An Agent-Oriented Approach to Automated Mission Operations

Walt Truszkowski
NASA Goddard Space Flight Center, Code 522.3
Greenbelt, MD 20771
Email: truszkow@kong.gsfc.nasa.gov

Jidé Odubiyi
Loral AeroSys, 7375 Executive Place
Seabrook, MD 20706
Email: jideo@groucho.aerosys.loral.com

Abstract

As we plan for the next generation of Mission Operations Control Center (MOCC) systems, there are many opportunities for the increased utilization of innovative knowledge-based technologies.

The innovative technology, discussed in this paper, is an advanced use of agent-oriented approaches to the automation of mission operations. The paper presents an overview of this technology and discusses applied operational scenarios currently being investigated and prototyped. A major focus of the current work is the development of a simple user mechanism that would empower operations staff members to create, in real time, software agents to assist them in common, labor intensive operations tasks. These operational tasks would include: handling routine data and information management functions; amplifying the capabilities of a spacecraft analyst/operator to rapidly identify, analyze, and correct spacecraft anomalies by correlating complex data/information sets and filtering error messages; improving routine monitoring and trend analysis by detecting common failure signatures; and serving as a sentinel for spacecraft changes during critical maneuvers enhancing the system's capabilities to support non-routine operational conditions with minimum additional staff.

An agent-based testbed is under development. This testbed will allow us to: (1) more clearly understand the intricacies of applying agent-based technology in support of the advanced automation of mission operations, and (2) to access the full set of benefits that can be realized by the proper application of agent-oriented technology in a mission operations environment. The testbed under development addresses some of the data management and report generation functions for the Explorer Platform (EP)/Extreme UltraViolet Explorer (EUVE) Flight Operations Team (FOT). We present an overview of agent-oriented technology and a detailed report on the operation's concept for the testbed.

1.0 Introduction

Major advances have been made in the process of automating mission operations over the last several years. However, in keeping with changing operational requirements and the need to more effectively realize cost and manpower savings in the area of mission operations, the necessity for more advanced automation technologies is clear. As examples of areas for continued improvement consider the following:

- Mission Operations Control Center (MOCC) software systems are currently developed using classical software engineering paradigms. To bring about added degrees of flexibility in how these systems could handle unexpected problems, the engineering of these systems along agent-oriented technology lines looks promising.

- Even with the increasing use of expert systems in support of telemetry monitoring and command constraint checking much reliance is placed on the manual intervention of operators. The use of agent-oriented techniques can effectively provide additional levels of automated support in handling these important types of operational activities and further reduce the need for manual interventions.

- With increasing automation of mission operations, there is a growing need for more advanced approaches to information handling. The use of agent-technology in support of the full range of information management functions will significantly reduce the growing possibilities of information overload on the part of operators.

It is becoming apparent that for future automated mission operations, more consideration will have to be given to the roles that distributed problem solving and computer-supported cooperative work will play. These increasingly important issues can be addressed by employing intelligent, distributed processes [6] found in a multi-agent based approach, described in this paper.

The rest of this paper presents an ontology [12], i.e., a conceptual framework for describing the mission operations domain, and an implementation framework for automating the operations in that domain. Our approach for dealing with the task of developing an agent-based mission operations environment is to first specialize by applying our agent methodology to automate the report generation function. Once this is accomplished we will then generalize and apply the agent-based approach to other functions in the MOCC as shown in Figure 1 below. Our approach for generating the agent-based report-generation solution is to employ an information agent model and define agent roles in a multi-agent environment required for this selected subdomain function.

