A Semantic Web Architecture for Advocate Agents to Determine Preferences and Facilitate Decision Making

Wolfgang Ketter
RSM Erasmus University
Rotterdam, the Netherlands
wketter@rsm.nl

Arun Batchu
netrii.com
Eden Prairie, USA
arun@acm.org

Gary Berosik
Thomson Reuters R&D
Eagan, USA
gary.berosik@thompsonreuters.com

Dan McCreary
Dan McCreary & Associates
St. Louis Park, USA
dan@danmccreary.com

Abstract
The world-wide-web (WWW) today consists of distinct, isolated islands of data and metadata. In the near future we expect the availability of a critical mass of data and metadata for use by intelligent agents that act on behalf of human users. These agents would identify, propose and capture new opportunities to assist human users in satisfying their goals, by traversing and acting on this semantically rich and abundant information. We envision a new class of agents, their networks and their communities that exist for the sole purpose of serving as their human "master's" Advocates – *Advocate Agents.* Advocate Agents learn a human's goals and preferences, collaborate with other agents, mine semantic content, identify new opportunities for action, propose them and finally transact them, while always keeping the human "in-the-loop." This paper discusses this class of distributed, intelligent, Advocate Agents, their potential uses, and proposed architectures and techniques that provide a conceptual framework for these networked agent societies to collaborate in the achievement of their human user's goals.

Categories and Subject Descriptors
D.2.11 [Software Architectures]; I.2.11 [Distributed Artificial Intelligence] *Intelligent* Agents, Multi-Agent Systems, Business Networks; K.4 [Computers and Society] Organizational Impacts, E-commerce

General Terms
Architecture, Algorithms, Economics, Standardization, Theory

Keywords
Preferences Elicitation, Personalization, Recommendation Agents

1. Introduction
Many managerial decisions involve multiple objectives. For instance when purchasing products via a real-time business network, a procurement manager may want to buy products from the retailer who offers the lowest price, the fasted shipping time, the lowest weight, and has the best reputation in terms of reliability, customer service, and trust. We can immediately see that the lowest price and fastest shipping time are conflicting objectives and we need to perform a cost-benefit analysis on which dimension(s) we are willing to make sacrifices. In addition, the problem is compounded by other factors. These include the sheer number of different online retail stores which might offer this kind of product, as well as other websites such as blogs and news sites that might offer information about the product, such as usability, and comparisons to similar products. The latter might even suggest buying a different product altogether. In such circumstances, a consumer is faced with an overload of information related to making a decision.

Herbert Simon coined the term bounded rationality [1] to refer to the fact that the human mind has limited cognitive abilities. Humans tend to use approximate methods, such as rules of thumb stemming from their own experience, to deal with most decision problems. Human decision makers often seek solutions that are satisfactory rather than optimal. These rules of thumb are refereed in the literature as heuristics [2] and are often well adapted to the structure of their knowledge about the environment. For example, when a prospective graduate student is looking for a PhD research position, he might focus his search on places which are ranked highly regarding their quality of research, and only then consider different characteristics such as income or the quality of life in the city where the university is situated.

The goal of our work is to research, develop, and use highly personalized agents to complement the cognitive limitations of the human mind to facilitate the decision making process (including the gathering of information and recommendation of actions). These agents work in a collaborative manner with users to accomplish their goals. To work effectively and efficiently with a human user, the agents must learn the human user's interests, habits and preferences (as well as those of their communities) [3]. In the online retail example, recommendations can be given as to what to buy (product-brokering) and from whom to buy (merchant-brokering), based on customer criteria. After the discussion of relevant related literature, we present a unique and innovative design of the business and technical architectures for Advocate Agents, and describe the challenges of implementing them. We conclude with a roadmap for future work.

2. Related Work

Much research is published on personalization and recommendation agents regarding customer preference modeling [4-6], based on prevalent Internet approaches. Currently there are three main streams of research, each dependent on how customer preferences are modeled: via vector similarity (collaborative and content-based filtering), probability (Bayesian), or association (correlation between user and item). One of the shortcomings of these research efforts is that they apply agents that represent clusters of people. That is, recommendations for a specific user are based on the preferences of many different users [7, 8] (such as Movielens.umn.edu and Amazon.com) rather than tailored to the needs of an individual user. Up until now agent actions have always been performed on the server side of the processing architecture rather than the client side (e.g. in the case of a typical online store application). As a consequence, these agents do not always act in the best interests of the customer. E.g., a retail agent might try to convince customers to buy more products that give the seller a higher profit margin. Further, such agents have little insight into the human user's real preferences since they can only observe those actions related to their competence and performed on their processing site. We envision a learning Advocate Agent, residing on the client side, which always supports and promotes the best interests of its master.

In [9], the authors present e-Butler, a first step toward a customer-centric architecture, but they have not yet implemented their architecture. These authors note that HTML is not fit for semantics, a limitation that is quickly disappearing with the advent of XHTML, micro-formats and presentation artifacts being moved into CSS-based display approaches. The authors promote XML, which is still a good recommendation, but they do not address RDF or other semantic technologies that we recommend in our proposed architecture. The e-Butler authors talk about XML services using SOAP, while REST[1] is the alternative web service approach that we suggest, since it can be more effective than SOAP-based approaches. In addition, our design leverages the power of social networks, and encourages the collation of temporary business relationships to increase negotiation power with agents from other businesses.

We envision every user having readily-available personal Advocate Agents that are part of a supportive multiagent information system [10]. Such agents are semi-autonomous, always reacting to additional input and feedback from the user. The currently most prevalent way to interact with a computer is via direct manipulation [11], a mode of operation in which the computer waits for specific commands from the user before performing responsive actions. Our vision is that we will be able to interact with a computer in a proactive manner, the same way we interact with people. This vision is based on agents that proactively collaborate with a user, in a mixed-initiative fashion, to predict the appropriate next steps that can speed up, and improve the quality of, a user's overall decision process. Such agents are sometimes referred to as expert or interface agents [3, 12]. Relevant examples of proactive email and web retrieval agents can be found in [13]. Those agents retrieve and present archived emails to a user if they are similar to current email being written. In other situations, similar agents retrieve archived web pages that are similar to the one currently being displayed.

