

Autonomy for SOHO Ground Operations

Walt Truskowski, NASA Goddard Space Flight Center (GSFC) Walt.Truskowski@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, Caltech Jet Propulsion Laboratory (JPL) sanda.mandutianu@jpl.nasa.gov

Abstract

The SOLAR and HELIOSPHERIC OBSERVATORY (SOHO) project [1] 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.

SOHO is designed to study the internal structure of the Sun, its extensive outer atmosphere and the origin of the solar wind. That view of the Sun is achieved by operating SOHO from a permanent vantage point 1.5 million kilometers sunward of the Earth in a halo orbit. SOHO is commanded from NASA's Goddard Space Flight Center (GSFC) in Greenbelt, Maryland (USA). Its data is retrieved via the NASA Deep Space Network (DSN) and routed to the Experimenters' Operations Facility (EOF) located at GSFC. There the SOHO experimenters will be able

to display on their workstations the images and measurements that are being produced by their instruments. From the EOF the experimenters will point the instruments aboard SOHO to a particular region of the Sun, or change the operating mode of the instruments. The SOHO scientists will use their instruments very much as an observer would at a ground-based observatory.

SOHO instruments produce a data stream of 200 kilobits per second, that can be transmitted continuously to the DSN stations of Goldstone (USA), Canberra (Australia) and Madrid (Spain), when each is visible from SOHO due to the daily rotation of the Earth.

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 subsystem. Figure 1 gives a generic view of a typical SOHO sub-community.

The Orbit Determination Problem

For real-time operations and mission planning purposes, the orbit of the spacecraft needs to be known. Orbit determination processes use theoretical models to relate observations over time in order to determine the position and velocity of the spacecraft

as a function of time. In the future, one might dynamically adjust the orbit to meet some conditions. Orbit determination is used by the navigation to maintain and control the orbit. The output of the procedure depends on the equation of motion used in the evaluation process. The equations of motions will be approximations due to modeling errors and computing approximations. The same equations of motions can be used to extrapolate in time. Because of the inherent errors, the tracking data interval as well as the extrapolation interval are limited. The orbit is generally determined at regular intervals.

There are three aspects of the orbit determination problem:

- the source and type of data,
- the algorithm for modeling the orbit,
- the computer system which processes the observation.

The observations used for orbit determination can be obtained by:

- tracking from the ground,
- tracking from space,
- autonomous systems on the spacecraft.

Accurate orbit determination using ground stations usually requires several passes. Observations are repeated during a transit of the spacecraft over the observation site, and the data collected is transmitted to a computing facility for processing. Observations from several tracking sites are usually necessary in order to minimize errors that are correlated with the location of the tracking station. Ground systems operate on historical data and therefore use propagated orbits.

When one attempts to determine an orbit for an object, one will always find some difference between the “observed” positions, and the “computed” positions (computed using the orbit having been determined). These differences are known as “residual errors”, or “residuals”. Ideally, they should represent the random errors in observation that inevitably happen in the real world. Estimation algorithms use different techniques, including Kalman filtering.

For orbit determination we are using [®]FreeFlyer [2]. This is a comprehensive orbital dynamics suite of software products developed by the a.i.solutions [3]. The main characteristics of [®]FreeFlyer are:

- automated orbit determination that can be optimized for real-time and product generation;

- precise measurement model supporting a wide variety of antenna types;
- tracking data processor that supports a wide range of tracking data types;
- advanced Kalman filter that automatically adapts to un-modeled error sources;
- extensive solved-for capability including tracking data, antenna, and satellite biases;
- automated tracking data editing including pattern recognition and sigma editing;
- graphical Display and Editing of Tracking Data.

SOHO Agents Model

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 [4], with some additions from the BDI (Beliefs, Desires, Intentions) model [5], 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) [6]. 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 defined as a 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.

