JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 2, April 2005

Using Autonomy Flight Software to Improve Science Return on Earth Observing One

Steve Chien,[*] Rob Sherwood,[†] Daniel Tran,[‡] Benjamin Cichy,[‡]
Gregg Rabideau,[§] Rebecca Castano,[**] and Ashley Davis[††]
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109

Dan Mandl,[‡‡] Stuart Frye,[§§] Bruce Trout, Seth Shulman
Goddard Space Flight Center, Greenbelt, Maryland 20771

and
Darrell Boyer[***]
Interface and Control Systems, Indialantic, Florida 32903

NASA's Earth Observing One Spacecraft (EO-1) has been adapted to host an advanced suite of onboard autonomy software designed to dramatically improve the quality and timeliness of science-data returned from remote-sensing missions. The Autonomous Sciencecraft Experiment (ASE) enables the spacecraft to autonomously detect and respond to dynamic scientifically interesting events observed from EO-1's low earth orbit. ASE includes software systems that perform science data analysis, mission planning, and run-time robust execution. In this article we describe the autonomy flight software, as well as innovative solutions to the challenges presented by autonomy, reliability, and limited computing resources.

I. Introduction

The Autonomous Sciencecraft Experiment (ASE), an autonomous software agent currently flying onboard the Earth Observing One (EO-1) spacecraft, demonstrates several integrated autonomy technologies that together enable science-directed autonomous operations. Using onboard data-processing, mission-planning, and robust execution, ASE commands the EO-1 spacecraft to react to the science-value of collected observations. A suite of data-processing "science" algorithms analyze incoming observations looking for dynamic science events, including volcanic eruptions, flooding, ice breakup, and cloud cover change. An onboard decision-making agent modifies the spacecraft observation plan to capture follow-on observations of high-value science events, or to delete observations with little scientific value. A robust goal and task oriented execution system executes mission plans, making adjustments to compensate for run-time anomalies and uncertainties. Together these technologies maximize science return through autonomous goal-directed exploration and data acquisition. This article describes the effort to develop and deploy ASE on EO-1.

Received 23 August 2004; revision received 15 December 2004; accepted for publication 15 December 2004.

[*]Principal Scientist, Autonomous Systems, 4800 Oak Grove Drive; steve.chien@jpl.nasa.gov.
[†]Experiment Manager, Autonomous Sciencecraft Experiment, 4800 Oak Grove Drive. Member, AIAA.
[‡]Software Engineer, 4800 Oak Grove Drive.
[§]Senior Software Engineer, 4800 Oak Grove Drive.
[**]Senior Member of the Technical Staff, 4800 Oak Grove Drive.
[††]Experiment Scientist, Autonomous Sciencecraft Experiment, 4800 Oak Grove Drive.
[‡‡]Mission Director, Earth Observing One Mission.
[§§]EO-1 Systems Engineer, Structure and Evolution of the Universe Division, Code 490. Member, AIAA.
[***] Senior Software Engineer/Controls Engineer, Interface and Control Systems, 122 Fourth Avenue.

The ASE onboard flight software includes several autonomy software components:

1) 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
2) Robust execution management software using the Spacecraft Command Language (SCL) [Interface & Control] package to enable event-driven processing and low-level autonomy
3) The Continuous Activity Scheduling Planning Execution and Replanning (CASPER)[4] 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. This software can 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.

The onboard planner (CASPER) generates mission operations plans from goals provided by the onboard science analysis module. The model-based planning algorithms 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 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.

A typical ASE scenario involves monitoring of active volcano regions such as Mt. Etna in Italy. (See Fig. 1.) Hyperion data have been used in ground-based analysis to study this phenomenon. The ASE concept is applied 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 infrared and near infrared 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.

The basic software architecture used by ASE on EO-1 has been previously described,[3,18] thus here we concentrate on the overall system – how the components work together to achieve the closed loop science response as well as how the software was modified to deal with the unique challenges of flight on EO-1 (most of which apply to other space missions).

Building autonomy software for space missions has a number of challenges, including the following:

1) Limited, intermittent communications to the agent. A typical spacecraft in low earth orbit (such as EO-1) has 8 x 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. A typical spacecraft CPU offers 25 MIPS and 128 MB RAM – far less than a typical personal computer. Our CPU allocation for the Autonomous Science agent 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 first provide background information:

1) describing the basic characteristics of the EO-1 mission and spacecraft; and
2) reviewing the basic ASE on EO-1 software architecture.

