 UNIVERSITY OF CALGARY	Course Number: SENG 609.22	Course Name: Agent-based Software Engineering
	Session: Fall, 2003	Department: Electrical and Computer Engineering
		Document Type: Tutorial Report

SENG 609.22 Agent Based Software Engineering

Course Instructor: Dr. Behrouz H. Far

Tutorial Report

Web Hunting Agent --- Agents and Web-services

Zhizhong Li

zhizli@ucalgary.ca

Web Hunting Agent --- Agents and Web-services

Zhizhong Li

Abstract

The world of services is evolving towards ‘web-services’, web concerned agents are becoming more and more popular. Agents, as well as many other technologies around the semantic web, have shown an increased maturity through standards and open-source. This tutorial presents these evolutions, positions agents, and introduces the open ecosystem currently in construction under the auspices of agencies. In the later part of tutorial we are talking about the Web Hunting Agent. The goal of this part is to introduce the reader to the basic elements of an intelligent agent, and then apply those elements to a Web search agent to provide the framework for the construction of a simple intelligent Web search agent. An overview of typical artificial intelligence search algorithms will be presented and performance metrics will be discussed.

1. Introduction

The world of services is evolving towards ‘web-services’, a simple concept where applications advertise their own capabilities, search for other applications on the web and invoke their services without prior design. These web-services can reason about their capabilities to combine services and negotiate. Although the ideal world of web services looks far-fetched, this article gathers some of the pieces of technology representing first steps towards this vision, in particular standards and open-source projects. However, due to the complexity of the web-services, there is a flurry of standards and software in competition and deployment becomes a key issue. We first describe an effort in deploying a freely accessible and open experimental environment for global web-services.

2. Evolution of E-services towards Web-services

We view web-services and their provision as containing the following key aspects:

2.1 Web-services

Web-services are flexible, Internet-based applications that allow companies to create new products and services faster than existing methods which consist of dynamic assembly of loosely coupled components (e-services, legacy data...). This is very different from the traditional hard-wired approach for developing applications. Fixed applications tend to resist change, whereas web-services assume that change is ever present. Web-services require research in: explicit representations of e-services and their capabilities; their re-use in different contexts to form new and dynamic services; the creation of a heterogeneous and competitive environment; reputation networks, negotiation, contracts ... Some benefits of web-services include faster time-to-market, convergence of

disparate e-business initiatives, significant reduction in total cost of ownership, easy to use software tailored for business people rather than IT staff.

2.2 Peer-to-peer computing

Peer-to-peer computing can be defined as “sharing of computer resources and services by direct exchange.” Current popular applications of peer-to-peer computing include distributed file sharing and distributed processing. However, these systems are just examples of what peer-to-peer computing can aspire to. As commercial peer-to-peer systems become developed and deployed, they will address applications such as dynamic integration and coordination of systems residing on arbitrary nodes of a network (ranging from wireless devices to server-class computers) in areas such as enterprise application integration, e-commerce, and network management. Agent-based peer-to-peer applications require research in: a new deployment and execution paradigm of pervasive computing; delegation of tasks and pro-active behavior; high level communication incorporating flexible interaction protocols, ontology and semantic models for communication; a virtual and pro-active representation of the user.

2.3 Mobile and Personal

The trend for personal information tools is towards small devices carried in the pocket or worn by users and connected nomadically through a wireless link. User interfaces predicated on browsing capabilities are no longer sufficient. Direct manipulation interfaces, as promoted by HTML and HTTP in their current forms, are not powerful enough to answer advanced user queries, nor do they lend themselves easily to helping build new Internet -based business models. Research is looking into a higher level of personalization and delegation of tasks, in aiming to provide a restricted but pertinent choice of services to the end-user. Awareness of the user’s context becomes one of the key elements to the success of these new interfaces.

