A Demonstration of Robust Planning and Scheduling in the Techsat-21 Autonomous Sciencecraft Constellation

Steve Chien[1], Rob Sherwood[1], Michael Burl[1], Russell Knight[1], Gregg Rabideau[1], Barbara Engelhardt[1] Ashley Davies[1], Paul Zetocha[2], Ross Wainsright[2], Pete Klupar[2], Pat Cappelaere[3], Derek Surka[4], Brian Williams[5], Ronald Greeley[6], Victor Baker[7], James Doan[7]

[1]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA {firstname.lastname}@jpl.nasa.gov
[2] Air Force Research Laboratory, Kirtland, Albuquerque, NM, USA, [3]Interface and Control Systems, Melbourne, FL, USA
[4]Princeton Satellite Systems, Plainsboro, NJ, USA, [5]Massachusetts Institute of Technology, Cambridge, MA, USA
[6]Arizona State University, Tempe, AZ, USA, [7]University of Arizona, Tucson, AZ, USA

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. We offer a demonstration of the planning, scheduling, and execution framework for this application.

1 Introduction

Robust planning, scheduling, execution, and mode identification in real environments is a challenging task. In general, each of these elements is difficult in their own right, and the fusion of these can be equally challenging. We offer a demonstration of the integration of each of these in the real task of flying a constellation of sciencecraft. Specifically, we demonstrate planning, scheduling, re-planning, science analysis, execution, and mode identification for the Techsat-21 mission. Our demonstration includes nominal operations as well as operations with anomalies. Our system is capable of generating its own science goals based on previous information-gathering activities. In general, the system as a whole provides considerable autonomy. The rest of this document describes the specific mission and its background, as well as our approach to addressing the challenges we face in flying such a mission.

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 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 (See Figure 1). The primary mission is one-year in length with the possibility for an extended mission of one or more additional years. 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.

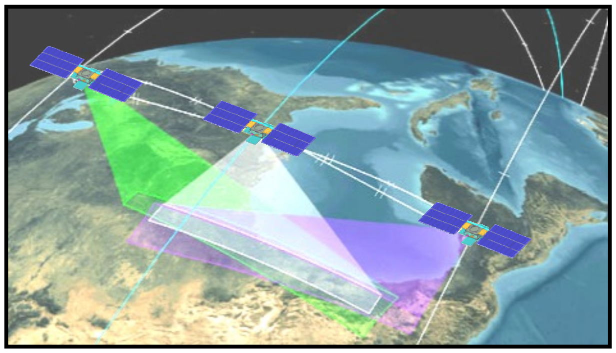
Figure 1 – Techsat-21 Configuration

The principal processor onboard each of the three TechSat-21 spacecraft is a BAE Radiation hardened 175 MIPS, 133MHz PowerPC 750 running the OSE 4.3 operating system from Enea Systems. OSE was chosen because it is inherently message passing based and particularly suitable for distributed applications. Each satellite will have 256 Kbytes of EEPROM for boot loads and 128 Mbytes of SDRAM. Communications will be through a Compact PCI bus. For storage of payload data and some large flight software components 8 disk drives per spacecraft will be used.

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, "interesting" features, and change relative to previous observations
- *Model-based mode identification and execution (MI-R)* that uses component-based hardware models to analyze anomalous situations and to generate novel command sequences and repairs.
- *Robust execution management software* using the Spacecraft Command Language (SCL) package to enable event-driven processing and low-level autonomy
- The Continuous Activity Planning, Scheduling, and Replanning (CASPER) *planner* that will replan activities, including downlink, based on science observations in the previous orbit cycles
- The ObjectAgent and TeamAgent *cluster management software* will enable the three Techsat-21 spacecraft to autonomously perform maneuvers and 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 aim-point on the next orbit cycle to identify and capture the full extent of a flood. Onboard science analysis would also enable capture of short-lived science phenomena at the finest time-scales without overwhelming onboard caching or downlink capacities. Examples include of this 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. Livingstone 2 performs model-driven estimation of spacecraft state and the Burton also accepts configuration goals from SCL. The ObjectAgent and TeamAgent 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.

One of the ASC demonstration scenarios 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) Initially, ASC has a list of science targets to monitor.
(2) As part of normal operations, CASPER generates a plan to monitor the targets on this list by periodically imaging them with the radar.
(3) During such a plan, a spacecraft images the volcano with its radar.
(4) Onboard, a reflectivity image is formed.
(5) The *Onboard Science Software* compares the new image with previous image and detects that the lava field has changed due to new flow. Based on

this change the science software generates a goal to acquire a new high-resolution image of an area centered on the new flow.

