Autonomy for SOHO Ground Operations

Walt Truszkowski, NASA Goddard Space Flight Center (GSFC) Walt.Truszkowski@gsfc.nasa.gov
Nick Netreba, a.i. solutions, Inc. netreba@ai-solutions.com
Don Ginn, a.i. solutions, Inc. ginn@ai-solutions.com
Sanda Mandutianu, Jet Propulsion Laboratory (JPL) sanda.mandutianu@jpl.nasa.gov

Abstract/Background

The SOLAR and HELIOSPHERIC OBSERVATORY (SOHO) project [SOHO Web Page] is being carried out by the European Space Agency (ESA) and the US National Aeronautics and Space Administration (NASA) as a cooperative effort between the two agencies in the framework of the Solar Terrestrial Science Program (STSP) comprising SOHO and other missions. SOHO was launched on December 2, 1995. The SOHO spacecraft was built in Europe by an industry team led by Matra, and instruments were provided by European and American scientists. There are nine European Principal Investigators (PI's) and three American ones. Large engineering teams and more than 200 co-investigators from many institutions support the PI's in the development of the instruments and in the preparation of their operations and data analysis. NASA is responsible for the launch and mission operations. Large radio dishes around the world, which form NASA's Deep Space Network (DSN), are used to track the spacecraft beyond the Earth's orbit. Mission control is based at Goddard Space Flight Center in Maryland.

The agent group at the NASA Goddard Space Flight Center, in collaboration with JPL, is currently involved with the design and development of an agent-based system to provide intelligent interactions with the control center personnel for SOHO.

The basic approach that is being taken is to develop a sub-community of agents for each major subsystem of SOHO and to integrate these sub-communities into an overall SOHO community. Agents in all sub-communities will be capable of advanced understanding (deep reasoning) of the associated spacecraft

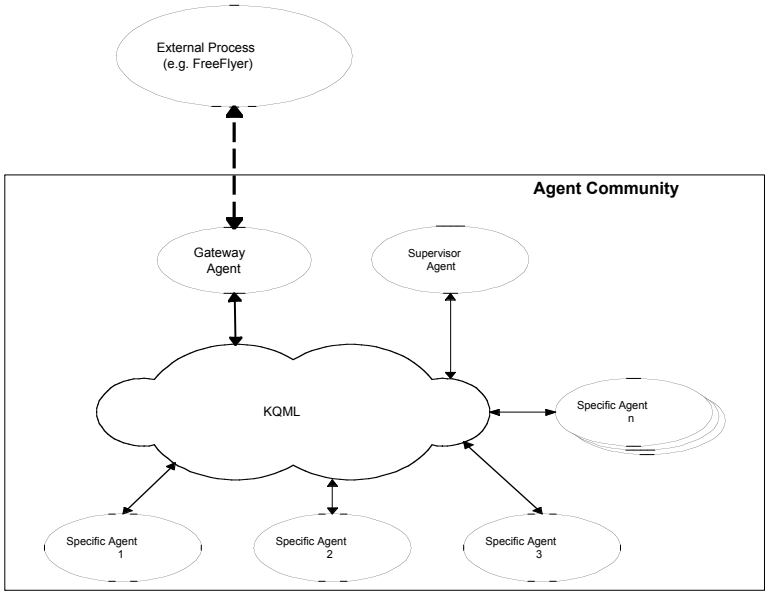

Figure 1. Generic View of a SOHO Agent Sub-Community

subsystem. Figure 1. gives a generic view of a typical SOHO sub-community. The Gateway Agent acts as an interface to the outside world, the Supervisor Agent coordinates the activity of the community, and the Specific Agents carry-on the detailed work associated with the SOHO subsystem.