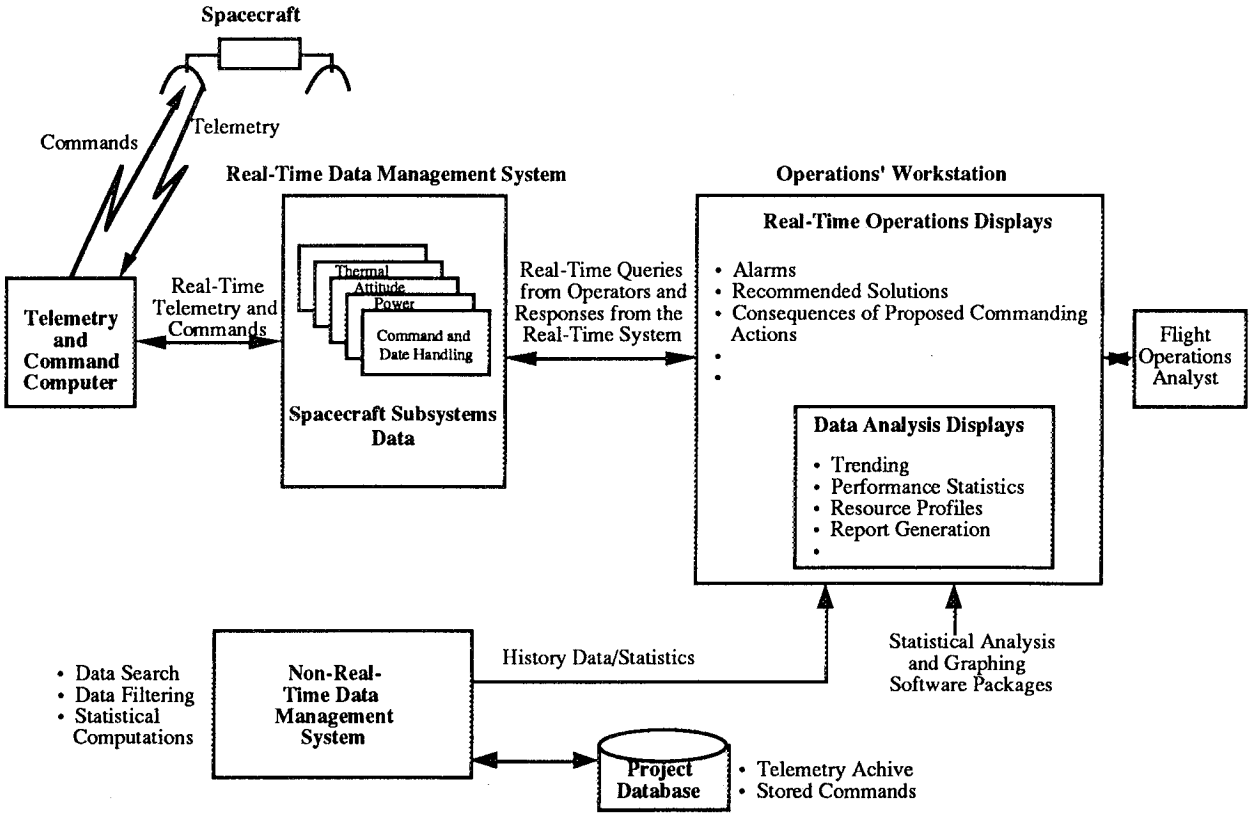


Figure 1: An Overview of a Spacecraft Mission Operations Control Center

We present related work on the use of agent-based approaches for automating information location and retrieval systems and the contribution of our investigation in proving the utility of agent-based technology in mission operations.

2.0 The EP/EUVE Report Generation Process

The EP/EUVE's operational environment is a heterogeneous network consisting of two MicroVAXs (VMS), a Sun workstation (UNIX), an HP-9000 workstation (UNIX), and i386 personal computers, using the X.25 and TCP/IP protocol.

