Lessons Learned from Autonomous Sciencecraft Experiment

Steve Chien, Rob Sherwood, Daniel Tran, Benjamin Cichy, Gregg Rabideau, Rebecca Castaño, Ashley Davies

Jet Propulsion Laboratory, California Institute of Technology

Dan Mandl, Stuart Frye[1], Bruce Trout[2], Jeff D'Agostino[3], Seth Shulman[4]

Goddard Space Flight Center

Darrell Boyer

Interface & Control Systems

Sandra Hayden[5], Adam Sweet[5], Scott Christa[6]

NASA Ames Research Center

Abstract.
An Autonomous Science Agent has been flying onboard the Earth Observing One Spacecraft since 2003. This software enables the spacecraft to autonomously detect and responds to science events occurring on the Earth such as volcanoes, flooding, and snow melt. The package includes AI-based software systems that perform science data analysis, deliberative planning, and run-time robust execution. This software is in routine use to fly the EO-1 mission. In this paper we briefly review the agent architecture and discuss lessons learned from this multi-year flight effort pertinent to deployment of software agents to critical applications.

1 Introduction

The Autonomous Sciencecraft Experiment (ASE) is currently flying autonomous agent software on the Earth Observing One (EO-1) spacecraft [19]. This software uses several integrated autonomy technologies to enable autonomous science. Multiple algorithms to detect the occurrence of science events based on remote sensing imagery analyze science data onboard. These algorithms are used to downlink science data only on change, and detect features of scientific interest such as volcanic eruptions, flooding, ice breakup, and presence of cloud cover. These onboard science algorithms are inputs to onboard decision-making algorithms that then modifies the spacecraft observation plan to capture high value science events. This new observation plan is then be executed by a robust goal and task oriented execution system, able to adjust the plan to succeed despite run-time anomalies and uncertainties. Together these technologies enable autonomous goal-directed exploration and data acquisition to maximize science return. This paper describes the Autonomous Sciencecraft Experiment (ASE) effort to develop and deploy the Autonomous Science Agent on the Earth Observing One spacecraft.

The ASE onboard flight software includes several autonomy software components:

- Onboard science algorithms that analyze the image data to detect trigger conditions such as science events, "interesting" features, changes relative to previous observations, and cloud detection for onboard image masking
- Robust execution management software using the Spacecraft Command Language (SCL) [10] package to enable event-driven processing and low-level autonomy
- The Continuous Activity Scheduling Planning Execution and Replanning (CASPER) [5] software that replans activities, including downlink, based on science observations in the previous orbit cycles

The onboard science algorithms analyze the images to extract static features and detect changes relative to previous observations. The software analyzes EO-1 Hyperion data to automatically identify regions of interest including land, ice, snow, water, and thermally hot areas. Repeat imagery using these algorithms can detect regions of change (such as flooding and ice melt) as well as regions of activity (such as lava flows). Using these algorithms onboard enables retargeting and search, e.g., retargeting the instrument on a subsequent orbit cycle to identify and capture the full extent of a flood. On future interplanetary space missions, onboard science analysis will enable capture of short-lived science phenomena. These can be captured at the finest time-scales without overwhelming onboard memory or downlink capacities by varying the data collection rate on the fly. 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 and resurfacing on Europa.

1 - MitreTek, 2 - Microtel LLC, 3 – the Hammers Company, 4-Honeywell, 5 – QSS Group, 6 - AerospaceComputing, Inc

The onboard planner (CASPER) generates mission operations plans from goals provided by the onboard science analysis module. The model-based planning algorithms enables 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 accepts as inputs the science and engineering goals and ensures high-level goal-oriented behavior.

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.

