Ausarbeitung zum Seminar

# *Business Intelligence - Teil II:*
# *Data Mining & Knowledge Discovery*

## Web Agents

Betreuer: Boris Stumm

Lehrgebiet Datenverwaltungssysteme

Bearbeiter: Christian Weber

# Gliederung:

# 1. Einleitung - Was sind Web Agents?

Das WWW ist heute als Informationsquelle nicht mehr wegzudenken. Immer noch im Wachsen begriffen, bietet es einen einfachen Weg, an gewünschte Daten zu gelangen. Um jedoch die gewaltigen Datenmengen effizient zur Informationsgewinnung nutzen zu können, braucht es fortgeschrittene Systeme, welche die notwendigen Aufgaben der Informationsbeschaffung, -auswertung und -präsentation bewältigen können. Zu diesem Zweck müssen zwangsläufig Erkenntnisse aus dem Feld der KI angewendet werden, etwa um das Suchen und Filtern der Informationen auf intelligente Art automatisieren zu können.
Systeme mit diesen Eigenschaften fasst man unter dem Begriff *Web Intelligence* zusammen[1]. Bedenkt man das Leistungsspektrum, das solche Anwendungen umfassen sollen, so dient es der Übersichtlichkeit, diesen Sammelbegriff in mehrere Untergruppen zu unterteilen. Das Web Intelligence Consortium (WIC) bietet beispielsweise eine Unterteilung in acht Untergruppen an [1].
Eine diese Untergruppen ist der Begriff der *Web Agents*. Sie können als handelnder Part des Ganzen gesehen werden, da sie die Komponenten im Gesamtsystem sind, die das Verhalten bestimmen. Je nach Aufgabe können sie vorhandene Techniken des *Web Mining and Farming* anwenden oder auch die Interaktion mit dem Benutzer realisieren. Sie sind es auch, die sich Konzepte der Künstlichen Intelligenz (KI) zueigen machen müssen, um sich bei der Informationsbeschaffung in einem heterogenen Umfeld zurechtzufinden und um sich flexibel an Benutzerwünsche anzupassen.

## 1.1. Angestrebte Szenarien für Web Agents

- Das WIC schlägt diese Tätigkeiten vor, die ein Agent erledigen kann[1]:
- Konversationsführende Systeme
- Filtern und automatische Verwaltung von E-Mails
- Globale Informationsbeschaffung
- Filtern von Informationen
- Navigationshilfe
- Empfehlungssysteme
- Erinnerungsagent
- Mechanismus zur Vermittlung zwischen Ressourcen

Das Einsatzgebiet eines Web-Agenten kann also darin liegen, mit einem Benutzer eine Konversation zu führen, um beispielsweise das Angebot auf einer elektronischen Einkaufsseite an ihn anzupassen. Ein Informationsbeschaffer soll es dem Benutzer ermöglichen, das Gesuchte in natürlicher Sprache zu beschreiben. Mit diesen Angaben soll er eine eigenständige Suche durchführen, die relevante Ergebnisse liefert.
Allen Einsatzgebieten ist die Tatsache gemein, dass sie eine gewisse Teilselbstständigkeit des Programms erfordern. Dies meint, dass ein Agent die Eingaben des Benutzers nicht (nur) wortwörtlich befolgen, sondern interpretieren soll. Im Falle einer Informationsbeschaffung wäre das eine Ausweitung der Suche auf synonyme oder verwandte Begriffe. Ein elektronischer Terminplaner hätte die Aufgabe, die angegebenen Termine entsprechend ihrer Priorität dynamisch zu verwalten usw.
Das erfordert Anleihen aus dem Forschungsgebiet der KI. Diese Ausarbeitung wird stellvertretend für diese Anleihen die Nutzung eines Bayesischen Netzwerks (siehe Abschnitt 3) vorstellen sowie den Einsatz eines intelligenten Systems, um Surfverhalten im Netz zu simulieren (siehe Abschnitt 4).

Viele der Aufgaben, die durch das WIC genannt werden, sind mit einfachen Verhaltenskonstrukten schwer zu realisieren. Spätestens bei geforderter Kooperation zwischen autarken Agenten wird der Bedarf nach ausgefeilteren Lösungen offensichtlich.
Web Agents sind als Problemlöser für spezielle Aufgaben konzipiert. Sie sind Werkzeuge, auf die Benutzer oder ihre Anwendungen zurückgreifen können, um ihre Ziele effizient zu erreichen. Sie befreien den Benutzer beispielsweise von der zeitraubenden Suche relevanter Dokumente und Daten. Möglicherweise führen sie die Suche sogar wirksamer durch, als es der Benutzer tun könnte, weil sie in der gleichen Zeit eine ungleich größere Menge an Dokumenten bearbeiten. Auch die systematische Zusammenfassung der Suchergebnisse könnte an solche Agenten abgetreten werden, mit denselben Steigerungen der Effizienz. In ihrer Rolle als Konversationspartner können sie sogar als Berater bei Problemen fungieren.

Kurz gesagt: Die Möglichkeiten von Dienstleistungen, die Web-Agenten realisiert würden, wären enorm. Um diese auszunutzen bedarf es allerdings mehr oder minder komplizierter KI-Konstrukte.

### 1.2. Verwandte Begriffe

Es existieren viele weitere Konzepte, die ebenfalls in die Richtung der automatisierten Aufgabenabwicklung tendieren, meistens jedoch einen Teilaspekt des Web Agent-Begriffs abdecken. Die Übergänge zwischen den Begriffen aus diesem Gebiet sind fließend und nicht einheitlich. Daher stellt die nachfolgende Auflistung nur eine grobe Übersicht ohne den Anspruch einer genauen Definition dar:

- Der *Mobile Agent* stellt zum Beispiel die Fortführung des Agenten-Gedankens dar, betrifft aber bereits Realisierungsaspekte; ein mobiler Agent ist nicht an die heimische Hardware gebunden, sondern bewegt sich in gesonderten Laufzeitumgebungen im Netz. Das bringt Vorteile und viele Schwierigkeiten [2], soll aber an dieser Stelle nicht weiter ausgeführt werden.
- *Intelligente Agenten* sind Helferprogramme, die bei komplexeren Anwendungen dem Benutzer einen intuitiven Zugang zur Dokumentation bieten sollen. Der MS Word Assistent ist z. B. ein solcher Agent. Diese Art von Hilfsprogrammen hat aber im Allgemeinen nichts mit Netzwerken zu tun [3].
- Das Ziel eines *Sozialen Agenten* ist es, einen Menschen bestmöglich zu simulieren. Er soll die Fähigkeiten besitzen, aus Interaktionen mit Personen zu lernen. Auch die Nachbildung von Emotionen gehört zu seiner Aufgabe. Ebenso sollen ihm Vermittlertätigkeiten zwischen Personen mit ähnlichen Interessen übertragen werden können [4].
- Auch die Begriffe *Web Robot*, *Web Wanderers*, *Web Crawlers* und *Spider* grenzen an das Konzept des Web Agents. Sie sind jedoch eher auf die Ebene der Dokumentensuche oder Ermittlung der Netzstruktur begrenzt. Dasselbe trifft auf die *autonomen Agenten* zu, die sich aber auf die Navigation zwischen speziellen Server beschränken [3].

## 2. Der allgemeine Aufbau eines modularen Agenten

Es existieren mehrere Ansätze für den Aufbau eines Agenten (siehe Abschnitt 3), doch in allen wird im Wesentlichen die folgende Arbeitsweise beschrieben:

- Um die ihm übertragenen Aufgaben zu realisieren, die von einem solchen Agenten gefordert sind, braucht er zunächst einmal einen Zugang zu den erforderlichen Informationen.
- Im zweiten Schritt muss eine Interpretation der Daten erfolgen. Der Anbieter stellt Daten unstrukturiert, semistrukturiert oder strukturiert zur Verfügung. Der Agent muss diese Daten nun als Informationen erfassen; zu diesem Zweck wird von ihm eine Ontologie verwendet werden.
- Der dritte Schritt der Bewertung findet vollständig innerhalb des Agenten statt. Die erhaltene Information muss auf ihre Relevanz geprüft werden. Nur die Informationen, die für die Aufgabenstellung des Agenten wichtig sind, sollen übernommen werden.
- Abschließend werden die Informationen in einem vierten Schritt so aufgearbeitet, dass sie dem Benutzer (oder dem übergeordneten Agenten) auf eine verständliche Art präsentiert werden.

Der zu Beginn geforderte Zugang zu Daten stellt nicht das Kernproblem für Agenten dar. Er ist vom heutigen Stand der IT und der WWW-Infrastruktur abhängig und ist eine Frage der technischen Realisierung. Für den Schritt der Präsentation kann gesagt werden, dass für dieses Problem bereits vielseitige Lösungen existieren, wie zum Beispiel die aktuell gängigen Browser. Ein Web-Agent kann die Darstellung der Lösungen an diese Programme delegieren.

