An Overview of Agent Technology for Satellite Autonomy

Paul Zetocha
Air Force Research Laboratory
Space Vehicles Directorate
Kirtland AFB NM, 87117
Paul.Zetocha@vs.afrl.af.mil

Lance Self
Air Force Research Laboratory
Space Vehicles Directorate
Kirtland AFB NM, 87117
Lance.Self@vs.afrl.af.mil

Abstract

To operate the majority of today's satellites is costly and manpower intensive. In order to reduce cost, enhance survivability, and increase responsiveness, technologies are needed which will make satellites more autonomous. Many artificial intelligence (AI) based prototypes have been developed with the above objectives in mind, however these prototypes are typically geared towards solving a particular problem with a particular technique such as identifying faults in the attitude control system using a model-based reasoning system. Generally these AI systems only look at a subset of the total information available and do not take a system approach to reasoning. To optimize the ability of an AI system to make correct decisions many factors need to be considered including the status of all subsystems, current and future mission objectives, and the status of components external to the satellite including ground facilities and other satellites. Intelligent agent technology provides a mechanism to integrate these various components and to reason at a higher level. The purpose of this paper is to give an overview of intelligent agents as they apply to space systems. The paper begins with a short introduction of what intelligent agents are and how they operate. Following this we will highlight particular areas where the use of intelligent agents may be advantageous. Next we highlight some of the intelligent agent efforts which are ongoing in the Space Vehicles (VS) Directorate of the Air Force Research Laboratory (AFRL). Following this we will present an overview of some of the ongoing efforts in the space community external to AFRL. We will then present an overview of some of the agent-based architectures which are publicly available and discuss their applicability to the space domain. Lastly we will summarize and discuss future directions in this area.

Introduction

Agent-based systems are goal oriented systems in which individual agents are given specific goals to achieve and in which the path to goal satisfaction is dependent on the surrounding environment. Agents have the ability to collaborate with one another to solve larger problems. This approach to satellite command and control provides the ability to integrate many of the more traditional AI approaches. Problems which may involve knowledge and cooperation between many components can be solved by applying the most appropriate problem solving technique to sub-components and integrating the individual solutions to form a complete solution.

One of the keys to the success of any reasonably sized agent-based system is the ability of the various agents to communicate with one another. Agents perform their tasks in cooperation with other agents, and an overall assessment of a situation is based on the status of multiple agents. Agents, whether they reside on the same platform or across multiple platforms, must be able to share knowledge amongst themselves. One such method developed to facilitate this collaboration is the Knowledge Query Manipulation Language (KQML). KQML, originally developed at DARPA, is a set of protocols and a language used by agents to exchange information [1]. One of the strengths of KQML is that it is language and operating system independent. KQML resides as an application on top of other communication protocols such as TCP/IP, while information to be shared is placed within KQML wrappers. KQML equipped agents communicate by maintaining a virtual knowledge base of information that might be relevant to other agents. Through these virtual knowledge bases, other agents make goals, beliefs, desires, requirements, and current status available for use. KQML messages are defined as *performatives*, which denote allowable operations that agents can perform on one another's virtual knowledge base.

A second key characteristic for agent success is autonomy. Autonomy is a characteristic that allows agents to have some measure of control over what they can do. Autonomy implies independence, for instance, an agent detects an event and acts accordingly without human intervention. This means that agents act against some pre-determined set of rules contained in a rule base, or some higher cognitive reasoning approach. Autonomy also implies knowledge of the environment in order to implement control. The agent knows where it is located 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 required decisions, given this state, to carry out assigned duties. If unable to, it should have the ability to report the state back to the user and/or move to some other state from which it will be able to carry out its assigned duties.

Agent Technology Applied to Space Systems

The use of agent technology has a wide range of applicability to space systems. It is most applicable for problems in which reasoning is a function of information from several disparate sources or for complex distributed systems in which cooperation amongst the various components is needed for problem solving. To this end the following areas are ideally suited for intelligent agent technology; fault detection, isolation, and resolution (FDIR); mission operations; intelligent planning and scheduling; data fusion and situation assessment for optical payload processing; executive satellite control; and distributed satellite systems.

In general, for solving all but trivial problems, when a human operator performs satellite anomaly diagnosis he/she takes the entire operations environment into consideration. Telemetry mnemonics for all subsystems need to be examined as there is generally a degree of dependence between mnemonics from different subsystems. In addition, current and future mission objectives as well as the states of applicable ground-based components need to be taken into consideration. Since reasoning is based on information from several sources, agent technology provides a natural mechanism to integrate the various information sources. The same justification applies to the use of agent technology for mission operations in general.

