Personal Agents on the Semantic Web *

Anugeetha Kunjithapatham, Mithun Sheshagiri, Tim Finin,
Anupam Joshi, Yun Peng

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore MD 21250 USA
{anu1,mits1,finin,joshi,ypeng}@cs.umbc.edu

Abstract. We describe an architecture for persistent personal agents (PPAs) designed to work on the semantic web. A PPA assists its user with everyday activities, such as maintaining a calendar, coordinating activities with other people, and answering simple queries on their behalf. Our PPAs use the semantic web languages RDF and DAML+OIL to define and use ontologies, to understand markup on web resources, and to encode the content in ACL messages exchanged with other PAs. We demonstrate the feasibility of our approach through a prototype which receives event recommendations, reasons about them, shares information with peer PPAs, and decides whether or not to add events to its user's calendar. The PPAs encode their knowledge in Jess and use FIPA languages, protocols and infrastructure to communicate with other agents. An extension to the FIPA framework allows the DAML Query Language DQL to be used to query other agents. A discovery mechanism has been developed that allows PPAs to find and negotiate information exchange with peer PPAs. We also present a simple approach to reasoning about the trust-worthiness of a query and allow a PPA to infer whether the requested information could be shared based on the user's privacy preferences.

1 Introduction

The Semantic Web is a vision to simplify and improve knowledge reuse on the Web. Efforts are underway to define the format and meaning of the language of such a Semantic Web. The structured data on the Semantic Web could serve both humans and computers, while a part of it will be formalized knowledge and will be used only by machines. The EU-NSF strategic workshop report on the semantic web[10] identifies "the applications for the masses such as intelligent personal assistants (like being the travel assistant)" as one of the key applications enabled by the semantic web. Personal assistants gather and filter relevant information and composes it into a coherent picture with regard to the user's preferences.

* This research was supported in part by DARPA contract F30602-97-1-0215.

To realize the real power of the Semantic Web, software agents that collect Web content from diverse sources, process the information and exchange the results with other agents need to be created. Software agents, characterized by their sense of autonomy, the agent's ability to control its own behavior to a certain degree and other social abilities such as the ability to exchange data with other agents, responsiveness to the environment, and pro-activeness are expected to perform roles on the Semantic Web similar to what an average user performed on the conventional web. The effectiveness of such agents will increase exponentially as more machine-readable web content and automated services become available. As described in [14], agents could help humans to cope with supposed information overload and to assist users in performing repetitive, common tasks.

A personal software assistant equipped with a model of its user's preferences operating on the Semantic Web can help to automate a wide range of activities. This could range from identifying content on the web useful to the user(from recommender agents) to managing the user's calendar. For the personal agents to successfully operate over the Semantic Web, it should possess the ability to process and manipulate the semantic markup, maintain an internal knowledge base and draw inferences from the accumulated knowledge. Inference is one of the driving principles of the Semantic Web, because it will allow the creation of software applications that derive a use from the Semantic Web data.

Personal agents offer a means for bringing the notion of personalization to the user's side with the ability to identify data directly from the Semantic Web and from other agents operating over the Semantic Web based on its internal model for the user. Thus the agent paradigm would allow a de-centralized, distributed, peer-peer type of an architecture in the place of conventional web-based information providers and recommender systems that act like centralized systems for disseminating information.

In this paper, we present our ongoing work involving Personal Agents and Semantic Web technologies. Our model architecture for a personal assistant that manipulates the user's calendar after reasoning over the information in its knowledge base and the information obtained from sources like the Xtalks system[9], peer Personal Agents and other Semantic Web resources, is described. Scenarios involving collaboration between peer Personal Agents or 'buddy' agents are envisioned and described in detail. The Personal Agent and the other agents in the multi-agent system are implemented using the Java Agent Development Environment(JADE)[3]. The Knowledge base of the agent consists of axioms and ontologies written in semantic mark-up languages, rules are expressed using the Java Expert System Shell (JESS)[13] and the agents interact using FIPA Interaction protocols and DAML+OIL[20] for the content.

Subsequent sections describe the notion of Personal Agents and Semantic Web in more detail, describes the functioning of the Personal Agent and the other agents in the multi-agent system. Agent discovery mechanisms that enables an agent to identify the communities that its user would want to exchange any information with or call for simple solicitation and recommendation of infor-

mation are explained. A formal and expressive querying mechanism that uses the DAML Query Language(DQL)[19] for query description is also illustrated. We also present a reasoning system that helps a PA reason over the trust-worthiness of a query and decide if the the requested information could be shared with the requestor based on the user's privacy preferences.