Subsystem engineers for EP/EUVE are responsible for daily monitoring of the satellite's subsystem performance, detection of anomalous subsystem behavior, weekly reporting of subsystem performance, generation of commanding products for subsystem operations, and continuing preparation for subsystem anomaly Detection, Isolation, and Resolution (DIR). These products reside on heterogeneous distributed computing nodes. Off-line analysis (Trend system) provides daily plots of over 600 mnemonics for visual checks of subsystem performance and trends. Subsystem engineers' performance is evaluated based on how well they handle a spacecraft anomaly, not on their daily activities. For example, based on an analysis of operator activities over a period of time, it was concluded that 90 percent of their time is spent performing daily routines. Each week, three of the Explorer Platform's engineers spend a total of 40 hours generating a weekly report on the performance of the system. The routine activities that consume most of the operators' time can be automated to allow them to spend time on more critical tasks.

Three categories of reports are generated by the Flight Operations Team (FOT) of the EP/EUVE system. The three subreports which correspond to the three subsystems of the EP/EUVE are the Modular Power Subsystem (MPS) subreport, the Command and Data Handling Subsystem subreport, and the Modular Attitude Control Subsystem subreport. Other subreports included in the MPS subreports include reports on the Battery Health and Safety, the Solar Array Performance, the MPS Heater Duty Cycle, the Critical MPS Events' Summary, and the Thermal System. The critical MPS Events' Summary is generated from the computer workstation which generates the Real-time and Trend data.

Adequate preparation for a spacecraft anomaly's DIR is the key to successful spacecraft flight operations. The level of preparedness depends on the amount of "spare time" a spacecraft subsystem engineer has to study the subsystem, and the time between anomaly detection and resolution. Automating the report generation process will allow the spacecraft subsystem engineer to devote their time to more productive mission operations such as early detection of anomalies, data analysis, and development of scenarios for anomaly prevention.

The subreports for the Command and Data Handling Subsystem result from collecting six other subreports. The subreports are orbit decay (EP/EUVE 's decrease in orbit periods), tape recorder performance, clock delta trends, transponder performance, Ultra State Oscillator frequency trends, and Modular Antenna Pointing Control.

3.0 An Agent-Oriented Solution to Support the Report Generation Process

An Agent-based FLight Operations AssociaTe (AFLOAT) is currently being prototyped to support the FOT in generating weekly reports. Each agent is an entity that can function semi-autonomously in an environment where other agents exist, accept instructions from a user, and communicate with other agents. In addition, it can be persistent, and can migrate from one node to another to process and retrieve information as requested. The agent can operate independently in the background without interfering with user's actions. An overview of agent-oriented technology and our approach for applying this technology to automate the EP/EUVE operations report generation process are described in the following paragraphs.

3.1 An Overview of Agent-oriented Technology

What is an agent? In the most general form, a software agent as opposed to a hardware agent (e.g., a robot) can be defined as an entity that enables a user to specify what the user wants leaving the process of how and when to accomplish it to the agent [3]. Huhns and Singh [5] present a more comprehensive definition for a software agent as an active knowledge-based computational entity that has knowledge, intentions, and mechanisms for perceiving, reasoning, acting, and communicating. An agent, in our initial prototype, is characterized by a subset of the capabilities of the agent in this comprehensive definition, as explained in paragraph 3.5.

3.2 Distinction between Agent-based Systems and other Computer System Services

There is general confusion on what agent-based systems are and how they differ from other computer system services such as Directory Assistance Programs and Information Brokers [5]. Directory Assistance Programs support interoperation between conventional software programs by accepting requests and routing them to appropriate programs for execution. Information Brokers or Distributed Object Managers such as the Common Object Request Brokering Architecture (CORBA) the Distributed Information Manager (DIM) for EOSDIS, and the Dynamic Data Exchange (DDE) programs, either statically or dynamically provide access to information making the source of the information transparent to the user. In addition to serving as directory assistants, they can also execute requests and return results. All these system services use procedures to communicate with other objects. True software agents use declarative directives that are more expressive to reason and communicate complex concepts with other agents instead of relying on procedural directives which are efficient but they are less expressive.