The concept behind an intelligent agent is synonymous with that of an autonomous planning system in that both are goal driven. With an autonomous planner, mission objectives or “goals” are given to the system and the planner has the responsibility of determining the sequence of steps necessary to achieve the desired objective(s). This would in general involve cooperation with other agents. An agent-based autonomous planner could be used in two capacities on-board a spacecraft. The first is in response to unknown satellite anomalies. These are anomalies which have not previously occurred or which have not been anticipated and for which an appropriate recovery procedure is not immediately known. The autonomous planner would generate the sequence of actions necessary to recover from the anomaly. The second capacity in which an agent-based autonomous planning system may be beneficial is with a surveillance type mission. New entities to be observed would be input into the system and the autonomous planner would determine the appropriate actions necessary to reconfigure the payload or spacecraft in order to meet the objectives. Related to this is the use of intelligent agents for data fusion and situation assessment with regards to optical payload data processing. The majority of efforts to automate target detection or observation type missions perform their analysis on an image by image basis. This often fails to sufficiently capture the scenario at hand. To obtain a true picture of what may be occurring it may be necessary to fuse the analysis of several images together. As an extension of this it may be beneficial to pass information from one satellite to another in a constellation in order to optimize effectiveness. For example for a surveillance mission one satellite may detect some entity of interest and track that object until it is no longer in its field of view. At that time the current state may be passed to a second satellite from which surveillance of the object of interest can continue. Agent technology provides a natural mechanism to fuse data and allow collaboration across multiple satellites.

Agent technology provides the flexibility to allow either distributed or centralized control. For use on-board a satellite one option is to have a top level executive controller agent which oversees the operation of several lower level executive agents. For example, lower level executive agents may monitor and control the processing of specific subsystems while the higher level agent may be used to facilitate collaboration between the lower level agents as well as agents which may be on the ground or on other satellites.

Agent-Technology efforts at AFRL

Two agent related efforts are underway within AFRL/VS. These include the Phillips Executive Agent-Based Controller Helper (PEACH) project and an effort involving cooperating intelligent agents for distributed satellite systems which is being accomplished in cooperation with the TechSat-21 program.

The objective of the PEACH project is to enhance the on-board intelligence and decision making of satellites using intelligent agent and executive controller technology with a target domain being surveillance. [2] Rather than processing surveillance payload data on the ground, processing is done on-board with payload functionality and configuration being a function of the processing results. Configuration and functionality is also dependent on several other factors including the status of other subsystems, mission objectives, ground interaction, and location of other satellites. As one example, if a spacecraft reorientation or extended sensor dwell time over a region is needed, then the appropriate hardware/software on-board the spacecraft could be tasked via agent technology. The initial PEACH prototype will focus on applying the executive controller / agent concept to a surveillance payload mission. Existing techniques largely focus on providing ground-based intelligent processing on a per image basis. This will be extended to include on-board image analysis by providing intelligent situation assessment based on multiple images. The configuration of sensor related hardware and software will be controlled based on intelligent decision making by the agent-based executive controller. Where appropriate, the architecture will have the capability to pass relevant information from one satellite to another. Subsequent work will involve integrating information from different sources and integrating an intelligent planning capability into the system which will allow for the autonomous generation of plans in response to changing mission objectives or the occurrence of anomalous conditions.

A PEACH prototype agent-based system for intelligent sensor control and target processing and detection is currently under development with the initial optical mission being a surveillance mission using spectropolarimetry. The initial framework has been developed in MATLAB with subsequent development to include moving the environment to a ground-based real-time flight environment. Incorporated in the framework is the capability to develop and simulate satellite components. In addition several artificial intelligence utilities are also being incorporated into the framework for use by the agents. Plans are underway to flight test aspects of the agent-based executive controller.

A second agent related effort at AFRL is investigating the use of cooperating intelligent agents for distributed satellite systems. This effort "involves modeling the individual satellites as intelligent agents and allowing them to work together in a cooperative fashion to solve-system level problems"[3]. As a proof of concept the technique has been applied towards solving the N-Queens and minimum queens classical computer science problems. The ultimate objective is to determine the applicability of an agent based approach to a large constellation of small satellites performing space-based radar in a cooperative fashion. AFRL is investigating this concept, along with a number of other issues, through the TechSat-21 program.

Agent-Technology efforts external to AFRL

One such effort is being conducted under a Small Business Innovative Research (SBIR) contract with NASA Goddard Space Flight Center (GSFC) with the objective being to "Investigate the feasibility of developing a distributed environment for onboard planning and scheduling."[4] Specifically, GSFC is investigating "lights out" pass plan generation, onboard instrument management, automatic "pre-established goal-directed science agenda" execution, and total spacecraft operations managment by software agents.