For the initial prototype we chose to use AgentBuilder [7], 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

SOHO Multi-agent System Architecture

The multi-agent system supporting the SOHO autonomous operations has been conceived as surrogate spacecraft subsystem analysts (agents) that can intelligently interface with their human counterparts in spacecraft management activities. This SOHO application will be developed in three phases: the Orbit Agent community, Attitude Agent

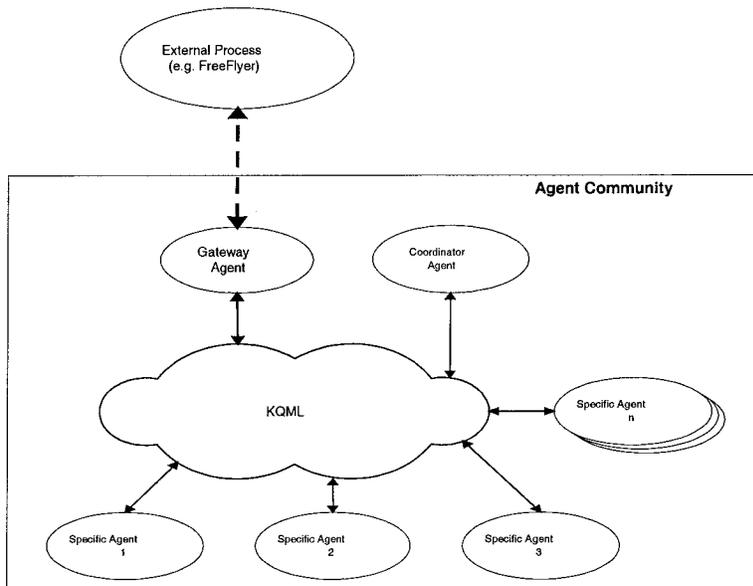


Figure 1. Generic View of a SOHO Agent Sub-Community

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.

community, and Telemetry Agent community. Other communities will be subsequently added for other spacecraft subsystems.

The implementation of an entirely ground-based guidance and control navigation system is both expensive and straightforward. The main reason to consider autonomous guidance and navigation is to reduce mission cost and risk. Another effect could be to extend the mission life, or to undertake missions which would not be possible without onboard autonomy.

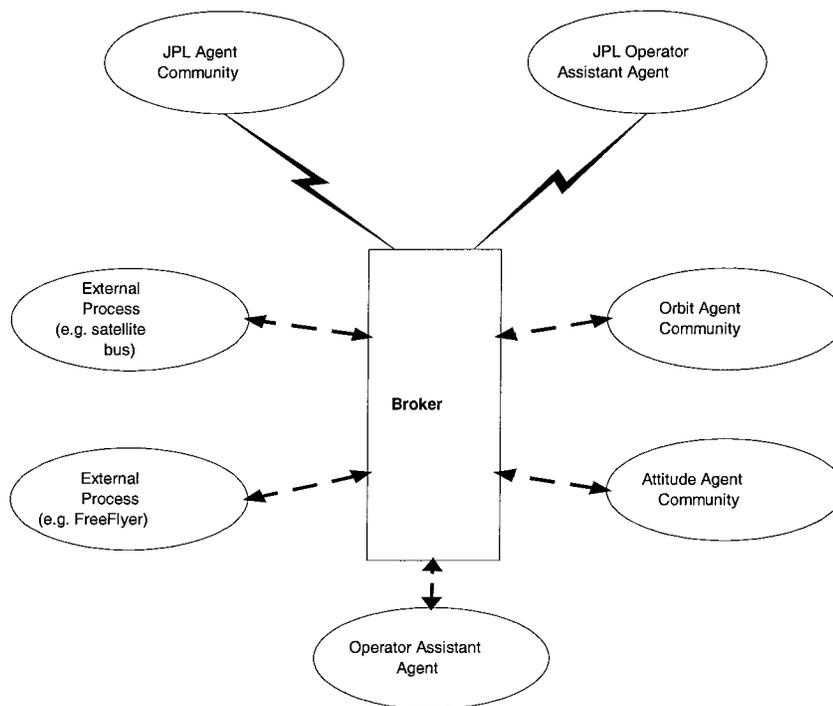


Figure 2. Overview of the initially-planned SOHO community

The *Orbit Agent Community* is a ground-based agent community. The agents within this community control the orbit determination process at the highest level. It has the following tasks:

- Detect and resolve tracking data errors
- Detect and correct errors in dynamics modeling
- Detect and report unplanned spacecraft maneuvers
- Generate alerts for errors the agent cannot resolve
- Provide orbit information to other agents in community
- Generate and distribute orbit products

The *Attitude Agent 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
- Generate and distribute attitude products

The *Telemetry Agent Community* will be an onboard agent community. It will:

- Detect and resolve scheduling errors
- Monitor recorder for hardware errors
- Detect and reject abnormal command sequences
- Predict possible hardware failures based on physical conditions

We have decided to use fine-grained agents and develop communities because we want to be able to demonstrate progressive autonomy concepts. This demo system is available via the web. We plan on demonstrating the migration of agents into the planned communities.

The *Orbit Determination Coordinator Agent* coordinates all high-level activities within the orbit determination community. It monitors the overall process from getting tracking data, processing it for orbit determination and distributing the products of the orbit determination to subscribed customers. Its knowledge include the delivery schedule of the tracking data and the delivery location of the tracking data.

The *Orbit Determination Process Agent* monitors the actual process of orbit determination. It is responsible of the Kalman filtering process and can detect and

obtain more accurate results. The agent is also able to make decisions about the orbit determination solutions that pertain to their age or the exact process

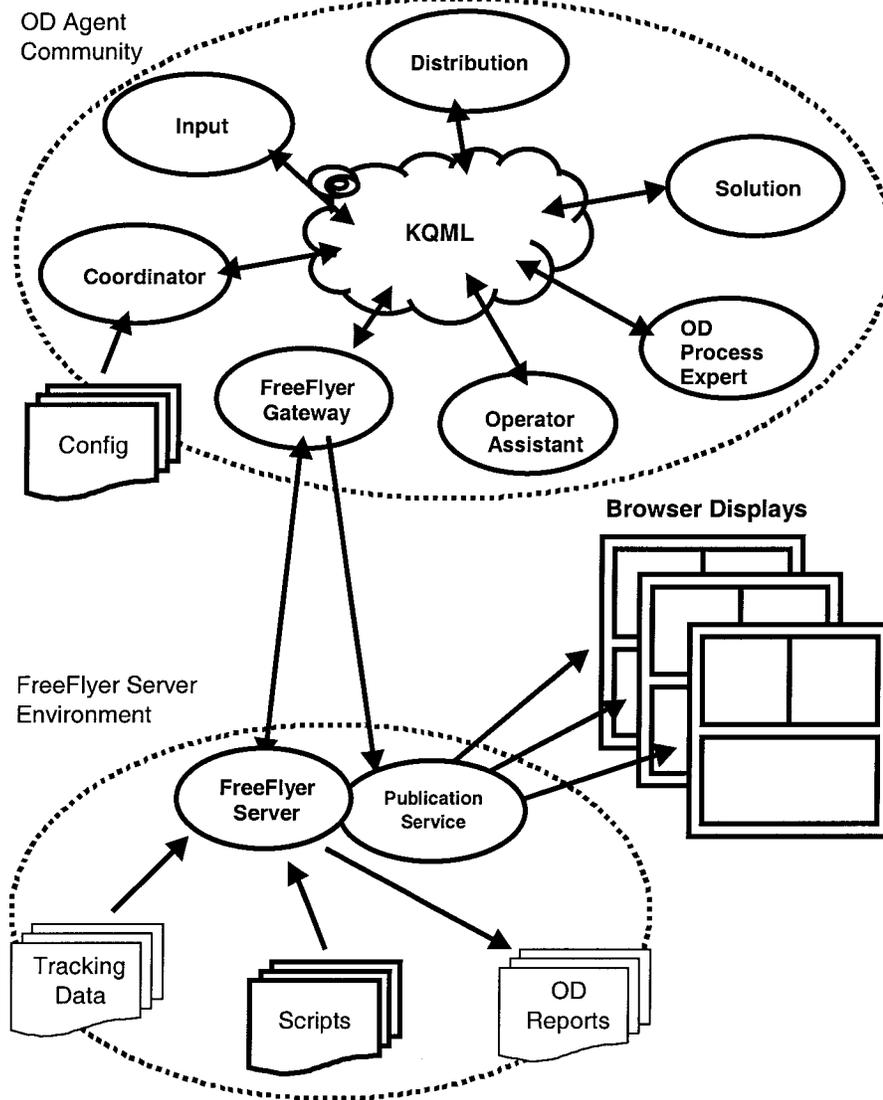


Fig. 3 Orbit Determination Community

handle certain process faults. For instance, when range residuals become abnormal, it can enable adaptive filtering and reprocess the data. If this is successful, it can inform about the problem and its resolution, for traceability and possible further analysis. If the agent cannot resolve the problem, it can choose to page a human operator.

The *Solution Agent* keeps track of the quality of the orbit determination solution. The solution knowledge for the spacecraft being monitored accumulates in its knowledge base. This knowledge is mainly used to

by which they were arrived at.

The *FreeFlyer Gateway Agent* provides the interface to the external FreeFlyer server. It abstracts the services of the [®]FreeFlyer software at the higher level, as required by the agent community communication protocols.

The *Input Agent* monitors the inputs to the orbit determination process. If tracking data doesn't arrive on time or is of poor quality, this agent can notify the community.

