispolitik, sodaß verschiedene Preise für unterschiedliche Gruppen möglich sind.

? Leichte Verständlichkeit: Dieses System ist leicht zu verstehen und auch Managern leichter zu kommunizieren, die sich nur selten mit Sicherheits-Politiken befassen, aber versiert darin sind, Risiken zu bewerten und Preise festzulegen.

? Abrechnung: Dieses System kann leicht dazu verwendet werden, eine Abrechnung je nach Nutzung von bestimmten Ressourcen zu integrieren: Berechtigungen müssen nicht nur aus bestimmten Zugriffserlaubnissen bestehen, sondern können auch quantitative Elemente umfassen, etwa CPU-Zeit, Datentransfervolumina oder die Nutzung von Spezial-Hardware. Agenten müssen nur für die von ihnen tatsächlich verwendeten Dienste bezahlen und Server können ihre Ressourcen so am nutzbringendsten einsetzen.

Es existieren jedoch auch einige Nachteile dieses Systems:

? Keine Kredit-Klassifizierung: Agenten können nicht nach ihrer Kreditwürdigkeit klassifiziert werden. Es macht keinen Sinn, von einem Agenten höhere Preise zu verlangen, wenn es wahrscheinlicher ist, daß er überhaupt nicht bezahlen wird (Eine Klassifizierung nach Zahlungsweise ist jedoch sehr wohl möglich).

? Große Entscheidungsanzahl: Es müssen viele Einzelentscheidungen getroffen werden, um ein Sicherheitsprofil zu erstellen. Einzelne Berechtigungen sind für viele verschiedene Gruppen von Besitzern und Codehersteller zu untersuchen und bewerten. Es ist daher nötig, eine sinnvolle Anfangskonfiguration bereitzustellen, sodaß in vielen Fällen nur kleinere Anpassungsmaßnahmen notwendig sind, bzw. ganze Gruppen von Berechtigungen auf einmal manipuliert werden können.

? Keine Delegierung: Berechtigungen können von einem Agenten nicht an einen anderen weitergegeben werden. Eine Aufgabe ist daher entweder selbst durchzuführen oder an einen selbständigen Agenten zu übertragen, welcher die notwendigen Berechtigungen bereits besitzen oder separat für sich erwerben muß.

Bei dem implementierten Sicherheitssystem handelt es sich, im Gegensatz zu dem normalen Java-System, um ein dynamisches System von Berechtigungen: Viele Berechtigungen sollten nicht strikt durchgesetzt werden, sondern können auch Begrenzungen darstellen. Ein Beispiel ist die Menge an temporärem Festplatten-Speicher, die ein Agent verwenden darf. Auch der Agent selber ist oft nicht in der Lage, im Voraus zu bestimmen, welches Ausmaß sein Bedarf annehmen wird. Eine sinnvolle Lösung für dieses Problem ist, einen Grundbedarf zu kaufen und eine zusätzliche Quote, abhängig vom Vertrauen zu erhalten. Wird diese Quote überschritten, so muß der Agent Platz nachkaufen. Da über das Zertifikat des Agenten auch der Besitzer genau bekannt ist, besteht immer die Möglichkeit, den Besitzer direkt heranzuziehen (die Bezahlung ist daher gesichert). Da dies jedoch mit zusätzlichen Kosten und Mühen verbunden ist, muß der Agent für dieses Risiko extra bezahlen. Hat er in der Vergangenheit immer regelmäßig bezahlt, wird der Preis für ihn sinken, während andernfalls der Preis steigt und die Quote, für die Vertrauen entgegengebracht wird, sich verkleinert.

Um sowohl den statischen Aspekt der Vertrauenswürdigkeit (Code des Agenten) als auch den dynamischen zu integrieren, werden beide Elemente über Zertifikate zum Inhalt der Identität eines Agenten gemacht. Das Vertrauen in einen Agenten hängt von dem Vertrauen zu beiden Kategorien ab:

? Hersteller des Programmcodes: Der Hersteller des Programms eines Agenten hat es in der Hand, bestimmte Aktionen auf bestimmte Art durchzuführen, z. B. unter sparsamer Verwendung von Ressourcen. Er ist jedoch auch in gewissen Grenzen in der Lage, „gefährliche“ Einstellungen oder Parameter durch Benutzer zu erkennen und Schaden zu verhindern. Der Hersteller wird dadurch identifiziert, indem die Signatur des „packages“, in dem sich der Code befindet, überprüft wird. Dies dient auch dazu zu erkennen, ob der Code nachträglich abgeändert wurde, z. B. durch Viren.

? Besitzer des Agenten: Vielfach läßt sich auch vollkommen normaler und problemloser Code durch entsprechende Parameter des Benutzers dazu verwenden, unerwünschte Ergebnisse hervorzurufen. Dies kann einerseits technische Probleme verursachen (z. B. hohe Netzwerkbelastung), andererseits aber auch auf inhaltlicher Ebene (etwa Aussagen über andere Personen oder Agenten) Schwierigkeiten hervorrufen. Daher ist auch dieser Aspekt zu integrieren, was über die Identität des Agenten erfolgt, in der ein Zertifikat des Besitzers enthalten sein und welche mit dem zugehörigen privaten Schlüssel signiert sein sollte.

Beide Zertifikate müssen nicht unbedingt vorhanden oder von einer anerkannten Zertifizierungsstelle ausgestellt sein. Dies wird jedoch in den meisten Fällen dazu führen, daß nur sehr geringe oder gar keine Berechtigungen an solche Agenten vergeben werden. Dies zu ermöglichen ist jedoch sinnvoll, da es hiermit möglich ist, daß ein Agent anonym (d. h. ohne seinen Besitzer bekanntzugeben) Vorerhebungen durchführt und erst nach einer Entscheidung einem einzelnen Server (bzw. anderen Agenten) seine wahre Identität preisgibt.

Wie unter den Nachteilen angeführt, sind viele Entscheidungen zu treffen, wenn eine Sicherheits-Policy implementiert werden soll. Um dies in der Praxis zu erleichtern steht eine zusätzliche Unterteilung zur Verfügung. Berechtigungen werden indirekt in einer Matrix gespeichert, an deren Achsen jeweils verschiedene Gruppen von Vertrauen in Besitzer und Code-Hersteller notiert werden (Siehe Abbildung 1). In manchen Zeilen oder auch Spalten werden die selben Berechtigungen notiert werden, was durch eine Abbildung auf Gruppen vereinfacht wird. Jede dieser Gruppen besteht wiederum aus zwei Teilen: Berechtigungen, welche ein Agent dieser Gruppe auf jeden Fall erhält (für normalen Betrieb notwendig oder

zugestanden), und optionale Berechtigungen. Für letztere ist entweder ein Preis zu entrichten, oder, zur besseren Auslastung des System, zumindest eine Reservierung vorzunehmen (Preis=0).