2 Background and Related Work

2.1 Personal Assistants

A Personal Assistant(PA) is a software agent that acts semi-autonomously for and on behalf of a user, modelling the interests of the user and providing services to the user or other users and PAs as and when required. It is unobtrusive but ready when needed and rich in knowledge about the users and their areas of work [32]. This is the generalized notion of a Personal Agent from the agents standards body, Foundation for Intelligent Physical Agents (FIPA) [11].

Learning Personal Agents have been used for the information filtering from the WWW [18], [2], [29]. In case of the WebWatcher [2] and the agent described in [29] the agent tries to find an "interesting" link in a Web Page that has already been pre-selected by a user. Similarly in News-weeder the user is subscribed to news-groups which are of interest to the user and have a large proportion of relevant articles.

In [27] the authors investigate how a Personal Agent could be structured to acquire a user profile, which enables it to distinguish between relevant and irrelevant documents in text form on the WWW. This user profile is then used to accomplish the task of notifying users about conference announcements and requests for proposals that match their research interests. WebMate [8] is a personal software agent that accompanies a user when he browses and searches and provides intelligent help. It learns user interests incrementally and automatically provides documents that match the user interests

The RETSINA (Reusable Environment for Task-Structured Intelligent Networked Agents) Calendar Agent(RCAL) [28], works symbiotically with Microsoft's Outlook 2000 and the Semantic Web. It can parse and reason about schedules, such as conference programs or recurring appointments that are marked up on the Semantic Web. RCAL can import and store schedules within Outlook 2000 and refer to these events to check if they have been updated, or to see if the user is free at a given time slot.

In general, the functions of a Personal Agent can be viewed as carrying out one or more of the following activities: Managing a user's diaries, filtering and sorting email, managing a user's desktop environment, managing a user's activities, plans and tasks, locating and delivering multimedia information, recommending entertainment, purchasing desired items, and, planning travel. The reference architecture of a Personal Agent as described by FIPA is shown in Figure 1

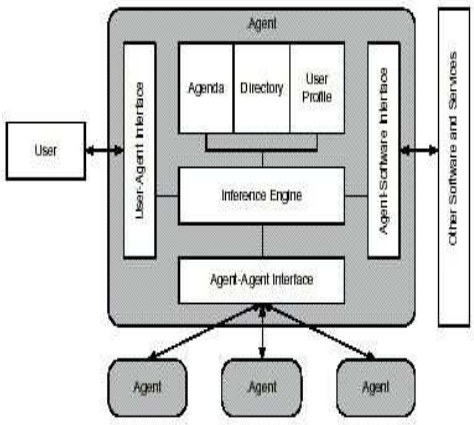

Fig. 1. FIPA Personal Assistant Reference Model

2.2 DAML and the Semantic Web

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The primary component of such a Semantic Web is the content available. Extensible Markup Language(XML)[6] and the Resource Description Framework(RDF)[21] form the underlying basis for representing such content. XML allows users to add arbitrary structure to their documents but says nothing about what the structures mean. Meaning is expressed by RDF, which encodes it in sets of triples, each triple being rather like the subject, verb and object of an elementary sentence. The third basic component of the Semantic Web comprises collections of information called ontologies. In philosophy, an ontology is a theory about the nature of existence, of what types of things exist. An ontology, in the context of the Semantic Web, is a document or file that formally defines the relations among terms [5]. The different layers of the semantic web as adapted from [4] are shown in Figure 2

The DAML+OIL[20] language is an extension to XML and the Resource Description Framework (RDF). The language provides a rich set of constructs with which to create ontologies and to markup information so that it is machine readable and understandable. It leverages and extends the expressability of RDF and RDF-Schema (RDFS) [33].

ITTalks [9] is a web portal offering access to information about talks, seminars and colloquia related to information technology (IT). It is organized around domains, which typically represent event hosting organizations such as universities, research laboratories or professional groups, and which are represented by independent web sites. ITTalks utilizes DAML+OIL for its knowledge base representation, reasoning, and agent communication. With information denoted in a semantically machine-understandable format like DAML+OIL[20], the com-

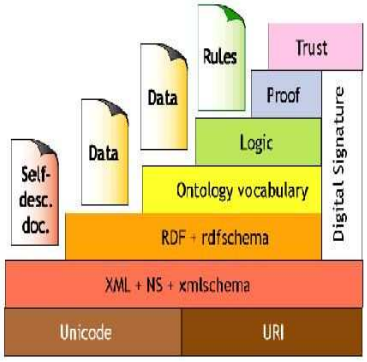

Fig. 2. Layers on the Semantic Web

puter can deduce additional information, a task which is difficult in a traditional database system.