3.3 Agent Types

An agent's behavior may vary along a spectrum of factors ranging from a controlled learning process to self-learning, controlled behavior to full independence, and simple to complex interactions. An agent's capability may be simple or complex; its interaction with its environment may be reactive or planned (i.e., deliberative). Reactive agents [2] are robot-like with very limited internal reasoning mechanisms while deliberative agents [4] have substantial reasoning capabilities. The agents in a multi-agent system may or may not coordinate their activities. All the agents may be identical or each may be unique, and they may communicate either by directed message passing or broadcast. The number of agents may range from a single agent to thousands. As you will see in paragraph 3.5, the agents in our prototype will be able to learn; each has some degree of independence. The agents can interact with their environment with deliberative reasoning, and communicate with one another through direct message passing and multicast via shared memory.

3.4 Essential Architectural Issues of Multi-agent Systems

To be successful in developing a multi-agent based system, the following four architectural issues must be addressed and the fifth issue is optional: (1) an approach must be established for describing, decomposing and distributing tasks among the agents; (2) a format must be defined for interaction and communication between agents; (3) a strategy must be formulated for distributing controls among agents in which a local control strategy demands that agents communicate only their results, or centralized control where one agent assigns all the tasks, or a predefined mixed results/tasks share control; (4) a policy must be made for coordinating the activities of agents, either by competition through negotiation as in ContractNet Protocol or cooperation through centralized or distributed planning; and (5) a rationale should be established for maintaining truths, i.e., consistent beliefs and conflict resolutions among agents [6] or mental states or trusts [10]. Our

current architecture, described below, can address the five architectural problems. Our initial goal is to resolve the first four issues making the resolution of the fifth issue a long-term goal.

3.5 A System Overview of an Agent-based Solution to Automate Mission Operations

Our objective is to develop AFLOAT as a multi-agent based system where a user interface agent interacts with the user to accept user requests, collaborates with other agents at a local host or over the Internet in locating, retrieving, and presenting the information to the user in appropriate form, with the correct amount and level of detail, and at the right time. An implementation framework for AFLOAT consists of an architecture for a software agent, a methodology for implementing the interactions between the user and a user interface agent, collaboration between multiple agents, and an approach for making background software agents specialize in data retrieval from distributed information sources. User-to-agent and agent-to-agent interaction issues are resolved by developing a communication protocol, a language format, and an agent migration process across networked computer systems. Our strategy for information location and retrieval is based on the premise that domain-dependent keywords used by the user will form an index to the information in the domain and to the specialized agent. If the key word does not exist, then retrieval is not possible, and the user interface agent will issue appropriate advice. Knowledge in AFLOAT can be stored as rules, objects, cases (examples), models, and programs. Each agent has access to a set of support services such as: creating, destroying, managing, or monitoring the activities of spawned agents; mechanisms for message transport; directory of other agents; information processing and presentation; and system performance monitoring.

In addition to supporting on-line and off-line flight operations of the EP/EUVE report generation process, agents in AFLOAT can also support the spacecraft platform and instrument Fault Detection, Isolation, and Recovery (FDIR) services.

Our architecture for automating mission operations has been designed to address the top four basic architectural issues and to be extensible enough to accommodate the fifth. The implementation framework is based on a deliberative agent architecture, depicted in Figure 2. The architecture has structural elements for data storage, coordination, and monitoring of activities between agents, execution of internal and external functions, inter-agent communication, and interface with other domains in the MOCC.