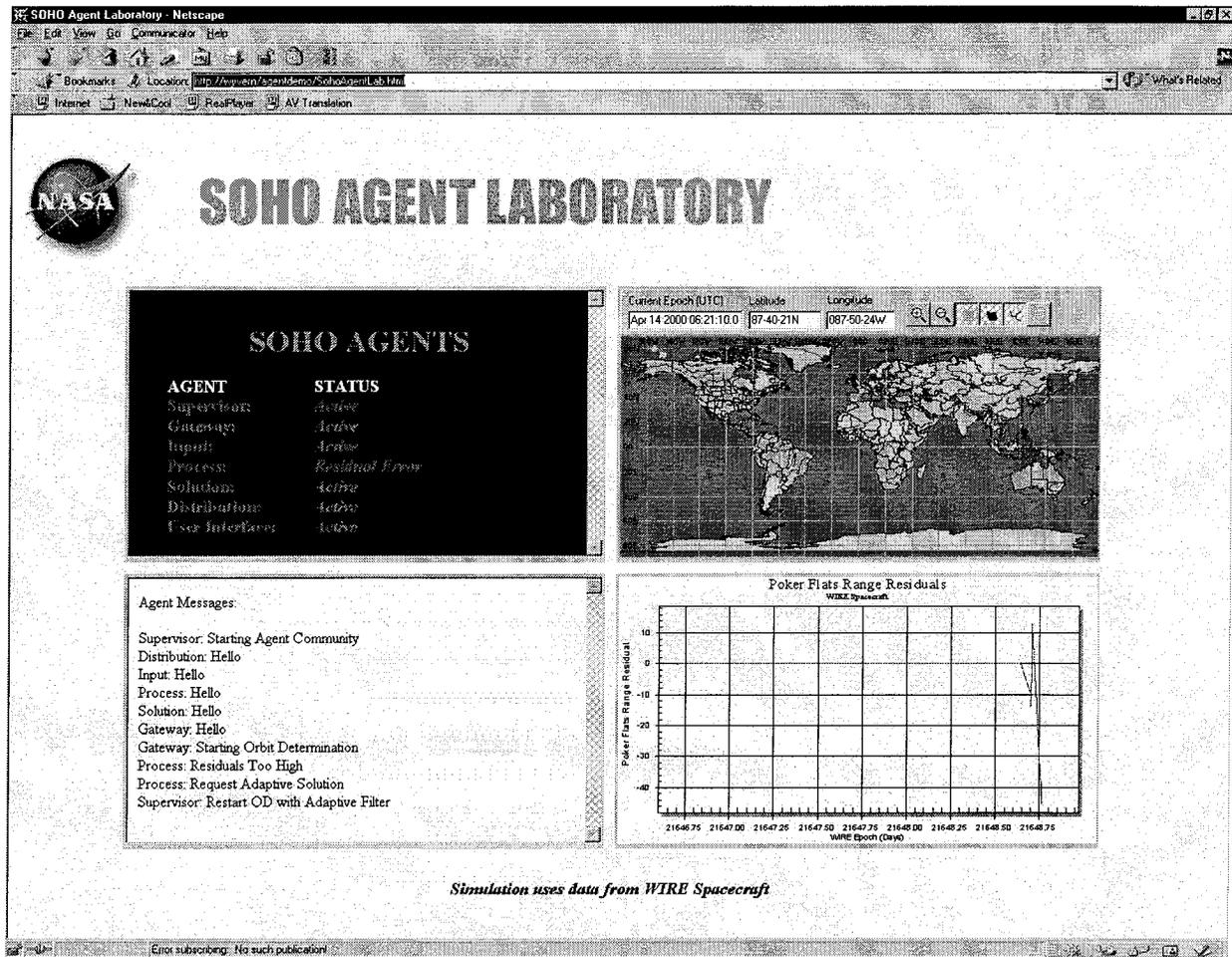


Fig. 4 A SOHO Prototype Screen

The *Distribution Agent* responsibilities are the generation of orbit determination products and the subsequent delivery to external entities. A typical orbit determination product could be an ephemeris file needed by another agent community or an external user. An ephemeris file is a time history of a spacecraft's position and velocity. This agent controls processes associated with the delivery of the products and detects failures and attempts corrective actions and request operator intervention where appropriate.

The *Operator Assistant Agent* acts as a liaison between the agent community and the human operators. Its responsibilities are: monitor the overall orbit determination process, ensures the appropriate feedback to the operator, facilitates the transmission of the operator's commands.

Final Remarks and Future Work

A generic architecture for a multi-agent system for SOHO mission operations has been presented. A prototype for the Orbit determination community has been currently implemented. The approach described in this paper has proven to be an efficient, flexible and robust way to manage the complexity of the problem of orbit determination. The tools used to build the prototype, i.e. Agent Builder, have well served the purposes of a rapid prototyping development cycle. We are currently investigating agent building FIPA-compliant tools for further development, such as JADE [8].

Orbit and attitude sensing are inherently related. In many cases, they are using the same sensors and the same actuators. The attitude control sensors control the spacecraft during orbit maneuvers. In addition,