The agent system described in this work derives and complements the work described in [9]. While [9] describes the notion of Semantic Web portals and agents improving the utility of each other, this work describes a Personal Agent architecture in the context of Semantic Web portals and its notion of being a semantic recommender residing at the user side and performing tasks on the user's behalf including collaboration with peer Personal Agents.

2.3 Rule Based Systems for Agents

[34] describes an end-user programming system that makes it easy for users to state rules for their agents to follow. The system automatically determines conflicts between rules and guides users in resolving the conflicts. The authors propose that much of the knowledge that such an agent needs can be expressed as rules of the form when these conditions are true, take these actions. They represent rules using CLASSIC, a description logic which permits users to create structured descriptions of sets of objects (known as concepts) and individual objects.

DAMLJessKB [15] facilitates reading DAML files, interpreting the information as per the DAML language, and allowing the user to query on that information. The software leverages the existing RDF API (SiRPAC) to read in the DAML file as a collection of RDF triples. It uses Jess (Java Expert System Shell) as a forward chaining production system, which exercises the rules of the DAML language on the data facts. The basic flow of this library is as follows as stated in [15]:

- *Read in Jess rules and facts representing the DAML language*
- *Have RDF API read in the DAML file and create SVO triples*
- *Take triples and assert into Jess' rete network in VSO form, with some slight escaping of literals and translation*

- *Have Jess apply the rules of the language to the data*
- *Apply the agent's rules, queries, etc*
- *Serialize relevant facts back to DAML*

3 System Design

Our Personal Agent architecure provides the basic infrastructure for manipulating the user's schedule, an internal representation of the agent's knowledge in its knowledge base(KB) and reasoning with information in its KB obtained from sources like the semantic web, peer Personal Agents and other forms of recommender agents. The utility of such a Personal Agent is in making the knowledge available through such sources of direct value to the end-user.

To demonstrate the utility and working of such a Personal Agent, complex scenarios involving recommender agents from the semantic web and peer Personal Agents were designed. In the current model of the web, various types of recommender systems are prevalent that recommend different things ranging from research papers [24][23] to TV content [17]. In many such systems, the user profile information is developed through the use of implicit machine learning techniques or by collecting explicit user preferences or in most cases, a combination of both. Such systems are limited by the amount of information that the user provides voluntarily. It would be reasonable to assume that an user would not want to divulge his complete set of preferences in a particular domain for want of privacy and other security considerations such as the amount of trust the user places on such a recommender system. And infact such systems are bound to be constrained by the scalability factor, in terms of the amount of preferences it could store.

One good solution to this problem would be to introduce the notion of a Personal Agent that resides at the side of the user, is trust-worthy and has a more complete model of the user's preferences in a particular domain making it much more capable of delivering the correct recommendations to its user. The model here would be that a third-party recommender system is not aware of specific user preferences but works with a more general model with another level of filtering being performed by the user's personal agent. A simple example of such a situation would be a talk-recommender system like "Xtalks" [35] being aware of the fact that the user is very interested in talks in the area of wireless computing but not the fact that the user never likes to attend talks by Mr. Foo Bar on the subject.

Also, conventional web-based information providers and recommender systems act like centralized systems disseminating information. The Personal Agent model allows a de-centralized, distributed, peer-peer type of an architecture. For instance, a system like 'Xtalks' would recommend talk announcements only to registered users. But a peer-peer multi-agent model would provide capabilities for even unregistered entities to receive the information. The multi-agent system designed and implemented to demonstrate these ideas consist of the following agents - the user's personal agents, recommender agents like the 'Xtalks' Agent and information agents like the 'Mapquest' Agent.

3.1 The Xtalks Agent

This agent acts as a third-party recommender agent that recommends talk announcements to registered users. Users register their Personal Agents with the Xtalks agent to have them receive talk announcements. Also, users can express their interests, schedule and location constraints through a DAML profile. The Xtalks agent monitors the Xtalks [35] system for new talk announcements. When new talk announcements get added to the system, the Xtalks agent informs registered Personal Agents about the talk announcement.