We then provide information on how our software dealt with three key aspects of software agents for spacecraft.

1) We describe how our onboard planning software can generate mission plans despite the limited EO-1 CPU processor (our allocation is about 4 MIPS)
2) We describe how the ASE telemetry was designed to provide sufficient information to track the ASE software performance within very limited bandwidth
3) We describe our layered, redundant agent and how it enables additional agent safety – critical to the operations of mission with costs over $100 Million dollars.

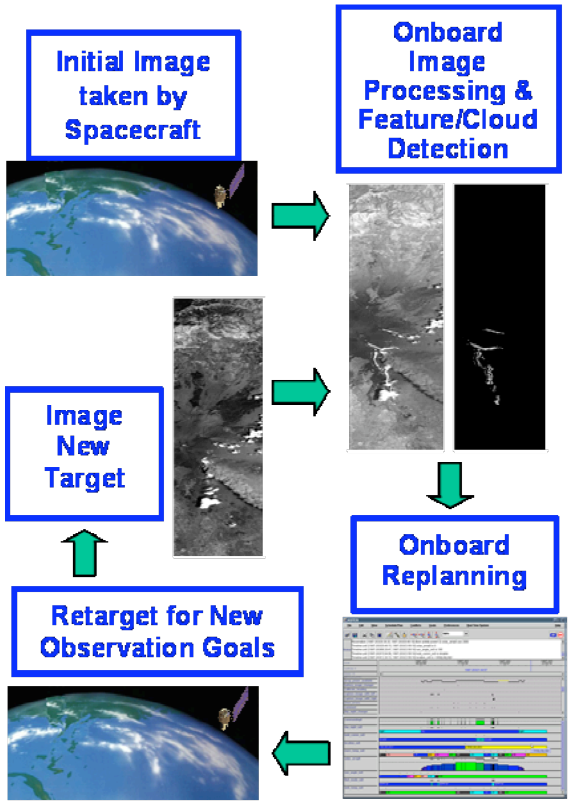

Initial Image taken by Spacecraft
Onboard Image Processing & Feature/Cloud Detection
Image New Target
Onboard Replanning
Retarget for New Observation Goals

Fig. 1 Autonomous science scenario

II. The EO-1 Mission

The Earth Observing-1 (EO-1) satellite is the first mission in NASA's New Millennium Program Earth Observing series.[8] Designed as a testbed for the next-generation of advanced land imaging instruments, EO-1 was launched on a Delta 7320 from Vandenberg Air Force Base on November 21, 2000 into a 705 km circular sun-synchronous orbit at a 98.7 degree inclination. This orbit follows a 16-day repeat track, with at least 5 day and 5 night over-flights per 16-day cycle separated by less than a 10-degree change in viewing angle.

EO-1 carries three instruments: the Advanced Land Imager (ALI), the hyper-spectral Hyperion Imager, and the Atmospheric Corrector (AC). The ALI combines novel wide-angle optics with a highly integrated multispectral and panchromatic spectrometer to demonstrate spatial and spectral resolution comparable or improved from Landsat at substantial mass, volume, and cost savings. The Hyperion is a high-resolution imager capable of resolving 220 spectral bands (from 0.4 to 2.5 µm) with a 30-meter spatial resolution. The instrument typically images a 7.5 km by 42 km land area per image and provides detailed spectral mapping across all 220 channels with high radiometric accuracy (ASE uses the products from the Hyperion instrument for onboard science processing). Finally the EO-1 AC provides the first space-based test of an Atmospheric Corrector (AC) - designed to compensate for atmospheric absorption and scattering, allowing for increased accuracy of surface reflectance estimates. Together the three instruments collect over 20-Gbits of science data to the onboard solid-state data recorder for each observation.

The EO-1 spacecraft has two Mongoose M5 processors. The first M5 is used for the EO-1 command and data handling functions. The second 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 secondary WARP M5 processor. This provides an added level of safety for the spacecraft since the ASE software does not run on the main spacecraft processor.

