Agency Systems

(Standardization & Architecture)

Dr. Abbas M. Al-Bakry
Babylon University-College of Science
Dep. of Computer Science
E. Mail: abbasmoh67@yahoo.com

Abstract

In this paper, we describe a principled methodology of agent-based systems architecture and standardization. We illustrate the use of this methodology on an example of a supply chain management system. Using the methodology, it is possible to assess architectural decisions on software qualities such as performance, modifiability, and security. The focus for the methodology is risk identification, evaluation, and mitigation in the early stages of system design. We describe a test bed to analyze and evaluate agent-based systems in manufacturing enterprises that is based in part on this methodology.

This paper provides an overview of the standardization effort in the area of specifying standards for heterogeneous, interpreting agent-based systems. We describe the types of software agents, which are of interest.

Keywords

Intelligent agent, Agency, Protocols, Multi-Agent, Standardization, Agent Architecture.

1 Introduction

The amount of information available on the World Wide Web is increasing on a daily basis. General-purpose search engines and browsers provide valuable assistance to users in locating general information relevant to their needs. However, finding information for a narrow query in a specific domain has become more and more difficult with the growth of the Web, and thus frequently resembles a search for a needle in a haystack. Individuals spend more and more time filtering out irrelevant information returned from general purpose search engines while searching the Web. It is not uncommon for a user to obtain thousands of hits, which match her query but belong to irrelevant domains.

This is termed the low precision problem, as defined in information retrieval [Sal83].

Many approaches have been suggested to tackle the low precision problem on the Web [Ben99], [Goo03], [Ben99], [Cha99]. One such approach is to restrict the search into a pre-defined domain. Several searches service, most notably Yahoo! [Mcc79], allow users to specify the domain in which to evaluate the query. Such a restriction narrows the search space,

hence increases the precision. However, the domain hierarchy is manually (or at best semi manually) consuming, and coverage is extremely limited. Major search engines, such as AltaVista, index about two orders of magnitude more Web pages than Yahoo! [Sea03].

Intelligent agents are a powerful Artificial Intelligence technology, which shows considerable promise as a new paradigm for mainstream software development. Agents offer new ways of abstraction, decomposition, and organization that fit well with our natural view of the world and agent oriented programming is often considered a natural successor to object oriented programming [Exc01]. It has the potential to change the way we design, visualize, and build software in that agents can naturally model "actors" – real world entities that can show autonomy and proactiveness. Additionally, social agents naturally model (human) organizations ranging from business structure & processes to military command structures. A number of significant applications utilizing agent technology [Cha99]have already been developed, many of which are decidedly non-trivial. An intelligent agent is one, which is able to make rational decisions, i.e., blending proactiveness and reactiveness, showing rational commitment to decisions made, and exhibiting flexibility in the face of an uncertain and changing environment.

Despite their promise, intelligent agents are still scarce in the market place 1. There is a real technical reason for this: developing intelligent agent software currently requires significant training and skill.

2 Software Agents

2.1 Characteristics of Agency

There is no single definition of what constitutes a software agent; instead, researchers define such a computational entity through the ascription of certain characteristics to the notion of agency. That is, what properties is it desirable for agents to possess? Agent researchers, for example, [Woo95], have defined a loose taxonomy of characteristics that agents might have, broadly split around the question of intelligence.

Strong agents are those agents, which typically have an explicit semantic representation of their goals, and the environment in which they interact. They are intelligent in the sense that their rational behavior can be expressed and reasoned about, as can their interactions with other strong agents. These kinds of agents try to embody the ongoing work of Artificial Intelligence (AI) that began as early as the 1950s.

Weak agents are not so demanding in their requirement for AI-style reasoning and semantic representations and processing.

They typically possess four key characteristics, each to a greater or lesser degree depending on the task that is being undertaken:

- ***Autonomy*** is the capability of an agent to act independently to achieve its goals.
- ***Reactivity*** is the capability of an agent to perceive its environment and react to it.
- ***Social ability*** is the capability of an agent to communicate with other agents.
- ***Proactivity*** is the capability of an agent to take the initiative to achieve its goals.

The classifications of strong and weak agency are not intended to be binding, but only represent a mechanism for typifying common features that may exist across agent-based systems. However, the attribution of these characteristics does serve to differentiate the agent paradigm from other paradigms, such as object-orientation.

2.2 Standardization for Agents

As a matter of standardization for software agents, FIPA (finding Intelligent Physical Agent) is most concerned with the characteristic of communication (social ability) between agents. If two agent-based systems that are developed by different organizations are founded upon the same model of communication (in terms of syntax and semantics), then they can ***interoperate*** [1]. This is what makes standardization a matter of importance since communicating agents need to have an agreed agent communication language (ACL) to use for enabling interoperability. Indeed, communication plays such a key role within the standardization effort of software agents that autonomy, intelligence and the other characteristics of agency can often become of secondary concern. Yet, intelligent communication does play a part in an agent communication language.