(6) The addition of this goal to the current goal set triggers CASPER to modify the current operations plan to include numerous new activities in order to enable the new science observation. During this process CASPER interacts with ObjectAgent to plan required slews and/or maneuvers.

(7) SCL executes this plan in conjunction with several autonomy elements. Mode Identification assists by continuously providing an up to date picture of system state. Reconfiguration (Burton) achieves configurations requested by SCL. And ObjectAgent and TeamAgent execute maneuvers planned by CASPER and requested at run-time by SCL.

(8) Based on the science priority, imagery of identified "new flow" areas; are downlinked. This science priority could have been determined at the original event detection o based on subsequent onboard science analysis of the new image.

As demonstrated by this scenario, onboard science processing and spacecraft autonomy enable the focus of mission resources onto science events so 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.

The ASC concept has been selected for flight on the Techsat-21 mission and the necessary software is currently being matured and brought into flight readiness. Key Techsat-21 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

There are two components of the onboard science software, the *image formation module* and the *onboard science algorithms*. The image formation module forms a (possibly reduced resolution) SAR image onboard the spacecraft from the raw phase history (demodulated I and Q returns). In the ASC mission concept, 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. Our baseline calculations estimate that forming a 15 km diameter spot size (dependent upon grazing angle) 10-meter by 10-meter resolution image can be formed can be formed in at best 18 seconds (with full processor utilization). A 2-meter resolution would require approximately 45 minutes. Both these timescales are 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 demonstrating 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 specific types of features such as volcanoes, lakes, and iceberg fragments. Figure 2 shows results from a prototype lava cone recognition algorithm under development for ASC flight. Output from such a module could be used to downlink higher resolution data around items of interest or by dowlinking a summary catalog of the interesting features. 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.

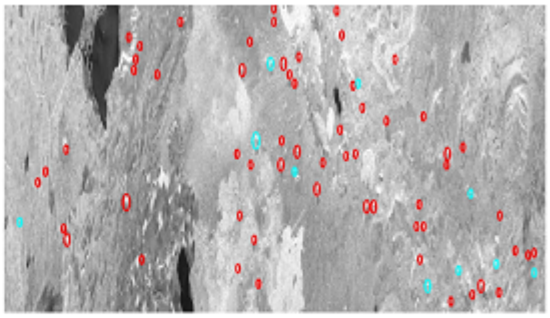

Figure 2 - Automatically identified lava cones in X-SAR image of Lava Beds National Monument, CA

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 ensure 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. 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 successive X-SAR radar images indicating lava flow on Kilauea volcano, Big Island, Hawaii. The changes in the highlighted areas of the image are indicative of lava flow that occurred in between images. This is the type of change detection that our algorithms will perform onboard Techsat-21.

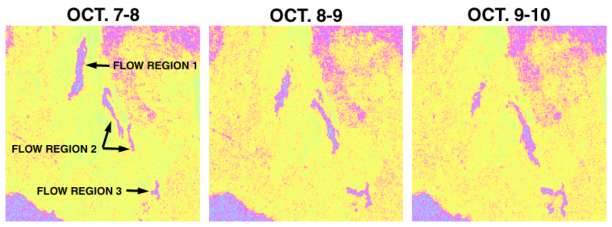

Figure 3 - Hawaii Lava Flows

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 PowerPC 750 flight 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, exclude that region from the downlink. 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. 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 aim-point 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).e.

3. Robust Execution

Techsat-21 will fly The Spacecraft Command Language (SCL) to provide robust execution. SCL is a software package that integrates procedural programming with a real-time, forward-chaining, rule-based system. A publish/subscribe software bus allows the distribution of notification and request messages to integrate SCL with other onboard software. This enables either loose or tight coupling between SCL and other flight software as appropriate. Dynamic messages are supported to allow for future growth as ever-smarter software 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. Ground-based absolute time script scheduling is equivalent to the traditional procedural approach to spacecraft operations based on time. In the ASC concept, SCL scripts will also be planned and scheduled by the CASPER onboard planner. The science processing agents, cluster management software, and SCL work in a cooperative manner to generate new goals for CASPER. These goals are sent with 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.

SCL is a mature software product, and has successfully flown on Clementine I and ROMPS. SCL has also been used in a wide range of ground-based control and operations contexts. As such it represents a good basis for integrating the multiple ASC autonomy functions: onboard science, mode identification and reconfiguration, planning, and constellation management.

