The Techsat-21 Autonomous Sciencecraft Constellation Demonstration

Steve Chien, Rob Sherwood, Michael Burl, Russell Knight, Gregg Rabideau,
Barbara Engelhardt, Ashley Davies, Jet Propulsion Laboratory
(firstname.lastname@jpl.nasa.gov)
Paul Zetocha, Ross Wainright, Pete Klupar, Air Force Research Laboratory, Kirtland
Pat Cappelaere, Interface and Control Systems
Derek Surka, Princeton Satellite Systems
Brian Williams, Massachusetts Institute of Technology
Ronald Greeley, Arizona State University
Victor Baker, James Doan, University of Arizona

Keywords: planning, autonomy, scheduling, model-based reasoning

Abstract

The Autonomous Sciencecraft Constellation flight demonstration (ASC) will fly onboard the Air Force's TechSat-21 constellation (an unclassified mission scheduled for launch in 2004). ASC will use onboard science analysis, replanning, robust execution, model-based estimation and control, and formation flying to radically increase science return by enabling intelligent downlink selection and autonomous retargeting. Demonstration of these capabilities in a flight environment will open up tremendous new opportunities in planetary science, space physics, and earth science that would be unreachable without this technology.

1 Introduction

There is an increasing desire in many organizations, including NASA and the DoD, to use constellations or fleets of autonomous spacecraft working together to accomplish complex mission objectives. The Air Force Research Laboratory (AFRL) has initiated the TechSat-21 program to serve as a proof of concept mission for a new design paradigm for space missions. This paradigm seeks to reduce costs and increase system robustness and maintainability by distributing functionality over several micro-satellites flying in formation. The distributed functionality includes processing, command and control, communications, and payload functions. A chief objective is for the system of micro-satellites to in effect function as a "virtual" satellite, which can be controlled and tasked as a single satellite.

TechSat-21 is scheduled for a late 2004 launch and will fly three satellites in a near circular orbit at an altitude of 600 Km and will accommodate a one-year mission lifetime. (See Figure 1.) During the mission lifetime the cluster of satellites will fly in various configurations with relative separation distances of approximately 100 meters to 5 Km. One of the objectives of TechSat-21 is to assess the utility of the space-based, sparse-array aperture formed by the satellite cluster. For TechSat-21, the sparse array will be used to synthesize a large radar antenna. Three modes of radar sensing are planned: synthetic aperture radar (SAR) imaging, moving target indication (MTI), and geo-location.

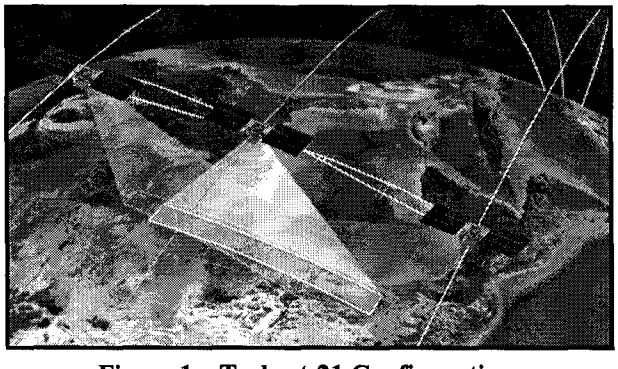

Figure 1 – Techsat-21 Configuration

The ASC onboard flight software includes several autonomy software components:

- *Onboard science algorithms* that will analyze the image data, generate derived science products, and detect trigger conditions such as science events and change relative to previous observations

- *Model-based mode identification and recovery* using the Burton system that will enable timely identification and reconfiguration of system state
- *Robust execution management software* using the SCL package to enable event-driven processing low-level autonomy
- The CASPER *continuous planner* that will replan activities, including downlink, based on science observations in the previous orbit cycles.
- The ObjectAgent *cluster management software* will enable the three techsat-21spacecraft to perform high precision formation flying to form a single virtual instrument

The onboard science algorithms will analyze the images to extract static features and detect changes relative to previous observations. Prototype software has already been demonstrated on X-band radar data (from shuttle missions) to automatically identify regions of interest including: regions where change such as flooding, ice melt, and lava flows as well as feature recognition such as crater and volcano recognition. Such onboard science will enable retargeting and search, e.g., shifting the radar aimpoint on the next orbit cycle to identify and capture the full extent of a flood. Onboard science analysis would enable capture of short-lived science phenomena at the finest time-scales without overwhelming onboard caching or downlink capacities. Examples include eruption of volcanoes on Io, formation of jets on comets, and phase transitions in ring systems. Generation of derived science products (e.g., boundary descriptions, catalogs) and change-based triggering will also reduce data volumes to a manageable level for extended duration missions that study long-term phenomena such as atmospheric changes at Jupiter and flexing and cracking of the ice crust on Europa.