3.2 The Mapquest agent

The 'Mapquest' agent built around the Mapquest [22] system provides information about distance, driving time and driving directions between two addresses. The required information is scraped from the Mapquest [22] system, massaged into a form suitable for inter-agent communication and sent to the requesting agents. The Mapquest agent exposes its service using the FIPA Request Interaction Protocol[1]. In the future, one could assume that web-pages from Mapquest [22] are augmented with semantic markup or the information being available as a web-service thereby eliminating the need for an intermediary agent to convert web-page content into agent-friendly representations.

3.3 The User's Personal Agent

The central entity of the system is the user's Personal Agent which reasons with all the knowledge input from different sources and arrives at meaningful conclusions on behalf of the user. The Personal Agent is equipped with a '*brain*' that essentially stores various types of information in the form of triples and comprises of a rule-based reasoning engine that manipulates the information in its KB to draw meaningful conclusions. In addition to interacting with recommender systems like Xtalks, the personal agent also collaborates with peer Personal Agents both recommending and receiving recommendations about talk announcements and arriving at a conclusion based on the following factors.

- *User's interest in the talk announcement topics*
- *User's schedule constraints (both in terms of availability and feasibility to attend the talk)*
- *Decisions of other Personal Agents the user would like his Personal Agent to interact with in arriving at a decision*

User's interest in the talk: The user agent is assumed to have a good model of the user's preferences for the particular domain, in this case, topics in Computer Science. It is available to the Personal Agent through an explicit DAML profile in which the user's interests are represented as topics in the ACM topic hierarchy[25]. The talk announcements are also available as DAML URIs with the topics marked up from the ACMtopic hierarchy. Rules specifying the following statements are loaded in the *brain* which help the agent in determining the interest of the user in that talk

- *If the user is interested in a particular topic in the hierarchy, he is also interested in all the sub-topics of that topic*
- *If the user is interested in a particular topic and the topic of the talk happens to be the same, then the user is interested in the talk.*

To determine all the sub-topics of a topic in the ACM hierarchy, a rule of the form, *"The sub-topics of a sub-topic of a (parent) topic are also sub-topics of the (parent) topics"* is inserted in the reasoner .

User's schedule constraints: The following conditions are checked to determine the feasibility of scheduling the talk on the user's calendar.

- *Whether the user has an empty slot in his calendar for the period of that talk*
- *Whether the talk location is reachable in available time from the location of the previous appointment of the user*
- *Whether the next appointment of the user (after the talk) is reachable from the location of the talk in available time.*

The time calculations are computed through interaction with the Mapquest Agent.

Peer-Personal Agents Interaction: The notion of interaction between peer Personal Agents draws its roots from the concepts of instant-messaging and its popularity in helping to form online communities. This agent interaction scenario tries to mimic the real world phenomena of forming buddy lists and engaging in group messaging. The user specifies the set of so called 'buddy-agents' which represent the group of Personal Agents of the user's buddies.

As in any instant-messaging/buddy-list application, the formation of buddy-list is preceded by a set of subscription and reply messages that establish the identity of the buddies with each other. The user could specify his list of buddies through publicly accessible DAML-URIs that specify agent details such as names and transport addresses. A Personal Agent can locate and subscribe to peer Personal Agents through the DAML URIs thereby expressing their willingness to both recommend and accept talk recommendations from the peer agents.

Thereafter, Personal Agents can exchange recommendations and decisions among themselves. The following are the responses an agent can send in the reply to a request from another Personal Agent regarding a decision to attend a particular talk.

- *if talk does not match interest, response value is 0.*
- *if talk matches interest but there is schedule conflict, response value is 1.*
- *if talk matches interest and passes Buddy Recommendation Test(*), response value is 2.*
- *if user confirms about his willingness to attend talk, response value is 3.*

(*) The buddy recommendation test succeeds if average score received from buddy agents exceeds a threshold. In this case the threshold could be the median value of 1.5.

There is a possibility of an occurrence of a dead-lock. A simple scheme employed to avoid deadlocks is, whenever an agent receives a query about a talk that it itself is waiting for replies from buddy-agents, then it immediately returns a special value of 1.5 (middle of all levels). This might also be used when the agent receiving the recommendation is not aware of the talk that is being recommended and hence does not really have a decision. Note that this request for a decision could itself act as a talk recommendation when it happens that the receiving agent is not aware of the talk.

In addition to multi-agent interactions to arrive at a decision in scheduling a talk, the Personal Agent can also be loaded with special rules to over-ride the default behavior.