In [14], the authors propose that the current hypertext document-centric web will evolve to include an infrastructure of machine-readable documents that can be understood and acted upon by intelligent agents. In the last seven years we have seen significant developments in progress toward the realization of this vision of the Semantic Web. We believe that conditions are now ripe for an explosion of useful Internet-based tools that go beyond simple keyword search and human-driven information mashups. We foresee a day in the near future when sophisticated personal agents will run continuously in the background on our behalf, exploring, monitoring, filtering, mining, collaborating and presenting relevant information for our utility, while flexible trust mechanisms will act to appropriately constrain the autonomous authority of those agents.

3. Advocate Agent Business Architecture

Advocate Agents do not merely facilitate product or merchant search, they "get to know their master," meaning the human decision maker, through initial profiling (sex, age, nationality, preferences, etc) and by observing the behavior of the decision maker (and their communities) while performing goal-oriented actions (e.g., visiting an online retail site, time spent, links clicked, dates visited, websites visited previously, etc.). Based on this information, the agents recommend actions (buy this product, buy from this merchant, read this article, etc), and autonomously execute some of the decisions the consumer delegated to it (make a dentist appointment; buy airline ticket; bid on eBay.com, etc.). In addition to reducing information overload, customers that use software recommendation agents reduce their workload and improve the effectiveness and efficiency of their decision making [15]. These agents represent the preferences of the consumer as his or her highly tailored "software avatar" in the Internet.

We define the business architecture of Advocate Agents as an abstract model that satisfies the requirements of Advocate Agents, independent of selected implementation technologies. Business architectures transcend and outlive concrete, related realizations of the vision and technical architectures. While technology changes at a rapid rate, this abstract model changes less rapidly, in tune with the rate at which the definition and scope of Advocate Agents change. Figure 1 displays our business architecture for Advocate Agents, the modules of which are described below.

3.1 Architecture Module Descriptions

We describe the key modules necessary for an Advocate Agent and its collaborators to succeed in serving its master. We describe each module and the role it plays in the overall architecture.

3.1.1 Observation Module

It is clear from the requirements that an Advocate Agent is a personal champion of a single master. In order to provide a personalized service, an agent will need to learn its master's preferences. To learn, it must observe. The observation module provides the sensory organs of the Advocate Agent. The module's architecture must allow multiple such organs to collect information (e.g. web-browsing behavior, e- mail, instant-messaging etc.) and send it on for further processing.

[1] *REpresentational State Transfer, http://en.wikipedia.org/wiki/Representational_State_Transfer*

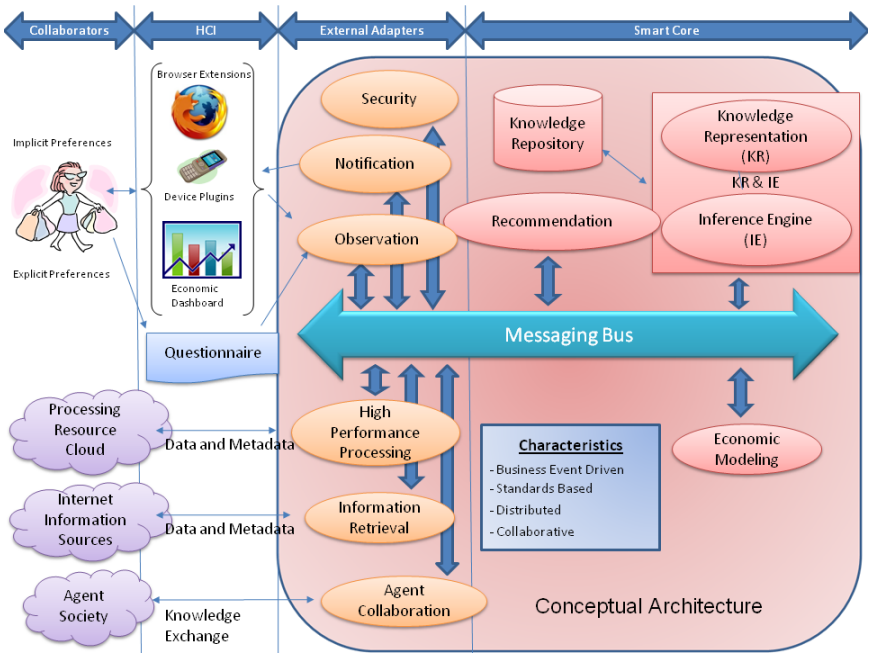

Figure 1: Advocate Agents Business Architecture

3.1.2 *Information Retrieval Module*

We envision the techniques of *Information Retrieval* to play a key role. This module is necessary for the agent to convert potentially vast amounts of unstructured but valuable information into useful knowledge.

3.1.3 *Knowledge Representation and Inference Module*

This module is responsible for processing raw information into semantically precise knowledge. This knowledge representation module is also responsible for collaboration with other agents in knowledge acquisition and knowledge sharing activities.

3.1.4 *Agent Collaboration Module*

In order for Advocate Agents to succeed in fulfilling their goals, they will have to collaborate with other Advocate Agents. This key module supports agent cooperation capabilities in dealing with the exchange of information and knowledge.

3.1.5 *Recommendation Module*

This module comes up with recommendations for the master. Triggered by either new preferential knowledge about its masters or new facts, this is the module that searches for new solutions that might get filtered into recommendations.

3.1.6 *Notification Module*

When recommendations need to be conveyed to the master (in preferred ways), this module seeks out the most effective (possibly multiple) means of performing the notifications, and carries them out.