One of the dominant methods of implementing intelligence within agents is known as the ***beliefs, desires*** and ***intentions*** architecture [Mcc79]. This views an agent as having an explicit set of beliefs about itself and the rest of the world, a set of desires for goals to be achieved and a set of intentions for carrying out actions to achieve those goals. Given that the act of communication is an action (known within the agent community as a ***communicative act***), then the semantics of such an action can be given in terms of:

- what an agent believes to be true,
- what an agent believes about the state and knowledge of another agent, and,
- what an agent desires to make another agent believe regarding a fact of mutual importance.

That is, the meaning of communication between agents can be explained in mentalistic terms that are expressed in predicate logic, such as what an agent believes to be true and what it desires to achieve.

3 Distribution Techniques of Control Applications

3.1 Protocols from packets to objects and agents

A basic condition for distributed computing to develop is the possibility of efficiently exchanging data. The packet switching technique, implemented with the now ubiquitous Ethernet, was a first step. However, for efficient and perennial programming to take place, higher levels of software protocols were necessary to reach the so called application layer from which operations usual in one's computer could as well be attainable on a remote one: managing files, executing a programs. The TCP/IP protocols at once brought network and distance independent programming opportunities.

In the following sections we will give a schematic overview of the not always straightforward path towards higher and higher abstraction and standardization levels in distributed programming.

3.1.1 The ISO/OSI stack versus Internet

The 7 layers protocol stack of the ISO/OSI (International Standard Organization/ Opened System Interconnection) model for communication is a standard in computer education. It was to supersede the less structured UDP/TCP-IP (User Datagram Protocol/ Transport Control Protocol- Internet Protocol) 4 layer protocols. However the TCP/IP stack and utilities was included for free, as soon as the early 80's, in all Unix-like platforms. For that reason, ISO protocols where only accepted in large organizations, or for long distance applications. By the early 90's TCP stacks and utilities were available for DOS based PC's, letting them communicate with Unix systems. At that time we could already propose an ***Open Machine-tool Controller*** linked in a transparent manner to CAD/CAM (Computer Aided Design / Computer Aided Manufacturing) sources [Rad93].

3.1.2 Distributed application programming in the IP world

The Internet protocol (IP) routes packets of data up to 64 Kbytes large, and supports essentially two transport level protocols:

- TCP is a reliable stream mode connection based protocol (partners negotiate to open a communication session on a particular ***port number***), near to the OSI TP4 protocol, well applicable for applications such as file transfer (FTP) or remote connection (Telnet). No limit is set to size of transmitted data. Fragmentation and packet numbering take place for sizes larger than 64 Kb;

- UDP is a connectionless protocol with no flow control or error recovery, applicable for local and small volume interactions such as data value reports, it is used in the trivial file transfer protocol (TFTP).

4 Multi-agents in the manufacturing domain

The multi-agent system is a concept originated in the Distributed Artificial Intelligence area, and can be defined as a set of nodes, designated by agents. In spite of several existing definitions, it is possible to define an agent as a component of software and/or hardware, which is capable of acting and decision making in order to accomplish tasks. In the manufacturing systems domain, an agent is a software entity, that represents manufacturing system entities, such as physical devices and tasks.

4.1 Motivation to use multi-agents

The multi-agent systems represent a suitable technology to support the distributed manufacturing environment, since the manufacturing applications present characteristics like being modular, decentralized, changeable, ill-structured and complex, for what the agents are best suited [Par98]. Analyzing the benefits of multi-agent technology it is possible to conclude that it fulfils some of main requirements of the actual distributed manufacturing systems: autonomy (an agent can operate without the direct intervention of external entities, and has some kind of control over their behavior), co-operation (agents interact with other agents in order to achieve a common goal), reactivity and pro- activity (agents perceive their environment and respond adaptatively to changes that occur on it). Last, agents can be organized in a decentralized structure, and easily reorganized into different organizational structures [Lei01].

4.2 Agentification

One important point when agentifying manufacturing components is the process of connecting the physical controller to the agent. This could be an easy task if every physical component was controlled directly by its own agent. However, outdated legacy controllers with close architectures control most of existing physical components. To integrate these legacy components in the agents' framework it is necessary to develop a software wrapper to hide the details of each component. The wrapper acts as an abstract machine to the agent supplying primitives that represent the functionality of the physical component and its local controller. The agents access the wrapper using local software interface (proxy), where all the services of the wrapper are defined. Figure 1 shows a high level representation for an operative agent indicating how the wrapper integrates a manufacturing component (robot).