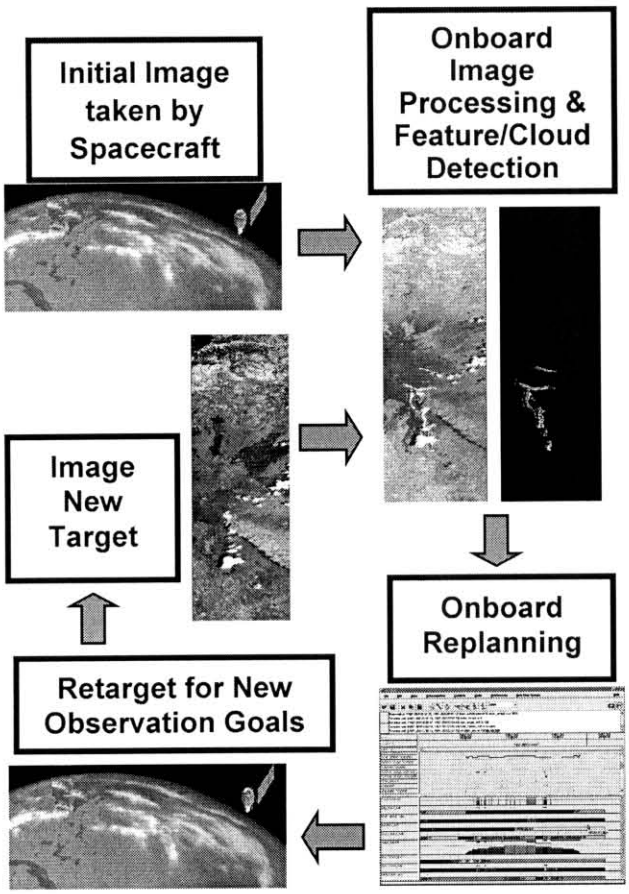

Figure 1. Autonomous Science Scenario

A typical ASE scenario involves monitoring of active volcano regions such as Mt. Etna in Sicily, Italy. ASE has already been used to perform similar demonstrations. The ASE concept is described as follows:

1. Initially, ASE has a list of science targets to monitor that have been sent as high-level goals from the ground.
2. As part of normal operations, CASPER generates a plan to monitor the targets on this list by periodically imaging them with the Hyperion instrument. For volcanic studies, the infra-red and near infra-red bands are used.
3. During execution of this plan, the EO-1 spacecraft images Mt. Etna with the Hyperion instrument.
4. The onboard science algorithms analyze the image and detect a fresh lava flow, or active vent. If new activity is detected, a science goal is generated to continue monitoring the volcanic site. If no activity is observed, the image is not downlinked.
5. Assuming a new goal is generated, CASPER plans to acquire a further image of the ongoing volcanic activity.
6. The SCL software executes the CASPER generated plan to re-image the site.
7. This cycle is then repeated on subsequent observations.

Building autonomy software for space missions has a number of challenges.

1. Limited, intermittent communications to the agent. A typical spacecraft in low earth orbit (such as EO-1) has 8 10-minute communications opportunities per day. This means that the spacecraft must be able to operate for long periods of time without supervision. For deep space missions the spacecraft may be in communications far less frequently. Some deep space missions only contact the spacecraft once per week, or even once every several weeks.
2. Spacecraft are very complex. A typical spacecraft has thousands of components, each of which must be carefully engineered to survive rigors of space (extreme temperature, radiation, physical stresses). Add to this the fact that many components are one-of-a-kind and thus have behaviors that are hard to characterize.
3. Limited observability. Because processing telemetry is expensive, onboard storage is limited, and downlink bandwidth is limited, engineering telemetry is limited. Thus onboard software must be able to make decisions on limited information and ground operations teams must be able to operate the spacecraft with even more limited information.
4. Limited computing power. Because of limited power onboard, spacecraft computing resources are usually very constrained. An average spacecraft CPUs offer 25 MIPS and 128 MB RAM – far less than a typical personal computer. Our CPU allocation for ASE on EO-1 is 4 MIPS and 128MB RAM.

5. High stakes. A typical space mission costs hundreds of millions of dollars, any failure has significant economic impact. The total EO-1 Mission cost is over $100 million dollars. Over financial cost, many launch and/or mission opportunities are limited by planetary geometries. In these cases, if a space mission is lost it may be years before another similar mission can be launched. Additionally, a space mission can take years to plan, construct the spacecraft, and reach their targets. This delay can be catastrophic.

In the remainder of this paper we describe the ASE software architecture and components. We then discuss how the issues of reliability and performance affected the software architecture. For a more in-depth discussion of the validation and testing process used for ASE see [17].

2 The EO-1 Mission

Earth Observing-1 (EO-1) is the first satellite in NASA's New Millennium Program Earth Observing series. EO-1 was launched on a Delta 7320 from Vandenberg Air Force Base on November 21, 2000. Its orbit allows for 16-day repeat tracks, with 3 over flights per 16-day cycle with a less than 10-degree change in viewing angle.