3.1.7 *High Performance Processing Module*

There are many elements of this environment that need to perform large-scale information processing tasks such as web crawling, indexing, search result relevance determination, sentiment analysis, relationship extraction/assessment and related natural language processing. To allow Advocate Agents to be usably responsive, much of this type of processing will need to take place in parallel, in real-time, and to reach successful completion as quickly as possible. The High Performance Processing module is responsible for coordinating the execution and completion of these tasks. It is expected that at least some of this need will be met by cluster-based, highly parallel and fault-tolerant "cloud computing" approaches [16].

3.1.8 *Security*

Support for security on Advocate Agent platforms are provided by the security elements of this architecture. Typical security infrastructure capabilities would include support for trusted security domains, digital signature handling and public key encryption/decryption. Security support would be provided to agent execution platform elements via messaging bus services as well as to the agents themselves via directly linked software library components.

3.1.9 *Economic dashboards*

Users have different roles and associated goals (professional and private) depending on their situation. Economic dashboards provide a dynamic perspective of information relating to those different roles and give rapid feedback regarding Advocate Agent performance on specific goal-oriented tasks. We envision economic dashboards as non-intrusive user interfaces that facilitate tactical and strategic decision making processes. Dashboards present summarized views of historical economic and newly gained knowledge, based upon data harvested from the web using information retrieval and drill-down capabilities. One goal is to find out if the human decision maker prefers the agent's findings and choices compared to their own findings. This requires validating the goodness of recommendations suggested by the agent. Users can give explicit feedback to the system to evaluate the goodness of the findings and suggestions. Such

explicit feedback, representing user preference, is then translated into business rules that enable an agent to know how to best reason on behalf of the human user in similar future situations. We envision using a continuous feedback range of numeric values, e.g. from 1 (not satisfied) to 5 (very satisfied), to record human preferences.

We hypothesize that, over time, a human decision maker will delegate an increasing number of decisions to Advocate Agents. We define this as *adjustable autonomy*. There is a switching point for the adaptation of a new technology by users, usually based on its usefulness and how smoothly the new technology integrates into daily life. We envision non-intrusive, proactive agents which will work according to the user's preferences even when they have no immediate assignment. Examples of proactiveness include looking for relevant new trends or recent news, informing the user about interesting ongoing auctions such as eBay.com, good deals, etc.

3.2 Infrastructure

In order for Advocate Agents to succeed, mature and scalable infrastructure elements will be needed.

3.2.1 Communications

Fundamental to meeting business requirements is for agents to communicate with their peer agents, information sources and their human masters. Without an empowering means to communicate, agents will be severely limited in providing optimal recommendations. The distributed and multi-module computing nature will require a unifying mechanism – a platform where existing and new modules can be plugged in and out and be able to communicate to each other while avoiding point-to-point knowledge. We propose that the concept of a service *Bus* is a good solution for the problem.

3.2.2 Standards Infrastructure

Standards for communication protocol and format will both be very important for seamless and straightforward exchanges. Such standards will allow new knowledge to be created and easily flow among elements of the agent environment. Standard communications simplify and support flexible information and knowledge fusion.

3.2.3 Maintaining Transformation Rules

In order for Advocate agents to be successful, an entire developer community will need to participate in its success. But individual developers cannot create components that interoperate without some standards. In order for Advocate agents to be applied, other services will need to be transformed to use formats that are usable by those Advocate agents. Transformation rules can be defined and applied to assist in enabling this.

3.2.4 Computing Infrastructure

Advocate Agents will need to have as small an execution-time footprint as possible in order to realize the principle of unobtrusiveness. Apart from the communication infrastructure, the observation modules will need sensors in all human user electronic touch-points, including wired and wireless devices, web browsers and electronic mail. Processing will get intensive at times, so sufficient hardware resources will be needed.

3.3 Agent Communities and Cloud Support

We envision a proliferation of Advocate Agents that constantly communicate with each other and form collaborative communities that collectively answer questions.

3.3.1 Collective Intelligence

Collective Intelligence [15] is a powerful emergent phenomenon in organic networks such as ant colonies, social networks in humans and the Internet. A similar phenomenon for Advocate agents is possible with our proposed architecture. Traditional data mining methods based on static data many not be effective for highly dynamic data. The challenge is that, as new assertions arrive, the training set is constantly altered by an agent [17]. There is, however, no existing standard for classifying assertions in web pages. In order to effectively build and process local and global semantic knowledge stores and related information structures, Advocate Agents need to rely on a wide range of text and information processing technologies that support what is termed *collective intelligence* [18]. These technologies can provide agents with the ability to collect, relate, summarize, assess and act upon relevant facts and opinions expressed as assertions embedded in web content and user behavior. Several of these technologies are described in the following paragraphs.

Foundational natural language processing technologies that support this include tokenization, part-of-speech tagging, syntactic and semantic parsing and tools that ease the definition and execution of flexible processing pipelines. Such processing pipelines successively extract and enrich information to support agent tasks, and the processing components of such pipelines are often strong candidates for high performance processing parallelization. Other advanced techniques that build upon these fundamentals are also available to support agent tasks. These include making recommendations, discovering information groups and classification categories, assignment of information and documents to existing categories and ontologies, focused search and relevance ranking of results, solution optimization and document filtering.

There is a range of relevant statistical, machine learning, information retrieval and text mining techniques that can be used to address these tasks. Collaborative and other filtering approaches can be used to develop group preference based recommendations. Hierarchical, k-means and enhanced word sense (e.g. via WordNet) clustering techniques can be used to identify previously unknown categories of information. Relevance ranking schemes allow effective ordering of search results. Simpler statistical techniques such as Naïve Bayes classification are often the most efficient and effective for certain classification tasks. K-nearest-neighbor techniques can help make statistically "best" prediction choices. There is also a range of more computationally intensive statistical techniques including kernel methods, support vector machines and conditional random fields that are favored for certain types of classification. This is just a partial list of the *collective intelligence* techniques that Advocate Agents can apply to fulfill their goals. We expect the full range of these techniques will be regularly applied by Advocate Agent systems.

3.3.2 Agent Metadata Usage

Our architecture enables agents to participate in a market where the barriers are very low to non-existent. Such an architecture will

empower such markets to exhibit the characteristics of a free market allowing the most optimal recommendations to be formed and made to an Advocate Agent's master.