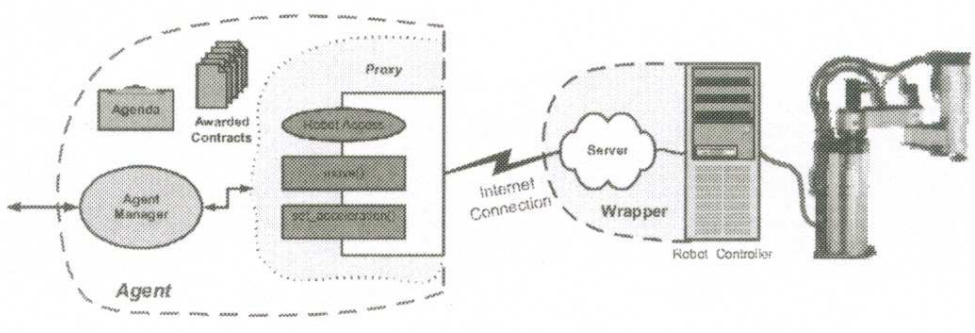

Fig. 1 Physical component integration

Wrappers for the NovaFlex's subsystems were developed using DCOM, not because of particular superiority but because (1) all the computers available to control the NovaFlex are running Microsoft operating systems (Windows95, 98, and NT), (2) C++ Builder, the used development language, has good tools to develop DCOM applications, and (3) developers were better trained on the Microsoft environment. In effect any of the distributed object architectures could be used because the intention was to prove that distributed objects are adequate to integrate physical controllers with agents (agentification) and are even better than the old two-tier architecture.

5 Knowledge-Based Agent Architecture

Knowledge-Based system is an important agency systems used a knowledge agent for acquiring and/or extracting a knowledge from its resources. The knowledge agent architecture contains the following components (Figure 2):

1. An agent manager, that sets up new knowledge agents and restarts existing agents. It is responsible for managing multiple concurrent connections for reading from the Web and serves multiple agents.
2. One or more Knowledge Agents who perform domain-specific Web search.
3. Each knowledge agent has an associated knowledge base, which contains domain-specific information to be used for searching. The KB is updated continuously throughout the use of the agent.

5.1 The Knowledge Base

The knowledge base contains a bounded collection of ranked sites and an aggregate profile of the textual content of these sites. Sites in the knowledge base are those which have proven to be highly relevant to a majority of the queries submitted to the agent.

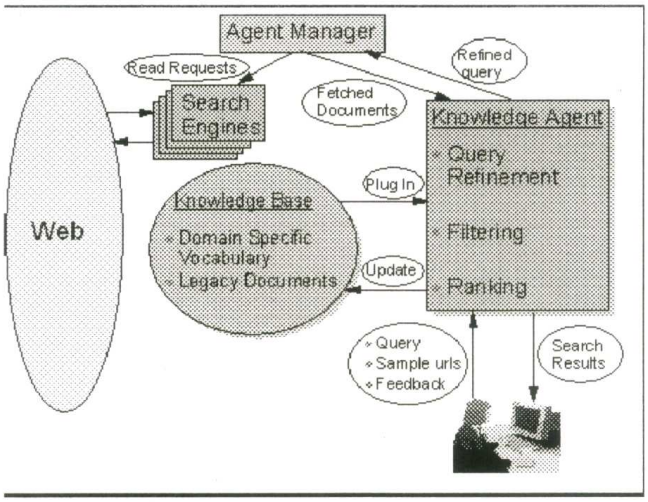

Fig. 2 The knowledge agent architecture.

The rationale for this is that sites which are consistently relevant to the users' queries are deemed as central to the domain. The textual profile contains all of the words, which appear in the sites, after deletion of stop words and a mild stemming process, along with their number of appearances. Each word in the profile is associated with a list of its lexical affinities. The flow of sites into and out of the KB is regulated using an evolutionary adaptation mechanism. Sites fight for the right to be included in the agent's KB. Each site is assigned a history score reflecting its relevance to the domain through the life of the agent. A combination of the history score and the relevance score for a specific query determines which sites are inserted and removed from the KB.
The KB is a pluggable component. It can be saved to a file and restored from one, and thus knowledge can easily be transferred from one user to another.