These three drivers are bringing a new set of requirements for the engineering of the web-services and place a premium on the ability of software components to exhibit a certain degree of autonomous goal-seeking behavior in dealing with the task they have been engineered to perform. In next section we describe the semantic stack allowing to reason over web-services, the section after focuses on agents, their

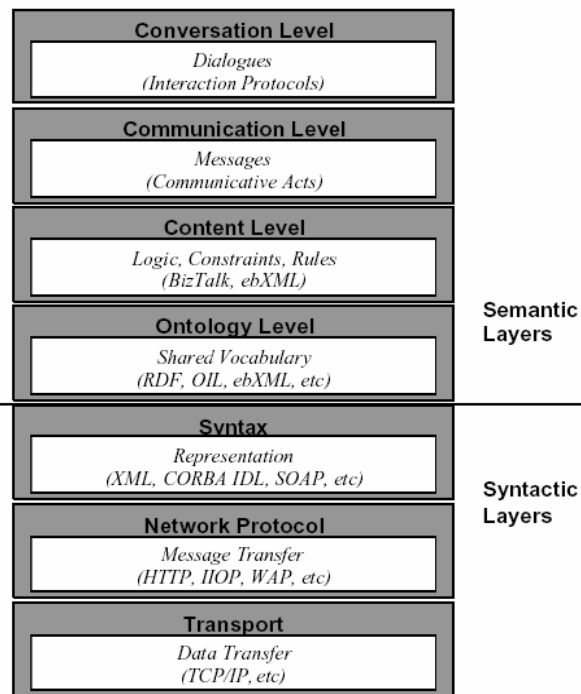


Figure 4: Communication technology stack

autonomy and proactive behavior leading towards an active web.

3. The Semantic Layers of Web-services Communications

Web-services require more infrastructure to realize all of their potential benefits than their existing static counterparts. There are mainly semantic layers below (see Figure 4).

Dialogs and Interaction protocols define message interactions between agents at the conversation level, that is, when two or more agents agree to exchange messages using a specific interaction protocol. The interaction protocol shows the messages that each agent can send and receive at each stage of the conversation. *Communicative acts* define standard, application-independent methods for passing semantic messages between agents. A communicative act is a verb-utterance providing context for the contents of a message, for example, request, inform, etc. *Content languages* are used to express the actual content of a message. *Ontology* is a vocabulary of terms and their definitions and relations that are applicable in the current problem domain.

4. An implementation of Web-services agent: Web Hunting Agent

The World Wide Web has become a vast resource of information. The problem is that finding the information that an individual desires is often quite difficult, because of the complexity in organization and the quantity of information stored. Information is stored in uniquely named files located within uniquely named directories on a Website identified by a series of four numbers or a corresponding symbolic name.

4.1 Search Algorithms

In the study of artificial intelligence, problems can often be reduced to a search. An agent can usually generate all of the possible outcomes of an event, but then it will need to search through those outcomes to find the desired goal and execute the path (sequence of steps) starting at the initial, or current state, to get to the desired goal state.

There are two basic classes of search algorithms: uninformed and informed. Uninformed, or blind, searches are those that have "no information about the number of steps or the path cost from the current state to the goal." [8, p.73] These searches include: depth-first, breadth-first, uniform-cost, depth-limiting, and iterative deepening search. Informed, or heuristic, searches are those that have information about the goal; this information is usually either an estimated path cost to it or estimated number of steps away from it. This information is known as the heuristic. It allows informed searches to perform better than the blind searches and makes them behave in an almost "rational" manner. These searches include: best-first, hill-climbing, beam, A*, and IDA* (iterative deepening A*) searches [8].

There are generally four searching strategies: breadth-first, uniform-cost, best-first, and A*. Here we only discuss about the breadth-first searching method.

Breadth-first search is one of the two most common search algorithms every computer science student learns (the other being depth-first search). The approach of breadth-first search is to start with a queue containing a list of nodes to visit. A node is a state or a value; in programming it is usually represented as a structure containing particular information about the environment or domain of the problem. The algorithm starts by placing the initial state of the problem into the head (beginning) of the queue. The search then proceeds by visiting the first node and adding all nodes connected to that node to the queue. When viewed as a tree graph, it would move from left to right from the current node (usually represented as a circle on a graph) along links (represented as connecting lines in between the node circles) to connected nodes, adding the connected nodes to the queue. As the head node of the queue is visited it is removed. The search then moves to the next node on the queue and continues until the goal is reached. This search will always find a solution if it exists [8]. Figure 1 provides a graphical example.

