The Semantic Web Paradigm for a Real-Time Agent Control (Part I)

Vasile Mazilescu

Dunarea de Jos University Galati, Romania

17. February 2010

The Semantic Web Paradigm for a Real-Time Agent Control (Part I)

Vasile MAZILESCU

Department of Accounting and Economic Informatics
University Dunărea de Jos of Galati
vasile.mazilescu@ugal.ro

Abstract. For the Semantic Web point of view, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning. Adding logic to the Web, the means to use rules to make inferences, choose courses of action and answer questions, is the actual task for the distributed IT community. The real power of Intelligent Web will be realized when people create many programs that collect Web content from diverse sources, process the information and exchange the results with other programs. The first part of this paper is an introductory of Semantic Web properties, and summarises agent characteristics and their actual importance in digital economy. The second part presents the predictability of a multiagent system used in a learning process for a control problem.

Keywords: Semantic Web, agents, fuzzy knowledge, evolutionary computing
Jel code: C63, C 88

1. Introduction

When discussing the Semantic Web (SW), it is important to get one thing clear from the start: this is not a new version of the Internet. The SW technologies will allow machines to make *inferences*. Pioneering this new approach to the Web is Tim Berners-Lee, the original worldwide web inventor. With health care and economic data, and databases of environmental information, all marked up in machine-readable codes, the SW could search for connections between where the sick people live, and any contextual environmental and economic information that might have contributed to the illness. Researchers are formulating projects that are likely to transform everyday use of the web, and potentially ways in which the internet can be used for better educational effectiveness.

The amount of information available via networks and databases has rapidly increased and continues to increase. Existing search and retrieval engines provide limited assistance to users in locating the relevant information that they need. Autonomous, intelligent agents may prove to be the needed item in transforming passive search and retrieval engines into active, personal assistants. Intelligent agents can improve the performance of short-term information retrieval in an existing search or retrieval engine [1,3,4].

This first part of paper is divided in two sections: section 2 is an introductory of SW properties, and section 3 summarises agent characteristics and their actual importance in digital economy.

2. On the Semantic Web

One of the projects that is developing involves the integration of public transport information using RDF. The project requires the integration of timetable information and route plans, and would also incorporate specific geographical information recorded at specific points in time to provide relevant travel information to the user as and when required, rather than leaving the traveller to consult a variety of timetable information from a variety of sources. Additionally, RDF describes objects and their relationships, rather than documents and the way they are displayed. This means it is easy to reuse information described in RDF for different devices such as mobile phones, and for presentation to people with different capabilities, such as those with cognitive or visual impairments. Using a similar framework, it is possible to extrapolate that in the near future schoolchildren will be able to extract far more data from a networked computer or wireless device, far more efficiently, to complete tasks. Based on a few specific search terms, library catalogues could be scanned automatically. Students could also be directed to relevant discussion lists and research groups, all in formats and on platforms they are most comfortable with or are most convenient.

Perhaps also relevant to the educational sector is Internet Relay Chat (IRC), a tool used by the Semantic Web development community to manage distributed working. IRC is a chat protocol where people can meet on channels and talk to each other. The semantic web community is enhancing this by writing robots that can help to log the chat when members are away, and a real-time chat-based tool that allows them to create and annotate links on a web page by typing in a chat room. Such tools have been used to support development in a community that is geographically and culturally widely distributed. IRC tools' usefulness comes both from their ability to enable many people to work together and distribute information about their work while separated in time and space, and also because they enable real-time support and discussion from the community. From an educational perspective, these qualities suggest that IRC and related tools could work well within education, for project discussion, remote working, and collaborative document creation. As demonstrated by the rising popularity of video-conferencing, schools are increasingly becoming interested in widening the boundaries within which students work. The incorporation of SW technologies could enable them to work across distributed locations in communities of learning and content creation within and outside of the classroom confines.

Whether or not the SW as a concept remains unclear, it is clear that a shake-up of the web is required to make it more meaningful, respond faster to questions, and join up disparate information objects and sources automatically.

The SW is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The first steps in weaving the SW into the structure of the existing Web are already under way. In the near future, these developments will usher in significant new functionality as machines become much better able to process and "understand" the data that they merely display at present. The essential property of the WWW is its universality. The power of a hypertext link is that anything can link to anything.