5.2 The Search Process

The search process (Figure 3) starts with the user entering a query and ends with the agent returning a ranked set of (hopefully highly relevant) sites to the user. The system supports two kinds of queries, text queries and sample-url queries. A text query is a keyword-based query such as those typically submitted to general purpose Web search engines. The user's query is automatically refined in the context of the agent's domain by adding to each of the keywords in the query its most notable lexical affinities as found in the profile of the KB. The refined query is submitted to the user's choice of one or more search engines. The results returned by the search engine(s) to the refined query are called the root set of sites.
A sample-url Query is a query which specifies a few (typically 1-5) seed urls, and whose purpose is to find a community of sites which are closely related to the seeds.

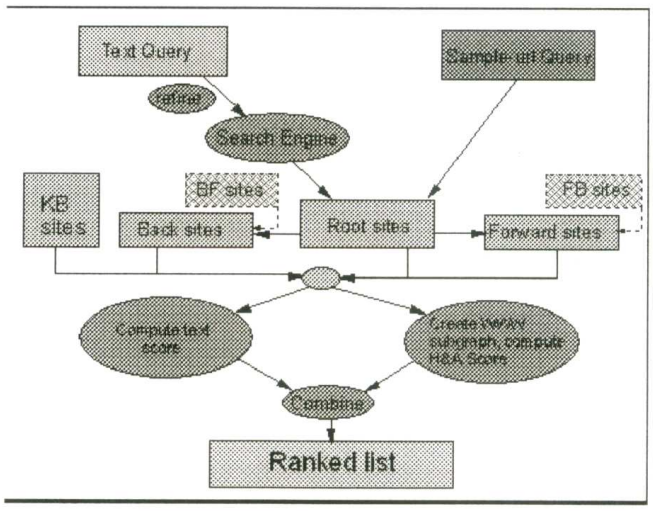

Fig. 3 The search process applied by the agent.

Similar services are offered by Excite's "More like this" [Exc01] and by Google's "Google-Scout" [Goo03]. Both services receive as input a single site, while we allow an arbitrary number of seeds. In sample-url queries, the user-supplied seed sites assume the role of the root set of sites, as if they were returned by a search engine in response to some textual query. The seed sites are read, and their combined content serves as a pseudo query for the purpose of evaluating the textual content of other sites in the search process. The collection of root sites is expanded by following the hyperlinks surrounding them, and by adding the KB sites. The breadth of the expansion is controlled by a user-defined link expansion factor that specifies how many pointed/pointing sites will be added in each expansion stage.

References

[Cha00] Charlton, P, Cattoni, R, Potrich, A and Mamdani, E, Evaluating the FIPA Standards and its Role in Achieving Cooperation in Multi-Agent Systems. In: Proceedings of HICS-33 on Software, 2000.

[Exc0] Excite Inc. Excite search. http://www.excite.com/, 2001.

[Goo03] Google Inc. Google search engine. http://www.google.com/, 2003.

Sal83] G. Salton and M. J. McGill. Introduction to Modern Information Retrieval . Computer Se-ries, McGraw-Hill, New York, 1983.

[Ben99] I. Ben-Shaul, M. Herscovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, V. Soroka, and S. Ur. Adding support for dynamic and focused search with fetuccino. In Proceedings of the Eighth International WWW Conference, pages 575–587. Elsevier, 1999.

[Ibm00] IBM Almaden Research Center Clever. http://www.almaden.ibm.com/cs/k53.clever.html,2000 .

[Lei01] Leito, P. & Restivo, F., An Agile and Cooperative Architecture for Distributed Manufacturing Systems, to appear: Proceedings of Robotics and Manufacturing'2001 International Conference, Cancun, Mexico, 21-24 May, (2001).

[Mcc79] McCarthy, J, Ascribing Mental Qualitiesto Machines. In: Philosphical Perspectives in Artificial Intelligence, Ringle, J (Ed), Harvester Press, 1979.

[Par98] Parunak, H. Van Dyke, What can Agents do in Industry, and Why? An Overview of Industrially Oriented R&D at CEC, Industrial Technology Institute; CIA'98, (1998).

[Rad93] Raddadi, M., Razafindramary, D., Boissier, R. & Ponsonnet, R., Spécification temps-réel et réalisation d'un directeur de commande numérique ouvert, évolutif pour machine-outil. Revue RAPA, **6** (3), Hermès Paris, (1993).

[Cha99] S. Chakrabarti, B. Dom, and M. ven den Berg. Focused crawling: A new approach to topic-specific web resource discovery. In Proceedings of the Eighth International WWW Confer-ence, pages 545–562. Elsevier, 1999.

[Sea03] Search Engine Watch. Search engine watch. http://www.searchenginewatch.com, 2003.

[Yah01] Yahoo Inc. Yahoo! http://www.yahoo.com, 2001.

[Woo95] Wooldridge, M and Jennings, N R, Intelligent Agents: Theories and Practice. In: Knowledge Engineering and Review, 10(2), 1995.