3.3.2.1 Web Crawling for Gathering

The key to this is a very simple database with just two data types: documents and links. Links are all of the same type (a Universal Resource Identifier or URI). Each successive page causes a search that itself generates new searches. If each page has on the average of N links in it the number of successive iterations searched scales as: Pages Crawled = $(N)^{interactions}$ .The average number of links per page is typically between three and four depending on what document types are included.

3.3.2.2 Resource Description Framework for Representation

We are now seeing another generation of applications being created to enhance our lives - the web mashup. In this type of application, if you are not happy with the way a web site arranges information, you can use the page information to harvest other facts. For example, consider Wikipedia, and a situation where each page has a database representation similar to the structure of the web page, but instead of simple pages and links, the data is rendered into another type of network – one in which each link is a fact about the page. These facts can be stored in a mirrored version of Wikipedia data (e.g., DBPedia[2]) using a format called Resource Description Framework (RDF[3]). RDF takes simple facts (e.g. from a wiki page) and converts them into assertions in the form of triples:

[Subject] [Predicate] [Object]

For example, a statement on a page about Berlin Germany such as: *Berlin is the Capital of Germany* becomes:

[Berlin] [Capital-Of] [Germany]

Storing these facts in a uniform structure allows queries to be performed on this dataset. As with web crawlers, an initial query can fan out and extract additional facts from pages, but there is one important difference. Unlike linked web pages, RDF statements require that each fact and link be a URI (e.g., the [[Capital of]] relationship is a URI in the above example). By forcing precise and unique relationships to also be unique URIs, RDF queries can look for relationships between triples that have not been joined together. This simple and repeatable rule of triples can generate a complex, arbitrarily large "system of knowledge". This semantic "web join" capability allows us to join otherwise unrelated data stored in general web page content.

RDF triples are stored in structures called "triple stores" analogous to a single relational database table with three columns. Triple stores have their own query language called SPARQL[4], which facilitates the creation of "mashups" of data. It does this by allowing discovery of nodes in separate semantic graphs that represent the same object. This in turn allows new facts to be generated from the RDF triples. This ability to *generate new-from-existing knowledge* in a machine-readable manner is the catalyst that allows Advocate Agents to be smart, creating many exciting possibilities. The new facts are semantically precise, unpolluted by human-related web artifacts, such as visual elements, and are "interactionable" by automatons - Advocate Agents in particular. Table 1 compares the three knowledge handling approaches, suggesting ways in which agents that use the public internet will require new tools.

	Relational	**Web**	**Semantic Web**
Designed for	Storing and Joining Tabular (Rectangular) Data	Linking Documents	Inference of Distributed Assertions
Metaphor	Tables	Hypertext	Graphs
Search	SQL	Keywords, meta data	SPARQL
Search Engine	RBDMS vendors	Google, Yahoo, Microsoft	Triple Store

Table 1. Document Web and Semantic Web Comparison

3.4 Security and Privacy

For an open and collaborative world to thrive, the citizens participating in that world need to feel safe and secure. The era of social networking in the digital world has ushered in new challenges pertaining to the privacy of and security of information. The next era of Advocate Agents working on behalf of their human masters introduces even more challenges. We suggest a few models that will help in address these challenges.

3.4.1 Social Trust and Verification

Digital social impersonation is a significant problem in human social networks where a human being presents a benign and friendly interface to another to win her or her trust, but with malicious intent. With agents, we predict that the same phenomenon will happen and we need to develop ways to counteract these impersonations. In social networks, trust is usually built on friend-of-a-friend (FOAF) connections. Using the Semantic Web, FOAF seems a natural fit to develop trust networks. The tolerance of the trust levels can be then tuned from a very low 1 (direct) degree of separation to *n* degrees of separation. Organizations like http://www.Garlik.com are doing interesting work in this arena.

3.4.2 Private and Public Firewalls

Every agent should be able to identify ("have a notion of") the private and public information allowed by the human master to be used by agents in support of knowledge sharing and learning tasks. Each human user will have different degrees of tolerance and boundaries of what she considers private or public. Therefore the agent infrastructure needs to accommodate these notions. It is possible that an agent's capability to learn and acquire new, relevant knowledge pertaining to satisfaction of the master's goals could be affected by the policies of privacy firewalls.

3.4.3 Behavioral Inferences

Another manner of protecting the privacy of sensitive information is to anonymize the subject and object of sensitive elements and assess its affects on semantic inference quality. For example, if it can be inferred that people who like solid state drives also like shopping on amazon.com, it might be that the personal information of Wolf and Arun (whose behaviors led to the

[2] DBPedia, *RDF representation of Wikipedia, http://dbpedia.org.*

[3] RDF Specification , http://www.w3.org/RDF/

[4] *Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/*

inference) may be insignificant and hence anonymized without affecting the quality of the inference.

3.4.4 Central Authorities and Open Identifications

Another approach that can be borrowed from the web world is the concept of *Certificate Authority (CA)*. It is possible that a trust system can be developed where Advocate Agents have a means of trusting a certificate issued by a *CA*, a chain of CA's, or both. Of relevance is a recent surge in *open* identity systems, such as http://OpenId.net, that apply a decentralized identification mechanism based on open standards. Advocate Agents could tap into this approach and use a mutual trust system to build communities.

4. Advocate Agent Technical Architecture

Here we describe several of the current and future technologies that will be required to achieve the vision of Advocate Agents.