Die eigentliche Schwierigkeit liegt in den Schritten zwei und drei. Erhält der Agent lediglich unstrukturierte Daten, die keinerlei maschinenlesbaren Hinweis auf ihre Struktur enthalten, wird deren Interpretation deutlich erschwert oder unmöglich. Ein Beispiel für solche Daten ist ein reines Textdokument ohne Metadaten. Hier wird ein Agent sehr wahrscheinlich mit einer Sprache konfrontiert, die nicht eindeutig und sogar abhängig vom kulturellen Kontext ist [5]. Der Agent muss in diesem Fall einen komplexen Interpreter beinhalten, was neben dem Zusatzaufwand auch zu Einzellösungen führt, die einen allgemeinen Standard bei der Interpretation der Daten behindern können. Der Schritt der Bewertung ist ebenfalls von diesem Problem insofern betroffen, als dass er die Interpretationsergebnisse als Grundlage für seine Arbeit hat. Eine nicht-standardisierte Interpretation führt demnach auch zu einem nicht-standardisierten Bewertungsvorgang.
Werden hingegen die Daten jedoch vom Anbieter strukturiert angeboten, zum Beispiel durch eine semantische Beschreibung gemäß eines Standards, fallen die oben genannten Probleme weg. Da der Agent nun weiß, in welcher Art die Daten angeboten werden, wird die Bewertung deutlich leichter, weil der Bedarf nach einem komplexen Parser entfällt.
Eine solche einheitlich strukturierte Darstellung der Daten im WWW, die eine automatische Verarbeitung durch Agenten ermöglicht, wird als *Semantic Web* bezeichnet. Als Grundlage dafür dient das sogenannte *Resource Description Framework (RDF)*, welches unter anderem das XML-Format als Syntax einsetzt [6]. Auch Konzepte wie die *Web Service Definition Language* (WSDL) zur systematischen Beschreibung von Diensten, in Kombination mit der *Universal Description, Discovery and Integration of Web Services* (UDDI) als „Yellow Pages“, unterstützen dieses Ziel.
In einem solchen Semantic Web sind die Eigenschaften von sowohl Webdokumenten als auch kompletten *Web Services* beschreibbar. Der Begriff Web Service steht hier für Programme, die durch das Netz aufrufbar sind und nicht nur statische Informationen anbieten, sondern auch Aktionen in der virtuellen oder realen Welt anstoßen können [7]. Elektronische Buchungen sind dafür ein gutes Beispiel. Alle Handlungen eines Agenten, außer vielleicht die reine Informationsbeschaffung, haben mit der Nutzung solcher Services zu tun.

Ein möglicher Standard zur Beschreibung von Web Services ist *DAML-S* (DARPA Agent Markup Language - Service). DAML-S basiert auf DAML+OIL (für Ontology Interference Layer), eine durch die KI inspirierte Ontologiesprache [5].

## 2.1. Einführung der DAML-S als mögliche Grundlage für Agenten

Die DAML Services Coalition beschreibt die vier Haupteigenschaften, die für Web Services gelten müssen, um von automatischen Agenten nutzbar zu sein [7]:

- Automatische Auffindbarkeit des Web Services: Der Service soll von einem Programm, für welches seine Dienste von Nutzen sind, automatisch ohne explizites Zutun des Benutzers gefunden werden können. *Die semantische Beschreibung des Dienstes muss dessen Eigenschaften und Handlungsmächtigkeit umfassen.*
  Beispielsweise soll ein Benutzer in der Lage sein, einen Reiseagenten mit der Suche nach einer günstigen Flugverbindung zwischen zwei Städten zu beauftragen. Der Agent kann dann unter Beachtung der gegebenen Randbedingungen (Preis, Komfort etc.) genau die Web Services herausfinden, die Angebote mit solchen Eigenschaften liefern.
- Automatischer Aufruf des Web Services: Der Service soll direkt durch das aufsuchende Programm ausführbar sein. *Die Kennzeichnung durch DAML-S muss zu diesem Zweck eine deklarative computerinterpretierbare API anbieten*, um die Ausführung des Dienstes zu ermöglichen (Eingabe, Form des Aufrufs, Ausgabe).
  Um beim obigen Beispiel zu bleiben, soll der Benutzer nicht mehr gezwungen sein, die gefundenen HTML-Seiten aufzusuchen und die Formulare selbst auszufüllen. Auch das soll vom Agenten übernommen werden, der dem Benutzer anschließend den Ausgang der Transaktion mitteilt.
- Die Komposition von Web Services für komplexere Aufgaben: Das ausführende Programm soll ein höher geordnetes Ziel durch die Ausführung mehrerer Dienste erreichen. Zu diesem Zweck soll es dem Programm möglich sein, für die gewählten Dienste automatisch eine bestimmte Reihenfolge festzulegen. Um dies zu unterstützen, müssen für die einzelnen Dienste wiederum *Vorbedingungen und Nachbedingungen für ihre Inanspruchnahme bereitgestellt werden*.
  Will der Benutzer aus dem bisherigen Beispiel die Reise zu einer Konferenz umfassend planen, so ist dafür bisher ein spezielles Programm nötig, welches mit den genau spezifizierten Diensten in ebenso genau spezifizierter Reihenfolge kommuniziert. Ein allgemeiner Agent sollte hier dynamisch – neben den bereits oben genannten Aspekten – auch den Ablaufplan dynamisch anpassen können.
- Automatische Überwachung der Service-Ausführung: Um die Transparenz während langer Vorgänge zu erhöhen, sollten Deskriptoren über den aktuellen Zustand bei der Ausführung des Dienstes bereitstehen.
  Wenn die Reisevorbereitung eine lange und umfassende Transaktion ist, könnte der Benutzer beispielsweise wissen wollen, ob zu einem bestimmten Zeitpunkt das Hotel bereits reserviert ist oder nicht, während andere Vorgänge noch laufen. Dieses Ziel wurde jedoch in der aktuellen Form von DAML-S noch nicht definiert.

Wie bereits in der dritten Eigenschaft beschrieben, können solche Web Services einfach oder komplex realisiert sein. Eine einfache Realisierung meint, dass der Service durch ein Programm dargestellt wird, welches während seiner Bearbeitung auf keine weiteren (Web)Programme zugreift. Eine komplexe Realisierung bedeutet demnach, dass der aufgerufene Web Services einen oder mehrere untergeordnete Services verwendet, um seine Aufgabe durchzuführen.

An dieser Stelle ist bereits absehbar, dass der Begriff Web Agent, wie er im ersten Abschnitt beschrieben wurde, mit dem eines komplexen Web Services verwandt ist. Beide sind in der Lage, ihre Aufgaben mit untergeordneten Instanzen durchführen zu können. Deren Resultate sind für sie einschätzbar und kombinierbar, und werden in einer verarbeiteten Form zurückgegeben an die aufrufende Instanz. Der Unterschied zwischen den beiden Begriffen ist, dass ein Agent als übergeordnete planende Instanz gesehen wird, während Web Services eher die Handlungsprimitive darstellen. Hierbei handelt es sich allerdings mehr um eine Richtlinie als um eine feststehende Definition.
Dieser Aspekt wird zu einem späteren Zeitpunkt noch einmal aufgegriffen werden, wo die Rolle der Web Services aus der Sicht der Agentenmodellierung betrachtet werden (siehe Abschnitt 2.3).

Um die Web Services in all den genannten Aspekten zu beschreiben, führt die DAML Service Coalition die Ontologiesprache DAML-S ein. Der Aufbau dieser Sprache basiert auf dem XML-Format. Eine detaillierte Beschreibung des DAML-S wurde den Rahmen dieser Ausarbeitung bei weitem sprengen. Statt dessen sei hier auf die Beispielbeschreibungen verwiesen, die auf der Webseite der DAML Service Coalition angeboten werden [8].
Die folgende Beschreibung wird sich daher auf die Kerngedanken der Sprache beschränken. DAML-S ist in drei konzeptionelle Bereiche unterteilt [5]: Das *Prozessmodell* (oder auch *Service Modell*), das *Profil* und das *Fundament* (*Grounding*).
Im Profil wird die Aufgabe des Dienstes beschrieben; es enthält all jene Daten, die zur Suche und Auswahl des Dienstes durch einen Agenten benötigt werden. Außerdem kann die Beschreibung zusätzliche Angaben über Input, Output, Vorbedingungen und Nachbedingungen umfassen. Zusätzlich können Angaben über den Anbieter, die Verlässlichkeit des Dienstes und andere allgemeine Eigenschaften mitgeteilt werden. Das Profil realisiert somit die erste und dritte Haupteigenschaft des Web Services.
Das Fundament sagt aus, wie der Dienst benutzt wird, d.h. auf welche Art der Agent mit ihm kommunizieren kann. Typischerweise schließt das ein bekanntes Kommunikationsprotokoll ein (SOAP , RPC , CORBA etc.) sowie für den Dienst spezifische Angaben wie die zu benutzende Portnummer. Die Beschreibung muss so gehalten sein, dass sie einen eindeutigen Weg für den Datenaustausch mit dem Dienst vorgibt.
Das Prozessmodell schließlich bildet die Arbeitsweise des Dienstes ab. Auch hier werden die Vorschriften über Eingabe, Ausgabe, Vorbedingungen und Konsequenzen aufgeführt, zusätzlich zu den Angaben für das Profil. Das Prozessmodell kann sich durchaus in mehrere Prozesse aufteilen, die untereinander durch einen Kontrollfluss verbunden sind. Da diese detaillierte Aufteilung für den Agenten sichtbar ist, hilft sie ihm bei einer tiefer gehenden Analyse, ob der Dienst geeignet für ihn ist. Darüber hinaus ist sie bei der Überwachung des Dienstes während der Ausführung hilfreich. Das Prozessmodell realisiert dadurch die zweite Haupteigenschaft für Web Services und bietet einen Ansatz für die vierte Forderung.
Das Prozessmodel ist das Kernstück von DAML-S und auch für den Aufbau der Agenten elementar wichtig [7]. Daher soll es näher beleuchtet werden, Abbildung 1 auf der folgenden Seite zeigt zusätzlich eine Übersicht des Modells.
Der zentrale Begriff des Modells ist der *Prozess*, der für die Abarbeitung einer Aufgabe steht. Jeder Prozess besitzt ein Profil, wie es oben bereits beschrieben wurde.
Es gibt drei Prozessarten: den *SimpleProcess*, den *AtomicProcess* und den *CompositeProcess*.
Ein SimpleProcess ist die Abstraktion von einem atomaren oder zusammengesetzten Prozess und wird als SingleStep-Ausführung wahrgenommen. Im letzteren Fall fungiert er als BlackBox-Sicht auf den darunter liegenden Prozess. Die Sicht auf den einfachen Prozess kann auf die Sicht des zusammengesetzten Prozesses ausgedehnt werden, ein atomarer Prozess bleibt jedoch immer verborgen.