The onboard planner (CASPER) will generate mission operations plans from goals provided by the onboard science analysis module. The model-based planning algorithms will enable rapid response to a wide range of operations scenarios based on a deep model of spacecraft constraints, including faster recovery from spacecraft anomalies. The onboard planner will accept as inputs the science and engineering goals and ensure high-level goal-oriented behavior for the constellation.

The robust execution system (SCL) accepts the CASPER-derived plan as an input and expands the plan into low-level commands. SCL monitors the execution of the plan and has the flexibility and knowledge to perform event-driven commanding to enable local improvements in execution as well as local responses to anomalies. Burton performs model-driven estimation of spacecraft state and also accepts configuration goals from SCL. The ObjectAgent cluster management software manages the maneuver planning and execution for the constellation. It accepts high-level constellation formation goals from CASPER and plans and executes these formations to support science (e.g. radar imaging) and engineering (e.g. downlink) activities.

To further illustrate the ASC concept, consider a volcano observation scenario that involves monitoring of lava flows in Hawaii. SIR-C radar data have been used in ground-based analysis to study this phenomenon. The ASC concept would be applied as follows:

(1) Select volcano region
(2) Radar image volcano
(3) Form reflectivity image
(4) Compare image with previous image (which may simply be a comparison of previous boundary position, if that was all that was returned)
(5) Determine area of new flow
(6) Downlink imagery of identified "new flow" areas; alternatively downlink a higher-level characterization of the flow

A more aggressive example would involve re-prioritize observation for high-resolution and/or repeat imaging. As demonstrated by this scenario, onboard science processing and spacecraft autonomy enable focus of mission resources onto science events such that the most interesting science data is downlinked. In this case, a large number of high priority science targets can be monitored and only the most interesting science data (during times of change and focused on the areas of change) need be downlinked.

Current status of the ASC concept is that it has been baselined for flight on the Techsat-21 mission, with the following development schedule. Key design reviews occur in Spring 2001 to Spring 2002, with final delivery of the spacecraft and software in September 2003. Nominal launch date is September 2004. The NASA New Millennium Space Technology 6 Project has selected the ASC concept for a Phase-A award.

2 Onboard Science Algorithms

2.1 Science Experiments

The ASC demonstration will be grounded in a range of Earth-science based experiments that have direct relevance to space science missions throughout the Solar System. The common foundation for these experiments will be the following elements:

- Onboard conversion of raw data to derived science products
- Detection of trigger conditions including recognition of both static features and dynamic change
- Reactions based on detected triggers

Onboard conversion of raw data to derived science products constitutes a form of intelligent data compression. Examples of derived products include compact descriptions of terrain boundaries (the outline of a flooded region or a fresh lava flow), histograms of heights, slopes, or reflectivities (volcano hypsometry), vector field descriptions of flows (wind vectors, ice migration), catalogs of features including type, location, and size information (lakes, sand dunes), and image data of regions of interest, particularly regions where change has occurred or which stand out from the local surroundings (flooding, lava, forest regrowth). We note that downlinking such summarizations is complementary with traditional data compression techniques.

Significant "events" will be identified both from single images and from pairs of images of the same area taken at different times. Examples of triggers defined on a single image include detection of region boundaries (coastlines, glacier edges), recognition of specific types of features (volcanoes, lakes, icebergs), and identification of generic, interesting regions that differ from the local surroundings. Examples of triggers defined based on change between a pair of images include detection of differences in the number of pixels of a particular type (water versus soil to identify flooding, new lava versus old lava to identify recent volcanism) and detection of changes in region descriptors such as size, centroid, boundary, or histogram.

We note that at the present time, the planned onboard experiments do not include interferometry using the data collected by the satellites in the constellation, due to the prohibitive amount of processing necessary and the limited inter-satellite bandwidth

A number of reactions to detected trigger conditions are possible, ranging from very conservative to very aggressive. Example reactions include prioritizing data for downlink, transmitting derived science products rather than complete raw images, discarding data when appropriate to achieve science goals, retargeting observations, and reconfiguring the constellation.