4 Model-based Monitoring and Reconfiguration

CASPER generates a mission level plan that includes a sequence of behavior goals, such as producing thrust. An executive is responsible for reducing these goals to a control sequence, for example, which opens the relevant set of valves leading to a main engine. A device, such as a valve, is commanded indirectly; hence, the executive must ensure that the components along the control path to the device are healthy and operating before commanding that device. Components may be faulty, and redundant options for achieving a goal may exist; hence, an executive must ascertain the health state of components, determine repair options when viable, and select a course of action among the space of redundant options.

Interpreting sensor information and generating sequences to handle a breadth of novel situations requires extensive reasoning about physical processes and state changing actions. A model-based executive performs this reasoning automatically from an onboard model [13]. In particular, a model-based executive is given a model of the spacecraft hardware, including a set of component models and a schematic that describes component interconnections. The executive uses the model to track planner goals, confirm hardware modes, reconfigure hardware, generate command sequences, detect anomalies, isolate faults, diagnose, and perform repairs.

A model-based executive receives a stream of hardware configuration goals and sensor information. It uses sensor information to infer the state of the hardware and then continually tries to transition the hardware towards a state that satisfies the current configuration goals. The model-based executive is reactive in the sense that it reacts immediately to changes in goals and to failures, that is, each control action is incrementally generated using the new observations and goals given in each state.

A model-based executive uses its model to determine the desired control sequence in three stages --- mode estimation (ME), mode reconfiguration (MR) and model-based reactive planning (MRP). ME and MR setup the planning problem, identifying initial and target states, while MRP reactively generates a plan solution. More specifically, MI incrementally generates the set of most likely state trajectories of the hardware that are consistent with the hardware model and the sequence of observations and control actions. This is maintained as a set of most likely current states. MR uses the hardware model and the most likely current state generated by ME to determine a reachable hardware state that satisfies the current goal configuration. MRP then generates the first action in a control sequence for moving from the most likely current state to the target state. After that action is performed ME confirms that the intended next state is achieved.

Model-based reactive planning is traditionally viewed as intractable for real world problems. We address this intractability through a set of model-compilation, causal analysis and online policy construction methods. The result is a model-based reactive planner that is sound and complete. It generates the first control action of a valid plan in average case constant time, and compensates for anomalies at every step. Finally, it will not generate irreversible, potentially damaging sequences except to effect repairs.

Our model-based execution framework is an enhanced version of the Burton system, described in [14] and under development at the MIT AI and Space Systems laboratories. This framework incorporates the Mode Estimation capabilities of Livingstone 1 and 2, described in [8,13] and developed at NASA Ames.

The marriage between the model-based executive and SCL provides a powerful hybrid execution capability with an expressive scripting language and an extensive capability to generate novel responses to anomalous situations.

5 Onboard Missing Planning

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 3).) 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 [7].

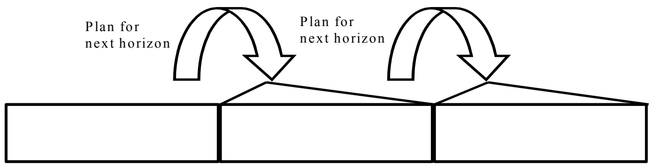

Figure 3: Traditional Batch Plan then Execute Cycle

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.

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:

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

- 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 [5] are invoked to remove conflicts and make the plan appropriate for the current state and goals

This approach is shown in Figure 4. 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 (D) in the goals relevant for the new planning horizon;
- The change (D) in the state; and
- The new (extended) planning horizon

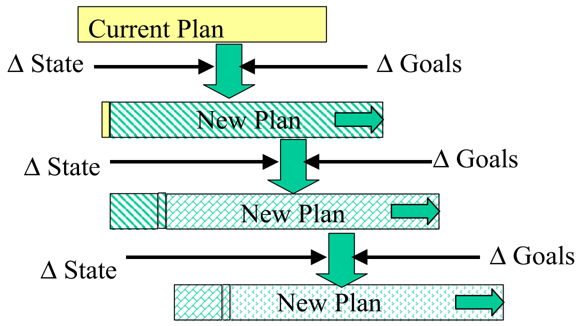

Figure 4 - Continuous Planning Incremental Plan Extension

In the ASC concept, CASPER is responsible for long-term mission planning in response to both science goals derived onboard as well as anomalies. In this role, CASPER must plan and schedule activities to achieve science and engineering goals while respecting resource and other spacecraft and constellation operations constraints. For example, when change is detected in an image, CASPER plans a response. If it is appropriate to take a more detailed image of the change area, CASPER will modify the operations plan to include the necessary activities to re-image. If this includes changing the formation of the constellation, the cluster manager will be consulted. Other required activities, such as callibration of the radar, acquisition of the image, and subsequent science processing are all planned by CASPER.