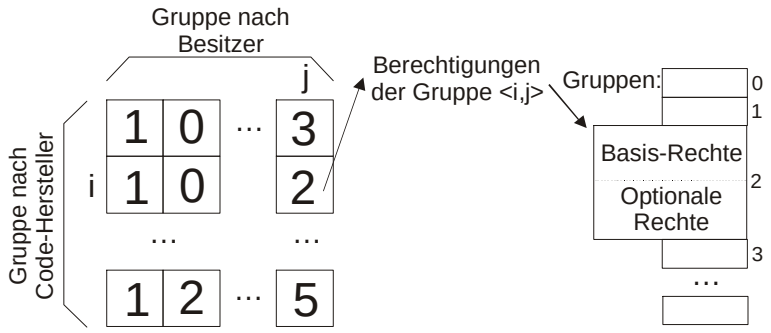

Abbildung 1: Zuordnung von Rechten nach Besitzer und Code-Hersteller

Ein Beispiel für die Achseneinteilung könnte sein (für Besitzer und Hersteller gleich, jedoch auch asymmetrische Gruppen möglich):

1. Zertifikat vorhanden, aber Prüfung fehlgeschlagen (Modifikation, abgelaufen, etc.)
2. Kein Zertifikat
3. Zertifikate vorhanden, aber technisch nicht prüfbar (Algorithmus unbekannt, keine Verbindung zur Zertifizierungsstelle, …)
4. Zertifikat vorhanden, aber nicht von einer bekannten Zertifizierungsstelle ausgestellt
5. Zertifikat vorhanden und von einer bekannten Zertifizierungsstelle ausgestellt, aber keine Widerrufsprüfung möglich (oder aus Zeit-/Effizienzgründen nicht durchgeführt)
6. Zertifikat vorhanden, von einer bekannten Zertifizierungsstelle ausgestellt, Widerrufsprüfung erfolgreich durchgeführt

Diese Einteilung würde theoretisch zu 36 Gruppen führen, doch können etwa alle 11 Felder, bei denen eine der beiden Achsen der Gruppe 1 entspricht, mit der selben Gruppe belegt werden (welche vermutlich keine einzige Berechtigung enthält).

3.1.4. Mobilität

Wie bereits oben erwähnt, handelt es sich bei der implementierten Form um sogenannte „schwache Mobilität“. Dennoch wird der gesamte Programmcode mitgenommen, sodaß es einem Agenten auch möglich ist, völlig fremde Rechner zu besuchen. Es ist daher das Programm-Paket für einen Agenten jeweils nur auf dem Rechner zu installieren, auf welchem der Agent gestartet werden soll.

Jeder Agent hat seinen gesamten Code mitzunehmen. Dies führt einerseits dazu, daß bei vielen Agenten Code mehrfach übertragen wird (Produzent-Agent und Konsument-Agent besitzen oft viel gleichen Code, zumindest für die gegenseitige Kommunikation), andererseits hat es zur Folge, daß keine Probleme mit verschiedenen Programm-Versionen entstehen. Jeder Agent führt seine eigene private Version mit sich. Selbst bei der Kommunikation müssen die Klassen beim Sender und dem Empfänger nicht identisch sein, falls der Empfänger zumindest die Version des Senders verstehen kann. Nimmt daher ein „alter“ Agent Kontakt mit einem „neuen“ Agenten auf, so muß der neue Agent seine eigenen Nachrichten in der alten Form codieren, um seinem Partner das Lesen dieser zu ermöglichen.

Technisch erfolgt die Code-Übergabe indem ein Archiv (JAR-Datei) aus allen Klassen erstellt wird, welches vom Code-Hersteller signiert wird. Da es sich hierbei um eine wichtige Funktion handelt, insbesondere auch für das Testen und die Verwendung eigener Klassen, ist diese Signierung auch über das Agenten-GUI möglich. Um bei einem Test dies nicht jedes

Mal durchführen zu müssen, können Agenten auch von dem lokalen Dateisystem (anstatt aus einem Archiv) gestartet werden. Soll ein solcher Agent auf einen anderen Rechner verlagert werden, so wird das gesamte Unterverzeichnis, in welchem sich die Hauptdatei (die Agenten-Klasse) befindet, in ein neues Archiv gepackt und ohne Signatur übertragen (sofern der Empfangs-Rechner einen solchen Agenten überhaupt zuläßt).

Die Übertragung des Programmcodes und des dynamischen Zustandes eines Agenten erfolgt verschlüsselt, nachdem sich beide Rechner dem jeweils anderen gegenüber mit einem Zertifikat ausgewiesen haben. Dies vermeidet auch den sogenannten „Man-in-the-middle-attack“, sodaß auch ein dazwischen geschalteter Rechner die Daten weder lesen noch verändern kann. Vor der eigentlichen Übertragung erhält der Zielrechner schon einige Informationen über den Agenten, um entscheiden zu können, ober er akzeptiert wird oder nicht: Name und Klasse des Agenten (mehr informativer Natur) sowie Code-Signatur-Zertifikate und die Identität des Agenten (worin das Zertifikat des Besitzers enthalten ist). Hierbei handelt es sich nur um eine grundlegende Entscheidung, welche später noch zurückgenommen und der Agent doch noch abgelehnt werden kann. Dies tritt dann ein, nachdem der gesamte Programmcode und der Zustand des Agenten übertragen wurde und entweder die Prüfung der Code-Signatur fehlschlägt oder der Agent nicht in der Lage ist, eine Botschaft mit seinem privaten Schlüssel zu signieren (womit er die Kenntnis von diesem nachweisen kann)[1].

Jedoch ist es auch einem Agenten möglich, eine Transferierung zu verhindern. Bevor diese tatsächlich durchgeführt wird, wird eine bestimmte Methode des Agenten aufgerufen, um ihm die Möglichkeit zu Vorbereitungsarbeiten (z. B. entfernen von Fenstern, etc.) zu geben. Er hat jedoch auch die Möglichkeit, dem Agentensystem zu signalisieren, daß er keine Übertragung wünscht. Hiermit kann u. U. auch eine Verzögerungszeit verbunden sein, etwa um ihm die Möglichkeit zu geben, bestimmte Arbeiten noch vorher abzuschließen. Im derzeitigen System (änderbar durch Subklassen des Agentensystems) ist dies auf jedem Rechner jedoch nur ein einziges Mal möglich und die Verzögerung ist nach oben hin begrenzt (entspricht aber sonst der gewünschten Zeitspanne). Verweigert der Agent auch beim zweiten Versuch die Verlagerung, so wird er terminiert. Diese harte Maßnahme ist nötig um zu verhindern, daß ein Agent das Agentensystem blockiert und es so etwa am Beenden hindert.

Um aus dem übertragenen Archiv wieder ein lauffähiges Programm zu extrahieren, wurde ein eigener Classloader implementiert, welcher auch für die Einhaltung der Sicherheits-Richtlinien dient. Teil-Programme werden von verschiedenen Orten und von verschiedenen Classloadern geladen, weshalb die korrekte Konfiguration hier für die Sicherheit besonders wichtig ist:

1. Klassen des Agenten-Systems und spezielle Utility-Klassen (Kryptographie): Diese werden als sicher angesehen und jeder Agent kann alle diese Klassen erzeugen. Sie werden für alle Agenten vom selben Classloader geladen und unterliegen nicht den Restriktionen, die für den Agenten gelten.
2. Gemeinsame Bibliotheken sowie JDK: Diese Klassen werden vom Agenten-Classloader geladen und unterliegen voll den Einschränkungen der Berechtigungen der Agenten. Sie werden (logisch) zu den eigenen Klassen des Agenten gezählt. Da sie von praktisch jedem Agenten benötigt werden sind sie lokal gespeichert, um zu verhindern, daß sie mit jedem Agenten bei einer Verlagerung mit übertragen werden müßten.

[1] Eine Signatur wird deshalb verwendet, da ein Agent zwar Signaturen herstellen können muß, aber nicht unbedingt Verschlüsselung zu beherrschen hat. Da Signaturverfahren nicht immer zur Verschlüsselung geeignet sind (z. B. DSA), wird hier eine Signatur herangezogen. Um damit keine Signaturen erschleichen zu können (gewünschter Text wird bei der Überprüfung eingereicht), werden sowohl am Anfang als auch am Ende zusätzliche Zeichen eingefügt, womit die sonstige Bedeutung (hoffentlich) zerstört wird.