One of this projects enabling technologies is AI planning and scheduling. Planning will consist of "finding a sequence of actions that would lead from the initial situation to the final one."[4] Further planning will proceed by selecting non-primitive tasks and decomposing hierarchically into sub-tasks by employing constraint propagation technology. Sub-tasks will be used to detect and resolve any conflicts which may exist. For example, a task may be to "send a picture to a ground station." In this case, sub-tasks would be: define picture object, define ground station, transmit picture. Using a decomposition method to refine a task helps resolve conflicts. Using the previous example, a conflict may occur when deciding which pictures are transmitted given a limited amount of bandwidth and time in which the satellite is within the ground stations view. Further decomposition of sub-tasks may occur until such time as a fundamental task is reached.

Another enabling technology which GSFC is addressing is the use of intelligent agents for distributed problem solving. They define an agent as "an entity which operates in an environment either autonomously or semi-autonomously interacting with other agents by means of communication" and distributed problem solving as an "approach employs a set of agents who communicate and cooperate with each other to achieve goals related to planning and scheduling." [4] Agents will communicate with one another via a message passing mechanism similar to KQML. They have defined sender, receiver, and identifier fields, however it does not follow the standard KQML format.

A prototype is being developed using Java socket based inter-agent communication. The feeling is that the distributed process approach increases agent efficiency and guarantees the plan and schedule quality. Possible applications might be to integrate GSFC's planning and scheduling technology, within other agent architectures such as NASA/JPL's Remote Agent or AFRL's PEACH program.

One of the first agents developed for intended use on-board a spacecraft was the Remote Agent Experiment (RAX) developed by NASA/AMES as part of NASA/JPL's Deep Space One (DS1) program. RAX basically consists of three components: an executive, an autonomous planner, and a model-based reasoning system [5]. The main function of RAX is to accept high level goals throughout its mission life and then have the autonomous planner develop a sequence of steps to achieve the goal. The model-based reasoning component would be used to solve anomalous conditions.

NASA/GSFC is interested in the use of intelligent agents for mission operations and has investigated agent communication languages [1]. NASA/JSC also has an interest in the use of agents for mission operations and has done some preliminary work in that area.

Commercial Agent-Based Architectures

Private companies and government agencies both have ongoing research projects concerning software agents. Their efforts are directed towards research issues such as: agent communication languages, agent collaboration, goal validation, agent granularity, agent formalization, individual agent learning, and learning by a community of agents. In the satellite community these research efforts are directed at advanced applications aimed at gaining fully autonomous ground and flight control systems.

Three tools are reviewed which ease agent development and are commercially available. Although these tools are not specifically targeted for satellite or space operations, they are useful for developing agents.

The first agent development tool is from **The Haley Enterprise**. Their development tool is called *ActiveAgentX* and, as the name implies, is an ActiveX control which supports Microsoft's Component Object Module (COM). This development tool "allows business rules to be encoded directly as production rules." [6] ActiveAgentX uses the same technology, *Rete Algorithm*, which is used in the *Eclipse* inference engine.

The *Rete Algorithm* is used in production rule systems. The production rules themselves can be organized for pattern matching. The Rete algorithm creates a decision tree that combines the patterns in all the rules. It uses a directed graph where nodes "represent patterns, and paths from the root to the leaves." [7] Each node has information about the preceding path(s) and facts which lead to this node. "This information is a relation representing the possible values of the variables occurring in the patterns in the path."[7]

ActiveAgentX can be used on decision support tools to deliver business rules directly to intranet users over web browsers running on Windows NT or 95. The tool can also be embedded within Java applets which run Microsoft Internet Explorer, or within Java applications which have been compiled using Microsoft Visual J++.

The second agent development tool is from **Reticular Systems Inc.**. Their suite of agent building tools is called *AgentBuilder* and is comprised of two major building blocks: the "Toolkit" and "Run-Time System." The Toolkit has tools for managing the agent development process while the Run-Time System provides the environment for executing the software agent. All *AgentBuilder* components are implemented in Java meaning they can be developed and executed on any Java virtual machine.

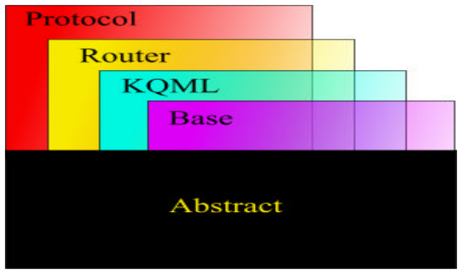
The Toolkit itself is made up of the following Project control tools: Ontology manager, Agency manager, Agent manager, and Agent debugger. The project control tools are for management of the agent development process. These tools include the dictionary and the repository. Agent developers can simultaneously develop agents and use these tools to manage the interaction between agents, and share information between agent projects.

The agency manager is used to construct an *agency* which is defined as "two or more agents that communicate and cooperate with each other to perform some task."[8] The agency manager lets the developer identify and define characteristics of the agents which are being developed. A run-time window is provided for starting, stopping, and examining agent states within the agency along with tools to specify messages and protocols between agents.