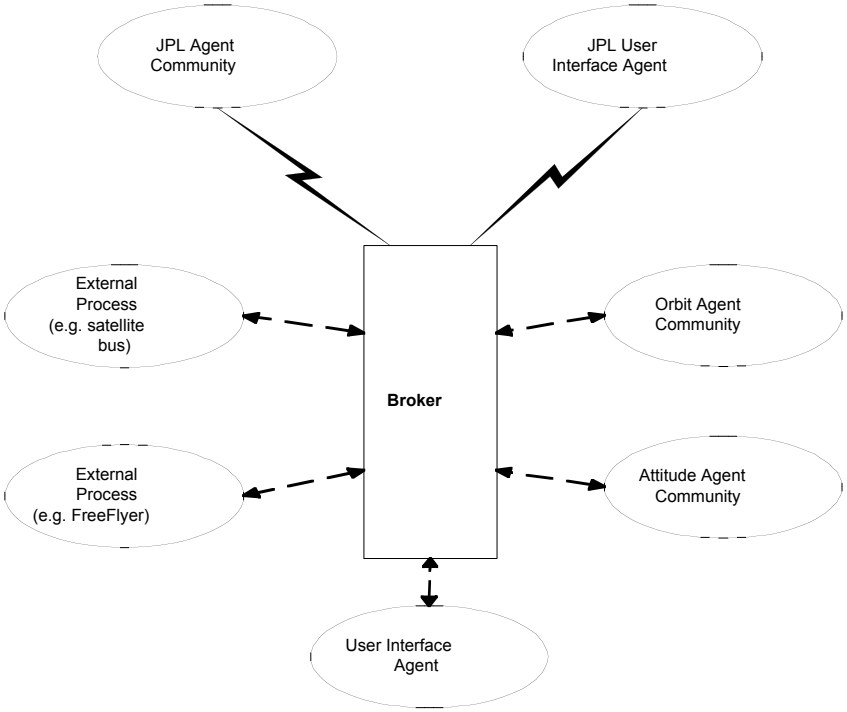

Figure 2. Overview of the initially-planned SOHO community

Figure 2. illustrates the planned infrastructure for the SOHO agent community residing between GSFC and JPL.

SOHO Agents in General

For the SOHO work an agent is considered as an entity described in terms of common sense modalities such as beliefs, or intentions. Intuitively, each agent represents an independent problem solver. Accordingly, an agent has a general problem solving mechanism and more specialized problem oriented capabilities. The agent has also to possess communication capabilities that make it able to communicate adequately with other agents. Agents are deliberative, in the sense that they use their knowledge in support of their reasoning.

The agent structure we use has been conceptually inspired by Shoham's agent-based programming approach [Shoham 1995], with some additions from the BDI (Beliefs, Desires, Intentions) model [Rao 1995], although the practical realization might differ in some aspects. The main idea is an analogy to those conceptual categories proved useful in designing the agent control architecture. It is assumed that the agent has to accomplish complex tasks that require higher level abstractions extracted from its behavior in relation with its environment, and *with other agents*.

The agent-based framework includes the agent state, and an interpreted programming language used to define agents. The agent state is defined by four basic components:

- *beliefs*: the information that the agent has about the environment and about some other agents;
- *capabilities*: the tasks that are supposed to be accomplished by the agent under given circumstances;
- *commitments*: the tasks that the agent is committed to achieve (usually communicated to another agent) at a particular time;

- *intentions*: the decisions about how to act in order to fulfill its commitments. This concept is necessary because it is assumed that the agent wouldn't be able in general to achieve all its commitments (desires).

It is assumed that this set of concepts are necessary to model agent behavior such as: deliberation, reactivity, interaction, and are flexible enough to be used at different levels of the agent architecture to describe the agent state. To be realized, the above mentioned concepts have to be represented by data structures in the agent architecture. Besides what was already mentioned, to complete the architecture, the agent might also have a repository of recipes (plans or rules) which specify the courses of actions that may be followed by the agent to achieve its intentions. The beliefs are updated from observations made of the world and as the effect of the interactions with other agents, generating new capabilities and commitments on the basis of new beliefs, and selecting from the set of currently active commitments some subset to act as intentions. An action has to be selected based on then agent's current intentions and knowledge (beliefs plus plans/rules).

The agent behavior is expressed in a declarative *agent language*, used by an interpreter that controls the agent execution cycle. The agent execution cycle consists of the following steps: processing new messages, determining which rules are applicable to the current situation, executing the actions specified by these rules, updating the mental model in accordance with these rules, and planning new actions.

In order to achieve *coordination* in a multi-agent system, the agents might have to negotiate, and they have to exchange information, i.e. they need to communicate. In multi-agent systems, the possible solutions to the communication range from no communication at all, ad hoc agent communication languages, and standard agent communication languages.

The de-facto agent communication language, used in the existing multiagent systems is KQML (Knowledge Query and Manipulation Language) [Finin 1992]. KQML is both a language and a protocol. It can be viewed as being comprised of three layers: a content layer, a message layer and a communication layer. The content layer is the actual content of the message, in a particular representation language. KQML can carry any representation language, including logic languages, ASCII strings, etc. The communicative actions such as asking for information, making something true about the environment, passing information to another agent, passing information to all agents are expressed using KQML performatives: ask, achieve, tell, broadcast, etc.

The agents have the ability to participate in more than one interaction with another agent at the same time. The structure to manage separate conversations is the protocol. The *protocol* provides additional structured contextual information to disambiguate the meaning of the message. In these conversations, an agent can play different roles depending on the context of the tasks or subjects. A role is the defined pattern of communications of an agent when implementing a protocol. An agent can assume several roles, when appropriate, depending on the protocol. A protocol state diagram can describe a protocol. Transitions in the state of a protocol represent changes in the state of a protocol, i.e. communications between agents. Defining a transition translates into creating a template for a KQML message that will be sent and received by the agents implementing the roles. The various fields in a KQML message will then be used by the message sending and handling rules in the appropriate agents. Inter-agent communication is regulated by intercommunication protocols.

The agents are heterogeneous and distributed. This means that they reside on different platforms and have different domains of expertise. In the current implementation, the interoperability is achieved by the system infrastructure rather than by the agents themselves. Agent interoperability is realized at two levels: knowledge level and interoperability mechanisms represented by the current distributed objects middleware (i.e., CORBA, RMI).