3. Agenten-Sammlung: Hierbei handelt es sich um ein Verzeichnis, in welchem sich die lokal installierten Agenten befinden. Soll ein neuer Agent gestartet werden, so wird hier danach gesucht. Wurde die Haupt-Klasse gefunden, so wird der Zugriff ab diesem Zeitpunkt auf das entsprechende Archiv oder Unterverzeichnis und die gemeinsamen Bibliotheken beschränkt. Sie unterliegen den Sicherheitsrichtlinien für die Agenten und werden durch den Agenten-Classloader geladen.
4. Temporärer Agenten-Code: Hier werden die empfangenen Code-Pakete von transferierten Agenten gespeichert. Jeder Agent hat nur auf sein eigenes Paket (neben den gemeinsamen Bibliotheken) Zugriff, welches nach seiner Beendigung (entweder Terminierung oder Verlagerung auf einen anderen Rechner) gelöscht wird. Auch dieser Code wird durch den Agenten-Clasloader geladen und unterliegt voll den Sicherheits-Richtlinien.

Die Identifikation eines Agenten-Systems (und damit von Rechnern, auf die ein Agent übertragen werden kann) erfolgt durch den Hostnamen des Rechners und eine Portnummer. Auf diese Weise ist es möglich, auf einem Host verschiedene Agentensysteme gleichzeitig laufen zu lassen, welche verschiedenen Einstellungen unterliegen können. Nicht vorgesehen ist ein Directory-System um festzustellen, welche Agenten-Systeme existieren, da dies in einem offenen System praktisch nicht möglich ist. Ein Agent (oder der Benutzer bei interaktiver Bedienung) muß daher selbst die Rechner kennen, auf die er sich hinverlagern möchte. Es ist jedoch auch ohne weiteres ein stationärer Agent denkbar, dem jeder ankommende Agent die ihm bekannten Systeme bekanntgibt, sodaß jeweils eine lokale Liste (zumindest bis vor kurzem bestehender System) entsteht.

3.1.5. Kommunikation zwischen Agenten

Die Kommunikation zwischen Agenten erfolgt ausschließlich über Message-Passing. Dies einerseits deshalb, da es eine Isolierung der Agenten voreinander erlaubt und so viele potentielle Sicherheitslücken von vornherein schließt, und andererseits, da ein direkter Methodenaufruf praktisch nur sehr erschwert möglich ist, da jeder Agent von einem eigenen Classloader geladen wurde (Methoden könnten zwar aufgerufen werden, doch würde ein Objekt einer bestimmten Klasse nicht als zu dieser Klasse zugehörig erkannt werden). Um daher Parameter zwischen Agenten auszutauschen, wird die Nachricht beim Sender serialisiert und beim Empfänger wieder zusammengesetzt. Hierbei stellt sich auch sofort heraus, ob der Empfänger (zumindest potentiell) in der Lage ist, die Nachricht zu verarbeiten: Er muß die benötigten Klassen in seinem eigenen Code-Package besitzen oder schon dieser Vorgang schlägt fehl.

Um lokal einen Agenten mit bestimmten Fähigkeiten zu finden ist es möglich einen lokalen Broadcast zu versenden. Dieser wird an alle lokalen Agenten weitergeleitet. Jedem Agenten steht es frei, darauf zu antworten oder nicht (sofern er die Nachricht überhaupt versteht). Auf diese Weise ist es den Agenten selbst überlassen, den (ihrer Ansicht und ihrem Erfahrungs-System nach) besten Agenten auszuwählen.

Aufgrund des Konzeptes von „Heavy-weight“-Agenten ist eine Kommunikation nur zwischen lokalen Agenten möglich, um den stärkeren Zusammenhang zu verdeutlichen. Eine Erweiterung zu einer Kommunikation mit entfernten Agenten wäre technisch ohne großen Aufwand möglich, doch stellen sich hierbei mehrere Probleme:

? Agenten müssen wissen, welche Agenten sich genau auf welchem Rechner befinden. Dies würde, da es sich um mobile Agenten handelt, eine „Registratur“ erfordern, die andauernd aktualisiert werden müßte. Da es sich um ein offenes System handelt, besteht natürlich rekursiv die Schwierigkeit, diese Registratur zu finden, was zu einer Hierarchie von Registrierungsstellen und einer Struktur ähnlich dem Domain Name System (DNS) führen würde. Als Alternative könnte das Weiterleiten von Botschaften an abgewanderte Agenten

verwendet werden, doch schlägt dies fehl, wenn ein Agentensystem beendet wird und bedeutet einen zusätzlichen Aufwand.

? Das finden von Agenten per Broadcast ist nicht mehr möglich. Es müßten dann nicht nur Agenten sondern auch Agentensysteme registriert werden, und die Anzahl der notwendigen Kommunikationsverbindungen würde exponential ansteigen. Bei einer größeren Anzahl von Agentensystemen wären diese nur mehr mit dem Absenden und Empfangen von Broadcasts beschäftigt und würden damit auch die dazwischenliegenden Netzwerke stark belasten.

? Es entstehen Sicherheitsprobleme, da nun ein Denial-of-Service Angriff auf einen Rechner möglich ist, indem er mit Nachrichten für enthaltene Agenten überschwemmt wird.

Natürlich steht es jedem Agenten frei, selbst eine Kommunikationsverbindung zu einem anderen Rechner (und damit auch zu anderen Agenten) aufzubauen, sofern er die hierfür notwendigen Berechtigungen besitzt.

Die Verarbeitung von Nachrichten durch die Agenten basiert auf Konversationen in Verbindung mit ereignisorientierter Nachrichtenverarbeitung. Hierbei handelt es sich um einfache Zustandsautomaten, welche jedesmal bei Eingang einer Nachricht diese als Eingabe erhalten und daraus bestimmte Aktionen ableiten und einen neuen Zustand annehmen. Dies ist notwendig, da ein Agent auch mitten während einer Konversation auf Benutzereingaben oder Aktionen des Agentensystems reagieren muß. Bei einer solchen Konversation kann es sich um sehr lang dauernde Aufgaben handeln, etwa die Beauftragung eines anderen Agenten damit, eine ganze Website zu laden, sodaß der Agent sonst u. U. für mehrere Stunden blockiert wäre; ein Zustand der keinem Benutzer zugemutet werden kann. Aufgrund dieser Eigenschaft ist es für einen Agenten auch möglich, gleichzeitig mehrere (verschiedene, aber auch gleiche) Konversationen mit einem oder unterschiedlichen Agenten zu führen und somit verschiedene Aufgaben zu einem Zeitpunkt in verschiedenen Ausführungsstufen zu bearbeiten.

Einem Agenten-Programmierer steht es jedoch frei, seinen Agenten auch so zu programmieren, daß eine bestimmte Verarbeitung ausgeführt wird und der Agent anschließend terminiert. In diesem Fall kann auf die Behandlung von Nachrichten vollkommen verzichtet werden oder etwaige Kommunikation mit anderen Agenten selbständig durchgeführt werden. Hierbei ist der Programmierer dafür verantwortlich, daß der Agent dem Benutzer die Möglichkeit gibt, Veränderungen vorzunehmen (soweit der Agent in so einem Fall überhaupt ein graphisches Interface besitzt; dies ist nicht obligatorisch). Dies ist beispielsweise für kurze Aufgaben sinnvoll, welche geringe Parameter brauchen, also besonders für Hilfs-Agenten, welche von anderen Agenten erzeugt werden.

3.1.6. Die Benutzeroberfläche (GUI)

Da es sich um eine Forschungs- und Testanwendung handelt, ist die Benutzeroberfläche eher spartanisch gehalten, beinhaltet jedoch die Möglichkeit, alle Funktionen des Agentensystems durchzuführen und (für Benutzer teilweise zu spezielle) Informationen darzustellen.