ASE uses the Hyperion hyper spectral instrument. The instrument typically images a 7.5 km by 42 km land area at 30m per pixel.

The EO-1 spacecraft has two Mongoose M5 processors. The first M5 is used for the EO-1 command and data handling functions. The other M5 is part of the WARP (Wideband Advanced Recorder Processor), a large mass storage device. Each M5 runs at 12 MHz (for ~8 MIPS) and has 256 MB RAM. Both M5's run the VxWorks operating system. The ASE software operates on the WARP M5. This provides an added level of safety for the spacecraft since the ASE software does not run on the main spacecraft processor.

3 The EO-1 Science Agent

The autonomy software on EO-1 is organized into a traditional three-layer architecture (See Figure 2.). At the highest level of abstraction, the Continuous Activity Scheduling Planning Execution and Replanning (CASPER) software is responsible for mission planning functions. CASPER schedules science activities while respecting spacecraft operations and resource constraints. The duration of the planning process is on the order of tens of minutes. CASPER scheduled activities are inputs to the Spacecraft Command Language (SCL) system, which generates the detailed sequence commands corresponding to CASPER scheduled activities. SCL operates on the several second timescale. Below SCL the EO-1 flight software is responsible for lower level control of the spacecraft and also operates a full layer of independent fault protection. The interface from SCL to the EO-1 flight software is at the same level as ground generated command sequences. The science analysis software is scheduled by CASPER and executed by SCL in batch mode. The results from the science analysis software result in new observation requests presented to the CASPER system for integration in the mission plan.

This layered architecture was chosen for two principal reasons:

1. The layered architecture enables separation of responses based on timescale and most appropriate representation. The flight software level must implement control loops and fault protection and respond very rapidly (within one second) and is thus directly coded in C. SCL must respond quickly (in seconds) nd perform many procedural actions. Hence SCL uses as its core representation scripts, rules, and database records. CASPER must reason about longer term operations, state, and resource constraints. Because of its time latency, it can afford to use a mostly declarative artificial intelligence planner/scheduler representation. CASPER is able to respond within 10s of minutes.
2. The layered architecture enables redundant implementation of critical functions – most notable spacecraft safety constraint checking. In the design of our spacecraft agent model, we implemented spacecraft safety constraints in all levels where feasible.

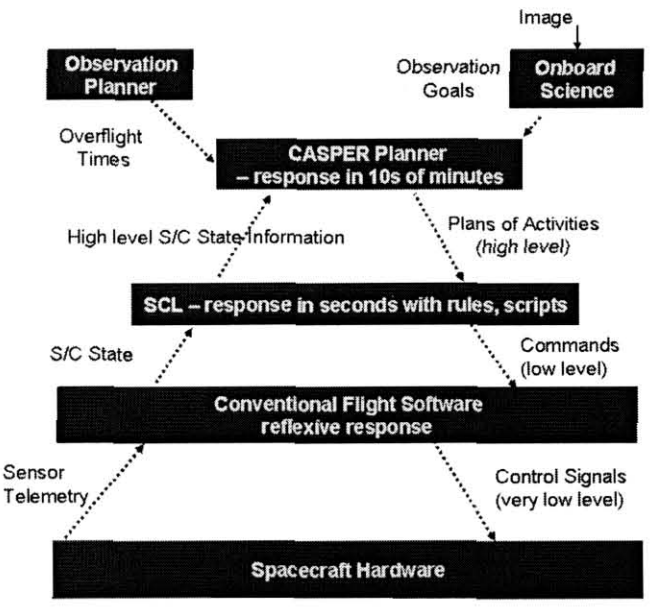

Figure 2. Autonomy Software Architecture

Each of the software modules operates at a separate priority level within the VxWorks real-time operating system onboard EO-1. The batch processes (Science)

have the lowest priority, with CASPER, L2, and SCL with increasing priority.

This agent architecture is designed to scale to multiple agents with agents communicating at either the planner level (via goals) or the execution level (to coordinate execution) (see clement, barrett)

We now describe each of the components of our architecture in further detail.