4.1 Trends

In the 1990's and the early 21st century, content was king. As the Internet expanded in scope geographically as well as in volume (number of nodes), content became more freely available over a wide variety of websites. The Internet started to develop a "long tail" [19] where boutique websites contained very specialized information. The ability to reach those sites became very important, so linkages among the sites became correspondingly important. Search engines like Google that leverage the rich relationships between the content became the hub of Internet activity. The resulting improvement of relevance and recall of sites by search engines meant that finding relevant information became easier than ever. Infrastructure improvements like ubiquitous broadband internet access all over the world meant that a massive number of diverse knowledge workers had access to highly specialized information sources. The human collective now needed a chance to exchange information - not only the ability to read (Web 1.0) but also the ability to exchange information on the web (Web 2.0). This urged websites to become more "humane", driving innovations such as the 'editable web' (the rise of the Wiki – e.g., Wikipedia), social bookmarking (e.g., http://del.icio.us), community sites (e.g., Flickr, MySpace and FaceBook), collaborative sites and technologies (e.g., Google Apps, Microsoft Sharepoint), and a host of other open source Content Management Systems.

The Web browser, which was initially designed as a document-centric window into the web, became an application platform of the web. Technologies like DHTML (Dynamic HTML) and AJAX (Asynchronous JavaScript and XML) make sites very interactive and user-friendly. Instant Messaging over the web is an essential way to connect with other users. User experience on the web has reached levels where the Web is the primary source of finding, providing and sharing information. All of the mentioned sites utilize the tremendous power of collective intelligence - the "wisdom of crowds". Standardization of empowering technologies like RSS and Atom Publishing Protocol (APP) permit "web mashups" (e.g. Google Maps and Yahoo Pipes). Web users have vast amounts of information at their fingertips; but therein is the dilemma. This very information has become as debilitating as Achilles' heel because of its sheer volume and the need for the human brain to process all this information while supporting personal goals and preferences.

Two other significant communities have been gathering momentum. One community realized that humans needed assistance from agents that competed and collaborated to meet desired goals (including, but not limited to, economic goals). Another community realized that we needed a precise way to capture essential knowledge, search it, share it and grow it via collaborative techniques such that it could be actionable by machines. Enterprises have typically used highly structured databases for such actionable knowledge but such technologies house only a very small fraction of the knowledge on the web and are mostly proprietary. They are information silos. Web 3.0's vision for the Semantic Web Community combines what we know works well on the world-wide-web with semantic technologies like RDF to capture precise relationships, RDF triple stores to persist them, SPARQL to query them, OWL (Web Ontology Language) to develop ontologies, XML to serialize knowledge, HTTP and other RESTful technologies [20] to communicate between systems, and syndication protocols (Atom, APP and RSS) to notify and deliver content. We believe that the Agent Community and the Semantic Community together offer a great platform for realizing and delivering the vision of Advocate Agents depicted in our technical architecture (see Figure 2).

4.2 The Browser is the Platform

We argued that the browser should be leveraged and that it is not just a window to the document web but a full-fledged platform. Recent developments offer hard testimony attesting to this proposition. Firefox, the world's second most popular browser, is free and open source. Its excellent extension architecture supports literally hundreds of available extensions. Since most personal interaction with the web happens through the browser, Advocate Agents stand to gain a tremendous insight into their master's behavior if they could operate as a browser extension. Such a extension would support several of the significant characteristics of Advocate Agents - the ability to observe, and non-invasive interaction/recommendation. AdaptiveBlue[5] is one relevant example of a Firefox extension.

4.3 REST - A Matter of Style

Representational State Transfer (REST) is an architectural style that is currently in use in the "human" web. While detailed discussions of the merits and constraints of REST are beyond the scope of this paper, RESTful applications are simpler, more scalable and can have a very small footprint - an ideal style for our vision of Advocate Agents where the multiplicity of such Advocate Agents can range from a single entity to an ultra-large-scale system (ULS). Since architectures based on REST tend to focus on Resources (a web page, an XML or RDF document, image etc.), they are termed Resource Oriented Architectures (ROA) or ROC (Resource Oriented Computing). We believe that the REST architectural style possesses the necessary properties to realize the business architecture of Agents. To support this, we cite the example of modern e-commerce web-services like Amazon.com's various API's, Google's GData and Flickr, as well as modern exchange protocols like the Atom Publishing Protocol which embrace REST styles. 1060 Research's NetKernel[6] is an excellent example of an advanced deep-REST engine used to

[5] *http://www.adaptiveblue.com/*

[6] *http://1060.org/*

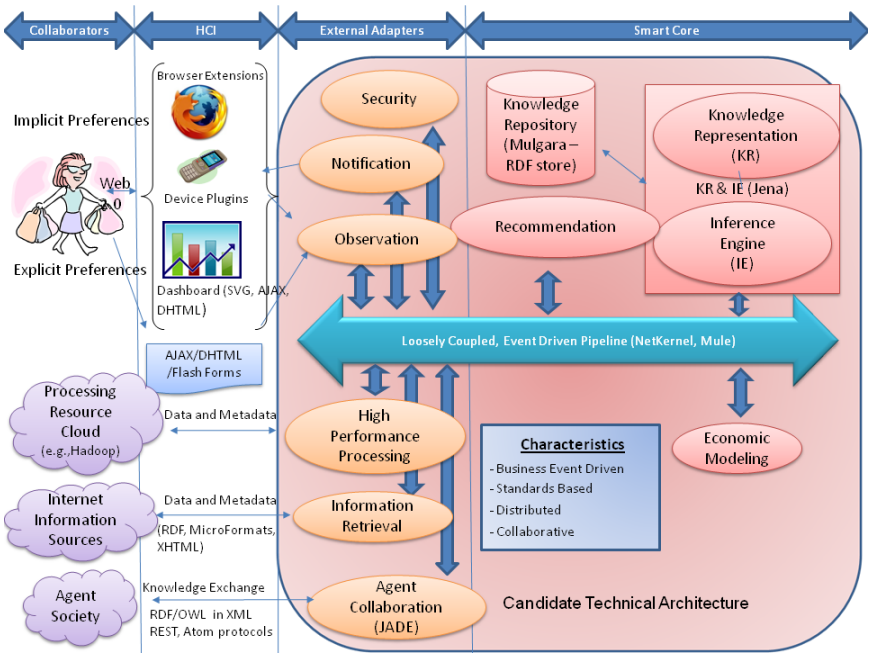

Figure 2: Technical Architecture for Advocate Agents

build scalable, high performance applications. Several of the authors have successfully deployed complex systems based on REST interfaces

4.4 Agent Framework

Although the browser is an excellent platform for an Advocate Agent to leverage the web interactions of its master, a separate small-footprint framework will be utilized to tap into other types of interactions that its master might have with the environment. These include E-mail, instant messaging and other mobile or desktop applications that provide additional contextual information for the Advocate Agent to be successful in achieving its goals. Such a framework will minimally have the modules we describe in the business architecture above. JADE [21] is one popular agent framework that can be utilized to construct Advocate Agents. Frameworks that embrace non-proprietary web standards will be prevalent and preferred. Utilizing open standards, agent framework implementations can proliferate, encouraging competition and collaboration which will work in favor of the Consumer - the human master.