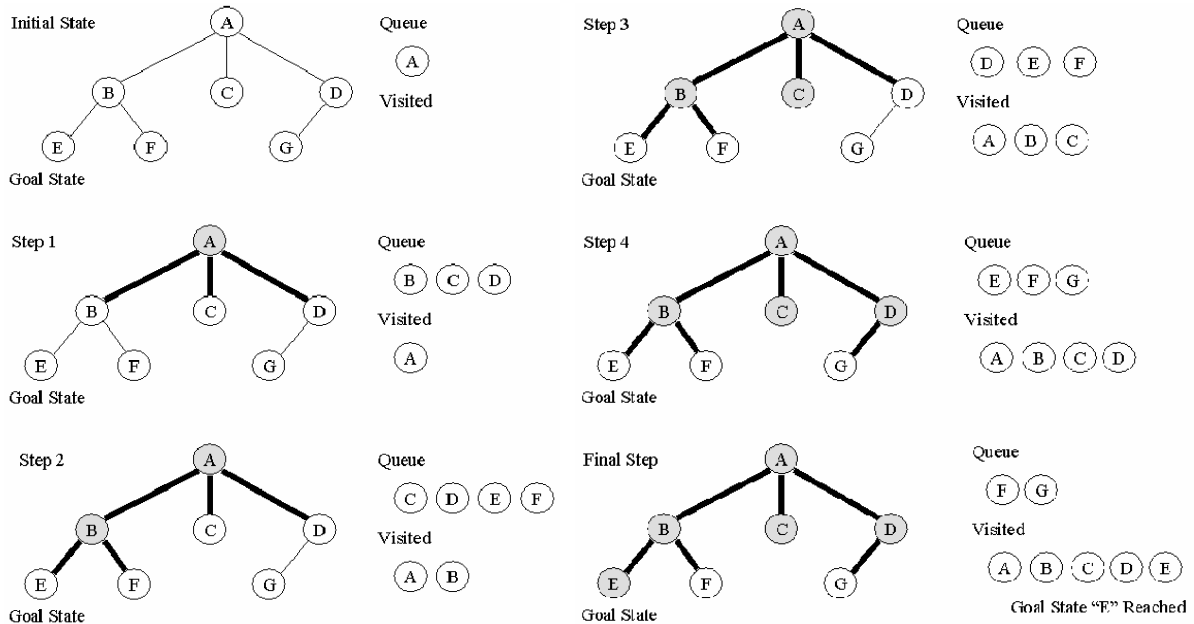


Figure 5: Breadth-first Search Example

4.2 Building a Web Hunter

The concept of what is needed to build an intelligent Web search agent should be clarified. From this point on the agent will be referred to as the Web Hunter. Given a target, the Web Hunter should proceed to look for it taking as many paths as are necessary. This agent will be keyword based. The method advocated is to start from a "seed" location (user provided) and find all other locations linked in a tree fashion to the

root (seed location) that contain the target. Blind URL searches based on random IP (Internet Protocol; i.e., 129.107.2.249) numbers would be a long and tedious search, but could be an exhaustive method for a complete Web hunt.

Implementation will require some knowledge of general programming, working with sockets, the Hypertext Transfer Protocol (HTTP), Hypertext Markup Language (HTML), sorting, and searches. There exist some agent shells for writing agents as well. The use of a sorting routine will be necessary for many of the search techniques. Using a more advanced, efficient sorting algorithm (e.g., shell or quick sort) will help improve the performance of the Web Hunter.