Über das Datei-Menü kann das gesamte System gespeichert und wieder geladen werden, um so mehrere zusammengehörige Agenten auf einmal zu speichern und später nicht wieder einzeln erzeugen zu müssen. Im Sicherheitsmenü können sowohl CA-Zertifikate importiert als auch Dateien signiert werden, wobei nur Besitzer-Zertifikate zur Verfügung stehen. Über das Agenten Menü können neue Agenten erzeugt werden, jedoch muß der genaue Name der Agenten-Klasse bekannt sein (Daher ist aber auch keine Registrierung bei der Installation eines neuen Agenten erforderlich). Über nachfolgende Dialoge können die Identität

ausgewählt bzw. neue Identitäten erzeugt werden. Hierzu können X.509 Zertifikate (für Besitzer) importiert sowie neue kryptographische Schlüssel erzeugt werden.

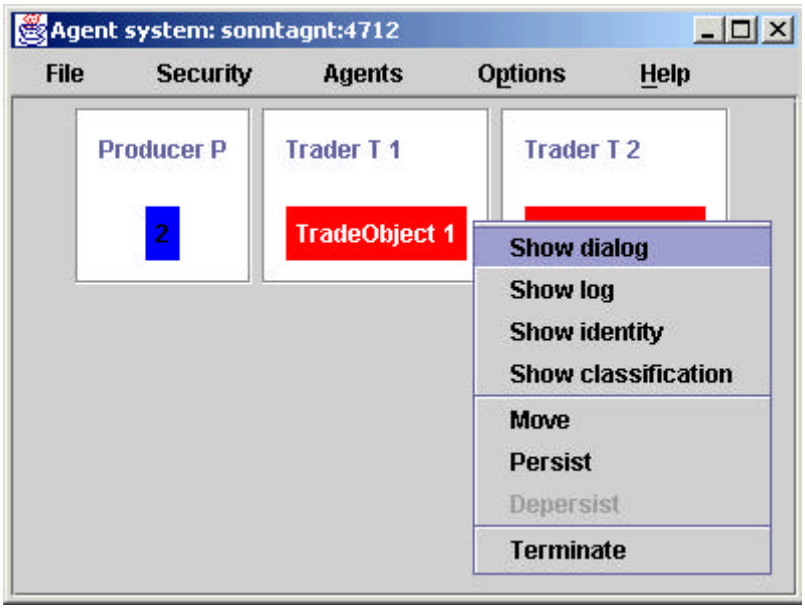

Abbildung 2: Agentensystem Benutzeroberfläche (mit drei Agenten)

3.2. Das Nachrichten-Verarbeitungssystem

3.2.1. Zielsetzung

Es wurde ein System zur Verarbeitung von Nachrichten aller Art entworfen, welches jedoch keineswegs universell für alle Arten von Nachrichten verwendbar sein soll. Zu berücksichtigen ist, daß unter „Nachricht“ viele verschiedene Konzepte versatnden werden. Das Grundkonzept entspricht E-Mail Nachrichten, welche vollständig (Multipart, Attachments, etc.) abgebildet werden. Es können jedoch auch andere Inhalte verwendet werden, wie etwa Artikel aus Newsgruppen (sehr ähnlich zu E-Mail), Webseiten (Seitentext als Nachrichteninhalt, Meta-tags als Header, …) aber auch selbst erzeugte Texte, wie etwa Börsenkurse, welche aus Webseiten extrahiert wurden. Nachrichten können aus verschiedenen Quellen stammen, wo sie von einzelnen Agenten (jeweils ein eigener pro Typ der Nachricht und Ursprung) abgeholt werden. Anschließend werden sie an andere Agenten zur Filterung weitergeleitet: Sie entscheiden für welche Nachricht welche Aktion gesetzt wird (oder ob keine erfolgt). Diese Reaktion wird dann je nach Komplexität oder Aufwand direkt ausgeführt oder an einen anderen Agenten übertragen.

In einem Agenten-System können gleichzeitig beliebig viele Filterungs-Systeme verwendet werden, sofern die dazu gehörenden Agenten unterschiedliche Besitzer-Zertifikate enthalten. Dies deshalb, da Quell- bzw. Filter-Agenten jeweils ausschließlich (dafür jedoch alle) Agenten des selben Besitzers akzeptieren bzw. beliefern.

3.2.2. Komponenten

Das System zur Nachrichtenverarbeitung besteht aus einzelnen Komponenten, welche durch Agenten realisiert wurden. Diese können separat erzeugt und transferiert werden. Grundsätzlich besteht eine dreiteilige Struktur (Siehe auch Abbildung 3):

1. Quellen: Diese Agenten sind dafür zuständig, Nachrichten aus externen Quellen in das System einzulesen und in das lokale Format zu konvertieren. Sie verteilen dann selbständig die Nachrichten an Agenten, welche sich vorher bei ihnen angemeldet haben.

Die Weiterlieferung kann entweder zu regelmäßigen Zeitpunkten erfolgen, aber auch durch Ereignisse ausgelöst werden.

2. Filter: Diese Agenten sind dafür zuständig, jeweils eine konkrete (wenn auch vielleicht komplexe) Filter-Aufgabe wahrzunehmen. Sie überprüfen eingehende Nachrichten darauf, ob sie den enthaltenen Regeln entsprechen und führen auf Grund dieser Entscheidungen dann Aktionen aus. Diese können in der Beantwortung der Nachricht, der Weiterleitung, der Information des Benutzers auf irgendeinem Wege oder einer selbst definierten Aktion bestehen.
3. Hilfs-Agenten: Hierbei handelt es sich typischerweise um Agenten, welche komplexere Aufgaben wahrnehmen und spezielle Aktionen durchführen, etwa den tatsächlichen Versand einer E-Mail. Nicht jeder Filter-Agent wird solche Hilfs-Agenten benötigen.

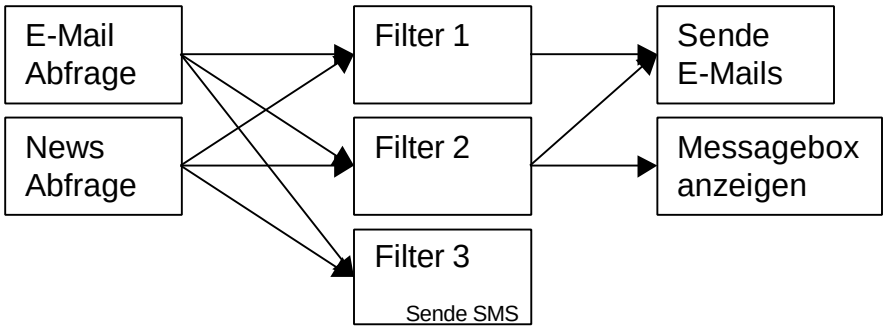

Abbildung 3: Beispiels-Konfiguration eines Nachrichten-Systems

Eine Erweiterung des Systems ist ohne großen Aufwand möglich, da die Konfiguration automatisch erfolgt. Jede Quelle gibt ihre Existenz bekannt und kann dann von Filter-Agenten abonniert werden. Analog dazu fragen Filter-Agenten, nachdem sie erzeugt oder verlagert wurden, automatisch um neue Quellen an. Sowohl Abonnement wie auch der Versand von Nachrichten wird nur durchgeführt, wenn der Besitzer der Agenten identisch ist. Es ist daher ausreichend, eine neue Quelle oder einen neuen Filter zu erzeugen, um diesen speziellen Agenten vollständig in das System zu integrieren. Voraussetzung dafür ist jedoch, daß alle Agenten in einem Agentensystem alle Nachrichten verarbeiten, bzw. an diese weitergeben. Ist dies nicht gewünscht, so wären die Agenten ohne großen Aufwand umzuprogrammieren, was jedoch ein viel komplexeres Benutzerinterface verlangen würde, um die dann notwendige Konfiguration des Systems vornehmen zu können.