Der AtomicProcess ist – im Gegensatz zum SimpleProcess – wirklich eine Ausführung, die in einem Schritt stattfindet. Da er der Grundbaustein der Verarbeitung ist, muss er mit einem Fundament verbunden sein, das den Nachrichtenaustausch regelt.
Ein CompositeProcess ist ein Konstrukt, das wiederum aus mehreren einfachen oder zusammengesetzten Prozessen besteht, verbunden durch die *Kontrollkonstrukte (Control Constructs)*. Die vorgegebenen Kontrollkonstrukte unterscheiden sich in der Abarbeitungsfolge der zugewiesenen Prozesse sowie dem Grad der Synchronisation.

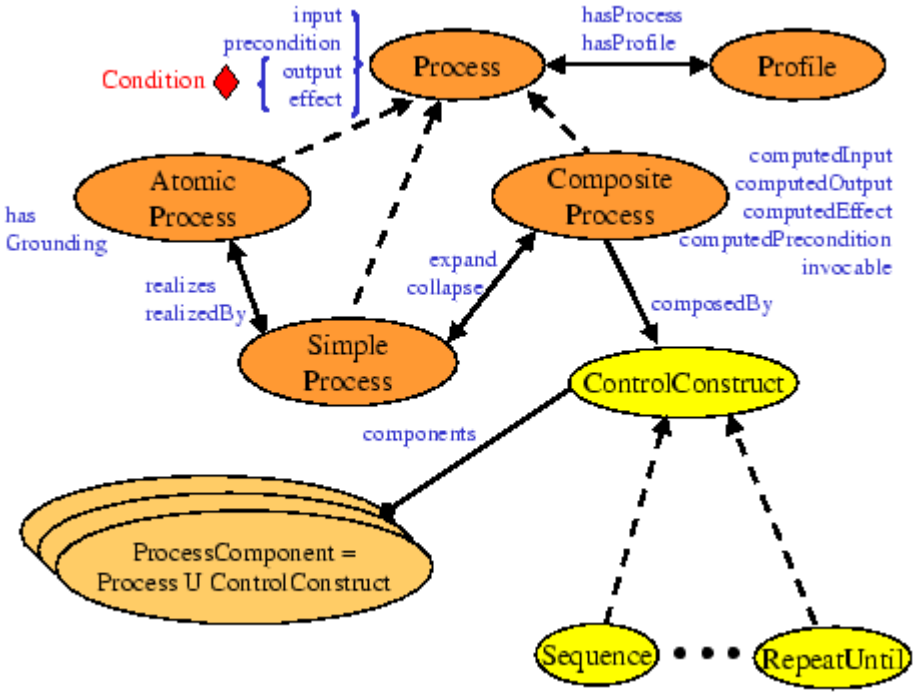


**Abb. 1.** Das DAML-S Prozessmodell der DAML Service Coalition

Mit dem Prozessmodell, das mit Hilfe der Kontrollkonstrukte eine beliebig komplizierte Bearbeitung abbilden kann, gibt DAML-S den Entwicklern von Agenten ein wirksames Werkzeug in die Hand, um Hierarchien zwischen Diensten oder auch Agenten darzustellen.

Nachdem DAML-S nun als eine mögliche Grundlage umrissen wurde, sollen nun die Entwurfsziele bei der Entwicklung der Web Agents vorgestellt werden.

## 2.2. Die Konzepte der Modularität und der Entscheidungsmodellierung

Bis jetzt wurden lediglich der gewünschte Leistungsumfang (in Abschnitt 1) und die prinzipielle Arbeitsweise der Web Agents erwähnt (siehe oben), aber nichts über die Vorgehensweise bei der Modellierung.
Es ist sinnvoll, einen Agenten – wie alle komplexen Programme – anpassbar und erweiterbar zu gestalten. Ein Agent, der vielen Benutzern längerfristig von Nutzen sein soll, muss nicht nur in der Wahl der Dienste flexibel sein, sondern unter Umständen auch in seinem Verhalten. Weiterhin fällt es leichter, die Arbeitsweise von Agenten zu verstehen, wenn sie als menschenähnliche Akteure beschrieben werden und ihnen modellierte Aspekte wie Ansichten, Intentionen und Fähigkeiten mitgegeben werden [5].
Der *verhaltensorientierte Entwurf* (*Behavior Oriented Design, BOD)* als Entwurfsmethodik für komplexe Agenten soll diesen Anforderungen genügen. BOD umfasst die beiden Konzepte der Modularität und der Modellierung der Entscheidungsfindung, die diese

Modularität und ihre Probleme berücksichtigt [5]. Durch diese Kombination soll ein Maximum an Mächtigkeit und Unabhängigkeit der einzelnen Verhaltensmodule entstehen. Die sogenannten *Basis Reactive Plans* sollen als robuster Mechanismus die „menschlichen" Fähigkeiten des Agenten nachbilden.
Warum ist die Modularität so wichtig? Bereits lange ist bekannt, dass dieses Konzept den Entwurf hinsichtlich der Flexibilität unterstützt. Dadurch ist sie unter anderem auch aus dem Gebiet des objektorientierten Entwurfs (Object Oriented Design, OOD) bekannt. Außerdem ist sie ein natürlicher Ansatz zur Abbildung einer Hierarchie, wie beispielsweise bei der bereits erwähnten Abbildung der Prozesse in Web Services.
Die Frage ist nun, wie eine sinnvolle Modularität erreicht wird. OOD besagt, dass ein Programm dergestalt zerlegt werden soll, dass jede entstandene Einheit nur noch einen Teil der ursprünglichen Variablen enthält, die aber inhaltlich zusammengehören. Darüber hinaus soll diese Einheit weiterhin in der Lage sein, diese Variablen zu bearbeiten. Im Falle des OOD sind diese Einheiten die Objekte, die die Variablen enthalten und Prozesse zu deren Bearbeitung besitzen.
Eine Haupterkenntnis aus dem BOD ist es, dass die Argumentation auch auf die Gestaltung von Agenten zutrifft. Während beim OOD die Variablen beliebige Werte darstellen, stehen sie im Kontext der Agentenmodellierung für das Wahrnehmungsvermögen (Inputvariablen) und das Gedächtnis (interne Variablen des Agenten). Betrachtet man vor allem diese Gedächtnisvariablen, so bilden sie einen dynamischen Zustand, das Kernstück des Agenten.
Wird der Zustand des Agenten nach dem Prinzip des OOD in Gruppen aus logisch zusammenhängenden Variablen zerlegt, können diese Gruppen und alle Aktionen, die von ihnen abhängen (Aktionen oder Messfunktionen), in jeweils einem Verhaltensmodul zusammengefasst werden [5]. Diese einzelnen Module bilden nun eine sinnvolle Zerlegung des ursprünglich monolithischen Agenten.
Dadurch steht die Entscheidungsmodellierung nun vor dem Problem, dass getrennte Module zuwiderlaufende Aktionen durchführen können. Ist der Agent zum Beispiel mit der Reservierung genau eines Flugtickets beauftragt und aktiviert dazu mehrere Unterprozesse zur Suche, darf nur ein Unterprozess die Bestelloperation abschließen.
Betrachtet man das System, das aus separaten Verhaltensmodulen besteht, stellt sich die Frage, wie ein solcher Konflikt erkannt und behandelt werden soll. Eine robuste Lösung besteht darin, für die Module eine gemeinsame *Umgebung* zu definieren und jedes Modul nach einem *reaktiven Basisplan (Basic Reactive Plan - BRP)* handeln zu lassen.
Mit der gemeinsamen Umgebung ist ein Kontext gemeint, über den die Module kommunizieren. Wenn ein Modul eine Aktion durchgeführt hat, gibt es eine Nachricht an den Kontext weiter. Dadurch erfahren die anderen Module von aktuellen Vorgängen.
Ein BRP besteht aus einer sequentiellen Liste, in der die möglichen Aktionen absteigend nach Priorität geordnet sind, jeweils an eine Bedingung gekoppelt. Das Modul trifft seine Entscheidung nun dadurch, dass es die Umgebung auf diese Bedingungen überprüft und die Einträge der Liste sequentiell abarbeitet. Sobald die erste Bedingung erfüllt ist, wählt das Modul die entsprechende Aktion. Als Ergebnis wird stets die Aktion mit der höchsten Priorität gewählt, die auf die aktuelle Situation sinnvoll anwendbar ist.
Es befähigt das Modul, sehr schnell Entscheidungen zu treffen [9], weil nicht etwa alle Zustände sämtlicher anderen Module berücksichtigt werden müssen, sondern nur der aktuelle gemeinsame Kontext. Werden die BRPs hinreichend restriktiv gestaltet, lassen sich Konflikte zwischen unabhängigen Aktionen vermeiden. Dieser oder ähnliche Ansätze werden zurzeit vorwiegend angewendet.

