INTRODUCTION TO SOFTWARE AGENTS

Lance Self
Air Force Research Laboratory
Space Vehicles Directorate
Albuquerque, New Mexico 87117
lance.self@vs.afrl.af.mil

ABSTRACT

Software agents are application programs that perform duties in an autonomous fashion. These applications can be used to increase productivity, better use existing bandwidth, and improve present and future software application programs. By using existing established methods, and adding a layer of intelligence, software agents can add problem solving abilities to an application program.

KEY WORDS

Software agent, autonomy, collaborative efforts, blackboard model

INTRODUCTION

Software agents have been described as everything from a panacea for building distributed application software to an unworthy step-child being falsely touted by the artificial intelligence community. The truth of agents, and their usefulness, lies somewhere between these two extremes. Agents are not the “magic bullet” capable of human like reasoning and actions. Difficult software applications will remain so to the software engineer who builds these applications.

Software agents do, however, move the building of certain software applications forward just as the object-oriented paradigm did. In fact, agents can be considered as an extension to, or as a natural progression of, object-oriented programming. Objects offer modularity, encapsulation, and polymorphism as building blocks for modeling the system entities. Software agents can build on these inert activities by using established algorithms to add a layer of problem solving abilities to the application.

AGENT DESCRIPTION

Before describing what software agents are it would be useful to define what a software agent is. Unfortunately, there are as many definitions of an agent as there are types of agents. Shoham defined agents as:

> "a software entity which functions continuously and autonomously in a particular environment, often inhabited by other agents and processes" [1]

Franklin and Graesser define an autonomous agent as:

> "a system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future" [2]

A general definition of software agents offered by Don Gilbert from IBM:

> "An intelligent agent is software that assists people and acts on their behalf" [3]

All these definitions define agents in terms of characteristics and attributes which are necessary to distinguish agent-oriented from other programming models such as object-oriented and procedure-oriented. The established models are inherently static and change their state only when responding to function calls or some other means of stimulation. They do not, by design, have control over when they will be invoked or by whom.

Practitioners are less concerned with definitions and more concerned with what an agent does and how an agent can be used. They see a potential and are trying to exploit this because they are faced with increasing demands on the software they are tasked to build. Practitioners expect a software agent to carry out some task without having to describe in detail *how* to accomplish that task. Agents can do this if they carry with them some knowledge of the context of the task to be accomplished. This is referred to as being goal oriented. Agents have knowledge of a goal and work towards accomplishing that stated goal. Alan Kay described an agent system that:

> "… when given a goal, could carry out the details of the appropriate computer operations and could ask for and receive advice, offered in human terms, when it was stuck. An agent would be a 'soft robot' living and doing its business within the computer's world." [4]

Practitioners also expect agents to act autonomously and have some built in measure of control over their actions and reactions to the environment in which they operate. Autonomy implies independence and knowledge of the environment. The agent knows where it is located within the system and knows what are the appropriate actions. Autonomy does not imply, however, complete and thorough knowledge. When appropriate, agents should "ask for and receive advice" from human controllers.

Describing what agents are is describing what agents do and what are their characteristics. It might be helpful to think of software agents as akin to their human counterparts such as a travel or business agent. They act on our behalf, have knowledge of our needs, have knowledge of our limitations such as how talented we are, and know what our goals and desires are. They also have knowledge of what methods are available to attain goals, what influences they can assert on our behalf, what they are capable of attaining, and when to seek help such as a travel agent, for example, might consult a consulate regarding travel to a foreign country.

Simply put, software agents are application programs that exhibit certain characteristics that distinguish them from other programming paradigms. They are capable of using existing technology to enhance and add another layer to an otherwise static application. Agents are autonomous, and sometimes mobile, pieces of software that act towards the accomplishment of some stated goal. They do so in such a manner as to seem to be operating "behind the scenes" with little human intervention or guidance.

AGENT CHARACTERISTICS

The following section describes certain characteristics that software agents may possess. Not all agents have all these characteristics. The inclusion of one or more characteristics may be essential to that agent depending on its stated goal. The exclusion of other characteristics may also be essential to that same agent.

Autonomy is a characteristic that allows agents to have some measure of control within their operating environment. Autonomy implies independence, for instance, an agent detects an event and acts accordingly without human intervention. Along with independence comes responsibility, that is the knowledge to act according to some pre-determined set of rules (perhaps contained in a *rule base* or some other higher cognitive reasoning approach), and not to exceed agreed bounds. Given independence, and responsibility, autonomous systems exercise their control towards some stated goal and do so in a proactive manner. Within bounds, they pursue a solution or goal using the

independence they are granted in an active manner without "asking permission" from a human controller. They do so until either the goal is accomplished or is determined to be unattainable.

Autonomy also implies knowledge of the environment in order to implement control. The agent knows where it is located, on a client for example, and knows what actions it is allowed to perform given these surroundings. The agent must also know its own state and should be able to make decisions, given this state, to accomplish the assigned goal. If unable to attain the goal the agent should have the ability to report back to the human controller and, perhaps, move to another state from which it will be able to attain the goal given its knowledge of the environment.

Agent *communication* skills is the ability of agents to report and exchange information required to accomplish the stated goal. An agent must be able to communicate with the user at a minimum, and with other agents when necessary. Communication with the user is essential if the agent is unable to attain the goal or has come up against a barrier which cannot be overcome internally. Communication with other agents may be required if a *collaborative effort* is needed to achieve the stated goal. Other reasons for communication skills are if changes are required such as a change in event priority, a plan of action, or operating schedules.