Our example Web Hunter design consists of four main phases: *initialization*, *perception*, *action*, and *effect*. In the *initialization* phase the Web Hunter should set up all variables, structures, and arrays. It should also get the base information it will need to conduct the "hunt" -- the target, the goal (the number of Websites containing the target, found in a timely manner), a place to start (a good hunter starts at a good entry location into the forest), and the method of searching, if more than one is available. The *perception* phase is centered on using the knowledge provided to contact a site and retrieve the information from that location. It should identify if the target is present and should identify paths to other URL locations (i.e., hyperlinks within the page). The *action* phase takes all of the information that the system knows and determines if the goal has been met (the target has been found and the hunt is over). If the hunt is still active it must make the decision on where to go next. This is the intelligence of the agent, and the method of search dictates how "smart" the Web Hunter will be. The *effect* phase is to list the location(s) of the target and to provide status and feedback to the user. Implemented together and connected to the Internet the phases form the Web Hunter.

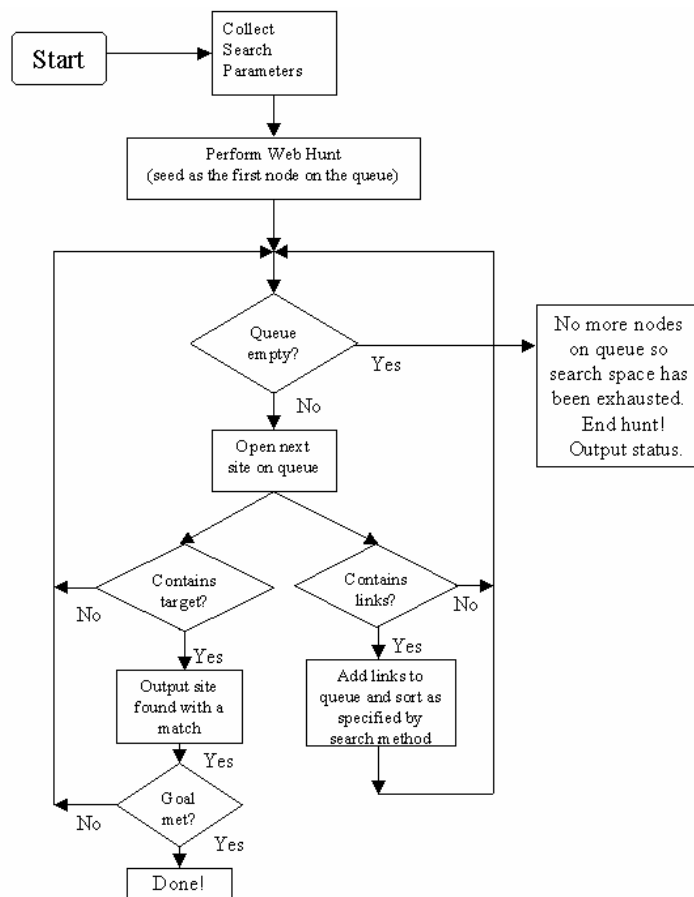


Figure 6: Web Hunter Basic Flow

4.3 Performance Metrics

An implemented Web Hunter will not perform like a commercial search engine. Search engines use database lookups from a stored database created by similar Web search agents (i.e., industrial spiders). Despite being a low-level utility, good performance will make the Web Hunter a more useful tool. Testing the speed of searches on a given target is a good metric. Wrap the perception-action-effect loop in a timing function, and test the speed of searches by utilizing different search techniques and sorting routines. Goal achievement is also important. Some seed values are better than others. Utilizing more seed Websites to execute sequentially after the previous has been exhausted may improve goal attainment. The program could even be modified to do seed lookups itself, utilizing search engines or random/sequential IP number generation. Use goal achievement as the metric for search ability. The use of time and goal achievement are examples of some simple metrics for performance; others, such as the validity of the hits may prove useful as well.