## 2.3 Konstruktion eines Agenten unter DAML-S

Betrachtet man nun das Konzept der Verhaltensmodule und vergleicht es mit dem der Web Services, stellt sich heraus, dass die beiden sich aufeinander abbilden lassen. Aus der Sicht eines Agenten haben Web Services die Gestalt eines Verhaltensmoduls, da sie in ihrer Beschaffenheit gekapselt sind; Web Services sind vom Agent nur wählbar und kombinierbar, aber nicht änderbar, genau wie autonome Verhaltensmodule.
Auf dieser Betrachtung aufbauend sind zusammengesetzte Web Services Mechanismen im Agenten, die eine Entscheidungsfindung auf der Ebene eines untergeordneten Ziels durchführen. Die Hierarchie zwischen Zielen ist damit eine simple Abbildung auf eine Hierarchie zwischen Web Services.
Die oberste Instanz in dieser Hierarchie aus Diensten, die das umfassende Ziel des Benutzers verfolgt, wird als User-Agent betrachtet. Alle untergeordneten Web Services verfolgen die Subziele, die vom Agenten daraus gebildet werden. Er unterscheidet sich von ihnen vor allem darin, dass seine Ziele dem Nutzer bekannt und durch diesen manipulierbar sind. Alle darunter liegenden Vorgänge sind dem Benutzer verborgen.
Die allgemeine Vorgehensweise auf der Ebene des User-Agents sollte so aussehen:

- Dem User-Agenten einen Mechanismus geben, mit dem er das Ziel des Users verstehen und zerlegen kann.
- Der Agent ordnet die Subziele in einer Reihenfolge, die er als effizient betrachtet
- Aufruf von tiefer liegenden Instanzen, die für ihn Blackbox-Verhaltensmodule sind, und Protokollierung des Vorgangs, um eingreifen zu können.

Eine grundlegende Entwurfsfrage ist die Entscheidung, ob der User Agent selbst ein Bestandteil des DAML-S sein soll oder nicht. Wird er in einer anderen Sprache spezifiziert, ist er isoliert von anderen Agenten und nutzt nur die beschriebenen Web Services als Untergebene. Dafür kann er aber einen für ihn günstigen Weg einschlagen, um die beiden ersten Punkte zu erreichen, ohne von DAML-S eingeschränkt zu werden. Lediglich seine Aufrufe, die seine Subziele erfüllen sollen, müssen konform zu DAML-S sein. Die Ergebnisse kann er wieder in eigener Weise protokollieren.
Der Agent kann aber auch selbst in DAML-S beschrieben werden. DAML-S bietet die Möglichkeit, mit dem Prozessmodell die Subziele zu ordnen. Die Modellierung der Entscheidungsfindung kann dort durch die Anpassung der Kontrollkonstrukte vorgenommen werden. Nebenbei ermöglicht das dem Agenten auch, in eine Aufrufhierarchie zwischen anderen Agenten eingefügt zu werden. Dort würde er als ein Web Service dienen. Dieser Ansatz vermindert die Entwicklung vieler redundanter Einzelagenten, indem er die Kooperation unterstützt. Im Gegenzug muss ein Weg gefunden werden, die charakteristische Arbeitsweise des Agenten in DAML-S zu formulieren.
Für diesen Fall wird von Bryson et al. folgende Umsetzung vorgeschlagen, um eine konforme Kodierung zu bewahren [5]:

- Die Protokollierung:
  Ein Datenhaltungsmodul oder eine ähnliche Komponente sind in DAML-S nicht definiert. Ohne jegliche Datenhaltung kann allerdings der Zustand des Agenten nicht gespeichert werden; darüber hinaus sind auch die Protokollierung der Ereignisse und das Wiederherstellen nach einem Fehlerfall nicht möglich. Daher müssen die anfallenden Daten durch ein separates Modul simuliert werden, das nur für Agenten zugreifbar ist.
- Die Anordnung der Subziele und die Reaktionen aufgrund der Protokollierung:
  Im Zuge der Verhaltensmodellierung wurden bereits die BRPs erwähnt. Sie bieten den separaten Modulen die Möglichkeit, sich untereinander zu koordinieren. Die Umsetzung

eines BRPs wird durch eine if-Verschachtelung der möglichen Einzelprozesse vorgenommen, eingebettet in eine umspannende while-Schleife. Eine direktere Abbildung, z.B. durch ein spezifisches Kontrollkonstrukt, ist nicht möglich.
Die Flexibilität eines Agenten kann weiterhin durch die Schachtelung solcher BRPs erhöht werden. Dahinter steht der Gedanke, dass der Agent auf eine kurzzeitige Änderung der Pläne reagieren kann. Hat er zum Beispiel gleichzeitig die Aufgaben der Datensuche und eines Terminplaners, soll er die Suche unterbrechen können, um dem Benutzer einen bevorstehenden Termin mitteilen zu können. Anschließend soll er mit der Suche fortfahren. Der untergeordnete BRP vermerkt regelmäßig seinen aktuellen Stand der Abarbeitung. Dann stößt er eine Überprüfung in der übergeordneten Liste/Anordnung an, ob er dort noch die höchste Priorität hat. Hat er sie nicht mehr, gibt er die Kontrolle ab. Erhält er sie wieder zu einem späteren Zeitpunkt, fährt ab dem vermerkten Zwischenstand fort. Dieser Ansatz ist aber im Gegensatz zum reinen BRP nicht weit verbreitet.

Neben diesen Vorschlägen zur Umsetzung wird eine explizite Möglichkeit zur Datensicherung empfohlen [5]. Auch eine direkte Abbildungsmöglichkeit für BRPs als Kontrollkonstrukte wird gefordert. Außerdem verlangen Bryson et al. ein Kontrollkonstrukt, dass ein schnelles asynchrones Anstoßen von sequentiellen Aktionen ermöglicht, ohne auf deren Rückmeldungen zu warten.

Zusammenfassend lässt sich über den Aufbau für Web-Agenten sagen:
Eine Ontologie ist als Unterbau für Agenten notwendig, um im Netz effizient zu suchen oder Aktionen anzustoßen. In unserem Fall wurde DAML-S als Beispiel angeführt, es können aber ebenso andere Standards verwendet werden.
Wenn DAML-S als Werkzeug gewählt wird, kann laut Bryson et al. ein Agent komplett in dieser Sprache beschrieben werden. Im Falle seiner Einbettung würde er anderen Agenten ermöglichen, ihn in eine Hierarchie aus Diensten einzugliedern.
Der Agent arbeitet modular; Probleme der Synchronisation zwischen Modulen, deren Aktionen sich ausschließen, sollen durch BRPs oder äquivalente Lösungen vermieden werden.

# 3. Architekturen für Agenten

Abschnitt eins fasst die Ziele zusammen, die momentan für die Web Agenten angestrebt werden, während Abschnitt zwei das Komponentenkonzept als Werkzeug vorstellt. Dieser Abschnitt wird nun diese Erkenntnisse verbinden und sich auf allgemeine Architekturen konzentrieren. Aus dem breiten Aufgabenspektrum sind hier Vorschläge für ein konversationsführendes, ein beratendes und ein informationsbeschaffendes System aufgeführt.

## 3.1. Architektur eines Such-Agenten

Dieser Ansatz konzentriert sich auf eine der ursprünglichen Aufgaben, für die Web Agents gedacht sind: aus einer gewaltigen Datenmenge heraus sinnvolle Ergebnisse zu finden. Dazu nutzt der Agent nun eine ontologische Basis [10], wie sie beispielsweise DAML-S benutzt. Prinzipiell ist dieser Agent ein Koordinator von Suchmaschinen-Diensten. Abbildung 2 zeigt ein mögliches ontologisches Modell als Beispiel. Der Agent durchsucht das Netz nach den Objekten Person, Wissenschaftler, Autor und Publikation. Als Suchparameter besitzt er die grau unterlegten Begriffe Name, Homepage usw.

Um die Arbeitsweise des Agenten mit dem Modell zu erläutern, sollen hier Begriffe aus der objektorientierten Sichtweise verwendet werden. Die Basiselemente der Ontologie werden als *Klassen* bezeichnet. Die Hierarchie zwischen den ontologischen Klassen wird mit den Relationen „ist“ und „hat“ zwischen den Klassen nachgebildet. Diese Relationen sind ontologiespezifisch und beziehen sich nicht auf die hierarchischen Relationen im objektorientierten Modell.
Lässt sich eine Klasse nicht weiter zerlegen, weil sie nur noch aus einem Literal oder einem Zahlenwert besteht, so nennt man sie eine *Grundklasse*. Darauf aufbauende Klassen werden *abstrakte Klassen* genannt. Eine Klasse, von der aus eine „ist“-Beziehung auf eine weitere Klasse verweist, ist eine *Vaterklasse*. Auch hier hat die Namensgebung eine rein ontologiespezifische Bedeutung.
Sind alle Werte in den Grundklassen einer abstrakten Klasse bekannt, so liegt ein *vollständiges Objekt* dieser Klasse vor. Sind einige Werte noch unbekannt, so haben wir ein *Teilobjekt*.