Architecture of AFLOAT's Deliberative Software Agent. Each of AFLOAT's software agents is deliberative, which means that it will reason before it acts. An architecture of such a software agent in AFLOAT is displayed in Figure 2. It addresses the issues that must be resolved in a deliberative multi-agent based system. The **coordinator** determines the type of coordination (task sharing or result sharing), and coordination policy (negotiation, shared memory, or an explicit domain-driven task delegation policy) that will be employed. In AFLOAT, agents coordinate their activities by sharing results, and an explicit domain-driven task delegation policy is employed since each agent is considered a specialist in a specific domain. The agent's coordinator module is also responsible for planning and scheduling the tasks of each agent. Each agent's **monitor** is responsible for monitoring interactions between agents, incoming and outgoing messages, the state of the agent, and maintaining a history of the agent's actions. Saving an agent's past actions aids it in learning by drawing from experience when presented with new tasks. The **external models** module of each agent maintains global functions that are accessible for use by other agents. Each agent must maintain its access rights to external information so as to aid the domain agents in the information retrieval process. The **internal models** module maintains functions (such as managing access to the skills of each agent or maintaining its message buffer) that are private to each agent and are not accessible to external agents except the AFLOAT executive agent. Each agent also has an **inter-agent communication module** which is responsible for validating inter-agent, semi-structured language format, sending outgoing messages, receiving incoming messages, and broadcasting messages to shared memory. The brain of each agent is its

information base where all the modules store their data and other information such as the name of the local system management agent (AFLOAT executive), buffers for incoming and outgoing messages, each agent's name, type, and state, and messages in shared memory. Communication with each agent is done by adding a message to its information base. Each agent can store knowledge as rules, objects, cases (examples), models, and programs. The structure of each agent, coupled with its behavior (i.e., capabilities) provides it with enough intelligence to respond effectively to information retrieval tasks delegated to it.

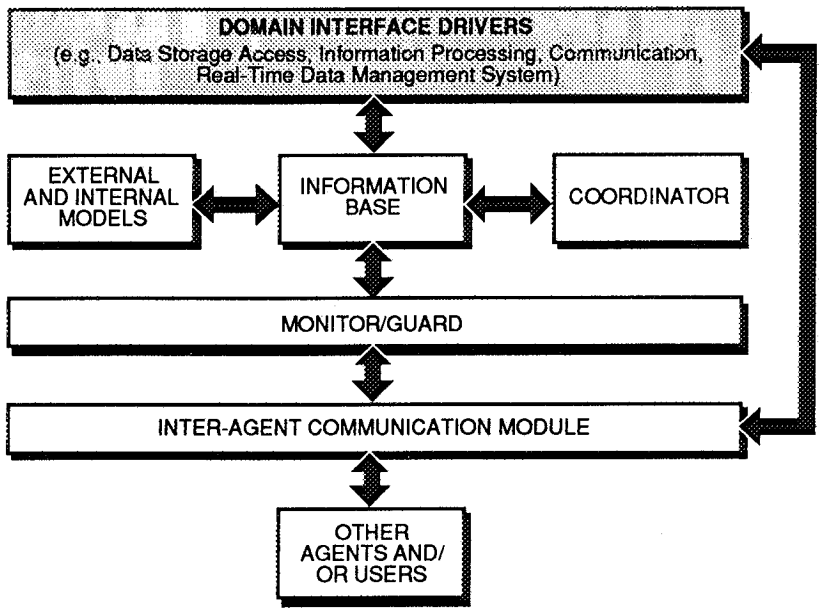
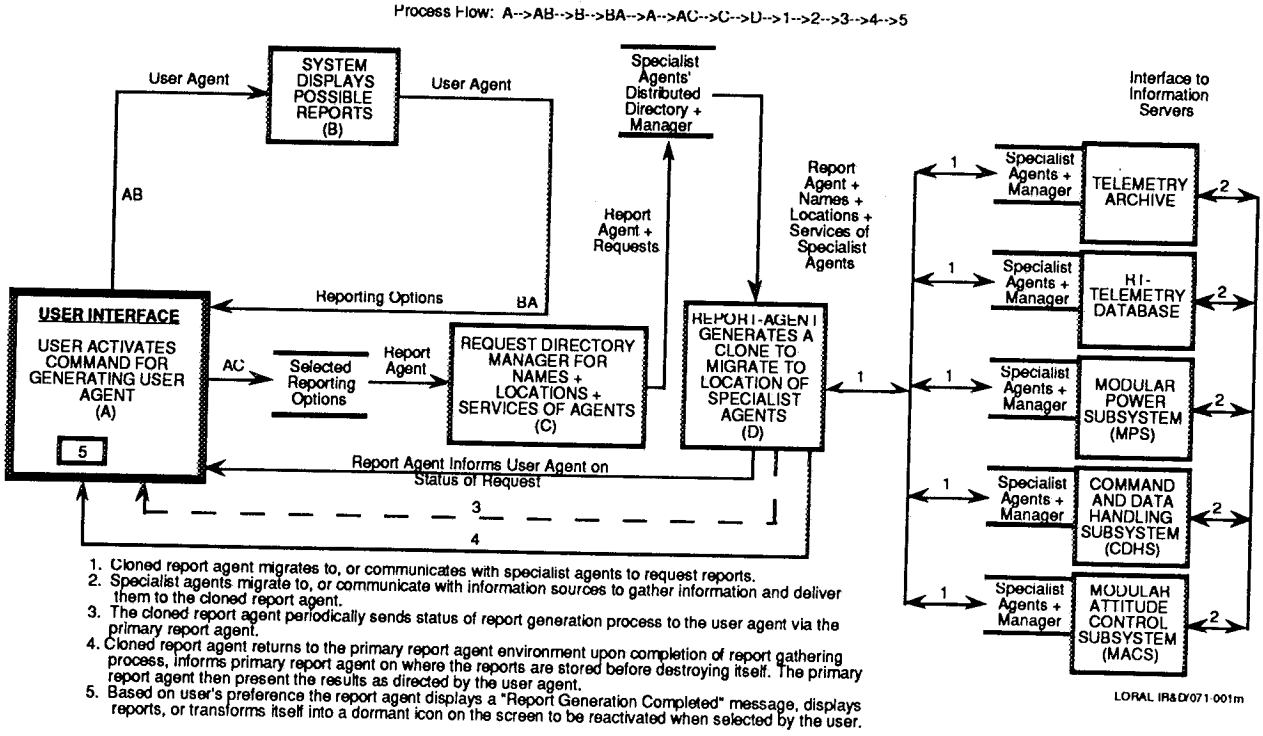