3.1 Onboard Science Analysis

The first step in the autonomous science decision cycle is detection of interesting science events. We are flying several science event detection modules including:

- Thermal anomaly detection – uses infrared spectra peaks to detect lava flows and other volcanic activity.
- Cloud detection [17] – uses intensities at six different spectra and thresholds to identify likely clouds in scenes.
- Flood scene classification – uses ratios at several spectra to identify signatures of water inundation as well as vegetation changes caused by flooding. (see Figure 4.)
- Change detection – uses multiple spectra to identify regions changed from one image to another. This technique is applicable to many science phenomena including lava flows, flooding, freezing and thawing and is used in conjunction with cloud detection.

Onboard detection of these science events enables ASE to monitor targets for extended periods of time for activity and automatically retarget when events are detected. For example, ASE might be used to monitor a dry riverbed acquiring 1 image every 16 days – but to increase the observation cadence to 5 images every 16 days when flooding is detected activity is detected.

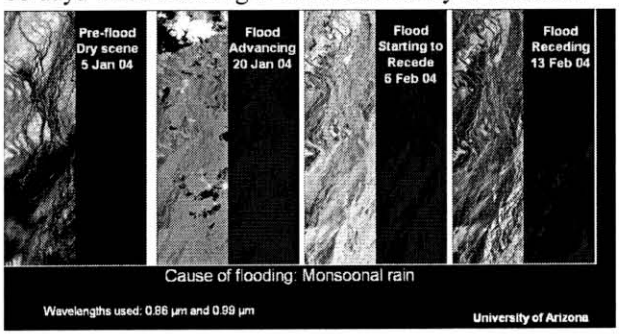

Figure 4. Flood detection timeseries imagery of Australia's Diamantina river with visual spectra at left and flood detection map at right.

Of particular interest is the study of Snow-Water-Ice-Land (SWIL) events. These algorithms are used to detect lake freeze/thaw cycles and seasonal sea ice. In this area, the ASE science team first manually developed classifiers. We later used scientist labeled data in conjunction with machine learning techniques to automatically develop improved classifiers [ref becky]. In particular, Support Vector Machines were used to develop classifiers that outperformed the scientist derived classifiers. It is these SVM classifiers that are currently being used for EO-1 operations.

3.2 Onboard Mission Planning

The CASPER [5] planner enables ASE to autonomously replan its future activities based on science event detections. CASPER is a deliberative, model-based AI planner which uses a local search approach [15] to develop operations plans.

Because onboard computing resources are scarce, CASPER must be very efficient in generating plans. While a typical desktop or laptop PC may have 2000-3000 MIPS performance, 5-20 MIPS is more typical onboard a spacecraft. In the case of EO-1, the Mongoose V CPU has approximately 8 MIPS. Of the 3 software packages, CASPER is by far the most computationally intensive. For that reason, our optimization efforts were focused on CASPER. Careful engineering and modeling were required to enable CASPER to build a plan in tens of minutes on the relatively slow CPU.

In the context of ASE, CASPER reasons about the majority of spacecraft operations constraints directly in its modeling language. However, ground operations still perform spacecraft orbit maintenance and momentum management.

3.3 Onboard Robust Execution

ASE uses the Spacecraft Command Language (SCL) [10] 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 design enables both loose or tight coupling between SCL and other flight software as appropriate.

Many aspects of autonomy are implemented in SCL. For example, SCL implements many constraint checks that are redundant with those in the EO-1 fault protection software. Before SCL sends each command to the EO-1 command processor, it undergoes a series of constraint checks to ensure that it is a valid command. Any pre-requisite states required by the command are checked (such as the communications system being in the correct mode to accept a command). Using SCL to check these constraints (while included in the CASPER model) provides an additional level of safety to the autonomy flight software.