Such rules could be of the following form: (These are merely examples and are in no way an exhaustive set of rules that can possibly be derived in such a situation. These examples are just to show the richness in adopting such an approach)

- *If the talk is on a specific topic, schedule the talk irrespective of other considerations.*
- *If the speaker of the talk is a specific person, schedule the talk irrespective of other considerations.*
- *If a specific buddy of mine decides to attend the talk, schedule the talk if the talk is of interest to me, irrespective of what my other buddies decide, provided there are no schedule conflicts.*

The Personal Agent also exhibits a pro-active kind of behavior in some situations. For example, if the user had specified his home location to be Baltimore, a system like 'Xtalks' would recommend talks to the user in and around Baltimore oblivious of a user's travel plans. In such situations, the Personal Agent monitors the user's calendar to figure out if it encounters a situation wherein the user has an appointment scheduled on the calendar that is away from the default home location, indicating that the user is travelling. It might also happen that the Personal Agent would have never received a talk recommendation for a talk happening away from the user's home location. In such a case, the Personal Agent queries the 'Xtalks' system for talks that could be happening at the new location.

A situation wherein there is some cancellation of a particular appointment from the user's calendar indicating that a potentially interesting talk could not have been scheduled because of earlier schedule conflict is also taken care of. Given the model of the Personal Agent, the decision of not scheduling such a talk because of schedule conflicts resides in its *brain*. So, the Personal Agent requests for information from the 'Xtalks' agent for that particular talk. Alternately, the Personal Agent adopts a lazy approach and request for talks during the period of the cancelled appointment and try to schedule it on the user's calendar. Such functionalities are modelled as additional plug-ins that augment the capabilities of the personal agent.

4 Key Implementation Insights

4.1 Agent Interaction Mechanisms

All the agents are implemented using the Java Agent Development Environment (JADE). JADE [3] is a software framework to develop agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. JADE can be considered an agent middle-ware that implements an Agent Platform and a development framework. Inter-Agent communication is through the FIPA-ACL [30] which specifies a standard message language by setting out the encoding, semantics and pragmatics of the messages. Each agent resides in its own individual platform with agent communication enabled using the Internet Inter-ORB Protocol (IIOP) and communicates with agents in other platform using the IIOP transport mechanism provided by JADE.

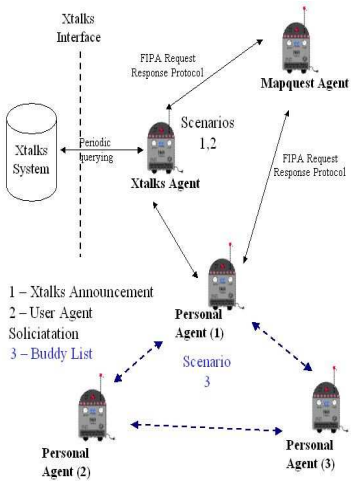

Fig. 3. Multi-Agent Scenario and Interactions

The Xtalks agent also exposes a query interface through an ontology, which defines the set of supported queries that an agent could pose to the Xtalks agent. Different Personal Agents exchange talk recommendations as defined by the buddy list ontology. Messages could be either talk recommendations, which are also used by the Xtalks agent or a 'query-if' in FIPA-ACL language act, asking the other agent whether (it believes that) a given proposition is true [31]. The sending agent requests the receiver to inform it of the truth of the proposition. In this case, the proposition is the sending agent's belief regarding the interest of the other agent (user) in the talk.

Thus all agent interactions use DAML+OIL as the content language in the FIPA-ACL message. A Personal Agent can acquire more knowledge from its peer

Personal Agents through queries. As a step towards framing a generalized peer to peer querying mechanism, we have developed an ontology[12] for representing a query in DAML+OIL, based on the DAML Query Language (DQL)[19]. Queries are framed using this ontology and the query descriptions are packed into FIPA ACL messages. The interaction between the agents is modeled after the FIPA Request Interaction Protocol[1]. This mechanism would enable peer personal agents to query each other about any information that an agent can store in its knowledge base. Examples of talk recommendation messages and the query-if messages that confirm to the buddy-ont [26] ontology are provided below.

```
Talk recommendation received from xtalks agent:
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:daml="http://www.daml.org/2000/10/daml-ont#"
xmlns="http://daml.umbc.edu/ontologies/buddy-ont#"
>
<Talk-Recommendation>
   <URL>http://www.ittalks.org/q.link/daml/ 20020613131516< /URL>
< /Talk-Recommendation>
< /rdf:RDF>

A buddy recommendation-request:
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:daml="http://www.daml.org/2000/10/daml-ont#"
xmlns="http://gentoo.cs.umbc.edu/subhash/buddy-ont#"
>
<Talk-Interest>
   <URL>http://www.ittalks.org/q.link/daml/ 20020613131516< /URL>
   <Value>3< /Value>
< /Talk-Interest>

< /rdf:RDF>
```