4.5 Semantic Service Bus Architecture

We propose using a variation of an Enterprise Service Bus (ESB)[7] [22] architecture for integrating a family of Advocate Agent services. In general, an enterprise service bus is a collection of services that can be used by any application within an enterprise, independent of computer language or operating system. The key difference between a standard ESB architecture and the one we propose is the use of a library of business-rule-driven transforms that map (and potentially convert) generic RDF triples into RDF statements that conform to Advocate Agent semantics. The ESB will provide several adapters for fitting a new module into the ESB. The ESB will allow different modules to communicate by configuring incoming and outgoing endpoints to enable information and knowledge to flow. Simple and complex pipelines can be thus created with each processing stage generating a semantic event. All these events can then be harvested for further utility as we describe further in our paper.

4.6 Pipelines in Event Driven Architectures

Content and metadata needs to be transformed in numerous ways before it can be captured as knowledge. Pipelining and deliberate, complete decoupling of message routing between component endpoints is a powerful and very flexible way to construct complex systems. As pipelining manifested itself in many ways over time in computing, it became clear that it is beneficial to trade off slight inefficiency in message handling for effectively infinite flexibility. As hardware and software resources become cheaper and human resources more expensive, the ability to quickly rearrange components in ways that are impossible to predict is very essential.

Pipeline architectures define processing components as a linear sequence of operations. The order and structure of these pipelines can be easily configured using declarative rules. Some ubiquitous examples of pipeline architectures are Unix pipes, functional languages, and the HTTP servers that make-up the modern web. Other examples include RSS/Atom mashups and XML pipelines such as those specified by the W3C XProc standard[8].

Event-driven-architectures (EDA) view every transformation or computation as an event that might be subscribed to, routed, vetoed, audited, inspected or in other ways processed before being dispatched to any other component (however small or large). This approach amplifies the flexibility and reliability of such pipeline architectures due to inherent decoupling. SEDA (Staged Event

[7] For example see Mule: An Open Source Enterprise Service Bus at http://mule.mulesource.org/

[8] For example set XProc: An XML Pipeline Language W3C Working Draft 29 November 2007 at http://www.w3.org/TR/xproc

Driven Architecture), an example of such an architecture, has been beneficially utilized in the past (e.g., Mule ESB). Complex Event Processing (CEP) engines exist to maximize and leverage information event storms (instead of getting drowned by them). Since Advocate Agents are primarily infomediaries [23, 24] and will process vast amount of knowledge with specialized modules and complex routing, we feel that Pipelines and EDA should be seriously considered for application to such systems.

4.7 XML Exchange and Integration

Despite inherent verbosity, markup languages have been hugely successful because they are both human-readable yet easily parseable by machines. XML (Extensible Markup Language) has become far more than just a way of delimiting comma or tab separated files. XML has become an entire ecosystem of declarative languages and tools to process them. XML Schema are commonly used to efficiently validate form and structure. XML is now the most common way to express domain specific languages (DSLs). The new standard for HTML, XHTML, is a DSL expressed in XML. Because of this, any XML processor can process XHTML (or for that matter produce or consume it). XML documents are ubiquitous. They pass to and from the web pages as they are used in AJAX applications. They are the primary serialization mechanism for web services, be it SOAP, XML-RPC or REST. Most electronic data interchange formats are XML based. XML has become so ubiquitous that we don't even notice it. For Advocate Agents to fully realize their power, their architectures need to embrace XML comprehensively - a distinct trend being seen in the enterprise world.

Advocate Agents will need to communicate process and store XML natively. The tooling for XML has matured significantly to reduce the overhead of XML plumbing. Web service consumers and producers are pre-built and available in every modern language. Standards like XPath[9] allow sophisticated, complex queries on XML documents. XForms[10] is a standard that allows complex forms to be built using a declarative style. The symmetry of XSLT, a full-featured declarative language for processing XML, allows pipelines to be built easily. XQuery[11], a recent standard, allows XML mashups to be constructed very quickly[12]. High performance, native XML databases like the popular, open source, native XML database eXist[13], and most major commercial databases, virtually eliminate the overhead of converting (shredding) XML to relational structures. XML IDEs like Stylus Studio, Oxygen and XMLSpy make short work of XML development, further reducing the learning curve required to effectively apply XML. From many architectural viewpoints (functional, development, operations), XML adoption to support Advocate Agents is highly recommended.

[9] http://www.w3.org/TR/xpath

[10] XForms Specification, http://www.w3/TR/XForms

[11] XQuery Specification, http://www.w3.org/TR/xquery

[12] For an excellent example of using XQuery to mashup data see: http://en.wikibooks.org/wiki/XQuery

[13] http://www.exist-db.org/

4.8 Knowledge Processing and Information Retrieval

These paragraphs describe detailed workings of the Knowledge Representation and Inference Engine module in our technical architecture (Figure 2).