4.4 Web Hunter Applications

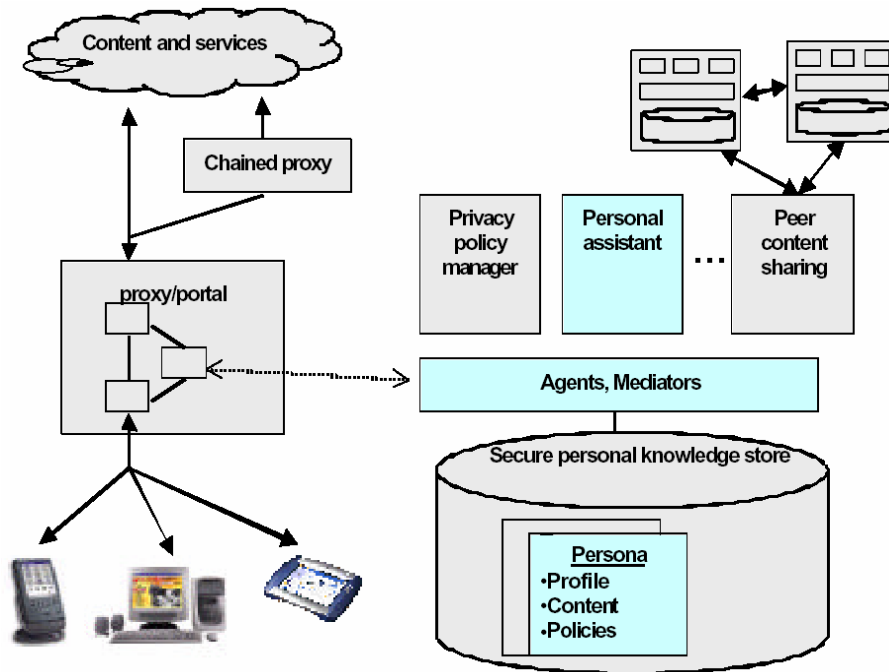
After creation of a Web Hunter, the agent could be used in many applications. An entire search engine system could be created using a dictionary word generator and a sequential IP number generator to perform an exhaustive search on a target. The results could be stored in a database for later retrieval. Provided with an HTML interface and its own Website, it could be another Web search engine. The Web Hunter could also be modified to be a Document Hunter or Record Hunter to go through files, instead of Websites, looking for targets. It could be used to hunt through any data in any environment it can access. The Web Hunter and its core, search, could be a useful tool for many applications.

4.5 Future Hunters

There are numerous possibilities for improvements in current Web agent technology. The complex nature of the dynamic World Wide Web makes productive planning and searching difficult tasks. Further research in advanced search methods and planning strategies could be employed in Web robots. There are also gaps in the human-computer interface. There is some disparity between the way people query for information and the way computers store information. Future Web Hunters may benefit from natural language processing research. If people could interact in a way that was more natural to them and the agent could understand their request better, then a better search could be performed. Searching can move from a syntactic basis to a conceptual or semantic basis allowing the agent to search for ideas and not just words. If the agent understands concepts then machine learning could be utilized to understand user behavior and could refine searches in a custom manner, tailored for the specific user based on their profile. Improvements in planning and search as well as integration of natural language processing and machine learning could improve the intelligence of Web agents.

5. HP's Blueprint for Future Active Web-services

The key to success of future systems is the ability of hosting an ecosystem of technologies that is capable of evolving with time and of welcoming new bricks of technology into an integrated world, of being deployed in decentralized networks like the internet, or centralized ones like the telecommunications, or in spontaneously created ad-hoc networks. In each case, the management and configuration of these services will change, but the global functional structure remains the same. Figure 5 gives a functional view of future web-services.



The user accesses web-services through any existing appliance: PDA, phone, laptop, voice mail, or pager and communicates either with proxies and portals (current static manner), or through agents or mediators as part of a middleware transforming the web-services into an active environment.

6. Conclusion

Different bricks of future web-services are reaching maturity mainly through standard efforts in various domains and open-source implementations. Web-services now need some landscaping to allow each of these initiatives to reassess its position in this new ecosystem.

The use of software agents and search are becoming important tools in the information age and especially in information environments such as the World Wide Web. Creating a Web search agent may seem like a daunting task, but most of the difficulty in beginning a project is not knowing where to start. RFCs, FAQs, standards, and other resources that are available are important tools that all programmers should be familiar with. The

creation of an intelligent Web search agent is not a difficult task and the framework provided is a good basic basis for creating a Web Hunter. But before the hunt begins, programmers should ensure they adhere to netiquette and hunt safely.