The Query consists of a Query Pattern, an answer KB pattern, an indication of the may-bind/must-bind variables and optionally consists of a query premise, an answer bundle bound size and a justification request. A Query Pattern is viewed as a collection of DAML+OIL expressions, in which some literals and/or URIRefs are replaced by variables. An answer to the query contains a binding of a URIRef or a literal to each of the must-bind variables of the query pattern and zero or more of the may-bind variables. The variables are bound to terms which occur in the DAML+OIL language or in the answer KB; the answerKB is the knowledge base which contains the query pattern with all the variables in the query replaced by the inferred bindings if a binding exists, and all other variables replaced by new RDF blank nodes.

4.2 The Personal Agent's Reasoning System

Our Personal Agent architecture is independent and easily extensible, with new capabilities being added seamlessly. The personal agent can interpret information in the DAML+OIL[20] language. Internally, DAMLJessKB[15] is used to read in DAML file and interpret its contents which are later asserted into Jess[13]. The

original version of DAMLJessKB represents the interpreted facts in the *(PropertyValue subclass man person)* format. In our system, an alternative representation of the facts has been introduced. Ordered facts such as *(property subclass man person)* are actually equivalent to an unordered fact such as *(property (_data subclass man person))*, i.e., the slot data is all stored in one multi-slot. Matching data in multi-slots is less efficient than matching data in normal slots because, there exists an extra indirection through an array. Hence we chose to store the facts in an unordered format using normal slots, which looks like *(property (p subclass) (s man) (o person)))*. We specifically chose Jess as the rule inference engine since it is optimized for these kind of unordered facts. Also, Jess is written in Java and hence easily integrates with the Personal Agent and can easily be packaged and deployed. The key idea with our Personal Agent architecture is that user could download and add components developed by third-party vendors that the user trusts which can augment the functionality of the Personal Agent. For example, a Personal Agent with components designed for interaction with a 'Xtalks' agent could be augmented with an 'Amazon' component that helps it interact with an 'Amazon' agent to receive different kinds of recommendations such as books, movies etc., The most rudimentary way of envisioning a plug-in would be as a JADE behavior that the agent can pick-up and execute at runtime. In such a situation, a plug-in manager would simply search a preconfigured directory for new behaviors and add them to the list of behaviors currently executed by the Personal Agent.

The Personal Agent interacts with the user through Microsoft Outlook. The user's calendar is assumed to be stored on Microsoft Outlook. Additionally, any appointment that the Personal Agent schedules for the user is stored on the Microsoft Outlook calendar for the user. The java-based agent interacts with Microsoft Outlook using Bridge2Java [7]. Bridge2Java is a tool that allows Java programs to communicate with ActiveX objects. It allows easy integration of ActiveX objects into a Java Environment. Using the Java Native Interface and COM technology, Bridge2Java allows an ActiveX object to be treated just like a Java object.

Another useful user-interface extension explored and developed with the Personal Agent through Microsoft Outlook is the 'Outlook Today Extensions for Xtalks' [16]. This downloadable component offers a user interface that displays both the list of most recent talks announcements available from Xtalks and those scheduled by the Personal Agent in the user's calendar serves to show the filtering done by the Personal Agent through its internal reasoning process.

Exchange of information among peers: The Personal Agent possesses various kinds of data in its KB: some persistent data, that is present all through its lifetime and some dynamic data acquired through various interactions and reasoning during its lifetime. This information might be useful to the peer personal agent's and hence could be shared with the interested party. While interactingwith a peer, the PA will encounter a situation when it would have to determine the trust-worthiness of the requested information exchange. Though its impossible for a PA to come up with the decision emulating it's user's pref-

erences, a simple and straight-forward mechanism that would allow the PA to reason over the information requested and the user's privacy preferences stored in its KB could help the PA come to a conclusion. In the subsequent paragraphs, we briefly describe the mechanism that we implemented for the PAs to reason over in-coming queries to determine the trust-worthiness of the potential information exchange. The mechanism allows the PA to decide if it should send a relevant answer to the query or to send a refuse message.

The facts in the PA's Knowledge base are categorized as *sharable, non-sharable* or *sharable with the user's consent.* In our model architecture, most of the personal information, past/present appointments and class schedule information of the user are classified as *sharable* facts. *Non-sharable facts* consist of confidential information. Facts such as the current location (determined by sensors), future appointments etc. are categorized as facts *sharable with user's consent.* The PA on receiving a query responds based on the type of information requested.