Following a one-year primary mission, EO-1 entered extended mission in January of 2002 having surpassed all original technology validation goals. To date, EO-1 has collected over 5,000 successful science observations, far beyond the original success criteria of 1,000 observations.

III. Autonomy Software Architecture

The autonomy software on EO-1 follows a traditional three-layer architecture (See Fig. 2) consisting of planner, executive, and skills layers. At the highest level of abstraction, the Continuous Activity Scheduling Planning Execution and Replanning (CASPER) software manages mission planning functions. CASPER schedules science activities while enforcing spacecraft operations and resource constraints. The duration of the planning process is on the order of tens of minutes. Activities scheduled by CASPER are passed as inputs to the Spacecraft Command Language (SCL) executive system, which generates the corresponding detailed command sequences. SCL operates on the several second timescale. Below SCL the EO-1 flight software represents the "skills" layer - responsible for lower level control of the spacecraft including 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 and is thus directly coded in C. SCL must respond (in seconds) quickly and 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.
2) The layered architecture enables redundant implementation of critical functions – most notably spacecraft safety constraint checking. In the design of our spacecraft agent model, we implemented spacecraft safety constraints in all levels where feasible.

It is worth noting that our 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).

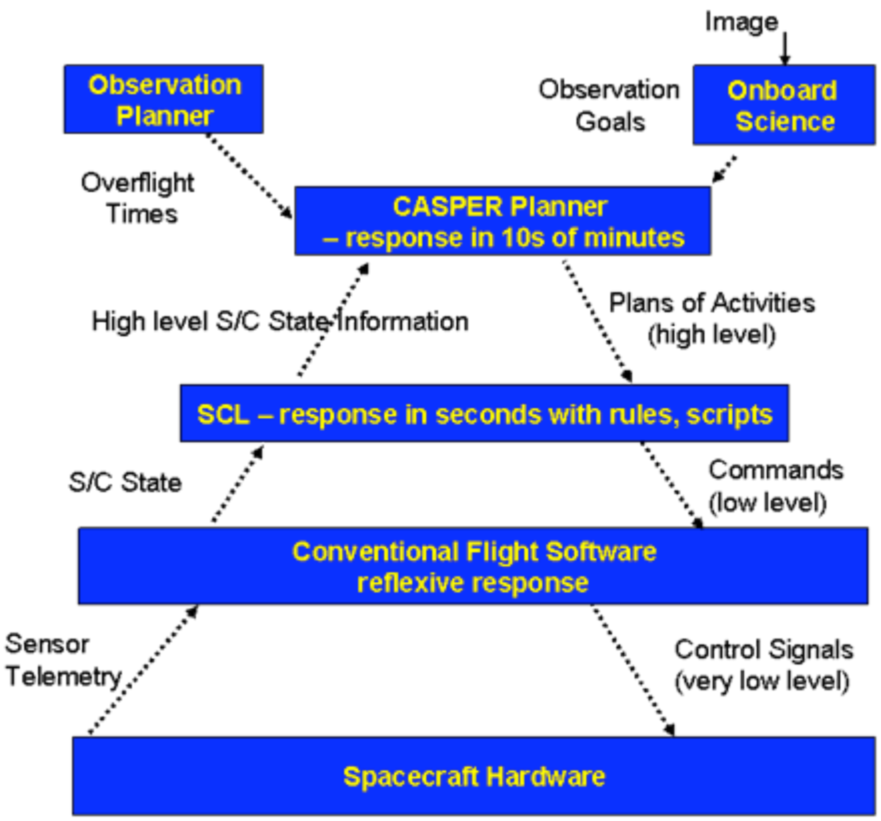


Image
Observation Planner
Observation Goals
Onboard Science
Overflight Times
CASPER Planner – response in 10s of minutes
High level S/C State Information
Plans of Activities (high level)
SCL – response in seconds with rules, scripts
S/C State
Commands (low level)
Conventional Flight Software reflexive response
Sensor Telemetry
Control Signals (very low level)
Spacecraft Hardware

Fig. 2 Autonomy software architecture.