Das System besteht zur Zeit aus folgenden Agenten, kann jedoch leicht erweitert werden, insbesondere im Hinblick darauf, neue Filter-Agenten zu erzeugen:

? E-Mail Abfrage Agent: Dieser Agent ist in der Lage, von einem E-Mail Server über das POP3 Protokoll Nachrichten abzufragen, wobei nur neue Nachrichten (i.e. Mails, welche dieser Agent noch nicht weitergeleitet hat) an andere Agenten verschickt werden. Eine Speicherung erfolgt nicht, sodaß ein später hinzukommender Filter-Agent nur diejenigen Nachrichten erhält, welche ab diesem Zeitpunkt hinzukommen, jedoch nicht solche, welche von bereist früher erzeugten Agenten bereits verarbeitet wurden.

? Agent zur Anzeige der Anzahl ungelesener Mails: Dies ist ein einfacher Beispiels-Agent, mit dem das Konzept des Abonnements von Daten dargestellt wird. Seine einzige Aufgabe ist es, die Anzahl der ungelesenen Nachrichten in seinem Benutzerinterface darzustellen. Er nimmt keine Verarbeitung des Nachrichteninhalts vor.

? Agent zum Versand von E-Mails: Dieser Agent versucht, eine oder mehrere ihm übergebene Nachrichten als E-Mail abzusenden. Zu diesem Zweck werden bekannte Mailserver, als auch aus den Adressaten herausgefilterte Rechner, zur Übertragung

getestet. Ist beim Entgegennehmen eines Auftrags auf dem lokalen Rechner der Mail-Support nicht installiert, so begibt sich der Agent auf andere ihm bekannte Rechner, um es dort zu versuchen und anschließend zurückzukehren. Erfolg und Mißerfolg werden im Erfahrungs-System gespeichert, sodaß zu einem späteren Zeitpunkt erfolgreiche Rechner zuerst besucht werden. Nach dem Absenden (bzw. der Rückkehr) erfolgt eine Benachrichtigung des Agenten, der den Auftrag erteilte.

? Agent zur Anzeige einer Messagebox in einem anderen Agentensystem: Dieser Agent nimmt Aufträge zur Anzeige einer Messagebox auf einem anderen Rechner entgegen. Der beauftragende Agent kann in der Zwischenzeit andere Aufgaben erfüllen, aber auch warten. Zu diesem Zweck erzeugt der beauftragte Agent einen (Programmcodemäßig sehr kleinen) Subagenten, welcher versucht, sich selbst auf den anderen Rechner zu verlagern und dort die Nachricht anzuzeigen. Kehrt dieser Subagent erfolgreich zurück oder schlägt die Verlagerung fehl, so erfolgt eine Nachricht an den beauftragenden Agenten (Erfolg bzw. Mißerfolg).

? Agent zum regelmäßigen Überprüfen von Webseiten auf Änderungen: Dieser Agent verwendet einen weiteren Agenten dazu, eine oder mehrere Webseiten regelmäßig abzufragen. Für die empfangenen Seiten wird anschließend ein Hashwert berechnet und mit dem Wert der vorigen Abfrage verglichen. Ergibt sich eine Veränderung, wird die Webseite in eine Nachricht umgewandelt und an die Filter-Agenten weitergeleitet. Dies ermöglicht einerseits eine Benachrichtigung des Benutzers von Änderungen, aber auch die Extraktion von einzelnen Informationen, welche dann weiterverarbeitet werden könnten.

Nicht mehr zu den Basis-Agenten sondern ein konkretes Anwendungsbeispiel ist der Agent zum Durchsuchen von Nachrichten nach einem Schlüsselwort mit Versand einer SMS oder der Anzeige einer Messagebox auf einem anderen Computer bei Erfolg. Hierbei handelt es sich um einen komplexeren Agenten, welcher auch eine komplette Benutzeroberfläche (Dialog) besitzt. Es kann ausgewählt werden, an welchen Provider und welche Telefonnummer eine frei bestimmbare Nachricht gesandt wird, wenn in einer empfangenen Nachricht eine bestimmte Zeichenfolge enthalten ist. Zum Versand der SMS werden öffentliche Gratis-SMS-Dienste verwendet. Die entsprechenden Webseiten werden geladen, sowie das darin enthaltene Formular identifiziert, dieses ausgefüllt und anschließend abgeschickt. Dies erfolgt nach generellen Richtlinien, sodaß auch Änderungen des Seitenlayouts oder der Formularfelder keine Behinderung darstellen.

3.2.3. Funktionsumfang

Zur Filterung der Mails stehen die folgenden Funktionen zur Verfügung, welche beliebig in Form eines Baumes kombiniert werden können. Bei vielen besteht die Möglichkeit, die Überprüfung entweder auf das gesamte Element oder Teile zu beziehen (Suche nach Teilstrings) sowie die Überprüfung unter Außerachtlassung der Groß-/Kleinschreibung durchzuführen. Die wichtigsten Bedingungen sind:

? Und, oder, nicht; wahr, falsch: Einfache Verknüpfungen und Konstanten, um mehrere Filterbedingungen zusammenzufassen.

? Inhalt der Mail (Body text): Der gesamte Textinhalt der Mail wird durchsucht, wobei eine einfache Substring-Suche erfolgt. Dies ist das wichtigste Kriterium, doch ist es in vielen Fällen nur in Kombination (siehe obige Verknüpfungen) sinnvoll (Wort1 enthalten und Wort2 enthalten oder Wort3 enthalten, aber nicht Wort4).

? Titel der Mail (Subject): Analog dem Inhalt der Mail. Hier wird nur der Titel durchsucht.

? Datumsvergleich (z. B. Absende-Zeitpunkt): Das Absendedatum der Mail kann mit einem anderen Datum verglichen werden, etwa um festzustellen, wie alt eine Mail ist. Dies ist

besonders beim Neustart wichtig, um etwa alte E-Mails grundsätzlich nicht mehr zu behandeln.

? Absender-Vergleich: Der Sender der Nachricht wird mit einem vorgegebenen Wert verglichen. Dies erlaubt es etwa, die Mails von bestimmten Personen bevorzugt zu behandeln, kann aber auch zum automatischen ignorieren von Werbe-Mails (Spam) verwendet werden.

? Adressaten-Vergleich: Analog dem Absender-Vergleich, jedoch wird hier die E-Mail Adresse des Empfängers ausgewertet. Dies ist nur dann sinnvoll, wenn entweder das verwendete Mail-System für verschiedene Adressen eine einzige Mailbox besitzt, oder wenn im Nachrichten-Verarbeitungssystem mehrere Quellen-Agenten existieren.

? Auswertung von Mail-Headern: Beliebige Header-Elemente werden mit vorgegebenen Zeichenfolgen verglichen. Eine Anwendung hierfür ist etwa, um festzustellen ob die Mail verschlüsselt und/oder signiert ist.

? Anzahl der Anhänge: Es werden die Anhänge gezählt, welche mit der Nachricht verbunden sind. Dies kann etwa als Information in eine Zusammenfassung integriert werden.

? Größe der Mail: Die Größe der Mail in Bytes gibt Aufschluß darüber, welche Aktionen mit ihr sinnvollerweise duchgeführt werden können, aber auch etwa um die zu reservierende Menge an Festplatten-Platz für eine lokale Speicherung zu bestimmen.

? Anzahl der Textzeilen: Es werden die Zeilen im ersten Textteil der Mail gezählt und zurückgegeben. Besonders lange oder kurze Mails können so gesondert behandelt werden (z. B. macht es bei langen Mails keinen Sinn, diese per SMS weiterzuleiten, hier müßte eine Auswahl getroffen werden).