orbit and attitude requirements are often combined to produce pointing and mapping results to satisfy mission requirements. Our goal should be to reduce the total cost and risk of attitude and orbit determination.

Traditionally attitude determination and control and the guidance and navigation systems are separate functions, realized by different systems. Getting these functions to work together constitutes one of the main challenges of developing autonomous navigation and orbit control, and therefore, fully autonomous spacecraft. The planned addition of the other agent communities will help to address this integration.

We also plan to develop the requirements and operational scenario for a Reusable Agent Repository (RAR). The RAR will support the cost-effective development of agent-based communities to support future mission operations systems. A generic version of the agents developed for the SOHO multi-agent system will be used to initially populate the RAR.

Acknowledgements

This work has been performed by the agent group at the Goddard Space Flight Center, and a.i.solutions in collaboration with Caltech-Jet Propulsion Laboratory, under contract with the National Aeronautics and Space Administration.

References

1. SOHO, <http://sohowww.estec.esa.nl/>
2. ®FreeFlyer, <http://www.ai-solutions.com>
3. a.i. solutions, <http://www.ai-solutions.com/freelyer/index.asp>
4. 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.
5. 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. 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.

7. AgentBuilder- User's Guide, Reticular Systems, Inc., 1999.
8. JADE, <http://sharon.csel.it/projects/jade/>