2.2 Onboard Science Software

There are two components of the onboard science software, the *image formation module* and the *onboard science algorithms*. The image formation module will be responsible for forming a (possibly reduced resolution) SAR image onboard the spacecraft from the raw phase history (demodulated I and Q returns). Previous NASA missions that have included SAR instruments have simply downlinked the raw phase histories and performed image formation on the ground. In our experiments, we only need to form a few images per orbit cycle (in contrast to a global mapping mission such as Magellan); hence, the necessary processing can be carried out onboard.

The TechSat-21 radar subsystems are single polarization (transmit and receive horizontal) and can be operated in both spotlight and strip-map modes. The radar transmits a sequence of linear frequency modulated (LFM) chirps centered in X-band. Range compression is achieved by de-modulating the returned pulses (shifting to baseband) and FFTing. In spotlight mode, the azimuth compression, which enables the cross-range position of scatterers at a given range to be determined based on doppler shift, also reduces to FFT processing. Thus, the complexity of spotlight processing is approximately that of a 2-D FFT: to obtain an n pixel by n pixel image, the complexity is $O(n^2 \log(n^2))$. By processing only a portion of the chirp bandwidth and a shorter aperture, a reduced resolution image (in both range and cross-range) can be obtained with reduced processing requirements. In strip-map mode, the azimuth compression is more complicated, but is also well understood and computationally feasible.

For TechSat-21, the altitude of the spacecraft will be 600 km and antenna beamwidth will be on the order of 1 degree leading to an approximately 15 km diameter spot size (dependent of course upon grazing angle). Order of magnitude calculations indicate that a 10-meter by 10-meter resolution image can be formed with 600 million real operations. On the 33-MHz RISC payload processor, this workload translates to 18 seconds or 0.3 minutes per image. For 2-meter resolution, the image formation process would require approximately 45 minutes, which is considerably less than the 90 minute orbit decision cycle for downlink.

Once the image has been formed, the *onboard science algorithms* can then analyze the SAR image(s) to create derived science products and detect trigger conditions, such as change relative to a previous orbit cycle. For example, fresh lava and old lava have very different backscatter properties; thus, new lava flows can be easily detected and localized. Likewise, water has very different backscatter characteristics than soil, enabling detection of flooding.

We are currently investigating several methods of converting images into derived science products. The derived products will in effect be summarizations that are significantly more compact than the raw image (or phase history) data. Intensity and texture-based segmentation (in common use for ground-based processing) will be evaluated for effectiveness in generating terrain boundary descriptions and region summarizations (e.g., a flooded region will be described by an average radar cross-section and a polyline outlining its boundary). Statistical pattern recognition techniques [1] [3] will be used to identify and produce catalogs of specific types of features such as volcanoes, lakes, and iceberg fragments. (See Figure 2.) Recently developed discovery techniques [2] will also be applied to identify "interesting" regions that differ from their local background leading to a compact description of an image in terms of subimage patches and locations.

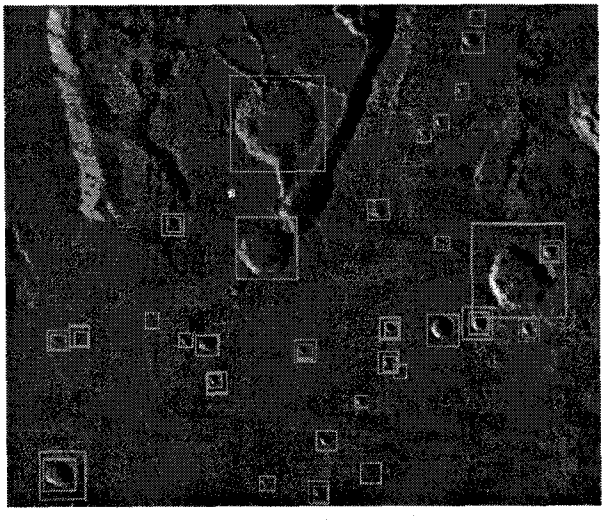

Figure 2 - Automatically identified craters in a Viking Mars Image

In addition to calculations based on a single image, the onboard science analysis software will include change detection algorithms that compare images of the same region taken at different times. The change detection capability is particularly relevant for capture of short-term events at the finest time-scale resolutions without overwhelming onboard caching systems and for compressing long-term "monitoring" observations in which changes are infrequent. For space science missions, example applications include tracking atmospheric changes on Jupiter, Neptune, or Triton (from optical image data), tracking ice plate movement on Europa, monitoring known (and identifying new) volcanoes on Io, capturing fine time-scale events such as jet formation on comets or phase transitions in ring systems, and detecting new cratering on planets and moons.