- *If the information is categorized as sharable, the PA replies with a relevant answer immediately*
- *If the information is categorized as not sharable, the PA sends a refuse message*
- *If the information is categorized as sharable with user's consent, the PA sends a mail to its user about the request and replies according to the user's response.*

We also framed various other rules based on the user's relationship with the requestor and some other rules based on the requestor' role. To enable the PA to identify the appropriate rules to execute, we have come up with a set of rules and their order of execution. Some of the implemented rules based on the relationship with the requestor are mentioned below:

- *If the requestor is a family member share any requested information.*
- *If the requestor is a friend share the sharable information, schedule information and appointments.*
- *If a colleague share only sharable information.*
- *If Advisor, share SSN, schedule information along with sharable information.*
- *If academic department staff member, share sharable information, SSN, class schedule etc.*

In the system, the above mentioned rules are expressed as Jess rules[13]. A *Cache* component is made use of to keep track of the rejected queries, to allow the PA determine the urgency of the query, possibly based on the number of times the PA got the same query. For example: A simple rule like *'if the query is asked for the third consecutive time, answer to the query with user's consent'* is applied to exploit the information in the cache.

4.3 Agent Discovery Mechanism

Before collaboration among a community of agents, an important step in forming such a community is agent discovery. We have implemented a discovery mechanism which in many ways is similar to buddy look-up like mechanisms in popular instant messaging systems. Instant messaging applications provide a feature to search people on the web, based on theirs names, email-ids etc. However all of these instant messaging systems rely on databases for finding matches. Our discovery mechanism uses existing infrastructure in the form of search engines. Search engines are engineered to handle massive amounts of information and by using search engines to locate agents our technique inherits scalability. Currently our technique can discover agents based on their owner's name.

Our approach requires that the agent owner has a homepage and his/her homepage shows up as a document in the first 'n' results(in our implementaion, we took n=10), when the owner's name is entered as a query in the search engine such as Google. Further we also require a HTML META tag to be embedded in the homepage. This tag contains a pointer to the user's profile in DAML+OIL. The entire mechanism consists of two phases- Agent Discovery and Agent Subscription. For the purpose of simplicity we refer to the person initiating the discovery as the user and the person who owns the agent being discovered as the owner. The following steps are involved. (1) User enters the name of the agent's owner using the agent's GUI (2) The name is forwarded to Google as a query. (3) The DAML profile of the owner is obtained from his/her homepage using the META tag. (4) The profile contains name, organization and other details about the agent owner. It also contains agent contact information (ip + port). (5) All matching results are displayed to the user. (6) Based on the information gathered the appropriate agent/owner is chosen. (7) A FIPA Subscription Request is sent to the owner's agent. (8) The owner's agent on receiving the request sends its owner an email. This mail is in the form of a HTML. The e-mail contains user's details and hyperlinks to capture the owner's decision. (9) In reponse to the owner's response a corresponding FIPA INFORM message is sent back to the user's agent.

5 Conclusion

The Semantic web is a vision to augment the current web with formalized knowledge and data that can be processed by computers thereby shifting the focus away from a human-centered interaction. Various technologies including agents, ontologies and information management are currently being developed to make the Semantic Web a reality. We have presented an example architecture for a Personal Agent working on the Semantic Web collaborating with recommender agents and peer Personal Agents. The Personal Agent operates using a rule-driven *brain* operating on Semantic Web data and the user profile. The multi-agent system interaction through DAML+OIL naturally extends the language for knowledge representation to being the language for communication. The system in general demonstrates the notion for future systems that would operate

on the Semantic Web and integrate well with day-to-day software that the user utilizes.

As the Semantic Web language evolves, agents and reasoners operating on them need to evolve constantly. The rule based JESS reasoner could be adapted to be more "fuzzy" to more closely model real-world decision making process. Though the agent operates with rigid rules, schemes by which a user can easily convey these rules in a language that the agent understands is the key for the success of any personal assistant.

The notion of intelligent Personal Agents never really took-off in the context of the web as it stands today. But the emergence of the Semantic Web promises to change that, for, the concepts of intelligent agents and the Semantic Web are a synergy. Effective and private access to user's desires, preferences, and habits coupled with information garnered from the Semantic Web promises to offer potent personal assistants such as the one described in this paper, that are bound to make life simpler for their masters.