For the SW to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning. Adding logic to the Web—the means to use rules to make inferences, choose courses of action and answer questions—is the task before the SW community at the moment. A mixture of mathematical and engineering decisions complicate this task. The logic must be powerful enough to describe complex properties of objects but not so powerful that agents can be tricked by being asked to consider a paradox. Fortunately, a large majority of the information we want to express is along the lines of "a hex-head bolt is a type of machine bolt," which is readily written in existing languages with a little extra vocabulary [4].

Two important technologies for developing the SW are already in place: eXtensible Markup Language (XML) and the Resource Description Framework (RDF). XML lets everyone create their own tags—hidden labels such as or that annotate Web pages or sections of text on a page. Scripts, or programs, can make use of these tags in sophisticated ways, but the script writer has to know what the page writer uses each tag for. In short, 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. These triples can be written using XML tags. In RDF, a document makes assertions that particular things (people, Web pages or whatever) have properties (such as "is a sister of," "is the author of") with certain values (another person, another Web page). This structure turns out to be a natural way to describe the vast majority of the data processed by machines. Subject and object are each identified by a Universal Resource Identifier (URI), just as used in a link on a Web page. (URLs, Uniform Resource Locators, are the most common type of URI.) The verbs are also identified by URIs, which enables anyone to define a new concept, a new verb, just by defining a URI for it somewhere on the Web.

A program that wants to combine information across the two databases has to know that these two terms are being used to mean the same thing. This program must have a way to discover such common meanings for whatever databases it encounters. A solution to this problem is provided by *the third basic component* of the SW, collections of information called *ontologies*. An ontology is a theory about the nature of existence, of what types of things exist. Artificial intelligence and Web researchers have adopted the term and for them an ontology is a document or file that formally defines the relations among terms. The most typical kind of ontology for the Web has a taxonomy and a set of inference rules.

3. An overview on agent technology

The real power of the SW will be realized when people create many programs that collect Web content from diverse sources, process the information and exchange the results with other programs. The effectiveness of such software agents will increase exponentially as more machine-readable Web content and automated services (including other agents) become available. The SW promotes this synergy: even agents that were not expressly designed to work together can transfer data among themselves when the data come with semantics. An important facet of agents' functioning will be the exchange of "proofs" written in the SW's unifying language (the language that expresses logical inferences made using rules and information such as those specified by ontologies).

Intelligent Agent. Computers need rules and instructions. Computers are very good at following rules. If we can explain our rules and patterns to computers, then we can design systems which can follow those rules. It is much more difficult to teach them how to find a pattern, but, within limited boundaries, one can teach computers to identify patterns, extract rules, and implement them. That is what intelligent agents are — software programs that can identify repetitive patterns of behavior, similarities between events or things, and changes in patterns over time. However, these programs are agents as well. An agent in the legal sense is one empowered to act on the behalf of another. Thus, an intelligent agent is one which can learn the patterns of behavior, or the rules regarding certain actions and transactions, and then act appropriately on behalf of its owner.

Definitions and Technologies. There are many definitions of an intelligent agent, how many agent systems exist. They are computer programs with a *knowledge base* and set of rules. Therefore, although many experts define intelligent agents more widely, we will stick with our more demanding definition. Most current researchers do agree on the following facets:

1. *Autonomy* is the first and foremost common criterion for agents. Autonomous agents use their knowledge of their owner's needs and interests to undertake tasks that their owner does repeatedly. The concept of proactiveness is closely related to the concept of autonomy. It emphasizes that agents do not simply act in response to their environment. They exhibit goal-directed behavior by taking the initiative. Proactiveness is usually considered a key element of autonomy. An operational definition for autonomy would be: agents operate without the direct intervention of humans or others and have some kind of control over their actions and internal state.

2. *Adaptiveness* is the second common criterion for an intelligent agent. Agents should be able to learn as they react to or interact with their external environment, so that their performance improves over time. The external environment may include the physical world, users (humans), other agents, or the Internet. Adaptive agents are sometimes called learning agents for this reason. Many researchers and developers believe that systems should adapt to people, instead of the other way around. Since it would be impractical to assume that we could predict all possible events in the external environment and encode all the knowledge about those events in advance, agents need learning capabilities. How they react to new circumstances can be programmed. What they learn cannot. The qualities necessary for adaptiveness are [4,5]:

• **Reactivity:** agents perceive their environment and respond in a timely fashion to changes that occur in it.

• **Social ability:** agents interact with other agents (and possibly humans) via some kind of agent-communication language such as KQML, a high-level language that agents can use to conduct conversations and exchange meaningful messages. KQML (Knowledge Query and Manipulation Language) is the de facto standard agent communication language nowadays.