Figure 2. Architecture of AFLOAT's Deliberative Software Agent

An Information Agent Model for Supporting Information Retrieval: Agents in AFLOAT are characterized by five "action-oriented" [9] capabilities: First, migration, is the ability of an agent to move to other nodes to process or retrieve information. This ability can support load balancing, improve efficiencies of communication, and provide unique services which may not be available at a local node. Second, semi-autonomy, is the ability to respond to a dynamic environment without human intervention, thus improving the productivity of the user. Third, spawning, is the ability to create other agents to support the parent agent, thereby promoting dynamic parallelism and thus fault-tolerance. Fourth, persistence, is the ability to recover from environmental crashes and support time-extended activities, thus reducing the need for constant poling of the agent's welfare and better use of the system's communication bandwidth. The fifth and final capability is interaction mechanisms for supporting agent-to-agent and user-to-agent interactions.

Operations Concept for AFLOAT Prototype: An operations concept for the AFLOAT testbed prototype is illustrated in Figure 3. It describes the procedures for using agents to locate, access, retrieve and present EP/EUVE reports or information located at remote information sources. To do this, the user generates a user agent. The user agent requests the system to display a set of reporting options. The user then selects one or more items from the list displayed by the system. Upon completing the selection process, the user agent generates a report agent and assigns it the responsibility of generating the reports. The report agent identifies specific subreports and requests the agents' directory manager (or name/skill server) for the names, locations, and services provided by agents that can support the generation of requested reports.