IV. Onboard Science Analysis

The first step in the autonomous science decision cycle is detection of interesting science events. In ASE a number of science analysis technologies are being flown including:

1) Thermal anomaly detection – uses infrared spectra peaks to detect lava flows and other volcanic activity.
2) Cloud detection[9] – uses intensities at six different spectra and thresholds to identify likely clouds in scenes.
3) Flood scene classification – uses ratios at several spectra to identify signatures of water inundation as well as vegetation changes caused by flooding.
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.
5) Generalized Feature detection – uses trainable recognizers to detect spatial features as sand dunes and wind streaks.

All of these science algorithms use the Hyperion instrument as the ALI data is not available for processing onboard. In addition, the onboard science algorithms had to work within several flight constraints (see Fig. 3):

1) Onboard data was only partially calibrated data (gain correction only)
2) Limited onboard processing capabilities restricts access to only 12 of the 220+ bands (any 12 bands may be selected)
3) Limited onboard processing capabilities necessitate the use of algorithms with low computational requirements per pixel

The utilization of onboard processing to select data for downlink and trigger future imagery is shown below.

1) The EO-1 spacecraft images a target and collects the full (220 band) instrument data over a 7.7km (across track) by 50+km (along track) swath. This data is stored directly on the solid-state recorder (SSR) onboard.
2) Onboard a portion of this data is read back into RAM (7.7km x 30km approx.) and 12 bands.

3) Onboard science algorithms run to detect the process of interest, producing a score based on the type of detector being run.
4) These science detection scores are used to: prioritize data for downlink; summarize or delete data; and/or to retask the spacecraft for subsequent observations of same or related phenomena.
5) If science images are deemed of interest, the full data from the observation (e.g. all bands and full swath) is available for downlink. However, in some cases greatly summarized data can be downlinked, optimizing the use of scarce downlink resources. For example, if volcanic activity is detected, it is typically 5-30 pixels of a 256x1024 image. By downlinking only key spectra and actual volcanic pixels (with their locations) we can achieve 3-4 orders of magnitude in data reduction.

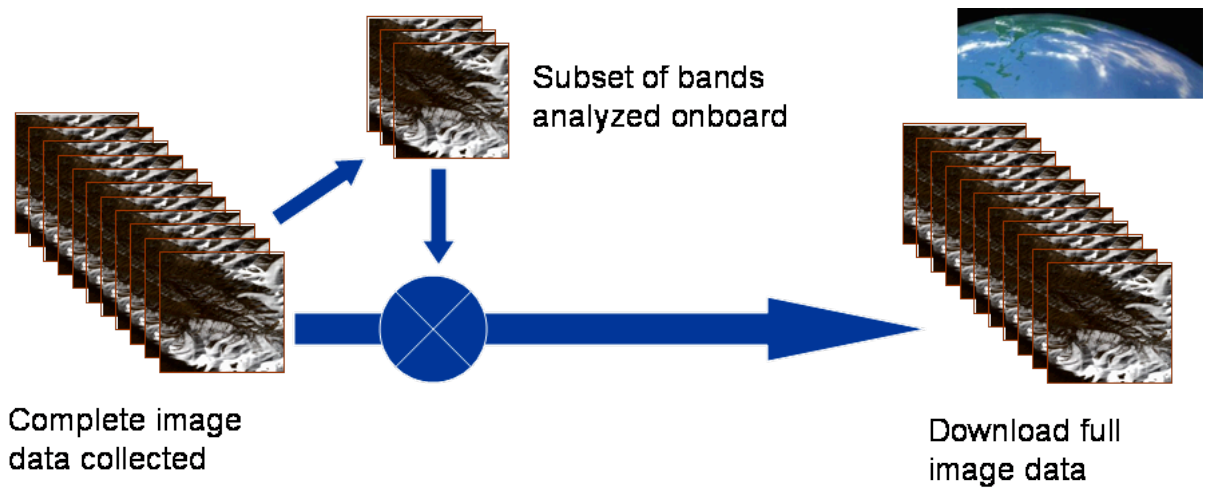

Subset of bands analyzed onboard
Complete image data collected
Download full image data

Fig. 3 Onboard science processing.