3. *Collaborative behavior* is the third commonly criterion for intelligent agents. It builds upon the concept of social ability mentioned above. Most of today's research concentrates on sets of agents or multi-agent systems (MAS). Each agent is given a discrete task. Sometimes they are parallel, such as finding the same information in different sources. They must work together to establish which agent will carry out each task, and how they will merge the information they collect for presentation to the user. Agents should be able to work in concert with other agents, possibly via an agent-communication language, to achieve a common goal. Agents may share knowledge and learning experiences in the problem solver process. This concept is important because a large portion of agent research is historically rooted in distributed artificial intelligence, that emphasizes task decomposition and distribution and collaboration among agents.

4. *Mobility.* This concept refers to the ability of agents to migrate in a self-directed way from one host to another on a network, such as the WWW, in order to perform their assigned duties. The duties may include gathering information at each host or balancing workload or traffic on the network, as will be presented in the second part of this article. Clearly it can be considered as an extension to the original concept of autonomy. Currently we are interested in building intelligent agents using machine learning techniques. Intelligent agents can either learn from the explicit training examples provided by the developers or from interactions with other agents, human or computer. They change or adapt their behaviour, based on the examples and the interactions.

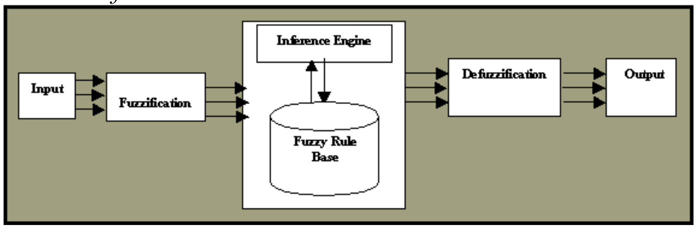
The agent functionality. Any of several technologies can design intelligent agents. All of them use some combination of statistical operations, artificial intelligence, machine learning, inference, neural networks, and information technologies. Agent systems are not plug and play. They need to be trained or taught. Most require examples of right answers or rules for appropriate behaviour. Typically, an agent system is implemented in several stages. First, one develops rules or training data. Once the agent system performs satisfactorily on the training data, it is ready to work on test data to make sure that it can extend what it has learned to unknown materials. A last step, but a continuing one, at least in theory, is to evaluate performance at several intervals. Agents should learn over time, and their performance should improve as they adapt to the user's needs, as well as to the kinds of information they navigate.

The Reasoning Technique. Of all the technologies used to build intelligent agents, the easiest to understand is rule-based reasoning, the basis for inference engines. Anyone who has ever set up an e-mail filter knows about setting up rules. These are usually some form of IF...THEN statements. Users can specify the rules or the agent systems can supply the rules, after training. Agents use the set of rules to decide which action or actions they should take. The problem with this approach is that the user needs to recognize the opportunity for employing an agent, take the initiative in programming the rules, endow the agent with explicit knowledge specified in an abstract language, and maintain the rules over time, as habits or events change. IBM's RAISE (Reusable Agent Intelligence Software Environment) is an example of rule-based reasoning. RAISE is the inference engine of IBM's Agent Building Environment (ABE) developer's toolkit. It can perform information flow functions: finding, searching, filtering, categorizing, storing, routing, and/or selectively disseminating information items. Prototype applications for RAISE include e-commerce shopping, customer service support, workflow on the Web and in Lotus Notes, news, and e-mail. One problem with rule-based systems is that users must keep them up to date manually. These systems cannot change by themselves. A second problem is that complex sets of rules may develop conflicting rules that the agent can't resolve. One can build knowledge bases based on a specific subject area or domain. These then serve as the basis for some inference mechanisms, including the rule-based reasoning techniques mentioned above. The major problem with such systems is that they require a large amount of work from the knowledge engineers. Furthermore, the knowledge of the agent is fixed and cannot be customized to the habits of individual users. In highly personalized applications the knowledge engineer cannot possibly anticipate the best aid for each user in each situation. The agents would have to modify their own behavior and extend their own knowledge, instead of relying on users to constantly modify, and possibly mess up, the rule bases and knowledge bases. Learning refers to this modification of behavior as a result of experience. That is the next step.

Statistical Analysis. The simplest learning technique that an intelligent agent can use is statistical analysis. It can determine the temporal or non-temporal correlation among events of interest. Charles River Analytics' Open Sesame and General Magic's Magic Cap are two such examples. The former periodically scans and analyzes the logs of user actions to find repeated sequences of actions. The latter recognizes frequently contacted people by their first names. EVA (evolving agent) technology uses statistical analysis to find terms that co-occur and should be added to a query.