The agent manager has tools to define an agents initial model and behavior. Initial beliefs, commitments, intentions, capabilities, and rules of behavior are among the constructs making up the initial model and behavior. There are also tools for planning and learning capabilities. The agent manager creates an "agent definition file" which is written in Reticular Agent Definition Language (RADL) and in turn interpreted and executed by the Run-Time System.

The Run-Time System is comprised of the agent program and the run-time engine which combine to produce an executable agent. The agent can then be deployed as an entity executing in the environment.

The final reviewed agent development tool is **JATLite**. JATLite (Java Agent Template Lite) is a suite of programs which provide the basic structure or template to create software agents which communicate over the Internet. Using JATLite, the developer builds agents which send and receive messages using KQML or some other communication language. The JATLite template gives the user Java classes which "facilitate agent construction."[9] JATLite does not, however, provide a built-in mechanism for creating agents with intelligence. It does provide a robust environment for agent communication and interaction. The JATLite architecture is organized into five layers as shown below.



The Abstract layer has the abstract classes required for JATLite implementation. JATLite assumes all connections are TCP/IP although other protocols such as UDP can be implemented.

The Base layer provides communication based on TCP/IP and the Abstract layer. The Base layer can be extended to allow inputs from sockets, output to files, and to give agents multiple message ports.

The KQML layer is used for storage and parsing of KQML messages. This layer is implemented with the KQML protocol for registering, connecting, and disconnecting.

The Router layer is used for agent name registration, message routing, and message queuing. All messages sent by agents are routed through this layer. The Router also stores messages.

The Protocol layer supports internet services such as SMTP, FTP, etc. for Java applications and applets.

Any or all of the above reviewed agent development COTS products could be used by private and government entities to develop and implement intelligent software agents. What arena these agents operate in are independent of the environment in which they were developed. Each tool has a different approach to agent development. Although the *ActiveAgentX* tool is primarily aimed at business, their approach could have impact on other areas. The second product is a full suite of agent development tools which could reach scientific as well as business communities. The final product supplies a basic environment in which the specific environment to develop agents is left up to the developer. All development environments could be used to develop robust, mobile agents applicable to the satellite or space fields.

Conclusions

The use of intelligent agents provides a mechanism to significantly enhance spacecraft autonomy and is applicable to a wide range of tasks including FDIR, autonomous planning, executive control, data fusion, and payload automation. Intelligent agents have applicability to both the satellite bus and payload. AFRL is exploring many of these aspects through in-house and contractual efforts. The current focus within AFRL/VS is the use of intelligent agents for surveillance payload automation. Included in this is the interaction with other subsystems in which the payload is dependent. A number of issues still exist with regards to intelligent agents with agent collaboration still being one of the key research issues. Several agent-based architecture prototypes have been developed which are generally geared towards a specific type of application and use a specific collaboration mechanism. Future research will likely help to standardize agent communication and assist in developing architectures which are more robust.

References

1. Dan R. Ballard, "Intelligent Agents and Agent Communication Languages for Mission Operations", Retcular Systems, Final Report for NASA contract NAS5-33264, June 1996.

2. Zetocha, Paul; Ortiz James, "*PEACH - An Agent for Increased Space Operations Automation*", Proceedings of the SpaceOps 98 conference, Tokyo Japan, 1998.

3. Skinner, James; Tollefson Mark; Rosenstock, Jeremy; "*Cooperating Intelligent Agents for Distributed Satellite Systems",* Proceedings of the AIAA Civil and Defense Systems Conference, Hunstville AL, Oct 1998.

4. Das, Subrata et al., "*A Distributed Environment for Onboard Planning and Scheduling*", Phase I SBIR Final Briefing by Charles River Analytics, Sep 1998.

5. Bernard, D.E.,; Dorais, G. A.; Fry, C.; Jr., E. B. G.; Kanefsky, B.; Kurien, J.; Miller, W.; Muscetolla, N.; Nayak, P. P.; Pell, B.; Rajan, K.; Rouquette, N.; Smith, B.; and Williams, B. C., *Design of the Remote Agent Experiment for Satellite Autonomy*, Proceedings of the IEEE Aerospace Conference, Snowmass CO, 1998.

6. "*ActiveAgentX: Business Rules with ActiveAgentX for Microsoft Component Object Model*", The Haley Enterprise Web site – http://www.haley.com

7. "*The Rete Algorithm*", UGAI Lectures – http://yoda.cis.temple.edu:8080/UGAIWWW/lectures/rete.html

8. "*AgentBuilder: An Integrated Toolkit for Constructing Intelligent Software Agents*", Reticular Systems, Inc. Home Page, http://www.agentbuilder.com/Documentation/WhitePaper, p. 43, Jan 1998

9. "*JATLite Overview*", JATLite Home Page, http://java.stanford.edu/java.agent