To detect change, we will test for statistically significant differences in derived descriptors such as region sizes, locations, boundaries, and histograms, as well as in the raw pixel data. The latter case is complicated by the need to insure that the two images are approximately co-registered. In part, the orbit repeatability and small absolute positional uncertainty of the TechSat-21 group will help insure approximate co-registration. If necessary, feature-based or image-based registration methods [Quakefinder] will be applied to fine-tune the registration. Also, since the magnitude of change necessary to initiate a trigger event can be specified as a parameter, some degree of robustness to image misalignment will be built in. For change detection, radar observations have the advantage that the illumination, target, and receiver geometry remains basically the same from pass to pass. (In optical imagery, irrelevant change caused by sun position complicates the change detection problem.) Figure 3 contains X-SAR radar images indicating lava flow on Kilauea volcano, Big Island, Hawaii. This is the type of change detection that our algorithms will perform onboard Techsat-21.

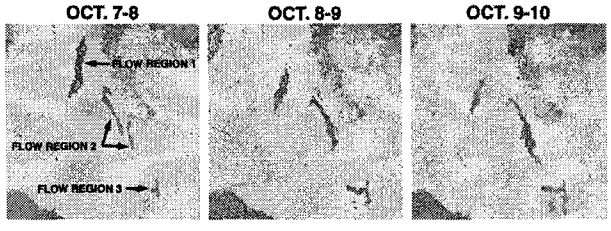

Figure 3 - Hawaii Lava Flows

All of the algorithms described scale linearly in the number of image pixels. Hence, image resolution can be selected appropriately to insure that computational and memory requirements fit within the onboard processing capabilities. For example, a previous study of the recognition algorithm in [1] indicates complexity on the order of 250 operations per pixel to reliably detect a particular type of Venus small shield volcano in the Magellan SAR data. Using this figure as a baseline, we could process approximately 10^5 pixels per second on the 33-MHz payload processor.

Detection of the image- and change-based triggers described here will enable a range of automated spacecraft reactions. On the conservative end of the spectrum, triggers can be used to prioritize data for downlink. For example, regions in which change was detected may be downlinked first (with TechSat-21, it will take a full four days to downlink the entire onboard cache of the three spacecraft). Early access to the "interesting data" would be especially valuable to the project scientist, potentially enabling a request for modification of the original observation plan.

A slightly more aggressive use of the trigger information involves actually "discarding data". For example, if nothing significant has changed in a region, don't bother to downlink that data. Although the scientist would never like to discard data, the realities of a finite onboard cache and constrained downlink bandwidth will sometimes force a discard to satisfy the primary objective. For example, if the science goal is to capture the fine temporal details of jet formation on a comet, the onboard cache will quickly overflow unless older data that doesn't contain the desired event is discarded or degraded to lower resolution.

A third, more aggressive, but potentially extremely rewarding, use of the trigger information that we will demonstrate onboard TechSat-21 is to autonomously retarget observations based on the trigger. For example, if an image indicates flooding in a region, subsequent orbits will employ the planner to close the loop onboard and use a modified radar aimpoint in an attempt to capture the full scope of the flooding. Similarly, since many geological features are spatially clustered (e.g., volcano fields, hydrothermal vents), detection of some features will be used to seed a broad area search (e.g., using the three spacecraft radars in a coordinated effort to look in the surrounding area for additional instances).

3 Robust Execution and Model-based Health Monitoring

Interface and Control Systems is providing to Techsat-21 an end-to-end integrated flight and ground system to support messaging, command & control, fault detection, isolation, and recovery for the constellation of nano-satellites.

A publish/subscribe software bus allows the distribution of notification messages to the registered listeners. The distribution of messages can be effected across the various namespaces. Each satellite has its own namespace or software bus. Applications can also make requests. Other applications or agents can register as handlers for those requests allowing a very loose coupling between all the agents. Dynamic messages are supported to allow for future growth as ever-smarter agents are added to the constellation in different satellites.