3.4 Model-based Diagnosis

More recently ASE has teamed with the NASA Ames Research Center to fly the Livingstone 2 Mode Identification and Diagnosis software [16] beginning in Fall 2004. Both L2 and CASPER use models of the

spacecraft separate from the reasoning engine: the models are tailored for a particular application without the need to change the software, allowing reuse of the advanced reasoning software across applications. The intent is that a trained subsystem engineer could build these models even at the design stage. The diagnostic capability of an on-board agent can then use the models to monitor the health of the spacecraft and detect faults. By reconfiguring the spacecraft, the agent may recover functionality and continue on to meet its mission goals. Mission operators are promptly informed of anomalies, as well as routine state transitions of the spacecraft. In any sizable system, there are multiple points of failure and ambiguity as to the true fault. If a fault occurs, L2 presents several hypotheses with their probabilities, rather than the minimal single solution as did L1. Common cause faults are isolated to their root cause, more likely than alternate hypotheses which have multiple independent concurrent faults. L2 diagnoses faults that impact multiple subsystems (such as power) and localized faults (such as a failed subsystem sensor) with equal ease, allowing the diagnostic capability to scale with increasing system complexity. The most significant advances of L2 over previous work which were demonstrated are:

- Multiple Hypotheses and Multiple Hypotheses with Backtracking - Capability to track multiple diagnostic hypotheses and revise hypotheses given new evidence (backtracking), important in any complex system;
- Diagnosis During Transients - Capability to monitor the spacecraft state and diagnose faults during transients, both under partial observability (before telemetry responses are seen) and whilst the physical dynamics of the system are settling out.

3.5 Status

The ASE software has flown in a series of increasing tests beginning in March 2003. Full autonomous science operations were first demonstrated in January 2004. As of February 2005, ASE software had been used to successfully acquire over 1000 science images and had operated as long as three weeks continuously. Our operations have been so successful the EO-1 flight operations team now uses the ASE software for normal operations.

4. Lessons Learned on ASE

One of the most important lessons learned in the flight of ASE was that the overall architecture and systems used were extremely synergistic and key to the success of the experiment. The two components that represent mission operations and spacecraft constraint knowledge (CASPER and SCL) have very different representations. CASPER uses an activity centered model that represents spacecraft states, resources, and timing requirements. SCL easily encodes procedures in the form of Do A then Do B then wait for condition C, then do E. These varied representations were invaluable in representing a wide range of operations situations and responses (this result is consistent with other experiences in automated planning [20, 21]).

Another important theme which was validated by flight experience is that encoding information in models rather than code yields many benefits. First, models are often more readable and directly represent the intent. This results in more rapid encoding, easier validation, and shared understanding of the overall system by larger elements of the team. Second, updates to the overall system that only require a model change are much easier. Code changes required that code patches be generated, uploaded and implemented (as patches to the executable binary onboard), a process expensive in effort, time, and filled with chances for mistakes. In the worst case, extensive code changes could require a complete binary image upload (see below). In contrast, model changes require an upload of a text or binary file and a restart of the ASE control software, a much easier process.

Another major challenge to the experiment was validation and version/upate management. Originally, we planned on using a significant simulation capability to test out many aspects of the agent software, models, and integration with the flight software [17]. However, while this was invaluable in validating early versions of the software and throughout integration, for later versions we found that this exhaustive testing took significant effort for little payoff. As a result, in later releases we tailored the validation and testing to the modifications to the software that were being made. This meant that the test scenarios were targeted at the specific modifications, enhancements, or extensions in the release. For example, when various instrument calibration activities were added to the ASE model, test cases were designed to focus on areas of the model that had been changed or added since the last version. Another challenge was balancing the release schedule against the ongoing flight testing. When flight testing revealed that a model or software should be updated, almost always an operations workaround existed. This meant that we had several options: 1. implement an operational workaround, 2. the software (or model) could be patched, or 3. wait and incorporate the fix at the next release upload. The factors to balance are: 1. operational workarounds restrict the capability of the mission and/or cause the operations team more work and/or increase the possibility of mis-operating the spacecraft; 2. software patches take effort and include their own risk of introducing errors into the flight load; and 3. major releases requires a long time to upload (3 weeks), required significant mission downtime for checkout (days), and required workarounds for significant periods of time (we have had only four versions uploaded in about 18 months of operation).

One adjunct to the testing/release process is that having a high fidelity testbed is invaluable to facilitate integration and pre-operational testing. Unfortunately, when we began working with the EO-1 mission, because of cost constraints they had only a single, small memory, not very high fidelity testbed. Because of cost constraints, the majority of their testing had been done on the spacecraft prior to launch. Because of this situation, ASE funded construction of three higher fidelity testbeds at significant cost (several hundred thousand dollars). However, the construction, delivery, debugging, and configuration of these testbeds was an extensive process which took 6 months for the delivery of the first testbed to 12 months before the last one was fully functional. Additionally, significant resources were devoted to upgrading the testbed simulation capability to make it more able to model aspects of flight that ASE would require. Even after this process was complete, the developed testbeds are far less than typical for a mainstream NASA mission such as Odyssey or the Mars Exploration rovers.