1. Cloned report agent migrates to, or communicates with specialist agents to request reports.
2. Specialist agents migrate to, or communicate with information sources to gather information and deliver them to the cloned report agent.
3. The cloned report agent periodically sends status of report generation process to the user agent via the primary report agent.
4. Cloned report agent returns to the primary report agent environment upon completion of report gathering process, informs primary report agent on where the reports are stored before destroying itself. The primary report agent then present the results as directed by the user agent.
5. Based on user's preference the report agent displays a "Report Generation Completed" message, displays reports, or transforms itself into a dormant icon on the screen to be reactivated when selected by the user.

Figure 3: Operations Concept for Agent-based FLight Operations AssociaTe (AFLOAT)

In addition to knowing the names, locations and services provided by the specialist agent, the report agent must also determine if there are restrictions to services provided at certain locations. If an access is restricted to information sources or there is an absence of unique services required by specialist agents, the report agent may request the reports remotely via message passing. If there are no such restrictions, the report agent generates and sends a clone with enough information necessary to generate the report to migrate to remote information sources, interact with agents with special skills, and retrieve the reports. Allowing the report agent to send its clone to retrieve reports while it stays at the user's environment adds some fault tolerance to the system. Therefore, if the cloned report agent fails, the primary agent has all the information needed to create another clone. Periodically, the cloned report agent informs the primary report agent resident at the user's environment on the progress of the report generation process. Steps 1, 2, 3, 4, and 5 in Figure 3 explain the interactions between the agents and the report generation process.

Development Environment and Status and Plans for AFLOAT Project: The development environment for implementing AFLOAT is the NASA/Johnson Space Center developed C-Language Integrated Production System (CLIPS) version 6.0 with CLIPsTOOL

software from KNOWARE Inc. (for building the user interface) running on a Sun SPARCstation with UNIX operating system, X-windows, and OSF/Motif Style Guide. The application of the defmodule construct (in CLIPS) which promotes the partitioning of knowledge bases will enable us to achieve agent independence. We have just completed Build 1 of the AFLOAT testbed. This build provides location transparency to information sources for generating reports on Battery Charge/Discharge ratios of the three batteries on the spacecraft. This build is also being used by two George Washington University researchers to investigate the issue of trust of automated systems. In their experiment, an operator is assigned a task that he/she must perform plus an additional task of monitoring the quality and number of faults correctly detected by the agents. The operator's trust level of the agent is based on the frequency and types of incorrect faults. Build 2 of AFLOAT will provide the users with the ability to generate reports on the operations of the three subsystems, i.e., the MPS, the CDHS, and the MACS from distributed information sources.

4.0 Related Agent-based Information Retrieval Systems

Several agent-based information retrieval systems are being prototyped at several research laboratories. Most of the research work attempts to resolve the fundamental architectural issues described earlier in paragraph 3.4. The research work of Amy Lansky at NASA/Ames [8], and Bond and Gasser [1] focuses on multi-agent planning and addresses the issues of coordination, synchronization, and control of multiple autonomous agents. Shoham's work [10] investigates the issue of an agent's mental states as they relate to beliefs, intentions, and capabilities. Other research on agent-based information retrieval similar to ours include the work by Kahn and Cerf [6] in which agents, called Knowbots, each hard coded to perform a specific task, are used to retrieve information from digital libraries. Etzioni's work [3] on Softbots employs software agents to perform different UNIX tasks to support a UNIX programmer. A very important contribution of his work is the ability of the Softbots to retrieve information with an incomplete request. Papazoglou and Laufmann [9] employ coarse-grained agents with a semi-structured language and message passing to support information retrieval from distributed information sources. The semi-structured language format is quite expressive and it can help the agents in communicating their goals, results, and states, thus facilitating coordination among the agents. Gio Wiederhold [12] employs very coarse-grained agents called mediators which can be used to filter data by resolving any mismatches in the data. A major contribution of the mediator approach is the merit of this architecture over integrated or federated agent-based system architectures. While it is more difficult to implement, the mediator architecture is easier to scale up and add new interfaces than the other two.