The SCL "smart" executive supports the command and control function. Users can define scripts in an English-like manner. Compiled on the ground, those scripts can be dynamically loaded onboard and executed at an absolute or delta time. This is equivalent to the traditional procedural approach to spacecraft operations based on time. Those scripts will also be scheduled autonomously by CASPER, the JPL onboard planner as new observations or goals are detected by the science processing agents, Object Agent and SCL in a cooperative manner using the messaging system.

Spacecraft telemetry from all satellites is gathered onboard and fed into the integrated expert system. Significant change in the data will trigger user-defined rules. Those rules can be used for fault detection, isolation and recovery (FDIR). In that case, rules can be used to execute recovery scripts. Another application of rules is for mission constraint checking to prevent operator errors or, more simply, command pre-processing.

The system will be enhanced in several areas by the addition of L2, a derivative of Livingstone that flew on DS1. L2 is a model-based diagnostic and control system provided by NASA Ames for determining Mode Identification and Mode Recovery (MI/MR). L2 has been tightly integrated with the SCL rule-based system. This leverages the best capabilities of each system. L2 receives notifications of commands and relevant observations. It can detect deviations from the expected behavior defined by the model. Nominal and fault transition modes are dispatched to the expert system which may have rules attached to trigger on mode changes. Complex fault recovery scripts can then be executed autonomously onboard the satellite. The scripts can be reloaded dynamically as ground operators learn about the system. TheL2 model will be enhanced from DS1 to model the behavior of the autonomous agents such as CASPER, OA and even SCL. The L2 model can expect the execution of specific scripts and wait for specific observables. This addresses a general problem of monitoring software components as well as hardware components and a specific problem seen on DS1, where a failure in the Smart Executive went undetected by Livingstone, and left the spacecraft's Ion Propulsion System (IPS) on longer than expected. Thus software failures, as well as hardware failures, will therefore be detected and recovered by L2.

L2 will be used for Mode Recovery and limited reactive planning. Using the model of the physical system, the commands and observations received over time, L2 can determine not only the likely states of the system but also the actions required to move the system into a desired configuration. This capability will only be used to formulate short plans to reconfigure the system under very specific circumstances defined by the model.

One advantage of using SCL in tandem with L2 is in the ability of mapping commands with scripts, modes with SCL database records. Mode changes will trigger rules that will execute complex scripts to

potentially reconfigure the system or safe the payload and/or the spacecraft. Since scripts are fairly small and can be reloaded very easily, the model does not have to change onboard to support the cluster manager functionality. Those same scripts can be used from the ground, by CASPER or L2 in a seamless manner.

ICS is also providing a graphical ground environment to assist in the development of the script/rules and even the modeling of the state transitions using UML. Fault trees can be easily created to support the fault management function. Scripts and rules are automatically generated with the proper documentation. All the information is encoded in SML (Spacecraft Markup Language) an XML subset to allow for rapid data interchange and publication among contractors on the web.

4 CASPER Planning Software

Traditionally, the majority of planning and scheduling research has focused on a batch formulation of the problem. In this approach, when addressing an ongoing planning problem, time is divided up into a number of planning horizons, each of which lasts for a significant period of time. When one nears the end of the current horizon, one projects what the state will be at the end of the execution of the current plan (see Figure 4). The planner is invoked with: a new set of goals for the new horizon, the expected initial state for the new horizon, and the planner generates a plan for the new horizon. As an example of this approach, the Remote Agent Experiment operated in this fashion [5].

This approach has a number of drawbacks. In this batch oriented mode, typically planning is considered an off-line process, which requires considerable computational effort, and there is a significant delay from the time the planner is invoked to the time that the planner produces a new plan. If a negative event occurs (e.g., a plan failure), the response time until a new plan is generated may be significant. During this period the system being controlled must be operated appropriately without planner guidance. If a positive event occurs (e.g., a fortuitous opportunity, such as activities finishing early), again the response time may be significant. If the opportunity is short lived, the system must be able to take advantage of such opportunities without a new plan (because of the delay in generating a new plan). Finally, because the planning process may need to be initiated significantly before the end of the current planning horizon, it may be difficult to project what the state will be when the current plan execution is complete. If the projection is wrong the plan may have difficulty.