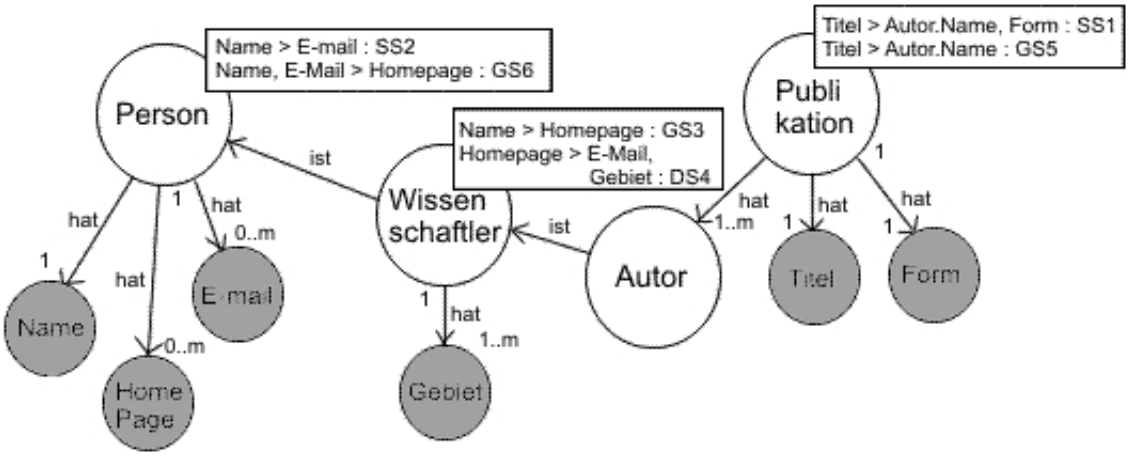


**Abb. 2** Beispiel eines Ontologiemodells

Den abstrakten Klassen sind Prozeduren zugeordnet, mit deren Hilfe der Agent die Entscheidungen trifft, welche Suche er durchführen lässt. Eine Prozedurbeschreibung hat die Form <input> → <output> : <procedure>. In der Procedure-Klausel steht die Webressource, die für die Durchführung zur Verfügung steht. Das hier betrachtete Modell betrachtet die Ressourcen Do*kumentensuche (DS)*, *spezielle Suchmaschinen(SS)* und *allgemeine Suchmaschinen (GS)*. Sie unterscheiden sich im Korrektheitsgrad der Informationen, die sie liefern. Webdokumente werden als die zuverlässigste Quelle erachtet, allgemeine Suchmaschinen als die unzuverlässigste. „Name → E-Mail : SS2“ liefert zum Beispiel alle E-Mail-Adressen zurück, die eine spezielle Suchmaschine SS2 bei einem Namen als Eingabe liefert.
Die Klauseln Input und Output geben die Klassen an, die der Ressource übergeben und zurückgeliefert werden. Das kann sowohl Grundklassen wie auch abstrakte Klassen beinhalten. Natürlich müssen die Ressourcen das gleiche Wissen über das ontologische Modell besitzen und eine abstrakte Klasse als Input bearbeiten können.
Sucht der Agent nach einer Klasse A und nutzt dabei eine ihr zugeordnete Prozedur, so ist dies eine *Eigensuche*. Er kann aber auch Prozeduren der Vaterklassen von A ausführen als eine *Vatersuche*. „Name → E-Mail : SS2“ ist in unserem Beispiel eine Eigensuche von Person, „Name → Homepage : GS3“ eine Vatersuche.

Die Arbeitsweise des Agenten besteht nun darin, aus der Eingabe des Benutzers ein anfängliches Teilobjekt abzuleiten und durch einen oder mehrere Bearbeitungszyklen übereinstimmende vollständige Objekte zu finden. Vollständige Objekte sind die Ergebnisse, die der Agent dem Benutzer liefert [10].
Die Ergänzung des teilweise unbekannten Objekts findet durch das Ausführen der Suchprozeduren statt, die das Ontologiemodell vorgibt. Sucht der Benutzer im obigen Beispiel nach einer Publikation und gibt nur Teile des Titels vor, wird der Agent eine der Suchprozeduren in Publikation starten, um die Namen der Autoren zu ermitteln. Die neuen Teilobjekte, die er nun erhält, versucht er im nächsten Arbeitsschritt wieder durch Suchen zu ergänzen, bis er vollständige Objekte erhält.
Ein Arbeitsschritt umfasst die folgenden vier Schritte:

- *Planung*: Der Agent entscheidet, welche Suche das aktuell betrachtete Teilobjekt am besten ergänzen wird. Die Wahl der Prozeduren findet nach drei Regeln statt:
    1. Eine Eigensuche hat Vorrang vor der Vatersuche, weil sie präziser ist.
    2. Ressourcen mit höherem Korrektheitsgrad haben Vorrang
    3. Es werden die Prozeduren bevorzugt, die einen hohen Zahl an Inputvariablen und gelieferten Outputs besitzen. Viele Inputvariablen sind ein Indikator dafür, dass spezifischere Antworten geliefert werden. Eine hohe Menge an Ergebnissen wird bevorzugt, weil dadurch möglicherweise mehr relevante Objekte gefunden werden
- *Suche*: Der eigentliche Suchvorgang durch die Suchmaschine. Beispiele für generelle Suchmaschinen sind AltaVista, Excite und HotBot. Bei BigYellow handelt es sich um eine domänenspezifische Suchmaschine.
- *Extraktion*: Nachdem Ergebnisse vorliegen, füllt der Agent das Teilobjekt mit Informationen auf, die aus den gefundenen Webdokumenten extrahiert werden. Dazu muss er Parser besitzen, die eine Interpretation der Seiten vornehmen können.
- *Integration*: Die Extraktion kann verschiedene Werte mit der gleichen Bedeutung ergeben. Im Integrationsschritt wird nach bestimmten Regeln eine Grundintegration zwischen den Informationen für Grundklassen durchgeführt. Bei der abstrakten Integration findet das auf höherer Ebene zwischen zusammengesetzten Objekten statt, wenn bestimmte Regeln zwischen den betreffenden Klassen erfüllt sind. Basis ist dann wieder die Grundintegration für die zugeordneten Grundkomponenten.

Zwischen jeden Verarbeitungszyklus ist das Agenten-Center als verwaltende Instanz geschaltet. Alle Teilobjekte werden im *Operationsspeicher* gespeichert, alle vollständigen Objekte im *Ergebnisspeicher*.
Das Center kann durchaus so konzipiert sein, dass ein übergeordneter Agent die Integration von Teilobjekten durchführt und anschließend das nächste Teilobjekt wählt, nach dem gesucht werden soll. Diese Suche kann er einem untergeordneten Agenten übertragen, der für dieses Objekt die Planung und die Extraktion übernimmt und das Ergebnis an das Center zurückmeldet.

## 3.2. Architektur eines konversationsführenden Agenten

Dieser Ansatz soll einen Agenten realisieren, der aus der Konversation mit dem User dessen Ziel herausfinden soll, anstatt lediglich auf Stichwörter während des Dialogs zu reagieren. Die Modellierung den Userziels findet durch ein Bayesisches Netzwerk statt [11].
Der Aufbau des Agenten ist in Abbildung 3 dargestellt. Der Benutzer richtet eine Frage in natürlicher Sprache an den Agenten. Diese Frage durchläuft einen *Filter* und wird dabei auf bestimmte Begriffe überprüft. Das Ergebnis wird an den *Intentionsprüfer* weitergereicht, der daraus das Ziel der Frage zu bestimmen sucht. Gelingt dies nicht, stößt sie den *Dialogmanager* an, der weitere Informationen beim Benutzer erfragt. Glaubt der Agent

schließlich, dass Ziel des Benutzers ausgemacht zu haben, sucht der *Antwortgenerator* schließlich in einer Datenbank oder in einem Skript nach der passenden Antwort.
Die Besonderheit dieses Agenten liegt im *Intentionsprüfer*, der die Konversation im Grunde steuert. Der Agent arbeitet mit einem vorgegebenen Set von Schlüsselwörtern, deren Abhängigkeiten untereinander mit Wahrscheinlichkeiten versehen sind.
Schlüsselwörter, die in der Eingabe des Benutzers enthalten sind, werden als Evidenzvariablen behandelt. Der Agent führt Wahrscheinlichkeitslisten für alle Schlüsselwörter in folgender Form: Das Schlüsselwort ist mit einer Wahrscheinlichkeit P das Ziel des Users (oder Teil davon), wenn bestimmte Evidenzvariablen gegeben sind. Weisen zum Beispiel vier Evidenzvariablen auf ein Schlüsselwort, so sind alle kombinatorischen Möglichkeiten zusammen mit einer zugeordneten Wahrscheinlichkeit gespeichert (was sich in der Praxis optimieren lässt).