While each of the research efforts described above address various aspects of the architectural issues of multi-agent systems, AFLOAT's architecture has been built as an extensible testbed and it can address all the basic architectural problems of a multi-agent-based system. In addition to its capability to automate distributed information retrieval, it can also support automation of other operations such as fault detection, isolation and recovery of satellite subsystems, and other domains. Whereas in a large majority of other multi-agent systems, the base prototyping language is either LISP or PROLOG which very often is not well received by the operations staff due to a lack of experienced programmers; AFLOAT is based on an expressive AI shell written in C with the UNIX operating system, making it readily portable to other platforms and acceptable to operations staff.

5.0 Conclusion

The distributed nature of the operations in a satellite MOCC calls for solution approaches to problems in the domain to consider the use of intelligent distributed modules instead of isolated intelligent systems. Such intelligent distributed modules have been modeled as a multi-agent system and prototyped as the AFLOAT testbed to support the automated report generation process, and described in this paper. An overview of agent-based technology has been presented with

essential architectural issues that must be addressed to successfully implement a multi-agent based system to support automated mission operations. We have shown how the architecture of each agent coupled with its behaviors (i.e., its capabilities represented as an information agent model), can be used to resolve basic architectural problems of multi-agent systems.

The use of multi-agent based designs is not limited to the mission operations domain. They can be employed in any environment where the user needs to delegate an associate to perform information management activities such as in telecommunications network management, software reuse management, and automated traffic incident management systems.

6.0 Acknowledgements

This work originated with a paper by Truszkowski and Moore [11]. We wish to thank Mike Moore for his major contributions to the development of the agent model and testbed concepts which are now being prototyped. The funding for this work is being provided by NASA Headquarters Code O (Office of Space Communications).

Bibliography

1. Bond, A. H., and Gasser, L. (1988). *Readings in Distributed Artificial Intelligence* , Morgan Kaufmann, San Mateo, CA.

2. Brooks, Rodney A. (April, 1991). *Intelligence Without Reason, Computers and Thought,* IJCAI-91, AI Memo No. 1293, pp. 1-27.

3. Etzioni, Ören, (1994, July). *A Softbot-Based Interface to the Internet,* Communications of the ACM, pp. 72-76

4. Ferguson, Innes A. (1992, May). *Touring Machines: Autonomous Agents with Attitudes,* IEEE Computer, pp. 51-55.

5. Genesereth, M. R., and Ketchpel, S. P. (1994, July). *Software Agents*, Communications of the ACM, pp. 48-53

6. Huhns, M. N., and Singh, M. P. (1994, May). *Distributed Artificial Intelligence for Information Systems* , MCC, Austin, TX 78759.

7. Knoblock, C. A., Arens, Y., and Hsu, C. (1994, May). *Cooperating Agents for Information Retrieval* , Second CoopIS-94 Proceedings, University of Toronto, Canada, pp. 122-133.

8. Lansky, Amy, (1994, April). *Data Analysis Assistant,* Recom/NASA Ames Research Ctr.

9. Papazoglou, M., Laufmann, S., and Sellis, T. K. (1992) *An Organizational Framework for Cooperating Intelligent Information Systems,* International Journal of Intelligent and Collaborative Information Systems, Vol. 1, No. 1, pp. 169-202.

10. Shoham, Yoav. (1990). *Agent-Oriented Programming, Technical Report,* STAN-CS-1335-90, Robotics Laboratory, Computer Science Dept., Stanford University, Stanford, CA.

11. Truszkowski, Walt, and Moore, M. (1992). *Towards an Information Ecology,* AIP Conference Proceedings 283, pp. 884-892.

12. Wiederhold, Gio. (February 1992). *Mediators in the Architectures of Future Information Systems*, IEEE Computer, pp. 38-49.