At the knowledge level the agents share their knowledge. They do this by interacting at run-time, based on a common set of models, grouped in a shared ontology. The substrate of this process is the concept of virtual knowledge; each agent must comply with the knowledge base abstraction when it communicates with other agents. This technology allows the agents to interact at the knowledge level, independently of the format in which the information is encoded.

The ontology represents the formal description of the model of the application domain. The model specifies the object in the domain, operations on them and their relationship. For a given application, several ontologies might be defined. For instance, ontology for navigation and control, a telecom ontology and so forth. Ontology can be represented as a semantic network, and has been implemented as a library of shared objects.

We chose to use AgentBuilder, [AgentBuilder, 1999] an integrated toolkit for constructing intelligent software agents. The agents have built-in capabilities for autonomous operation, monitoring their environments, reasoning, and communicating with other agents and users. The interactions between agents are defined by agent protocols. AgentBuilder, using specific protocol generation tools can automatically generate the protocols. The generated rules have to be further completed with more specific information. The protocols are implemented as behavioral rules, which encode the preconditions and actions, associated with state transitions in the state transition diagram of a protocol.

Current project Specifics.

Phase I will include an agent community to monitor the orbit determination process. The orbit determination process uses ground station measurements to determine where the spacecraft is in the orbit in a standard coordinate system. Agents will be used to monitor the input data, processing, and output products.

The Orbit Determination Agent sub-community is a ground-based agent community. It can:

- Detect and resolve tracking data errors
- Detect and correct errors in dynamics modeling
- Detect and report unplanned spacecraft maneuvers
- Generate alerts for errors which the sub-community cannot resolve
- Provide orbit information to other agents in community
- Monitor the generation and distribution of orbit products

In order to achieve the desired functionality of the orbit determination sub-community the specific agents included in that sub-community are as follows:

- Input Data Agent (monitors scheduling and data quality)
- Process Agent (monitors Kalman Filter process)
- Solution Agent (monitors the output)
- Distribution Agent (monitors product distribution)
- Supervisor Agent (maintains overall cognizance of the sub-community)
- Gateway Agent (interfaces with external processes)

The following table shows which agents will be needed to support the orbit determination functions.

	Gateway	Input	Process	Solution	Distribution	Supervisor
Interface with external processes (e.g. Orbit Determination Server)	X					
Detect and report tracking data scheduling errors	X	X				X
Detect and resolve tracking data errors	X	X				
Detect and resolve errors in the dynamics modeling	X		X			
Detect and report unplanned spacecraft maneuvers	X		X			
Generate alerts for errors the agent cannot resolve	X	X	X	X	X	

Provide orbit information to other agents in the community				X		X
Provide orbit information to other communities of agents (e.g. attitude)	X			X		
Monitor the generation and distribution of orbit products	X			X	X	

Table 1. Agent/Functionality Matrix

The agent community will be extended to include the attitude determination process for Phase II. The attitude determination process uses spacecraft telemetry (e.g. Earth/Sun sensor) to calculate the orientation of the spacecraft. Agents will be used to monitor the attitude determination process and take corrective action. Operator actions will be monitored to allow the agents to learn the trends in telemetry that cause anomalies.

The Attitude Agent sub-community will be a ground-based agent community. It will:

- Detect and resolve telemetry errors
- Detect and correct errors in dynamics modeling
- Detect and report unplanned spacecraft events
- Generate alerts for errors the agent cannot resolve
- Provide attitude information to other agents in community

Conclusion

This GSFC/JPL cooperative activity will not only provide a new degree of autonomy for SOHO operations but will contribute to the advancement of our understanding of agent community dynamics. Much will be learned from this experience.

References

1. [SOHO Web Page]: http://sohowww.estec.esa.nl/

2. [AgentBuilder 1999]: User's Guide, Reticular Systems, Inc., 1999.

3. [Finin 1992] Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McGuire, J., McKay, D., Shapiro, S., Pelavin, R., Beck, C.: Specification of the KQML Agent Communication Language (Official Document of the DARPA Knowledge Sharing Initiative's External Interfaces Working Group), Technical Report 92-04, Enterprise Integration Technologies, Inc., Menlo Park, California, 1992.

4. [Jennings 1998] Jennings, N. R., Sycara, K., Woolridge, M.: A Roadmap of Agent Research and Development. In Autonomous Agents and Multi-Agent Systems, 1, 275-306, Kluwer Academic Publishers, Boston, 1998.

5. [Rao 1995] Rao, A., Georgeff, M.: BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, June 1995.

6. [Shoham 1995] Shoham, Y.: CSLI Agent-oriented Programming Project: Applying software agents to software communication, integration and HCI (CSLI home page), Stanford University, Center for the Study of Language and Information, 1995.
7. [Tambe 1995] Tambe, M., Johnson, W., L., Jones, R., M., Ross, F., Laird, J., E., Rosenbloom, P., S., Schwamb, K.: Intelligent Agents for Interactive Simulation.