Another lesson learned was that infrastructure to operate ASE was much more important and greater effort than we had originally envisioned. In order to track what was going on onboard the spacecraft within the ASE software, it required significant engineering to figure out what was the minimal information necessary [22], how to summarize it appropriately, and how to automate getting it down from the spacecraft, and how to get it to the ASE team, and how to process, visualize, and analyze it. Additionally, as the usage of the ASE software has changed from "monitor every single aspect of its operations" to "monitor similar to conventional operations" to the move to more "lights out" operations, the levels of telemetry needed have changed and the amount of automation required to deliver telemetry has increased significantly.

One of the major advantages of going to the ASE software to operate the spacecraft has been the increase in flexibility of operations of the spacecraft. In the conventional operations mode, typically observation selections were made 6-12 days in advance. While observations could be changed at the last minute (1-3 days in advance) for targets of opportunity, the significant manual re-work required limited this to fewer science events. In the automated operations mode, nominal targets are selected in advance, but automated triggered observations replacements and scientist last minute requests 2 days in advance do not require major rework and hence have becomes commonplace.

One lesson learned is the sheer size and scope of operations. From the science selection, to engineering activities, to scheduling ground contacts, to the actual generation of command loads, and handling of anomalies, there is an immense number of small and large tasks involved in the spacecraft operations flow. Only by extended contact (multi-year collaboration) has it been possible to understand the operations flow enough to be involved in the end to end automation of EO-1 operations.

As a specific example of how the EO-1 mission has evolved consider how its staffing and science product efficiency have improved steadily over the mission lifetime. Originally, the intent was to fly EO-1 acquiring several images per day. Now, EO-1 is operated to acquire many (~16) images per day. In some cases, EO-1 acquires "dual collects" in which on a single orbit it acquires imagery of two targets, without even shutting down elements of the instruments in between images. This has enables EO-1 to increase its utilization efficiency to where it is currently acquiring about 120 scenes per day. This evolution is a reflection of the innovativeness and excellence of the EO-1 operations team. In some sense, this evolution is an indication as to how most of the low hanging fruit for automation of EO-1 had already been exploited prior to the flight of ASE.

Very important to the experiment success was long-term contact between the JPL AI and Goddard EO-1 Flight Operations teams. Throughout the experiment, many trips were made to GSFC where multiple JPL team members spent long periods of time at GSFC observing operations and for every major checkout of the software. This enabled the JPL personnel to understand in depth the spacecraft and operations constraints. This also enabled GSFC personnel to understand in depth how the AI software worked. This long term learning experience was critical to the view of both institutions working as one team (as well as the elements at Interface & Control). This enabled the group to understand the (possibly competing) motivations of team members. For example, when anomalies occurred it enabled the joint team to agree on reasonable steps to investigate. Nowhere was this more clearly illustrated than after anomalies. In two cases, where very serious anomalies occurred, the joint team was in agreement that if the anomaly could not be explained after specific tests that the flight test would continue despite the very real chance of recurring anomaly.

Specifically, in November 2003, during a flight test, the EO-1 Solid State Recorder (SSR) powered off near the end of a data acquisition under ASE control. During this image acquisition there was no ground contact with the spacecraft. The joint team agreed on analyzing the telemetry, re-running in testbed under varying conditions, and checking certain fault responses. The team also agreed that if all of these steps did not uncover the problem and three weeks had passed flight experimentation would continue. The team concluded that in this case more data would be needed to diagnose the problem and a test case was devised in which the spacecraft would image a ground station so that real-time telemetry would be available if the anomaly recurred. Luckily, after 2.5 weeks the cause of the anomaly was found. While the ground testbed had shown CPU loading during end of recording an image at only 80%, flight data showed that the CPU was pegged at 100%. Furthermore, it was discovered that if the memory scrub task was

starved for more than 8 seconds a fault response to power down the SSR was invoked. This response was discovered only by code inspection (e.g. it was not present in the available flight software documentation). This was suprising because the memory scrub task is not time critical in that it must cycle through memory some number of times every 24 hours but short term stoppage is not critical. This anomaly was resolved by patching the flight software response to log an error message rather than power down the SSR.