4.8.1 RDF Harvesting

Many researchers doing work on intelligent semantic web agents see their projects fall into several phases:

1. **Gather** - Gather assertion data from various sources using a variety of formats such as RDF statements, microformats, and RDFa
2. **Convert** - Convert data to a useable format such as RDF and store it in triple stores
3. **Analyze** - Analysis of this data to see if it is relevant to the objectives
4. **Recommend** – Modify product ranking and potentially create recommendations and notifications for their master

The first step depends on web page authors representing their data in some machine readable format such as microformats embedded in XHTML pages. Unfortunately, early web browsers were very forgiving in their ability to correctly use HTML that was not well-formed. As a result it became highly problematic to harvest machine-readable facts from many web pages. With the growth of standards such as XHTML and machine readable formats like microformats, and RDFa, it is becoming feasible to harvest content into RDF assertions and store the assertions into triple stores. Inference engines work on these triple stores to satisfy queries (SPARQL).

4.8.2 Microformats and RDFa

Several new technologies, such Microformats, allow intelligent agents to harvest facts or assertions on the web. Microformats are small additions to XHTML pages that make content semantically precise and machine readable. Microformats take advantage of the fact that all HTML tags have a **class** attribute that can be used to add semantic information to an HTML file without disrupting its display to a human viewer.

4.8.3 Business Rules

We propose a series of rule-based transformations that can harvest RDF resources and transform or link them to standards that are registered by one or more metadata repositories used by the semantic integration bus. These metadata repositories are similar to OWL[14] files that are currently being used by the semantic web community. The difference is that these repositories contain more information than is currently encoded in the OWL standard. This includes traceability data such as what person or agent added a data definition to the metadata registry and when these events occurred. This information, typically stored in an ISO/IEC 11179 metadata registry[15], can be quickly transformed to create validation schemas and user interfaces such as XForms.

[14] OWL Specification , http://www.W3.org/TR/OWL

15 ISO/IEC 11179 Metadata Registry, http://metadata-standards.org/11179/

4.8.4 Knowledge of User Preferences

User preferences will be stored as triples, e.g. the following statements might be stored in a user preference triple store:

[Wolf] [prefers Rating above][3]

[Wolf] [prefers Vendor][Amazon]

4.8.5 Knowledge of User Needs

An Advocate Agent learns of its master's needs via observations or direct questions. For example, after a user does several searches for computer hard drives, an agent might store the following assertion in its objectives triple store:

[Wolf] [needs Product] [Hard drive]

4.8.6 Knowledge of Available Products

The agent must combine knowledge of the user and user needs with data about where these products might be purchased, and about how keywords related to user needs also relate to product taxonomies. The following is a sample list of these assertions:

[Amazon] [has product] [Samsung Solid State Drive]

[Best Buy] [has product] [Samsung Solid State Drive]

[Samsung Solid State Drive] [has Rating above] [4]

[Samsung Solid State Drive] [is an instance of the class of products] [hard drive]

4.8.7 Deriving New Knowledge

Merging of RDF graphs produces new knowledge. Inference engines do the merging, given SPARQL queries. Advocate Agents generate new queries depending upon user preferences, observations and activities. As new product knowledge (e.g. solid state drive) comes up, new product matches might be discovered that the Advocate Agent non-intrusively suggests to its master. In this case, a (SPARQL) query for a hard drive that matches Wolf's preferences should return a series of products in ranked list similar to the way search engines return web search results in a page-rank order. The Samsung Solid State Drive would be listed in this product ranking and an [Explain] link near the product would allow the user to see in plain language why that product was ranked at a specific level.

4.8.8 Knowledge and Inference Architecture

The overall architecture for knowledge representation and inference engine is depicted in Figure 3.

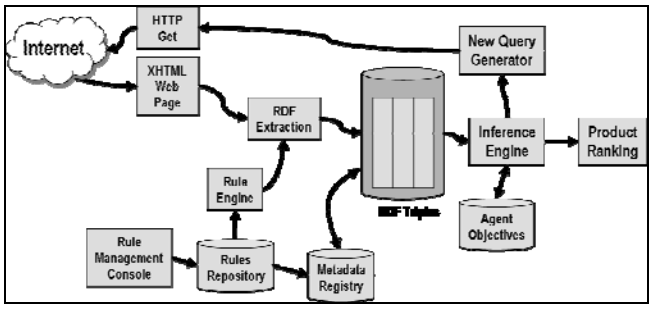

Figure 3: Knowledge representation and inference engine architecture.

Figure 3 shows the following interactions:

1. A user creates one or more objectives and saves the objectives in the **Agent Objectives** database.

2. The **Inference Engine** takes these objectives and uses this data to search its **RDF triple store** and construct new queries. It passes these queries to the **New Query Generator** which converts it to **HTTP Get** queries on the Internet, to look for possible product information, recommendations and other metadata.

3. These **HTTP Get** requests return well-formed **XHTML Web Pages** that are run through an **RDF Extraction** and harvesting process.

4. The RDF extraction process is governed by a **Rule Engine** that determines what RDF statements will be added to the **RDF triple store**.

5. The executed rules are stored in a **Rule Repository** that is consistent with the semantic bus discussed previously.

6. Changes to the **RDF triple store** may trigger new queries or may change the **Product Rankings** of products.

7. Users may change the rules at any time using a **Rules Management Console**. This is a set of **Rule Templates** generated by the Advocate Agent system.

These processes continue within the constraints defined by the user, such as network bandwidth limitations, CPU limitations or search fee budgets. The user is notified when product ranking goals have been achieved or resource limits have been reached.

4.8.9 Making Rules Visible

The current trend is to move away from hard-coded rules in procedural languages toward declarative rule-based systems. These systems store business rules in registries that can be created and updated by the user community. Users can view and change business rules via the economic dashboard. In addition users can ask questions such as "Why did you make this decision?" We propose storing rules for extracting data from other systems in a rules repository that is tightly coupled with the metadata registry. Rules can then be expressed in terms of a series of condition/action statements, as selection lists that reference values in the metadata registry. Advocate Agents must be able to perform independent research on their own without specific guidance from the user. They may begin with a small query and use the results to gain additional facts. These repetitively enriched facts are then used to gain additional information and make appropriate inferences until goals are achieved. One key difference between this and the way in which current web crawlers work is that Advocate agents autonomously make fact-grounded assertions and use them to reach specific, human-personalized objectives.