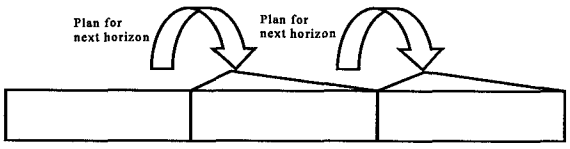

Figure 4 - Traditional Batch "Plan then Execute" Cycle

To achieve a higher level of responsiveness in a *dynamic planning* situation, we utilize a *continuous planning* approach and have implemented a system called CASPER (for Continuous Activity Scheduling Planning Execution and Replanning) [4]. Rather than considering planning a batch process in which a planner is presented with goals and an initial state, the planner has a current goal set, a plan, a current state, and a model of the expected future state. At any time an incremental update to the goals, current state, or planning horizon (at much smaller time increments than batch planning)[1] may update the current state of the plan and thereby invoke the planner process. This update may be an unexpected event or simply time progressing forward. The planner is then responsible for maintaining a consistent, satisficing plan with the most current information. This current plan and projection is the planner's estimation as to what it expects to happen in the world if things go as expected. However, since things rarely go exactly as expected, the planner stands ready to continually modify the plan. From the point of view of the planner, in each cycle the following occurs:

- Changes to the goals and the initial state first posted to the plan,
- Effects of these changes are propagated through the current plan projections (including conflict identification)
- Plan repair algorithms [6] are invoked to remove conflicts and make the plan appropriate for the current state and goals

This approach is shown in Figure 5. At each step, the plan is created by using incremental replanning from:

- The portion of the old plan for the current planning horizon;
- The change (Δ) in the goals relevant for the new planning horizon;
- The change (Δ) in the state; and
- The new (extended) planning horizon

[1] For the spacecraft control domain we are envisioning an update rate on the order of tens of seconds real time.

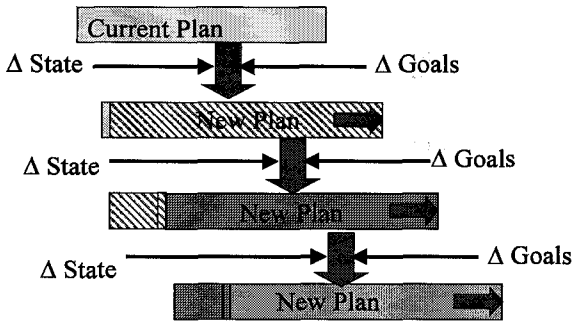

Figure 5 - Continuous Planning Incremental Plan Extension

5 ObjectAgent Cluster Management

Several new missions being proposed by NASA call for constellations or fleets of autonomous spacecraft working together to accomplish complex mission objectives. Some of the many advantages of using distributed satellite systems include greater performance, lower cost, and improved fault tolerance, reconfigurability and upgradability. Coordinating the activities of all the satellites in a constellation is not a trivial task, however.

Princeton Satellite Systems (PSS) is developing the ObjectAgent and TeamAgent systems under two Phase II SBIR contracts from the Surveillance and Control Division of the Air Force Research Laboratory's Space Vehicles Directorate to address this issue. The primary objective is to create an agent-based software architecture that is designed for autonomous, distributed systems and that is easy to use.

The ObjectAgent system is an agent-based real-time architecture for distributed, autonomous control. Control systems are decomposed into agents, each of which is a multi-threaded process. Agents are used to implement all of the software functionality and communicate via a flexible messaging architecture. Each message has a content field written in natural language that is used to identify the purpose of the message and its contents. Agents may be loaded at any time and have the capability to configure them when launched, which simplifies the process of updating flight software and removes the complexity associated with software patches. They will automatically seek out other agents who can provide them with the inputs they need. Decision-making and fault detection and recovery capabilities are also built-in at all levels.

The ObjectAgent software package provides a graphical user interface (GUI) based development environment for the design and simulation of multi-agent systems. This design environment simplifies the agent creation process and provides a common interface to a number of advanced control and estimation techniques. The Agent Developer GUI is shown in Figure 6.