In January 2005, with a new release of the software, an anomaly was experienced during an apparently idle portion of operations. Specifically, the CASPER planning software threw an exception, which invoked a fault response to restart the secondary processor running the ASE software (e.g. EO-1 reverted to normal control). Again, the joint operations team agreed to: analyze the telemetry, try to reproduce the fault in the ground testbed, and also to examine all areas of code that were changed or added in the new release. However, as the operations team was very comfortable with the ASE software, the team suggested that if the problem could not be diagnosed in two weeks autonomous operations should be restarted (note that at this time we had approximately 1 year of successful autonomous operations and 800 autonomous image takes). This anomaly was particularly vexing as it occurred in an apparently idle period of operations (again with no ground contact). After analyzing the telemetry and unsuccessfully trying to reproduce the error in the ground testbed, prospects for diagnosing the problem were not good. However, analysis of the new features added in the release revealed that a new onboard science data summarization algorithm was writing the science summary data to an incorrect address. The implemented address corresponded to RAM allocated to CASPER data structures, and in flight prior generated science summary products overwrote CASPER RAM – resulting in the later exception.

One interesting effect of the prolonged intermingling of the operations team is that certain members of the technology team acquired a strong risk aversion as operations continued. This has progressed to the point where for current operations members of the EO-1 flight operations team are often more willing to aggressively test and utilize the autonomy software.

10 Related Work, and Conclusions

In 1999, the Remote Agent experiment (RAX) [13] executed for a several days onboard the NASA Deep Space One mission. RAX is an example of a classic three-tiered architecture [8], as is ASE. RAX demonstrated a batch onboard planning capability (as opposed to CASPER's continuous planning) and RAX did not demonstrate onboard science. PROBA [14] is a European Space Agency (ESA) mission demonstrates onboard autonomy and launched in 2001. However, ASE has more of a focus on model-based autonomy than PROBA.

The Three Corner Sat (3CS) University Nanosat mission used CASPER onboard planning software integrated with the SCL ground and flight execution software [3]. The 3CS mission was launched in December 2004 but the spacecraft were lost due to a deployment failure. The 3CS autonomy software includes onboard science data validation, replanning, robust execution, and multiple model-based anomaly detection. The 3CS mission is considerably less complex than EO-1 but still represents an important step in the integration and flight of onboard autonomy software.

More recent work from NASA Ames Research Center is focused on building the IDEA planning and execution architecture [12]. In IDEA, the planner and execution software are combined into a "reactive planner" and operate using the same domain model. A single planning and execution model can simplify validation, which is a difficult problem for autonomous systems. For EO-1, the CASPER planner and SCL executive use separate models. While this has the advantage of the flexibility of both procedural and declarative representations, a single model would be easier to validate. We have designed the CASPER modeling language to be used by domain experts, thus not requiring planning experts. Our use of SCL is similar to the "plan runner" in IDEA but SCL encodes more intelligence. The EO-1 science analysis software is defined as one of the "controlling systems" in IDEA. In the IDEA architecture, a communications wrapper is used to send messages between the agents, similar to the software bus in EO-1. In the description of IDEA there is no information about the deployment of IDEA to any domains, so a comparison of the performance or capabilities is not possible at this time. In many ways IDEA represents a more AI-centric architecture with declarative modeling at its core and ASE represents more of an evolutionary engineered solution.

ASE was originally scheduled for flight on the Techsat-21 mission [18]. However this mission was cancelled and the software was adapted for flight on EO-1. The principal changes from the Techsat-21 to EO-1 are that the science payload was changed from a synthetic aperture radar (SAR) to a hyperspectral imaging device (Hyperion). This change requires significant alteration to the science targets and analysis algorithms. The basic software architecture and components (e.g. CASPER and SCL) have remained the same. This paper also reports on some of our experiences in getting the software to flight and operations.

ASE on EO-1 demonstrates an integrated autonomous mission using onboard science analysis, replanning, and robust execution. The ASE performs intelligent science data selection that will lead to a reduction in data downlink. In addition, the ASE will increase science return through autonomous retargeting. Demonstration of these capabilities onboard EO-1 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 reduced operations cost.