4.8.10 Leveraging Agent Societies

With the highly collaborative, standards-based architecture that we propose, we envision that cooperative agents will naturally gravitate towards communities not unlike human social networks. Each agent group's members will have a symbiotic relationship among themselves with respect to helping each other out.

5. Conclusions and Future Work

This paper presents an architecture for Advocate Agents - preference-based, user-centric, learning, infomediary agents that facilitate human decision making. The facilitation is supported by various machine learning, data mining, recommendation, and business intelligence techniques, as well as cutting edge architectures and technologies such as Enterprise Service Bus, Inference Engines, Processing Resource Clouds (e.g., Hadoop), RDF, RDFa, Microformats, SPARQL, Rule-based systems, XML Pipelines, XForms, XQuery, XPath, and XML. The main contribution of this paper lies in demonstrating feasibility of Advocate Agents by presenting an architecture that integrates all these technologies into a unique system and demonstrating that all the components of Advocate Agents can be built from already existing methods and elements.

We are currently working on a prototype of our architecture. In the next step we'll bootstrap preference models with initial customer profiles obtained through a questionnaire and collaborative filtering [25]. Afterwards we plan on observing individual users through a longitudinal study with many participants. The agent will record all relevant user actions. After gathering this observation data, we plan to reverse engineer the human user objective. With the help of data mining we are planning to design rules based on how good the human decision fits the underlying data. Then the agent will attempt to optimize the decision space and come up with a possible new recommendation for a user. Since the feature space of the environment's specific domain might be quite high, we plan on applying a method such as Principal Component Analyzes (PCA) to reduce the dimensionality of the space, so that a following optimization is faster and produces more intuitive results for the user. A critical question we must answer is: "Is an agent capable of optimizing individual real life human decisions?" In connection to this, we would like to determine whether the human decision maker prefers an agent's choice to his own or not. This means we must validate the goodness of recommendations suggested by the agent. Our first prototype is in collaboration with Dutch Flower Auctions – an Advocate Agent for a flower wholesaler. Current efforts involve researching this specific feature space and the connected preference space of individual flower traders in detail.

6. References

1. Simon, H.A., *Rational Decision Making in Business Organizations.* The American Economic Review, 1979. **69**(4): p. 493-513.
2. Gigerenzer, G. and P.M. Todd, *Simple Heuristics That Make Us Smart*. 1999: Oxford University Press.
3. Maes, P., *Agents that reduce work and information overload.* Communications of the ACM, 1994b. **37**(7): p. 30-40.
4. Kumar, R., et al., *Recommendation Systems: A Probabilistic Analysis.* JCSS, 2001. **63**(1): p. 42-61.
5. Schafer, J.B., J.A. Konstan, and J. Riedl, *E-Commerce Recommendation Applications.* Data Mining and Knowledge Discovery, 2001. **5**(1): p. 115-153.
6. Varian, H.R. and P. Resnick, *CACM Special Issue on Recommender Systems.* Communications of the ACM, 1997. **40**.
7. Adomavicius, G. and A. Tuzhilin, *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.* 2005.
8. Xiao, B. and I. Benbasat, *Consumer Decision Support Systems for E-Commerce: Design and Adoption of Product Recommendation Agents.* MIS Quarterly, 2007. **31**(1): p. 317-209.
9. Adomavicius, G. and A. Tuzhilin, *An Architecture of e-Butler: A Consumer-centric Online Personalization System.* International Journal of Computational Intelligence and Applications, 2002. **2**(3): p. 1-15.
10. Wellman, M.P., E.H. Durfee, and W.P. Birmigham, *The digital library as a community of information agents.* Expert, IEEE [see also IEEE Intelligent Systems and Their Applications], 1996. **11**(3).
11. Shneiderman, B., *Direct manipulation: A step beyond programming languages.* 1981.
12. Maes, P., *Social interface agents: Acquiring competence by learning from users and other agents.* Software Agents—Papers from the 1994 Spring Symposium, Technical Report SS-94-03, Etzioni, O., Ed, 1994a: p. 71-78.
13. Rhodes, B.J. and P. Maes, *Just-in-time information retrieval agents.* IBM Systems Journal, 2000. **39**(3&4): p. 685-.
14. Berners-Lee, T., J. Hendler, and O. Lassila, *The Semantic Web.* Scientific American Magazine 2001.
15. Haeubl, G. and V. Trifts, *Consumer Decision Making in Online Shopping Environments: The Effects of Interactive Decision Aids.* Marketing Science, 2000. **19**(1): p. 4-21.
16. Jeffrey, D. and G. Sanjay, *MapReduce: simplified data processing on large clusters.* Commun. ACM, 2008. **51**(1): p. 107-113.
17. Witten, I.H. and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. 2000: Morgan Kaufmann.
18. Segaran, T., *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. 2007: O'Reilly Media, Inc.
19. Anderson, C., *The Long Tail: Why the Future of Business is Selling Less of More*. 2006: Hyperion.
20. Richardson, L., S. Ruby, and D.H. Hansson, *RESTful Web Services*. 2007: O'Reilly Media, Inc.
21. Bellifemine, F., A. Poggi, and G. Rimassa, *Developing Multi-agent Systems with JADE.* Intelligent Agents VII: Agent Theories Architectures and Languages: 7th International Workshop, ATAL 2000, Boston, MA, USA, July 7-9, 2000: Proceedings, 2001.
22. Chappell, D., *Enterprise Service Bus*. 2004: O'Reilly Media, Inc.
23. Varun, G. and T.C.T. James, *E-commerce and the information market.* Commun. ACM, 2001. **44**(4): p. 79-86.
24. Hagel, J. and M. Singer, *Net worth*. 1999: Harvard Business School Press Boston.
25. Billsus, D. and M.J. Pazzani, *Learning collaborative information filters*. Proceedings of the Fifteenth International Conference on Machine Learning. 1998. 46-54.