Eine Erweiterung um zusätzliche Bedingungen ist jederzeit problemlos möglich, etwa die Suche nach regulären Ausdrücken oder die Auswertung von Prioritäts-Informationen im Header.

Wurde eine Nachricht aufgrund der obigen Bedingungen ausgewählt, so sind mit ihr unter Anderem folgende Aktionen standardmäßig möglich. Weitere können beliebig implementiert werden, siehe etwa der Versand von SMS als Aktion nach der Auswahl.

? Mehrfach-Aktion: Hierbei handelt es sich um eine Meta-Aktion. Mehrere verschiedene Aktionen werden hintereinander ausgeführt.

? Weiterleiten (Forward): Die Nachricht wird an eine andere E-Mail Adresse weitergeleitet.

? Feste Mail senden: Eine fest vorgegebene E-Mail wird abgesandt, wobei sowohl Sender als auch Empfänger fix sind.

? Feste Antwort senden: Eine fest vorgegebene E-Mail wird abgesandt, wobei der Empfänger immer der Sender der zu bearbeitenden Mail ist.

? Antwort zusammenstellen: Diese Aktion erlaubt es, in Verbindung mit dem Parser komplexe Antworten zusammenzustellen, welche auch aus Elementen der eingegangenen Mail bestehen können. Es wird eine Ergebnis-Nachricht vorgegeben, in welcher alle Zeichenfolgen des Musters „<…..>“ durch entsprechende Werte aus der zu bearbeitenden Nachricht ersetzt werden (Beispiel: „<FROM>“ wird durch den Absender der Nachricht ersetzt). Als Werte können die verschiedenen Elemente einer Nachricht (Adressaten, Titel, Datum, …) verwendet werden, aber auch (vorher durch gesonderte Spezial-Aktionen herausgelöste) Phrasen der Nachricht, welche durch bekannte Anfangs und End-Zeichenketten begrenzt sind. Hiermit lassen sich etwa Antworten auf ausgefüllte E-Mail Formluare erzeugen, bei denen der Benutzer z. B. Felder folgenden Typs ausfüllt: „Bitte

beschreiben Sie den Bug: [ ]“. Je nachdem was der Absender der Nachricht in der Markierung schrieb, kann dann die zu sendende Nachricht zusammengestellt werden.

? Mehrere Antworten zu einer Antwort zusammenfassen: Mehrere verschiedene Antworten werden zu einer einzigen Nachricht zusammengefaßt. Wenn aus einer Nachricht verschiedene Elemente extrahiert und einzeln beantwortet werden, so kann hiermit eine einzige Rück-Nachricht erzeugt werden, anstatt viele verschiedene Nachrichten zu senden.

? Antwort(en) absenden: Alle anderen Aktionen erzeugen nur Rücknachrichten, diese hier sendet sie dann tatsächlich als E-Mail ab.

Weitere Aktionen können bei Bedarf ohne Schwierigkeiten hinzugefügt werden, etwa damit das Absenden einer Antwort nicht per E-Mail erfolgt sondern in eine Newsgruppe, stattdessen ein Ausdruck erfolgt oder Nachrichten auf dem E-Mail Account gelöscht oder verschoben werden.

3.2.4. Beispiels Filter-Agent

Ein eigener Agent mit bestimmten Filtern und Aktionen kann sehr leicht erzeugt werden. Es ist dazu lediglich eine Subklasse von FilterMailAgent anzulegen und die Regeln zu spezifizieren. Sollen etwa alle Nachrichten, welche im Betreff das Wort „Agent“ enthalten an den Mailaccount „agent.list@mailing.list.at“ weitergeleitet werden (wobei der Rechner „mail.server.provider.at“ zum Versand der Mails verwendet wird), so kann hierzu das folgende Programm verwendet werden:

```
package FilterTest;

import MailAgents.MailChecker.FilterMailAgent;
import Filtering.Condition.*;
import Filtering.Action.*;
import Filtering.Rules.*;

public class ForwardTest extends FilterMailAgent
{
    public ForwardTest()
    {
        ruleList.addRule(
            new SimpleActionRule(
                new Filtering.Condition.SubjectTerm("Agent",false),
                new SendResponse(this,
new ForwardMsg("agent.list@mailing.list.at"),"mail.server.provider.at")
            )
        );
    }
}
```

3.3. Bücher-Suche im Internet

3.3.1. Zielsetzung

In diesem System wird ein Agent beauftragt, ein bestimmtes Buch zu suchen, wobei er mit relativ wenige Vorgaben auskommen muß: URLs von Buchanbietern im Internet und einer kurzen Beschreibung des Buches (Titel und Autor). Der Agent hat anschließend selbständig die entsprechenden Webseiten abzufragen, die Antworten zu untersuchen und die Ergebnisse zu vergleichen. Geplant, aber noch nicht implementiert, ist die Fortsetzung der Aufgabe, indem der Agent das Buch auch bestellt.

Der Vorteil der Implementierung dieses Systems über Agenten ist, daß die Auswertung mehrerer Webseiten längere Zeit dauern kann, jedoch ohne Benutzerinteraktion auskommt.

Im Gegensatz zu bestehenden System für Einkaufs-Vergleiche (Comparison-shopping; z. B. http://www.dealtime.com/), ist kein direkter Zugriff auf die Datenbanken der Anbieter oder eine genaue Definition der einzelnen Webseiten (Suchseite, Ergebnisseite für Einzelbücher, Ergebnisseite für Bücherliste, etc.) notwendig. Es werden die normalen Webseiten nach vordefinierten Grundsätzen durchsucht (z. B. eine Bücherliste besteht aus einer Tabelle oder ist durch horizontale Linien getrennt, …) und daraus die Informationen gewonnen. Aufgrund der vielseitigen Gestaltung der Webseiten ist es jedoch nicht möglich, die Agenten für alle Anbieter einzusetzen. Eine gewisse Anpassung ist daher für jeden zusätzlichen Verkäufer notwendig. Der große Vorteil ist, daß Veränderungen der Webseiten, z. b. graphischer Natur, den Code der Webseiten stark ändern und eine Neuerstellung bei festen Definitionen erfordern würden, die Struktur der Daten jedoch meist unverändert bleibt und so die Agenten ohne eine Anpassung auskommen. Ähnliches gilt für das Ausfüllen von Formularen: Der (interne) Name von Feldern kann sich ändern, die daneben stehende Bezeichnung wird jedoch viel geringeren Veränderungen unterliegen und immer in der Nähe des Felder bleiben.

3.3.2. Komponenten

Dieses System besteht, im Gegensatz zum Nachrichten-Verarbeitungssystem, aus einer geringeren Anzahl von Agenten, welche jedoch größere Aufgaben übernehmen. Ein oder mehrere Agenten werden mit dem Laden von Webseiten beauftragt, während ein einzelner Agent die Auswertung der empfangenen Seiten durchführt (Haupt-Agent). Eine weitere Aufteilung wäre möglich, etwa indem Webseiten nicht nur parallel geladen, sondern auch gleichzeitig ausgewertet werden. Dies ist jedoch nur dann sinnvoll, wenn dies verteilt auf mehreren Rechner erfolgen würde, da es sich hierbei um eine sehr rechenintensive Aufgabe handelt.

Im einzelnen besteht das System aus den folgenden Komponenten:

? Download von Webseiten: Hierzu wird der selbe Hilfs-Agent verwendet, der auch bei der regelmäßigen Überprüfung von Webseiten im Nachrichten-Verarbeitungssystem Verwendung findet.