The Arizona State University developed Snow-Water-Ice-Land (SWIL) algorithm is used to detect lake freeze/thaw cycles and seasonal sea ice. The SWIL algorithm uses six spectral bands for analysis. The classified pixels in the images are then used to drive a science event trigger. For example, in the images in Fig. 4, the ratio of ice to ice and water pixels (e.g., ice/(ice + water)) is used to detect the freezing and thawing of Round Lake in Minnesota. Figure 5 shows the use of the SWIl classifier to track sea ice breakup in polar regions.

Figure 6 shows both the visible and the infrared bands of the same image of the Mt. Etna volcano in Italy. The infrared bands are used to detect hot areas that might represent fresh lava flows within the image. In this picture, these hot spots are circled with red dotted lines.

The University of Arizona developed flood scene classification algorithm uses multiple spectral bands to differentiate between land and water. The results of the algorithm include are compared with land and water counts from a baseline image to determine if flooding has occurred (or is receding). If significant flooding has been detected, the image can be downlinked. In addition, a new goal can be sent to the CASPER planning software to image adjacent regions on subsequent orbits to determine the extent of the flooding. Figure 7 shows ASE data used to track a rare monsoonal flooding event in the Diamantina river basin in Australia.

The above examples utilize "expert-derived" classifiers. These are classifiers developed by ASE science team members by manual inspection of Hyperion images. While this process was effective, it required considerable effort by the science team. More recently, elements of the ASE team have applied machine learning techniques to the Hyperion image classification problem (Ref. 22 in preparation). First, a brute force band ratio search algorithm was applied to determine the best bands and thresholds for a simple single band ratio classification technique. Second, more sophisticated Support Vector Machine[20,21] learning techniques have been applied.

As seen in Fig. 8, the automated ratio method achieved results comparable to the human expert derived classifier. The SVM methods achieved results comparable or better than the human expert derived classifier.

Later flights will validate as many science analysis algorithms as resources allow. These flights will begin by validating change detection on multiple science phenomena, spatial feature detection on Aeolian (wind) features such as sand dunes, sand shapes, and wind streaks, and the Discovery algorithm.[1] Validating this portfolio of science algorithms will represent a valuable step forward to enabling future autonomous science missions.[5,6]