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. 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," Proc i-SAIRAS 2001, Montreal, Canada, June 2001.
3. S. Chien, R. Sherwood, M. Burl, R. Knight, G. Rabideau, B. Engelhardt, A. Davies, P. Zetocha, R. Wainright, P. Klupar, P. Cappelaere, D. Surka, B. Williams, R. Greeley, V. Baker, J. Doan, "The TechSat 21 Autonomous Sciencecraft Constellation", Proc i-SAIRAS 2001, Montreal, Canada, June 2001.
4. S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling, Breckenridge, CO, April 2000. (also casper.jpl.nasa.gov)
5. A.G. Davies, R. Greeley, K. Williams, V. Baker, J. Dohm, M. Burl, E. Mjolsness, R. Castano, T. Stough, J. Roden, S. Chien, R. Sherwood, "ASC Science Report," August 2001. (downloadable from ase.jpl.nasa.gov)
6. Davies, A. G., E.D. Mjolsness, A.G. Gray, T.F. Mann, R. Castano, T.A. Estlin and R.S. Saunders (1999) Hypothesis-driven active data analysis of geological phenomena using semi-autonomous rovers: exploring simulations of Martian hydrothermal deposits. EOS, Trans. Amer. Geophys. Union, 80, no. 17, S210.
7. E. Gat et al., Three-Layer Architectures. in D. Kortenkamp et al. eds. AI and Mobile Robots. AAAI Press, 1998.
8. Goddard Space Flight Center, EO-1 Mission page: http://EO-1.gsfc.nasa.gov
9. M. Griffin, H. Burke, D. Mandl, & J. Miller, "Cloud Cover Detection Algorithm for the EO-1 Hyperion Imagery," Proceedings of the 17th SPIE AeroSense 2003, Orlando, FL, April 21-25, 2003.
10. Interface and Control Systems, SCL Home Page, sclrules.com
11. M. C. Malin and K. S. Edgett, "Evidence for recent groundwater seepage and surface runoff on Mars," Science, 288, 2330-2335, 2000.
12. N. Muscettola, G. Dorais, C. Fry, R. Levinson, and C. Plaunt, "IDEA: Planning at the Core of Autonomous Reactive Agents," Proceedings of the Workshops at the AIPS-2002 Conference, Tolouse, France, April 2002.
13. NASA Ames, Remote Agent Experiment Home Page, http://ic.arc.nasa.gov/projects/remote-agent/. See also Remote Agent: To Boldly Go Where No AI System Has Gone Before. Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian Williams. *Artificial Intelligence* 103(1-2):5-48, August 1998
14. The PROBA Onboard Autonomy Platform, http://www.estec.esa.nl/proba/
15. G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations in the ASPEN System," Intl Symp Artificial Int Robotics & Automation in Space, Noordwijk, The Netherlands, June 1999.
16. J. Kurien and P. Nayak. "Back to the future for consistency-based trajectory tracking." In Proceedings of the 7th National Conference on Artificial Intelligence (AAAI'2000), 2000.
17. B. Cichy, S. Chien, S. Schaffer, D. Tran, G. Rabideau, R. Bote, Dan Mandl, S. Frye, S. Shulman, J. Van Gaasbeck, D. Boyer, "Validating the EO-1 Autonomous Science Agent," Intl Workshop on Planning and Scheduling for Space, Darmstadt, Germany, June 2004.
18. S. Chien, et al., "The Techsat-21 Autonomous Space Science Agent," International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2002). Bologna, Italy. July 2002
19. S. Chien, et al. "The EO-1 Autonomous Science Agent," International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2004). New York City, NY, July 2004.
20. T. Estlin, S. Chien, and X. Wang, ``Hierarchical Task Network and Operator-Based Planning: Two Complementary Approaches to Real-World Planning," *Journal of Experimental and Theoretical Artificial Intelligence*, 13:379-395, 2001.
21. David E. Wilkins and Marie desJardins "A Call for Knowledge-Based Planning" AI Magazine 22(1): Spring 2001, 99-115
22. D. Tran, S. Chien, G. Rabideau, B. Cichy "Flight Software Issues in Onboard Automated Planning: Lessons Learned on EO-1" International Workshop on Planning and Scheduling for Space (IWPSS 2004). Darmstadt, Germany. June 2004

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.