? HTML-Formulare parsen, ausfüllen und absenden: Diese Komponente erlaubt es, HTML-Formulare, welche als Zeichenkette vorliegen, in die einzelnen Felder zu zerlegen. Es werden jedoch nur diejenigen Attribute ausgewertet, welche für die Daten wichtig sind (inklusive versteckten Feldern); Hinweise zur graphischen Darstellung und Anordnung werden ignoriert, da es sich um eine rein interne Verarbeitung handelt. Felder können über verschiedene Elemente identifiziert werden, wie etwa den Namen, den Inhalt, der Gruppe der sie angehören (für Radiobuttons) oder dem daneben stehenden sichtbaren Text. Wurde das Formular ausgefüllt, so kann es an den darin angegebenen URL abgeschickt werden, wobei die Ergebnisseite zur weiteren Verarbeitung zurückgegeben wird.

? Preise identifizieren und auswerten: Ein wichtiges Element bei der Auswertung von Bücherlisten ist der Preis, da dieser immer beigefügt ist und es sich um ein relativ festes Schema handelt: Eine Zahl (eventuell mit „.“ oder „,“) und eine vor- oder nachgestellte Währungsangabe. Hierzu wurden einige Klassen geschaffen, mit denen Währungen definiert werden können. Durch diese Klassen ist eine Identifizierung eines Preises und die anschließend Auswertung möglich, wobei das für die jeweilige Währung spezifische Muster verwendet wird (vor- / nachgestellt, „.“ oder „,“ als Dezimaltrennzeichen). Über einen Umrechnungskurs ist auch ein späterer Vergleich zwischen Preisen in unterschiedlichen Währungen möglich.

? Ergebnislisten auswerten: Eine Liste von Büchern ist dadurch gekennzeichnet, daß sie einerseits mehrere Preise enthält (in gleicher Währung) und andererseits die einzelnen Bücher voneinander getrennt sind. Dies kann entweder durch horizontale Linien erfolgen, oft aber auch durch Tabellen, wo in jeweils ein bis drei Zeilen ein Buch beschrieben wird.

Dies erfordert es zu überprüfen, wie viele Zeilen zusammengehören und diese anschließend zusammenzufassen (was durch eine Numerierung der einzelnen Titel vereinfacht wird). Sind die Abschnitte, welche jeweils ein einzelnes Buch beschreiben, isoliert, so erfolgt die Auswertung wie bei einer Einzel-Ergebnisseite (siehe nächster Punkt).

? Einzelne Bücher auswerten: Auf einer Seite, welche nur ein einzelnes Buch erläutert, muß zuerst das gesamte Umfeld bereinigt werden. Dies bedeutet, daß Fußzeilen, Navigationsspalten, Header, Werbung, etc. zu beseitigen ist. Nach der Isolierung des eigentlichen Kerns der Seite, den Informationen zu einem Buch, werden diese ausgewertet. Dies ist teilweise recht einfach (ISBN Nummern werden immer neben dem Kürzel „ISBN“ angeführt), manchmal aber auch schwierig. Ein Beispiel hierfür ist der genaue Titel des Buches. Er steht normalerweise in der ersten Zeile der Buchbeschreibung, doch kann sich dort auch noch zusätzlich der Autor befinden. In diesem Fall ist zu berücksichtigen, daß zusammenhängende Teile oft mit einem einzigen Hyperlink belegt sind, wie es manchmal bei Autoren (zu einer Seite, auf der alle Bücher dieses Autors angezeigt werden) oder dem Titel (wenn es sich um einen Ausschnitt einer Liste handelt, einen Link zur Detail-Seite) der Fall ist. Durch diese Verbindung können zusammengehörende Teile identifiziert werden, und es ist so eine Trennung von Autor und Titel möglich.

? Vergleich von Büchern: Nachdem alle Bücher von den Ergebnisseiten gesammelt wurden, erfolgt der Vergleich. Ist die ISBN-Nummer bekannt, so wird diese als Vergleich verwendet, da es sich hierbei um eine eindeutige Identifizierung handelt. Ansonsten erfolgt der Vergleich nach Titel und Autor. Alle passenden Bücher von verschiedenen Anbietern werden anschließend dem Benutzer mit ihrem Preis präsentiert.

3.3.3. Beispiels-Abfrage

Als Beispiel wird die Suche nach dem Buch „Java Security“ von Scott Oaks verwendet (nur nach dem Titel, Autor nicht vorgegeben!). Dies ist zwar der genaue Titel des Buches, doch existieren einige Bücher mit ähnlichem oder gleichem Titel, sodaß jeweils eine ganze Liste gefunden wird. In der Liste sind noch einige Fehler enthalten, insbesondere beim Anbieter Barnes&Noble ist der Autor besonders gut „versteckt“. Sind die einzelnen Bücher jedoch einmal isoliert und wird ein *bestimmtes* Buch gesucht, so lassen sich die Elemente noch genauer isolieren (z. B. kann bei vorgegebenem Autor direkt nach einzelnen Namensteilen gesucht werden und so dieser exakt identifiziert werden).

Die Ergebnisse der einzelnen Anbietern sind:

? Amazon.com:

Scott Oaks (Paperback - May 1998): Out of Print--Try our out-of-print search service!
Price: USD 34.95
Scott Oaks (Paperback - May 1998): Java Security
Price: USD 27.96 (Detail link available)
Marco Pistoia, et al (Paperback): JAVA 2 Network Security
Price: USD 39.99 (Detail link available)
Jamie Jaworski (Paperback): Java Security Handbook
Price: USD 39.99 (Detail link available)

? Amazon.de:

Versandfertig innerhalb von 1 bis 2 Wochen: Inside Java 2 Platform Security von Li Gong
Price: DEM 90,05
Marco Pistoia(Herausgeber), u. a.: JAVA 2 Network Security
Price: DEM 119,32 (Detail link available)
Jonathan B. Knudsen: Java Cryptography
Price: DEM 69,23 (Detail link available)

Scott Oaks: Java Security
Price: DEM 81,54 (Detail link available)
G McGraw: Securing Java 2e
Price: DEM 79,64 (Detail link available)
Sean J. Harnedy: Web-Based Information Management: For the Enterprise
Price: DEM 145,59 (Detail link available)
Jamie Jaworski: Java Security Handbook
Price: DEM 119,32 (Detail link available)

? Barnes & Noble:

In Stock: 24 hours (Same Day): Java Security
Price: USD 27.96 (Detail link available)
In Stock: 24 hours (Same Day): Java 2 Network Security
Price: USD 39.99 (Detail link available)
In Stock: 24 hours (Same Day): Inside Java 2 Platform Security: Architecture, API Design, and Implementation
Price: USD 34.95 (Detail link available)
In Stock: 24 hours (Same Day): Web-Based Information Management: An Introduction to the Technology and Its Application
Price: USD 44.00 (Detail link available)
In-Stock: Ships 2-3 days: Java Network Security
Price: USD 39.99 (Detail link available)
In Stock: 24 hours (Same Day): Securing Java: Getting Down to Business with Mobile Code
Price: USD 34.99 (Detail link available)
In Stock: 24 hours (Same Day): Java Security Handbook
Price: USD 39.99 (Detail link available)

? Lion.cc:

Marco Pistoia, Duane F. Reller, Deepak Gupta, et al: Java Security
Price: ATS 542,00 (Detail link available)
Oaks, Scott: Java Security
Price: ATS 542,00 (Detail link available)
Kühnel, Ralf: Die Java 2 Fibel, m. CD-ROM
Price: ATS 361,00 (Detail link available)
Marco Pistoia, Duane F. Reller, Deepak Gupta, et al: Java 2 Network Security, w. diskette (
Price: ATS 788,00 (Detail link available)
Gong, Li: Inside Java 2 Platform Security
Price: ATS 564,00 (Detail link available)
Jaworski, James: Java Security Handbook
Price: ATS 542,00 (Detail link available)
Pistoia, Marco (Edt): Java 2 Network Security (Itso Networking Series)
Price: ATS 639,00 (Detail link available)
MacGregor, Robert S. (Edt): Java Network Security (The Itso Networking Series)
Price: ATS 509,00 (Detail link available)
Gong, Li: Inside Java 2 Platform Security : Architecture, Api Design, and Implementation (Java Series)
Price: ATS 449,00 (Detail link available)
McGraw, Gary: Java Security : Hostile Applets, Holes & Antidotes
Price: ATS 332,00 (Detail link available)
McGraw, Gary: Securing Java : Getting Down to Business With Mobile Code
Price: ATS 449,00 (Detail link available)
Fan, Joel: Cryptography and Security in Java (Advances in Object Technology Series , No 2 0)
Price: ATS 449,00 (Detail link available)
Jaworski, Jamie: Java Security Handbook (Sams Professional Series)
Price: ATS 509,00 (Detail link available)

Das am Besten passende Buch er einzelnen Anbieter:

Matching book from Amazon DE found:
Scott Oaks: Java Security
Price: DEM 81,54 (Detail link available)
Matching book from Amazon USA found:
Scott Oaks (Paperback - May 1998): Java Security
Price: USD 27.96 (Detail link available)
Matching book from Barnes&Noble found:

In Stock: 24 hours (Same Day): Java Security
Price: USD 27.96 (Detail link available)
Matching book from Lion.CC found:
Marco Pistoia, Duane F. Reller, Deepak Gupta, et al: Java Security
Price: ATS 542,00 (Detail link available)

Insgesamt das billigste Buch:

Cheapest book:
Scott Oaks (Paperback - May 1998): Java Security
Price: USD 27.96 (Detail link available)

4. Praktische Verwertung

4.1. Agenten-Framework

Das Basis-Framework für Agenten findet seine praktische Anwendung zusammen mit den beispielhaften Systemen, insbesondere dem System zur Verarbeitung von Nachrichten. Es ist auf mehreren Rechnern des Institutes installiert und wird zur Erledigung einfacher Aufgaben eingesetzt.

Weiters findet es Anwendung in der Forschung und wird weiterhin ausgebaut (Siehe unten).

4.2. Nachrichten-Verarbeitungssystem

Dieses System ist am Institut im praktischen Einsatz und erfüllt zur Zeit die folgenden Aufgaben:

? Überprüfung auf neue E-Mails und Anzeige der Anzahl der ungelesenen davon.

? Benachrichtigung bei wichtigen Nachrichten per SMS bzw. Messagebox.

? Weiterleiten einer Kopie von Nachrichten (Mehrfachbetreuung von Diplomanden).

? Kurze Rückmail im Falle längerer Abwesenheit.

Ein weiterer Ausbau des Systems für andere Aufgaben ist geplant (siehe unten).

In der Forschung findet es Anwendung bei der Untersuchung und dem Test des Vertrauensmodells, indem mehrere gleichartige Agenten um Kunden rivalisieren. Ein Beispiel hierfür ist die kurzfristige Benachrichtigung des Benutzers: Ist ein Agent, welcher eine Messagebox auf einem anderen Rechner anzeigt, sehr erfolgreich, so wird eher dieser verwendet. Schlägt dies jedoch öfters fehl, wird eine SMS versandt. Bei diesem Fall stellt sich jedoch insbesondere das Problem der Rückmeldung: Nur wenn eine Messagebox angezeigt oder eine SMS erfolgreich versandt wurde bedeutet dies noch nicht, daß der Benutzer den Hinweis auch tatsächlich erhalten hat.

4.3. Buch-Suche

Dieses System dient der Forschung und wird nicht praktisch eingesetzt. Hierfür entwickelte Teile (z. B. die Klassen zur Bearbeitung von Web-Formularen) werden jedoch in anderen Applikationen eingesetzt (Web-Formulare etwa beim Versand von SMS im Nachrichtensystem).

Es wird vor allem für die Weiterentwicklung der „Intelligenz“ der Agenten und die Untersuchung der Mobilität im Zusammenhang mit lang andauernden Aufgaben verwendet.

5. Zukunftsperspektiven

Das Projekt wird weitergeführt und vervollständigt sowie erweitert. In Bezug auf die Forschung soll das Vertrauensmodell näher untersucht werden, sowie der Schutz von Agenten vor dem Rechner, auf dem sie sich befinden, verbessert werden. Ein Ansatz dafür ist, gewisse (unveränderliche) Teile des Agenten zu signieren (Daten, nicht Code wie bereits jetzt). Dies würde zumindest die Grundeinstellungen sichern, sodaß jeder Empfangs-Server erkennen kann, ob dieser Agent verändert wurde oder nicht. Zur Sicherung der Nachvollziehbarkeit von Veränderungen wäre es auch möglich, wenn jeder Rechner alle Agenten signieren muß, die ihn verlassen. So kann zwar eine Veränderung nicht erkannt werden, doch kann zumindest im Nachhinein (falls eine entdeckt wurde) festgestellt werden, wo sie erfolgte.

Eine weitere geplante Erweiterung betrifft die stärkere Integration von Agenten in das WWW. Es soll ein Zwischenstück implementiert werden, sodaß über ein WWW-Formular Agenten parametrisiert und gestartet werden können. Diese erfüllen dann eine relativ kurze Aufgabe, kehren zurück und geben ihr Ergebnis über dieselbe Schnittstelle als Webseite zurück an den Auftraggeber. Dies würde einen einfacheren Zugriff für normale Benutzer erlauben und hat den wichtigen Vorteil, daß auch ein Zugriff von Rechnern aus möglich ist, auf denen das Agenten-System nicht installiert ist.

Durch Erweiterung soll die Intelligenz der Agenten gesteigert werden, indem ein vollständiges Regel-System schon im Basis-Framework integriert wird. Dieses soll als dritte Option neben der freien Programmierung und dem ereignisbasierten Modell (aber unter Integration dieses) angeboten werden.

Das Vertrauenssystem kann noch weiter verfeinert werden, insbesondere indem Strategien für das kontrollierte Vergessen integriert werden. Im Laufe längerer Zeit akkumuliert ein Agent sehr viele Aussagen, was sowohl die Größe für Übertragungen als auch die Rechenzeit negativ beeinflußt. Es ist daher notwendig, sowohl mehrere bekannte Aussagen zusammenzufassen (was mit einem gewissen Verlust an Genauigkeit verbunden ist), als auch bestimmte Daten vollständig zu löschen (z. B. wenn bekannt ist, daß dieser Agent beendet wurde).

Auch das System zur Verarbeitung von Nachrichten soll weiter ausgebaut werden, insbesondere da es auch praktische Anwendung findet. Besonderes Augenmerk wird darauf gelegt werden, dem Benutzer mehr Aktionsmöglichkeiten anzubieten, wie etwa die automatische Auswertung von Formularen oder die Speicherung in einer Datei als Liste. Jedoch auch neue Anwendungsgebiete sollen erschlossen werden, etwa die Überwachung eines bestimmten Börsenkurses: Es werden regelmäßig Webseiten (über Formularen) abgefragt, um den aktuellen Kurs eines Titels zu bestimmen. Verändert sich dieser um einen bestimmten Wert/Prozentsatz oder erreicht er einen vorgegebenen Wert, so erfolgt eine Benachrichtigung des Benutzers. Auch eine periodische Zusammenfassung ist geplant.

6. Weitere Informationen

Weitere Informationen über das Projekt und die Möglichkeit zum Download des Systems und der Dokumentation sind unter dem URL

http://www.fim.uni-linz.ac.at/telework/agents.htm

zu finden.