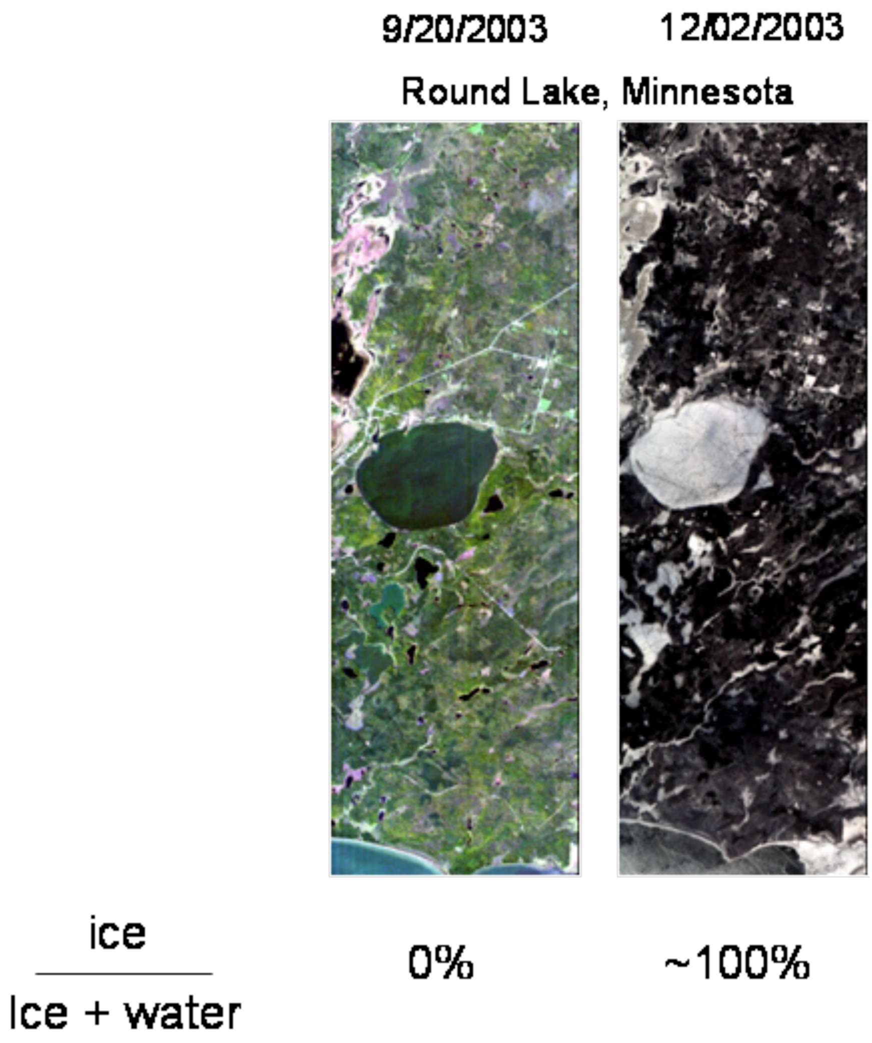


Fig. 4 Detection of lake freezing for Round Lake, Minnesota by classification of ice and water and computing the ration of ice to ice or water pixels. (Courtesy of NASA)

V. Continuous Planning

The spacecraft planning and scheduling process is traditionally performed as one step of ground operations to schedule science observations and downlink opportunities. The output of this process is a detailed sequence of commands to be issued to the spacecraft for execution. In order for the ASE to autonomously satisfy new science requests from image processing algorithms, this step is done onboard by the CASPER[4] planning software. CASPER is able to represent the operations constraints in a generic modeling language and reasons about these constraints to generate a detailed mission operation plan while respecting mission constraints and resources (see Fig. 9).

CASPER uses a local search[15] approach to develop the detailed operation plan. The main algorithm for planning and scheduling is based on a technique called iterative repair. During iterative repair, the conflicts in the plan are detected and resolved one a time, until no conflicts exists. A conflict is considered to be any violation of the mission or spacecraft constraints and resolved through several predefined methods. These methods include: moving, adding, removing, detailing, or abstracting a scheduled operation. The repair algorithm may use any of these methods in an attempt to resolve a conflict.

Figure 11 shows an example of the repair algorithm after introducing a set of new science observations into the plan.

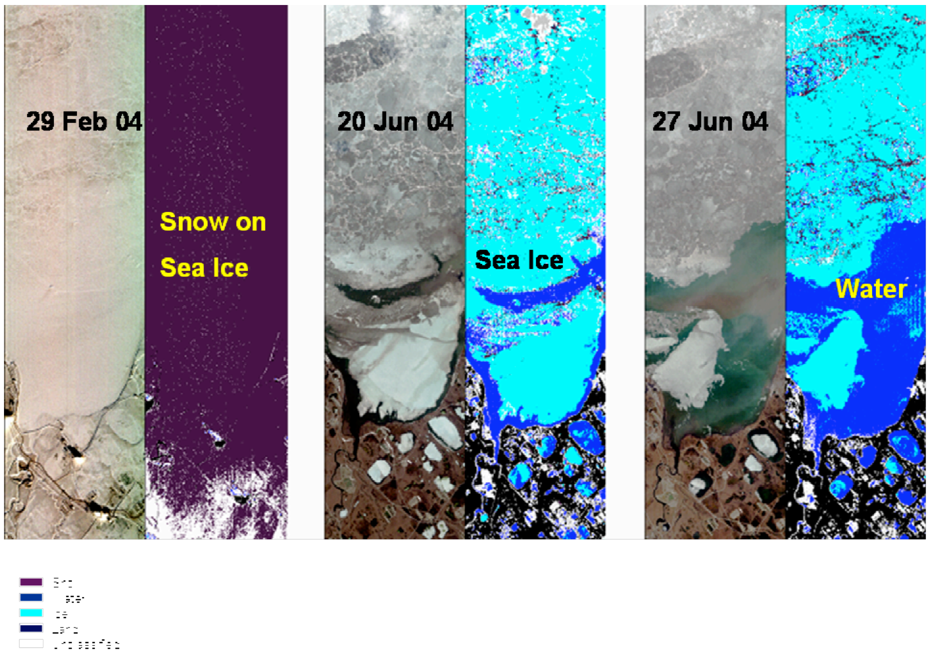


Fig. 5 Classification of snow, water, ice, and land used to detect sea ice formation and breakup in polar regions. (Courtesy of NASA)