Fuzzy Agents. When an agent needs to reason with imprecise or incomplete information, or the domain variables are expressible using linguistic variables, such as the words or adjectives we use to describe our world, fuzzy logic is a useful tool. Fuzzy logic is a form of logic used in some expert systems in which variables can have degrees of truthfulness or falsehood represented by a range of values between 1 (true) and 0 (false). Using fuzzy logic, we can design decision support or crisis management systems that offer a range of alternative actions to solve a problem [1,3,6]. A fuzzy system, based on fuzzy logic, is a collection of membership functions and rules that are used to reason about data. It resembles human decision making from uncertain and approximate information. It can be applied to systems whose information is inherently fuzzy to diagnose the problem, and find some fuzzy solutions. In fuzzy systems knowledge can be expressed using linguistic variables that are described by fuzzy sets. Fuzzy Systems usually consist of four components (as is presented in figure, a classic fuzzy inference system):

1. *Fuzzification Interface: maps crisp input values into Fuzzy sets (linguistic values).*
2. *Fuzzy Inference engine: receives inputs and evaluates all the rules to determine their truth-value. The two main steps in the inference process are aggregation and composition. Aggregation is the process of computing for the values of the IF (antecedent) part of the rules, while composition is the process of computing for the values of the THEN (consequent) part of the rules.*
3. *Fuzzy rule base: A collection of Fuzzy If-Then rules.*
4. *Defuzzification Interface: maps Fuzzy sets to a crisp output value. There are different methods of doing so, such as: Centroid, Bisector, Middle Of Max, and more.*



Neural Networks Neural networks consist of a set of interconnected nodes, like a web. Each node has a weight assigned to it. Like brains, neural nets need training by experience. Training sets of data, in the case of an information system, consist of two parts: the set of training data and the "right" answers extracted from that data. The neural net keeps trying connections until it gets the answer right. Neural nets are tricky.
They need training with large amounts of data in order to develop the right patterns. They can perform non-linear mappings between their input and output patterns. The most popular type of neural networks are three-layer, feed-forward neural networks, which consists of an input layer, an output layer, and a hidden layer. Each of these layers consists of one or more processing units (or neurons). Each unit in a given layer connects with all the units in the neighboring layers, but not with those in the same layer.
Each unit receives inputs from the units one layer below it and sends outputs to the units one layer above it. Each connection is associated with a weight, which, conceptually, represents the strength of the connection between this pair of units. Given a set of weights, the entire network can be thought of as a mapping from a set of input vectors to a set of output vectors. If we embed such a neural network in an information agent, the input vector could represent a set of query terms, while the output vector could indicate the "relevance" of the input vector to a certain information need.
Neural networks handle unstructured data or noisy data effectively. These are often difficult to process using rigid reasoning techniques. In agent systems, they can identify sequences of user actions, like statistical analysis, and train agents to automatically assign documents or Web pages to pre-defined categories.

Evolutionary Computing To expand the learning horizon and to create more intelligent agents, one needs a learning algorithm, such as a genetic algorithm, that can operate at a higher level and view things from an inter-agent perspective. By approaching the learning algorithm from two different levels — the local level of individual agents and the global level of inter-agent operation — we can ensure the optimization of each agent from local knowledge, while genetic algorithms will act as a driving force to evolve the agents collectively based on global knowledge. The goal is to construct a new generation of agents that benefit from the learning experiences of individual parent agents and the collective learning experiences of previous generations. The most popular examples of evolutionary computing are genetic algorithms. They work by maintaining a population of possible solutions (chromosomes, or agents in our case).

Successive evaluations of the performance of the agents determine which unfit set of agents to terminate, and which fittest set of agents to recombine to produce (or reproduce) possibly better agents.

References

1. Knapik M., Johnson J., Developing Intelligent Agents for Distributed Systems, McGraw Hill, 1998.
2. Wooldridge M., Jennings N.R., 1995. Proceedings of ECAI Workshop on Agent Theories, Architectures and Languages, pp. 1-32.
3. An introduction to ontologies: **www. Semantic Web.org/knowmarkup.html**
4. Simple HTML Ontology Extensions Frequently Asked Questions (SHOE FAQ): **www.cs.umd.edu/projects/plus/SHOE/faq.html**
5. DARPA Agent Markup Language (DAML) home page: www.daml.org/