Reference:

- [1] <http://www.agentlink.org/roadmap/contents.html>
- [2] A.M. Uhrmacher, P.A. Fishwick, and B.P. Zeigler, editors. *Special Issue: Agents and Simulation: Exploiting the Metaphor*, volume 89 of *Proceedings of the IEEE*, 2001.
- [3] S.D. Anderson. Simulation of Multiple Time-Pressured Agents. In *Proc. of the Wintersimulation Conference, WSC'97*, Atlanta, 1997.
- [4] H. Praehofer, J. Sametingger, and A. Stritzinger. Discrete Event Simulation Using the JavaBeans Component Model. In A.G. Bruzzone, A. Uhrmacher, and E.H. Page, editors, *1999 International Conference on Web-Based Modeling and Simulation*, volume 31, pages 107--112, 1999.
- [5] Andrew S. Tanenbaum. *Distributed Operating Systems*. Prentice Hall International Editions, 1995.
- [6] N. R. Jennings, K. Sycara, and M. Wooldridge. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 1(1):275--306, 1998.
- [7] <http://www.aaai.org/AITopics/html/agents.html>
- [8] Ingargiola, Giorgio P. "Search."
<http://yoda.cis.temple.edu:8080/UGAIWWW/lectures97/search>. 1 Dec. 1998.
- [9] www.agentbuilder.com/AgentTools
- [10] S. Hanks, M.E. Pollack, and P.R. Cohen. Benchmarks, Test Beds, *Controlled Experimentation and the Design of Agent Architectures*. AAAI, (Winter):17--42, 1993.
- [11] Y. Gil, M Hoffman, and A. Tate. Domain Specific Criteria to Direct and Evaluate Planning Systems. Technical Report ISI-93-365, Information Sciences Institute, University of Southern California, Marina del Rey, CA, 1994.
- [12] http://www.enel.ucalgary.ca/People/far/Lectures/SENG609-22/PDF/tutorials/2001/Agent_Based_Simulation.pdf
- [13] A.M. Uhrmacher. Dynamic Structures in Modeling and Simulation - a Reflective Approach. *ACM Transactions on Modeling and Simulation*, to appear.
- [14] <http://www.ecs.soton.ac.uk/~nrj/download-files/roadmap.pdf>
- [15] World Wide Web Consortium Website (www.w3.org),
- [16] N.R. Jennings and M. Wooldridge. Applications of Intelligent Agents. In N.R. Jennings and M. Wooldridge, editors, *Agent Technology : Foundations, Applications, and Markets*. Springer, 1998.
- [17] http://www.thesimguy.com/GC/papers/WMC02/G067_UHRMACHER.pdf

- [18] E.H. Page. *Simulation Modeling Methodology: Principles and Etiology of Decision Support*. PhD thesis, Virginia Polytechnic Institute and State University, 1994.
- [19] B. Logan and G. Theodoropolous. The Distributed Simulation of Multi-Agent Systems. *Proceedings of theIEEE*, 89(2):174--185, 2001.
- [20] Agentcities web site <http://www.agentcities.org/>
- [21] T. Montgomery and E. Durfee. Using MICE to Study Intelligent Dynamic Coordination. In *Conference on Tools for Artificial Intelligence*, pages 438--444, Washington, DC, 1990. IEEE.
- [22] Berners-Lee, T., R. Fielding, and H. Frystyk. ``Hypertext Transfer Protocol -- HTTP/1.0." *RFC:1945*. <http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1945.txt>. May 1996.
- [23] Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. McGraw-Hill, New York, N.Y., 1997.
- [24] Crossley, John. ``IP Network Index." <http://ipindex.dragonstar.net>. 22 Dec. 1997.
- [25] *Agentcities: a Worldwide Open Agent Network*. Steven Willmott, Jonathan Dale, Bernard Burg, Patricia Charlton, Paul O'Brien. AgentLink review, in review.
- [26] *System for document telenegotiation (negotiator agents)*.Rodrigez J.M and J. Sallantin. COOP'98: 3rd International Conference on the Design of Cooperative Systems, Cannes, France, May 26-29, 1998, pp. 61-66.