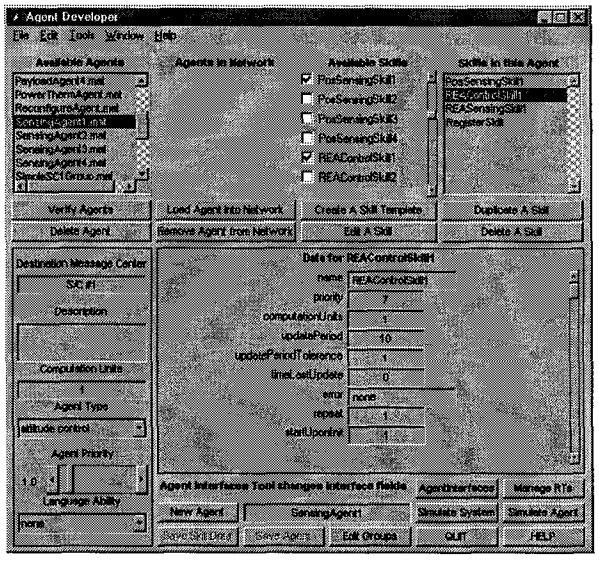

Figure 6 - ObjectAgent GUI

The TeamAgent system applies ObjectAgent to the problem of controlling multiple cooperative satellites. TeamAgent enables agent-based multi-satellite systems to fulfill complex mission objectives by autonomously making high- and low-level decisions based on the information available to any and/or all agents in the satellite system. The required spacecraft functions for the multiple spacecraft missions have been identified and the use of software agents and multi-agent based organizations to satisfy these functions have been demonstrated [7] [8]. Simulations of multi-agent systems for multiple satellites have been developed using TeamAgent to illustrate

collision avoidance and reconfiguration for a four-satellite constellation. Agents were used to monitor for collisions, reconfigure the fleet, optimize fuel usage across the cluster during reconfiguration, and develop a fuel optimal maneuver for reconfiguration.

During the first phase of development, ObjectAgent and TeamAgent were prototyped in Matlab. A complete, GUI-based environment was developed for the creation, simulation, and analysis of multi-agent, multi-satellite systems. ObjectAgent and TeamAgent are presently being ported from Matlab to C++ for implementation on a real-time system [9].

The current architecture is designed to run on top of Enea's OSE operating system. OSE is a message based operating system designed for distributed architectures with many features that facilitate. The current baseline processor is the PowerPC 750.

ObjectAgent/TeamAgent is scheduled to fly on the Air Force's TechSat-21 mission that will involve three satellites flying in formation and acting as a "virtual" satellite. ObjectAgent/TeamAgent will be used to build two elements of the flight software, the Cluster Manager and the Spacecraft Manager. The Cluster Manager is designed to perform relative control of the satellites in the cluster. This would include relative stationkeeping and estimation of the cluster center-of-mass and the relative positions of each satellite. This also includes cluster level commanding, health summarization, and fault detection. The Spacecraft Manager performs many of the functions that would normally reside on the ground including spacecraft level fault detection. The Spacecraft Manager manages above the spacecraft flight software while the Cluster Manager manages the Spacecraft Managers.

6 Conclusion

This important flight validation will enable radically different missions with significant onboard decision-making leading to novel science opportunities. The paradigm shift toward highly autonomous spacecraft will enable future NASA missions to achieve significantly greater science returns with reduced risk and cost.

7 Acknowledgement

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

8 References

[1] M.C. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, J. Aubele, and L. Crumpler, "Learning to Recognize Volcanoes on Venus," *Machine Learning Journal,* April 1998.

[2] M.C. Burl and D. Lucchetti, "Autonomous Visual Discovery", *SPIE Aerosense Conference on Data Mining and Knowledge Discovery,* (Orlando, FL), April 2000.

[3] M.C. Burl, W.J. Merline, E.B. Bierhaus, W. Colwell, C.R. Chapman, "Automated Detection of Craters and Other Geological Features," *iSAIRAS*, (2001).

[4] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran , "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling," *SpaceOps*, Toulouse, France, June 2000.

[5] A. Jonsson, P. Morris, N. Muscettola, K. Rajan, and B. Smith, "Planning in Interplanetary Space: Theory and Practice," *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge*, CO, April 2000.

[6] G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations in the ASPEN System," *International Symposium on Artificial Intelligence Robotics and Automation in Space*, Noordwijk, The Netherlands, June 1999.

[7] T.P. Schetter, M. E. Campbell, and D. M. Surka, "Comparison of Multiple Agent-based Organizations for Satellite Constellations," *2000 FLAIRS AI Conference*, Orlando, Florida, May 2000.

[8] T.P. Schetter, M. E. Campbell, and D. M. Surka, "Multiple Agent-Based Autonomy for Satellite Constellations," *Second International Symposium on Agent Systems and Applications*, Zurich, Switzerland, September 2000.

[9] Surka, D. M., M. C. Brito, and C. G. Harvey, "Development of the Real-Time ObjectAgent Flight Software Architecture for Distributed Satellite Systems," To be Presented *at IEEE Aerospace Conference*, Big Sky, Montana, March 2001.