For example, imagine that CASPER is trying to schedule several observations: Act-1 and Act-2 shown above. CASPER is continuously checking constraints – properties that must hold in order for an operations plan to be valid. An example constraint is that the plan must always use no more power than is available at that point in time onboard the spacecraft. In this case, CASPER would detect a conflict – which is a way in which a constraint can be violated. In this case the combination of Act-1 and Act-2 uses more power than is available. CASPER notes the conflict of "using more power than is available" over the time interval (b). CASPER then considers a set of repair methods – each repair method is a modification to the plan that may remove the conflict. One repair method for this type of conflict is to delete an activity using power during the conflict (e.g. either delete Act-1 or delete Act-2). CASPER will choose a repair method applicable to this conflict. If it chooses "delete" it may have further choices to make, such as which of Act-1 and Act-2 to delete.

CASPER may have many alternative choices in dealing with a single conflict. And there may be many conflicts in the plan. CASPER must attack each of these individually, searching possibly many alternatives to find a combination that works for the complete plan. In this way CASPER develops a mission plan by composing and modifying the many activities in the plan.

Below we show a mission plan generated by CASPER. At the top of the plan the black lines represent activities. The lines start at the beginning (in time) of the activity and the length of time indicates the duration of the activity (e.g. the end of the line represents the completion of the activity). The lower portion of the diagram indicates the states and resources tracked by CASPER. These include the spacecraft state (such as orientation, power, etc.), computer state (available space on the SSR, etc.), and communications availability (such as ground stations in view).

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 which only about 4 MIPS are available to the ASE software. Additionally, while EO-1 has considerable RAM for a spacecraft (256MB), the complexity of EO-1 operations can cause this flight constraint to become critical.

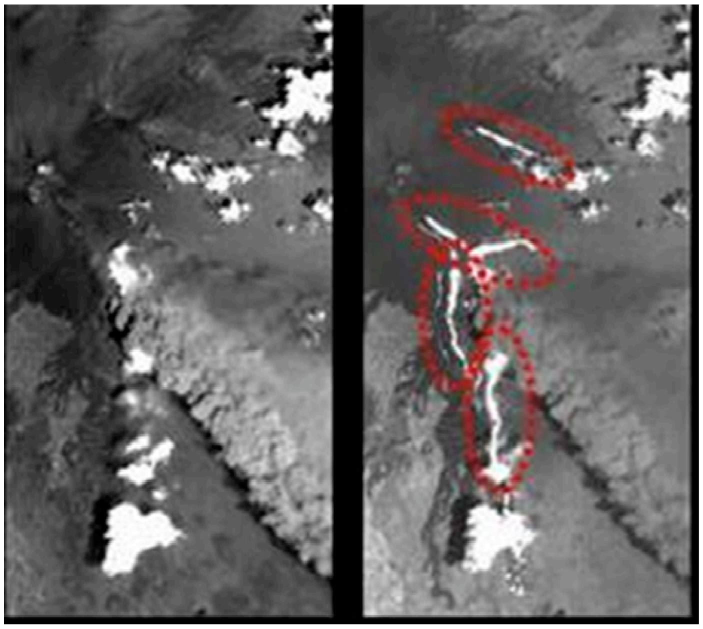


Fig. 6 Thermal anomalies associated with volcano activity at Mt. Etna, visual spectra at left and infrared spectra with labeled lava flows at right. (Courtesy of NASA)

CASPER plans within limited CPU resources by using a hierarchical, continuous[4] planning paradigm. Rather than attempt to plan out an entire week of operations in a single batch timeslice, it utilizes a long-term, more abstract plan for the longest planning horizon (one week), and plans at a detailed level for the next day of operations. As time proceeds forward, it incrementally replans for the new observations that fall within this one-day horizon (see Fig. 5). Consequently, CASPER CPU usage is spread more evenly than in a batch planning paradigm.

CASPER also utilizes incremental, continuous planning to deal with onboard RAM limitations.

1) One week of EO-1 ops includes about 100 science observations plus 50 S-Band/X-Band contacts.
2) The science observations alone for one week of operations represent approximately 7800 activities.
3) Represent a full week of operations at the activity/command level of detail would require approximately 224 MB of heap space in RAM.
4) Onboard constraints limit CASPER to 32 MB of heap space.