6 Acknowledgments

We would like to thank Subhash Kumar, a member of the UMBC alumni, currently with IBM, for his significant contributions to this work. The work presented in this paper has been extended over his Master's Thesis.

References

1. Fipa request interaction protocol specification. World Wide Web, `http://www.fipa.org/specs/fipa00026/XC00026F.html`.
2. Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. Webwatcher: A learning apprentice for the. In *AAAI Spring Symposium on Information Gathering*, pages 6–12, 1995.
3. Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade- a fipa-complient agent framework. In *Proceedings of PAAM'99*, pages 97–108, 1999.
4. Tim Berners-Lee. The semantic web vision, 2000.
5. Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 2001.
6. T. Bray, J. Paoli, and C. Sperberg-MacQueen. Extensible Markup Language. `http://www.w3.org/TR/1998/REC-xml19980210`, 1998.
7. Bridge2Java. World Wide Web, `http://www.alphaworks.ibm.com/tech/bridge2java`.
8. Liren Chen and Katia Sycara. WebMate: A personal agent for browsing and searching. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 132–139, New York, 9–13, 1998. ACM Press.
9. R. Scott Cost, Tim Finin, Anupam Joshi, Yun Peng, Charles Nicholas, Harry Chen, Lalana Kagal, Filip Perich, Youyong Zou, Sovrin Tolia, and Ian Soboroff. Ittalks: A case study in the semantic web and daml. In *proceedings of the International Semantic Web Working Symposium*, July 2002.

10. EU-NSF. Research challenges and perspectives of the semantic web - report from the joint european commission and national science foundation strategic workshop on the semantic web, 2001.
11. FIPA. World Wide Web, `http://www.fipa.org/`.
12. An Ontology for representing DAML Queries. World Wide Web, `http://daml.umbc.edu/users/dqlontology.daml`.
13. Ernest J. Friedman-Hill. Jess, the expert system shell for the java platform, 2002.
14. M. Hoyle and C. Lueg. Open sesame: A look at personal assistants, 1997.
15. Joe Kopena. Damljesskb, http://plan.mcs.drexel.edu/projects/legorobots/design/software/damljesskb/.
16. Subhash Kumar. Outlook today extensions for xtalks. World Wide Web, `http://daml.umbc.edu/download/`.
17. K. Kurapati, S. Gutta, D. Schaffer, J. Martino, and J. Zimmerman. A multi-agent tv recommender.
18. Ken Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc. San Mateo, CA, USA, 1995.
19. DAML Query Language. World Wide Web, `http://www.daml.org/2002/08/dql`.
20. DARPA Agent Markup Language and Ontology Inference Layer. World Wide Web, `http://www.daml.org/language`.
21. O. Lassila and R. Swick. Resource Description Framework. `http://www.w3.org/TR/1999/REC/rdf-syntax-19990222`, 1999.
22. Mapquest. World Wide Web, `http://www.mapquest.com`.
23. S. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S.K. Lam, A.M. Rashid, J.A. Konstan, and J. Riedl. On the recommending of citations for research papers. In *Proceedings of ACM 2002 Conference on Computer Supported Cooperative Work (CSCW2002), New Orleans, LA*, pages 116–125, 2002.
24. Stuart E. Middleton. Exploiting synergy between ontologies and recommender systems.
25. ACM Classification Ontology. World Wide Web, `"http://daml.umbc.edu/ontologies/classification`.
26. Buddy List Ontology. World Wide Web, `http://gentoo.cs.umbc.edu/subhash/buddy-ont`.
27. A. Pannu and K. Sycara. A learning personal agent for text filtering and notification, 1996.
28. Terry R. Payne, Rahul Singh, and Katia Sycara. Calendar agents on the semantic web., 2002.
29. M. Pazzani, L. Nguyen, and S. Mantik. Learning from hotlists and coldlists: towards a www information filtering and seeking agent, 1995.
30. FIPA ACL Message Structure Specification. World Wide Web, `http://www.fipa.org/specs/fipa00061/XC00061E.html`.
31. FIPA Communicative Act Library Specification. World Wide Web, `http://www.fipa.org/specs/fipa00037/`.
32. FIPA Personal Assistant Specification. World Wide Web, `http://ww.fipa.org/specs/fipa00083`.
33. S. Staab, M. Erdmann, A. M adche, and S. Decker. An extensible approach for modeling ontologies in rdf(s), 2000.
34. Loren G. Terveen and La Tondra Murray. Helping users program their personal agents. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*, volume 1, pages 355–361, 1996.
35. Xtalks. World Wide Web, `http://www.ittalks.org`.