6 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, ability to reconfigure, and ability to upgrade. However, Coordinating the activities of all the satellites in a constellation is a challenging task. Princeton Satellite Systems (PSS) is developing the ObjectAgent(OA) and TeamAgent(TA) systems to create an easy to use agent-based software architecture for autonomous, distributed systems.

In ObjectAgent, 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. Agents 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. ObjectAgent also 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 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 [6]. 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.

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 includes relative station-keeping, estimation of the cluster center-of-mass, and estimation of the relative positions of each satellite. This also includes cluster level commanding, health summarization, and fault detection as relating to the attitude and orbit of the spacecraft and constellation. 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.

These ObjectAgent and TeamAgent functions encompass both plan time and execution capabilities. At plan time, CASPER consults OA and TA on the feasibility and resource requirements to perform formation changes, maneuvers, and slews. At execution time, formation changes, maneuvers, and slews planned by CASPER and requested by SCL are performed by OA and TA. In this execution time function OA and TA perform closed loop control in their use of lower-level attitude and control software to achieve the desired goals.

Current efforts involve transitioning this demonstrated software from workstations to the Techsat-21 flight software environment (OSE Operating system on PowerPC 750s). The OSE operating system is a message passing OS designed for distributed architectures and thus facilitates the distributed agent based architecture.

7 Related Work and Conclusions

In 1999, the Remote Agent experiment (RAX) [10] executed for a few days onboard the NASA Deep Space One mission. RAX demonstrated a batch onboard planning capability but did not demonstrate onboard science. RAX also included an earlier version of the Livingstone and Burton mode identification and fault recovery software.

PROBA[12] is a European Space Agency (ESA) mission that will be demonstrating onboard autonomy. PROBA will be launching in 2001.

The Three Corner Sat (3CS) University Nanosat mission will be using the CASPER onboard planning software integrated with the SCL ground and flight execution software [6]. The 3CS mission is scheduled for launch in late 2002. 3CS will use onboard science data validation, replanning, robust execution, and multiple model-based anomaly detection. The 3CS mission is considerably less complex than Techsat-21 but still represents an important step in the integration and flight of onboard autonomy software.

ASC will fly on the Techsat-21 mission will demonstrate an integrated autonomous mission using onboard science analysis, replanning, robust execution, model-based estimation and control, and formation flying. ASC will perform intelligent science data selection that will lead to a reduction in data downlink. In addition, the

ASC experiment will increase science return through autonomous retargeting. Demonstration of these capabilities in onboard the Techsat-21 constellation mission 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.

Acknowledgement

Portions of this work were performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Other portions were performed at the Massachusetts Institute of Technology Space Systems and Artificial Intelligence Laboratories, under contracts from AFOSR and the DARPA Mobies program.

References

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.

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

M.C. Burl, W.J. Merline, E.B. Bierhaus, W. Colwell, C.R. Chapman, "Automated Detection of Craters and Other Geological Features *Intl Symp Artificial Intelligence Robotics & Automation in Space*, Montreal, Canada, June 2001

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.

S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," *Proc Fifth International Conference on Artificial Intelligence Planning and Scheduling,* Breckenridge, CO, April 2000.

S. Chien, B. Engelhardt, R. Knight, G. Rabideau, R. Sherwood, E. Hansen, A. Ortiviz, C. Wilklow, S. Wichman " Onboard Autonomy on the Three Corner Sat Mission," *Intl Symposium on Artificial Intelligence Robotics and Automation in Space*, Montreal, Canada, June 2001

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

J. Kurien and P.P. Nayak, "Back to the Future for Consistency-based Trajectory Tracking," *Proc. National Conference on Artificial Intelligence*, Austin, TX, 2000.

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.

rax.arc.nasa.gov, Remote Agent Experiment Home Page.

D. Surka, J. Mueller, M. Brito, B. Urea, M Campbell, "Agent-based Control of Multiple Satellite Formation Flying," *Intl Symposium on Artificial Intelligence Robotics and Automation in Space*, Montreal, Canada, June 2001

The PROBA Onboard Autonomy Platform, http://www.estec.esa.nl/proba/

B.C. Williams and P.P. Nayak, "A Model-based Approach to Reactive Self-Configuring Systems," *Proceedings of the National Conference on Artificial Intelligence*, Portland, Oregon, 1996.

B.C. Williams and P.P. Nayak, "A Reactive Planner for a Model-based Executive," *Proc. International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.