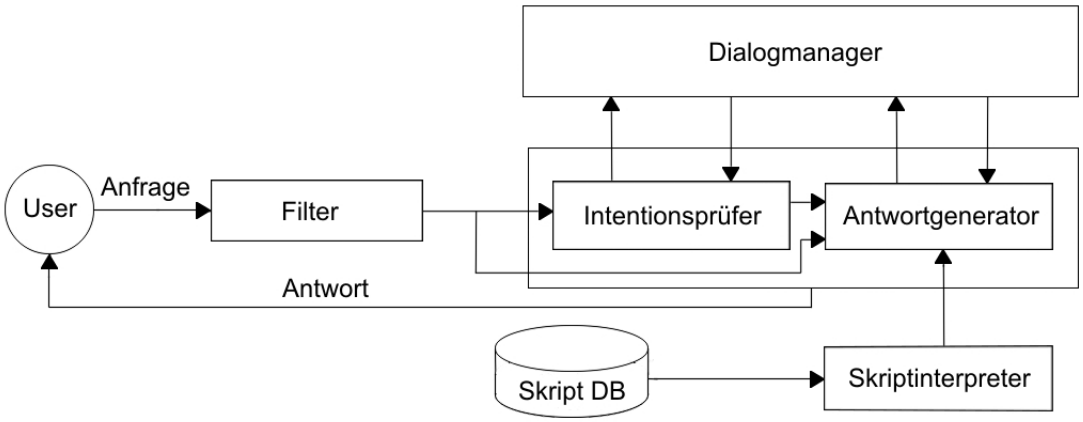


**Abb. 3.** Architektur eines konversationsführenden Agenten

Aus den Evidenzvariablen werden die Schlüsselwörter abgeleitet, für die die höchste Wahrscheinlichkeit vorgegeben ist, sofern sie einen gesetzten Grenzwert überschreitet und als hinreichend sicher erachtet wird. Dieser Vorgang findet rekursiv im Netzwerk statt, bis ein Ergebnis-Set an Schlüsselwörtern feststeht. Anhand dieses Sets wird nun über den Antwortgenerator ein Skript aufgerufen, welches den Benutzer entweder nach einer weiteren Eingabe fragt oder ihm eine Antwort aus der Datenbank liefert.
Die verwendeten Wahrscheinlichkeiten werden in diesem Modell von Lee et al. fest vordefiniert [11]. Der Agent könnte darüber hinaus durchaus so gestaltet werden, diese Wahrscheinlichkeiten im Zuge eines Lernprozesses umzuformen.

## 3.3. Architektur eines beratenden Agenten

Es gibt bereits eine Vielzahl von Empfehlungsagenten im Bereich des E-Commerce. Als eines der bekanntesten Beispiele sei hier der Agent von Amazon.com erwähnt, der passend zu den betrachteten Produkten weitere Vorschläge unterbreitet.
Empfehlungssysteme lassen sich in drei große Kategorien einteilen [12]: *kollaborative, inhaltsbasierte* und *demographische Empfehlungssysteme*.
Ein System mit einer kollaborativen Arbeitsweise speichert die Präferenzen eines Benutzers, sei es durch dessen eigene Vorgaben oder sein Einkaufsverhalten. Das System sucht nun die Benutzer heraus, die ähnliche Interessen haben und schlägt die Items vor, die sie interessant

fanden. Eine günstige Tatsache bei diesem Ansatz ist, dass der Agent keinerlei Wissen über die Eigenschaften der Produkte selbst zu besitzen braucht.
Das inhaltsbasierte Vorgehen betrachtet den Inhalt der Produkte, die für den Benutzer bisher interessant waren, und vergleicht ihn mit dem Inhalt anderer Produkte. Diese werden dann beim nächsten Besuch vorgeschlagen.
Arbeitet ein System mit demographischen Daten, dann geht es im Prinzip wie ein kollaboratives System vor. Die Ähnlichkeit zwischen Benutzern wird lediglich anhand demographischer Daten festgestellt. Die Vorschläge bestehen aus den Entscheidungen dieser Benutzer.
Es wird nun ein Agent beschrieben, der den Benutzer umfassend berät, indem er alle drei Verfahren gewichtet anwendet. Als Ergebnis kann der Agent in einem unbekannten Umfeld eingesetzt werden und sich dort anpassen, indem er die jeweils geeigneten Empfehlungssysteme bevorzugt.

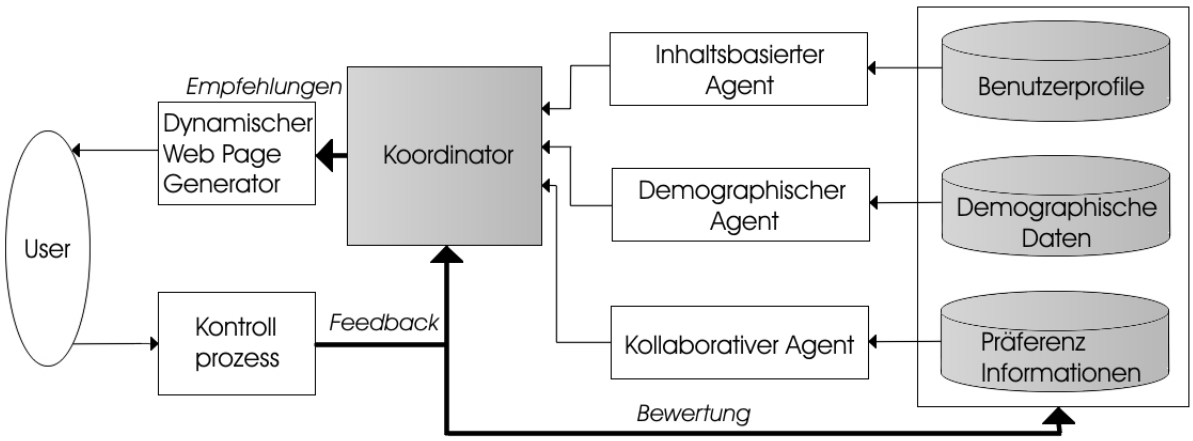


**Abb. 4.** Aufbau eines beratenden Agenten

Abbildung 4 zeigt den Aufbau des Agenten. Grundlage des Agentensystems sind drei *Datenbanken*, die für jeweils einen Agenten der drei genannten Kategorien die relevanten Daten über den Benutzer oder die Produkte speichern. Ein *kollaborativer Agent,* ein *inhaltsbasierter Agent* und ein *demographischer Agent* sind die Instanzen, die aus diesen Daten ihre Vorschläge generieren. Ihnen übergeordnet ist der *Koordinatoragent*, der alle drei Teilergebnisse erhält und die Vorschläge gewichtet. Anschließend teilt er die endgültigen Vorschläge dem Benutzer mit, beispielsweise mit einer dynamisch erzeugten Webseite. Wählt der Benutzer nun ein Item aus, egal ob es Teil der Vorschläge war oder nicht, dann wird ein Feedback an den Koordinator und an die Datenbestände weitergeleitet.
Mit Hilfe des Feedbackvorgangs soll hier die Arbeitsweise des Systems kurz umrissen werden. Hat der Benutzer ein Item gewählt, werden die Datenbestände aktualisiert. Das stößt beim Koordinator und den drei untergeordneten Agenten jeweils verschiedene Aktionen an.
Der Koordinator achtet darauf, ob das gewählte Item von einem der Untergebenen stammt. Ist das der Fall, so werden dessen Vorschläge in Zukunft mit einer größeren Gewichtung belegt. Durch diesen Mechanismus passt sich er Agent allmählich an die Umgebung an.
Der inhaltsbasierte Agent hat das Profil des aktuellen Users als einen Interessenvektor gespeichert. Aus der Beschreibung des eben gewählten Items erhält er einen Vektor, der die Zugehörigkeit des Items zu den verschiedenen Interessengebieten beschreibt. Anhand dieses Itemvektors aktualisiert das Userprofil. Mit dem aktualisierten Profil werden nun sämtliche vorhandenen Items verglichen, und durch eine Ähnlichkeitsanalyse zwischen Profil und entsprechendem Itemvektor wird ausgewählt.

Der demographische Agent erstellt seinen Eigenschaftsvektor für den Benutzer aus dessen demographischen Daten. Dann stellt er die ähnlichsten Benutzer fest und generiert die Vorschläge aus einem Durchschnitt ihrer Kaufentscheidungen, wobei dieser Durchschnitt gewichtet ist. Sehr ähnliche Benutzer haben einen größeren Einfluss als weniger ähnliche Benutzer. Das aktuell gewählte Item hat keinen Einfluss auf die Vorschläge. Es wird lediglich als Vorschlag für andere Benutzer in Betracht gezogen werden.
Nach diesem Feedback-Vorgang sammelt der Koordinator alle Vorschlagslisten ein und wählt eine gewisse Anzahl daraus aus. Diese Auswahl trifft er anhand der Gewichtung, die er für die Einzelagenten protokolliert. Zusätzlich zu der Gewichtung können auch andere Faktoren hinzukommen, die das Ergebnis beeinflussen sollen.

# 4. Anwendungsbeispiele aus dem Bereich der Web Agents

In diesem Abschnitt sollen Anwendungsaspekte für Web-Agenten beschrieben werden. Er umfasst ein Beispiel für ein konkretes Entscheidungsfindungsmodell sowie einen benutzerfreundlichen Ansatz zum Entwurf von Verhaltensszenarien, sowohl zwischen Agenten untereinander als auch zwischen Agenten und Menschen.