Autonomy and communication skills are central to the effectiveness of a software agent. Without autonomy an agent is powerless to exercise its obligation to act as a problem solving mechanism. Without communication skills an agent is muzzled in its responsibility to report back to the user and/or collaborate with other agents to achieve its goal.

Agents may need to *collaborate* to attain the stated goal. These efforts can be as simple as "I'll do X when you do Y" to more elaborate efforts such as adjusting receivers after gathering environmental data. All these efforts have in common the ability for one agent to communicate with other agent(s). The *de facto* standard for agent communication is Knowledge Query Manipulation Language (KQML) which is both a language and protocol. KQML is being used as a medium since agents are being developed in a variety of languages. Information on KQML can be found at several web sites such as http://www.cs.umbc.edu/agents/.

Mobility is the characteristic needed by agents to move from one platform to another working towards a directed goal. Mobility may or may not be an operative characteristic depending on the arena which the agent is used. In some applications the ability of an

agent to move, from client to server for example, may be crucial to the agent being able to carry out its assigned task. In another application the agent may spend its entire life residing on the client. Mobility assumes the agent is written in such a way, and in a language, that it can operate on a variety of platforms and in a variety of environments. Mobile agents can be constructed so as to reduce the bandwidth from a traditional client/server transaction. When the client needs data, or some sort of access, it can be made to migrate to the server, complete the transaction, then return back to the client.

Other attributes which an agent may possess are: *reactivity* (the ability to sense the environment and act), *temporal knowledge* (the ability to know ones state and identity over a period of time), and *learning* (the ability of an agent to learn and improve).

When combined, these agent characteristics have the potential to create powerful new application programs which, while adding a layer of complexity to the program, will at the same time expose new problem solving potential.

AGENT APPLICATIONS

Agents are being used in several areas. In some cases they are an integral part of the process, while in other cases they are part of an on-going test process. In either case, software agents are being used to tackle difficult software applications. In this section I present a few different application areas where software agents are being used today.

In Australia, at Sydney airport, a system called OASIS is undergoing tests. Within this system multiple agents represent aircraft and the air traffic control system itself. How the system works is when an aircraft enters the Sydney airspace, normally within a few miles of the airport, an agent is assigned to that aircraft. This agent may be running its own thread. Once assigned, the agent gathers knowledge of the real aircraft's goals which are a part of the planes flight plan. This information is made up of the planes tail number, people on board, fuel on board, destination, etc.. Given a goal, such as to land at Sydney, the aircraft's agents are responsible for collaborating with other agents (representing other aircraft), and agents representing the air traffic control system, to bring the plane in safety. This system is being fulfilled using the *Belief-Desire-Intention* (BDI) model.

In the medical field software agents are being used to monitor patients. The idea behind this application is patients are more likely to prosper when a team of specialists are able to share and exchange vital information about that patient relative to their field of expertise. Close monitoring is needed but not available by specialists. The nursing staff is able to

monitor patients closely but they do not have the expertise of the specialists. The role of the software agents is to distribute the information. Agents are organized into a hierarchy and the system as a whole is based on the *blackboard model*.

The blackboard model is where agents have certain information. They "write" this information to a common data structure or blackboard. Other agents check the blackboard for information they need and are capable of processing. When they find something they take that information from the blackboard and use it as they are designed to do. In this application patient information is shared among those doctors and nurses which need the information when they need it.

The entertainment area is also using software agents. Test projects have been setup for a user to interact with agents who play the role of actors in a movie. Humans interact with agents in which agents carry on a "dynamic narrative" with their human counterparts. These projects are an on-going development because agents are not at a point where they can carry on a conversation in a dynamic fashion.

NASA is using agents, on a restricted basis, to control their Deep Space 1 spacecraft, part of their New Millennium Program. Their approach uses an agent of the ground team on-board the spacecraft. This agent carries information on the high level goals. A planning and scheduling agent uses these stated goals, and knowledge of the spacecraft, to generate either time or event based events. These events are sent to the executive agent which distributes them to the appropriate system, such as thrust.

Because of the time involved in getting information back to the ground station, finding a solution to a problem or determining the correct set of commands, then sending those commands back to the spacecraft, using software agents is a viable option.

SUMMARY

This paper discussed agent description, characteristics, and their application. Agent use can be beneficial to the designers, engineers, and especially the software application programmers when they are faced with writing increasingly complex programs. Although not the "silver bullet" which has been promised, agent technology does take the state of the art a step forward. It does address dwindling bandwidth, learning technology, and can use certain algorithms in an effective manner to create powerful new software programs. Software agents are a viable option available to the industry today.

REFERENCES

[1] Shoham, Y., "An Overview of Agent-Oriented Programming," Software Agents, 1997, ed J.M. Bradshaw, Menlo Park, CA, AAAI Press.

[2] Franklin, S., and Graesser, A., "Is It an Agent or Just a Program? A Taxonomy for Autonomous Agents," Proceedings of the Third International Workshop on Agent Theories , Architectures, and Languages, 1996.

[3] Gilbert, Don, "Intelligent Agents: The Right Information at the Right Time," IBM Corporation, Research Triangle Park, NC, May, 1997.

[4] Kay, Alan, "Computer Software," Scientific American, 251 (3), 1984, 53-59.