## 4.1. Simulation des Surfverhaltens durch einen Agenten mittels agenten-basierter Charakterisierung von Regelmäßigkeiten im Web

Die Charakterisierungen der Regelmäßigkeiten im Web sind bereits bekannt. Die Beliebtheit von Webseiten ist durch Verteilungen zu beschreiben, die den Potenzgesetzen oder auch Zipfverteilungen folgen [13].
Die Idee hinter dem folgenden Algorithmusentwurf ist es, das Verhaltensmuster eines Users beim Surfen extrahieren zu können. Der Algorithmus soll sich in seiner Funktionsweise hinsichtlich Motivation und Entscheidungsfindung einem Benutzer annähern. Erfüllt er diese Anforderungen, so wird sein Surfverhalten ebenfalls die oben genannten Verteilungen erzeugen.
Der Surfer wird als ein Informationen beschaffender Agent nachgestellt. Er besitzt einen gewichteten Interessenvektor

$$VU_n = [vu_{n1}, vu_{n2} ... vu_{ni} ... vu_{nm}],$$

wobei $vu_{ni}$ ein Wert für das Interesse des Agenten n am Thema i steht.
Das Web wird auf ähnliche Weise durch einen Raum repräsentiert, der aus Seiten (Knoten) und Links besteht. Jede Seite besitzt einen gewichteten Informationsvektor VP, der in seinem Aufbau dem oben beschriebenen Interessenvektor gleicht. Weiterhin besitzen die Knoten Links, mit denen sie untereinander verbunden sind. In diesem Modell existiert ein solcher Link nur, wenn die verbundenen Seiten inhaltlich verwandt sind.
Der Link beschreibt dann diese Verwandtschaft zweier Knoten i und j durch den Euklidischen Abstand

$$L(i, j) = \left( \sum_{k=1}^{m} \left( vp_{ik} - vp_{jk} \right)^2 \right)^{1/2}$$

Der Aufbau des Raumes geschieht nun durch die Erzeugung einer Anzahl von Seiten und Links, wenn die Seiten eine hinreichend geringe Inhaltsdistanz haben.
Um zu simulieren, wie stark der Agent zum Weitersuchen motiviert ist, wird die Motivation als eine Variable dargestellt und bei jedem Schritt neu berechnet:

$$S_{t+1} = S_t + \Delta M_t + \Delta R_t$$

M stellt die Abnahme der Motivation dar und ist immer negativ, R ist ein belohnender Faktor. Mit jedem weiteren Schritt wird M kleiner und simuliert somit das Ermüden während der Suche. Die Größe R hängt davon ab, wie sehr der Informationsvektor der Seite mit dem des Agenten übereinstimmt. Der Surfalgorithmus kommt zu einem Halt, wenn S eine obere oder untere Grenze überschritten hat. Der Agent hat dann entweder das Gesuchte gefunden oder gibt auf.
Hat der Agent die Entscheidung getroffen, einem Link folgen zu wollen, so lassen Liu et al. ihn auf mehrere Arten wählen. Entweder wird angenommen, dass sich der Agent perfekt auskennt und den Link wählt, der ihn zu seinen stärksten Interessen führt. Oder aber die Wahl der Links wird durch Zufall bestimmt, wobei Gewichtungen gegeben werden. Die Entscheidungsfindung kann pro Seite variieren, je nachdem, ob der Agent die Seite bereits „kennt“ oder nicht.
Der Gesamtalgorithmus des Agenten lautet:

**While** $S_{\min} < S < S_{\max}$
Stelle die Links des aktuellen Knotens fest
Wähle daraus einen Link aus
Surfe zum Zielknoten
Aktualisiere die Motivation S und den Interessenvektor VU
**Endwhile**

Liu et al. berufen sich auf Experimente, laut denen dieser Agent dasselbe Surfverhalten erzeugt wie ein Benutzer [13]. Auf eine nähere Beschreibung wird hier verzichtet.
Wenn die Charakteristika beim realen Surfverhalten bereits bekannt sind, wofür ist dann ein Algorithmus gut, der in dieser Hinsicht keine neuen Ergebnisse liefert, sondern nur Ergebnisse hervorbringt, die bereits bekannt sind?
Eigentlich bringt die Erkenntnis über ein solches Modell nur dann Vorteile (für das Gebiet der Web Agents), wenn es sich auf das reale Netz übertragen lässt. Um das zu erreichen, müsste eine Abbildung realer Seiten und Links auf das Modell existieren. Das Konzept des Semantic Web wiederum hat solche Abbildungen zum Ziel.
Wenn also die Situation eines vollständig beschriebenen Semantic Web eintreten würde, dann könnte der Agent als „verlängerter Arm“ des Users dienen, wenn er mit genau den Interessen des Benutzers initiiert würde.
Die Nützlichkeit des Algorithmus bezüglich Web Agents bleibt an dieses „Wenn“ gekettet.
In seiner jetzigen Position ist er allenfalls im Rahmen von Benchmarks von Nutzen.

## 4.2. Die Programmiersprache Q zur Verhaltensmodellierung und zum Entwurf von Szenarien

Ishida et al. beschäftigen sich mit einer ganz anderen Problemstellung hinsichtlich der Web Agents [14]. Ihre Aussage ist, dass die Anwendungsszenarien für soziale Agenten, die sich auf die Kommunikation zwischen Mensch und Maschine konzentrieren, selten direkt von Programmierern des Agenten definiert werden. Während die Programmierer für die Modelle und ihre Umsetzung verantwortlich sind, werden die Szenarios von Fachleuten aus dem Einsatzgebiet des Agenten vorgegeben.
Um die Übertragung eines möglicherweise komplexen Szenarios in die nicht unbedingt intuitive Welt der Agentenmodellierung zu erleichtern, stellen sie die Sprache Q vor.
Q soll als Interface Nicht-Programmierern dabei helfen, den Entwurfsvorgang von Szenarios - und damit von Verhalten – “benutzerfreundlich“ zu gestalten. Es handelt sich dabei um eine Erweiterung der Sprache Scheme [15].

Q ist als Beschreibung für Interaktionen zwischen Agenten entworfen und trifft keinerlei Aussage über deren interne Funktionsweise. Daher kann statt einem Agent ebenso gut ein Avatar, also die Repräsentation eines Benutzers, der Kommunikationspartner sein. Somit ist Q auch für die Kommunikation zwischen Mensch und Maschine einsetzbar.
Die beiden Kernbegriffe von Q sind der *Hinweis* (*Cue*) und die *Aktion*.
Ein Hinweis ist ein Ereignis, das eine Interaktion anstößt. Außer dieser Interaktion besitzt es keinerlei Seiteneffekte. Eine Aktion steht das Bemühen eines Agenten dar, seine Umwelt zu verändern.
Ihre Syntax soll an einem Beispiel beschrieben werden (Abb.5).

```
(?hear „Hello“ :from Jerry)                          (Hinweis mit ?)
(!walk :from bus_terminal :to railway_station)   (Aktion mit !)
(!speak „Hello“ :to Jerry)
(?see railway_station :direction south)
```

**Abb. 5.** Einfache Abfolge von Hinweisen und Aktionen

Eine Aktion kann synchron oder asynchron stattfinden. Würde im obigen Beispiel die synchrone Aktion (!walk :from bus_terminal :to railway_station) durch eine asynchrone Aktion ersetzt (gekennzeichnet mit !!), führt dies dazu, dass der Agent noch auf seinem Weg dorthin die nächste Aktion ausführt.
Für komplexere Situationen, in denen mehrere Hinweise gleichzeitig betrachtet werden müssen, kommen *geschützte Kommandos (guarded commands)* zum Einsatz. Ein solches Kommando listet mehrere Hinweise auf, die beachtet werden. Zu jedem dieser Hinweise sind Aktionen definiert, alle anderen Hinweise sind während dieses Kommandos nicht von Belang. Trifft keiner der erwarteten Hinweise zu, wird eine alternative Aktion unter der Klausel (otherwise) durchgeführt.
Ein Beispiel hierfür (Abb. 6):

```
(guard
        ((?hear „Hello“ :from Jerry)
          (!speak „Hello“ :to Jerry)….)
        ((?see railway_station :direction south)
          (!walk :from bus_terminal :to railway_station)…)
        (otherwise
          (!send „I am still waiting“ :to Tom)…))
```

**Abb. 6.** Beispiel eines geschützten Kommandos

Ein solches geschütztes Kommando bildet einen Zustand ab. Ein Szenario kann beliebig viele solcher Zustände enthalten und somit einen Zustandsgraphen darstellen. Die Zustands-übergänge sind dann als Aktionen abgebildet.
Ein Beispiel für ein Szenario mit dem Eingangsparameter „message“ wäre (Abb.7):

```
(defscenario reception (message)
        (scene1  ((?hear ”Hello“ :from $x)
                 (!speak “Hello” :to $x) (go scene2))
                 ((?hear “Bye”)
                 (go scene3)))
        (scene2  ((?hear “Hello” :from $x)
                  (!speak “Yes, may I help you?” :to $x))
                  (otherwise (go scene3)))
        (scene 3 …))
```

**Abb. 7.** Konstruktion eines Szenarios anhand mehrerer Szenen

Wahlweise ist sogar ein rekursives Verhältnis zwischen Szenarios möglich.
Die obigen Beispiele zeigen eine allgemeine Ausprägung von Q. Jede Agentenarchitektur, die zu Q kompatibel ist, bietet eigene Hinweise und Aktionen an. Zur Erläuterung sollen die Desktop-Assistenten von Microsoft dienen, die Microsoft Agents. Sie erweitern Q zum Beispiel mit Aktion wie „!gesture“ oder „!fly“, um die Animationen der Assistenten auf dem Bildschirm darzustellen. Auch Aktionen zum Auftauchen und Verschwinden des Assistenten wurden in die Sprache eingebracht.
Bisher bietet Q zwar die Möglichkeit, das Verhalten eines Agenten in komplexe Szenarien zu realisieren, aber die Programmierung durch Reintext wie in den Abbildungen 5 bis 7 würde sehr schnell unübersichtlich werden. Außerdem ist nicht nötig, den gesamten Spielraum von Q auszunutzen, wenn man damit Szenarien für einen bestimmten Bereich beschreiben will. Daher führen Ishida et al. das Konzept der *Interaction Pattern Cards (IPC)* ein. Eine IPC ist eine domänenspezifische Beschreibung, auf welche Art Interaktionen stattfinden.

<table>
<tr><td>Card ID</td><td>14</td><td>Card Name</td><td colspan="2">Visiting Kimono Web site</td><td>Card Type</td><td>User Initiative</td></tr>
<tr><td rowspan="2">Opening</td><td colspan="6">Action</td></tr>
<tr><td colspan="6">Hm-hum, you are so enthusiastic.<br>Then, how about this page?<br>http://www.kimono.com/index.htm</td></tr>
<tr><td rowspan="6">Reactions to Users' Mouse Click Repeat</td><td rowspan="4">Mouse Click</td><td colspan="2">Cue</td><td colspan="3">Action</td></tr>
<tr><td colspan="2">http://kimono.com/type.htm</td><td colspan="3">There are many types of obi.<br>Can you tell the difference?</td></tr>
<tr><td colspan="2">http://kimono.com/fukuro.htm</td><td colspan="3">(GestureLeft)<br>Fukuro obi is for a ceremonial dress.<br>Use it at a dress-up party!</td></tr>
<tr><td colspan="2">http://kimono.com/maru.htm</td><td colspan="3">(Evaluate Card42)</td></tr>
<tr><td rowspan="2">No Reaction</td><td>Seconds</td><td colspan="3">Action</td></tr>
<tr><td>20</td><td colspan="3">(End of Repeat)</td></tr>
<tr><td rowspan="2">Closing</td><td colspan="6">Action</td></tr>
<tr><td colspan="6">Did you enjoy Japanese Kimono?<br>OK, let's move on to the next subject</td></tr>
</table>

**Abb.8.** Darstellung einer IPC unter MS Excel

In Abbildung 8 dient Excel als Erstellungsoberfläche für MS Agents. Die gezeigte IPC beschreibt das Verhalten eines Assistenten, der einen User beim Surfen zu der fiktiven Seite www.kimono.com führt. Die IPC schreibt dem Assistenten ein bestimmtes Verhaltensmuster vor. Er kann beim Öffnen der Seite eine Aktion ausführen, daraufhin eine bestimmte Zeit auf eine Reaktion des Benutzers warten und eine abschließende Aktion vornehmen. Die Wahl der Aktionen ist durch den Sprachumfang von MS Agents begrenzt. Die Hinweise, auf die der

Agent reagiert, sind die Aufrufe von bestimmten Webseiten. Die Aktionen bestehen aus Animationen für den Assistenten (GestureLeft), Ausgabetexten, Navigationsbefehlen innerhalb des Szenarios (End of Repeat) und Navigationsbefehlen zwischen Szenarien (Evaluate Card42).
Steht der Inhalt des IPC fest, wird daraus ein Szenario in Q erzeugt, danach findet die Übersetzung in eine Programmiersprache statt. Q selbst ist plattformunabhängig, daher ist diese zweite Übersetzung durch einen geeigneten Compiler notwendig.
In Verbindung mit dem Konzept der IPC stellt Q somit ein Interface zum Entwurf von Szenarien dar. Der komplette Entwurfsvorgang für Szenarios kann in drei Rollen unterteilt werden. Der *Szenarioschreiber* stimmt sich mit dem *Programmierer* ab, mit welchen Hinweisen der Agent arbeiten soll. Der *Interaktionsdesigner* erstellt ein System, mit dem der Szenarioschreiber bequem umgehen kann, während der Programmierer sich um die Abbildung der Hinweise und Aktionen in den Algorithmus des Agenten kümmert.
Die mit Q entworfenen Szenarien könnten beispielsweise für den Skriptteil des konversationsführenden Agenten aus Abschnitt 3.2 eingesetzt werden. Kommt der Agent in diesem Beispiel zu dem Schluss, dass eine bestimmte Anfrage vorliegt, kann er das als Event formulieren und die passende Antwort durch die Wahl eines gegebenen Szenarios bestimmen. Das ermöglicht eine klare Trennung zwischen den Aufgaben der Netzwerkprogrammierung und der Dialoggestaltung.

# 5. Ausblick

Das WWW als größte Informationsquelle der Welt ist immer noch im Wachsen begriffen. Zu Beginn dieser Ausarbeitung wurde der Bedarf an Programmen aufgezeigt, der hinsichtlich der Informationsfülle im Netz besteht. Das Spektrum dieser elektronischen Assistenten soll mehr als nur fortgeschrittene Suchmaschinen umfassen; selbst die Interpretation und die Aufbereitung der gefundenen Informationen soll durch sie vorgenommen werden. Das Gebiet der Web Agents ist eine weitere Ausprägung des allgemeinen Trends zur Automatisierung.
Abschnitt 2 zeigt mit DAML-S einen Ansatz, Web Services einerseits zu beschreiben und darüber hinaus auch zu klassifizieren. Es soll an dieser Stelle noch einmal erwähnt werden, dass das allgemeine Ziel eines maschinen*verständlichen* Webs ebenfalls durch andere Standards erreicht werden kann, die jeweils Teilgebiete abdecken und sich gegenseitig ergänzen. Der Trend in diese Richtung ist deutlich; zumindest WSDL, das zur reinen Beschreibung von Web Services dient, gewinnt an Bedeutung, ebenso UDDI als elektronische Registratur („Yellow Pages“).
Abschnitt 3 stellt allgemeine Entwürfe für Agenten vor, Abschnitt 4 detailliertere Ideen aus dem Bereich der Social Agents. Es sind bereits viele Ideen beschrieben worden, wie sich diese neue Generation von Software einsetzen lässt. Würde sich die Standards für die Erzeugung eines Semantic Web in dem Sinne durchsetzen, dass sie konsequent eingesetzt werden, wäre der Weg frei für die Portierung dieser intelligenten Systeme zu Web Agents in einer WWW-Umgebung.
Mit jedem Schritt der Automatisierung stellt sich die Frage, ob es sicher ist, als Benutzer einen weiteren Teil der Kontrolle an ein Programm abzugeben. Je mächtiger solche Agenten werde, und je sensibler die übertragenen Daten und Aufgaben, desto vertrauenswürdiger müssen sie für den Benutzer sein. Es ist anzunehmen, dass Agenten im Zuge der Automatisierung auch kritische Aufgaben ausführen werden.
Die Kommunikation zwischen einem mobilen Web-Agent und einem Benutzer muss berücksichtigen, dass beide Seiten unzuverlässig sind. Der Benutzer könnte den Dienst missbrauchen oder abhören, während der Agent ebenso dazu konzipiert sein könnte,

Informationen ohne Wissen des Benutzers an eine dritte Partei weiterzuleiten. Fälle dieser Art sind längst bekannt, und die Web Agents werden diese Probleme erben.
Daher wird das nächste Forschungsschwerpunkt nach der Realisierung von Web Agents bei deren Sicherheit liegen.

## Literatur:


[1] Web Intelligence Consortium:
About WI
http://wi-consortium.org/html/aboutwi.html
[2] Christopher Browne:
Christopher Browne's Web Pages
http://cbbrowne.com/info/agents.html
[3] Martjin Koster:
The Web Robots FAQ
http://www.robotstxt.org/wc/faq.html
[4] Hui Guo, Thomas Kreifelts, Angi Voss:
SoaP: Social Filtering through Social Agents
ECRIM Workshop Proceedings No. 98/W001 of the 5 th DELOS Workshop on Filtering and Collaborative Filtering
http://www.ercim.org/publication/ws-proceedings/DELOS5/guo.pdf
[5] Joanna J. Bryson, David Martin, Sheila A. McIlraith, Lynn Andrea Stein:
Agent-Based Composite Services in DAML-S: the Behavior-Oriented Design of an Intelligent Semantic Web
Ning Zhong, Jiming Liu, Yiyu Yao - Web Intelligence, Springer Verlag 2001 (p.37-580)
[6] W3C World Wide Web Consortium:
Semantic Web Definition
http://www.w3.org/2001/sw/
[7] The DAML Service Coalition:
DAML-S: Semantic Markup for Web Services, DAML-S (OWL-S) 0.9 Draft Release
http://www.daml.org/services/daml-s/0.9/daml-s.pdf
[8] The DAML Service Coalition:
DAML-S Example Archive
http://www.daml.org/services/examples.html
[9] Joanna J. Bryson:
Cross-Paradigm Analysis of Autonomous Agent Architecture
Journal of Experimental and Theoretical Artificial Intelligence, 12(2): 165-190, 2000.
[10] Yi-Jia Chen, Von-Wun Soo:
Ontology-Based Information Gathering Agents
Web Intelligence - Research and Developement: First Asia-Pacific Conference, WI 2001 (p.423-427)
[11] Seung-Ik Lee, Chul Sung, Sung-Bae Cho:
An Effective Conversational Agent with User Modeling Based on Bayesian Network
Web Intelligence - Research and Developement: First Asia-Pacific Conference, WI 2001 (p. 428-432)
[12] Myungeun Lim, Juntae Kim:
An Adaptive Recommendation System with a Coordinator Agent
Web Intelligence - Research and Developement: First Asia-Pacific Conference, WI 2001 (p.438-442)
[13] Jiming Liu, Shiwu Zhang, Yiming Ye
Agent-Based characterization of Web Regularities
Ning Zhong, Jiming Liu, Yiyu Yao - Web Intelligence, Springer Verlag 2001 (p.19-36)
[14] Toru Ishida, Hideyuki Nakanishi
Designing Scenarios for Social Agents
Ning Zhong, Jiming Liu, Yiyu Yao - Web Intelligence, Springer Verlag 2001 (p. 59-76)

[15] J. E. Laird, P. Rosenbloom:
The evolution of the Soar cognitive architecture
Mind Matters: A Tribute to Allen Newell, 1996