In order to deal with this restriction, CASPER performs detailed planning ~6 hours in advance, which uses approximately 16MB Heap Space (see Fig. 12). It represents the future observations at a more abstract level and does not reason about all of their command interactions. CASPER also deletes past activities from its plan database to reclaim the memory for other uses.

VI. Robust Task Execution

ASE uses the Spacecraft Command Language (SCL) [Interface & Control] 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 either loose or tight coupling between SCL and other flight software as appropriate.

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 relative time. Ground-based absolute time script scheduling is equivalent to the traditional procedural approach to spacecraft operations based on time. In ASE scripts are planned and scheduled by the CASPER onboard planner. The science analysis algorithms and SCL work in a cooperative manner to generate new goals for CASPER. These goals are sent with a messaging system.

Many aspects of autonomy are implemented in SCL. For example, many constraint checks redundant with fault protection are implemented in SCL. Before each command is sent from the autonomy software to the C&DH software by SCL, 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). SCL also verifies that there is sufficient power so that the command does not trigger a Low Bus Voltage and that there is sufficient energy in the battery so as to retain safe margins. Using SCL to check these constraints (while included in the CASPER model) provides an additional level of safety to the autonomy FSW.

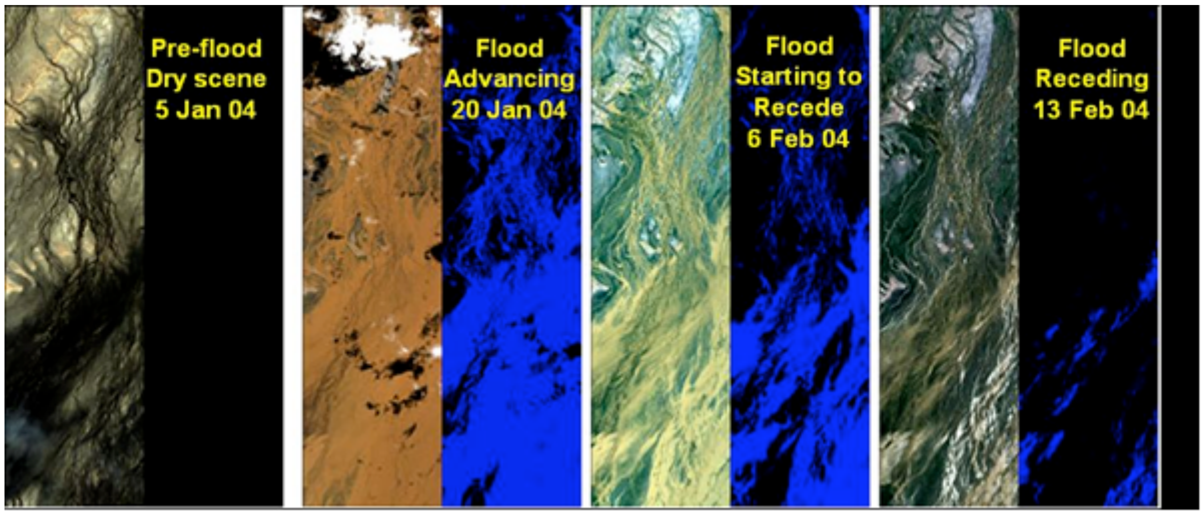


Fig. 7 Flood detection with visual spectra at left and flood detection map at right. Time series shown of same target. (Courtesy of NASA)

VII. Telemetry Management

On traditional ground planning systems, there are several ways of collecting data about the state and actions of the planner. Most systems provide a graphical user interface that allows ground personnel to immediately determine conflicts within the schedule, and the disk drive allows for large amounts of data to be stored for review. These are the methods we had used for developing and debugging issues with CASPER.

Collecting data for spacecraft operations is done much differently. ASE has two methods of collecting data: a telemetry packet and log files. Telemetry is output by each subsystem at various frequencies (a packet every 1 to 8 seconds) and provides information about the health and state of the spacecraft. The telemetry values are stored on the spacecraft local recorder and automatically downlinked during each ground contact. While in ground contact, the real-time telemetry data produced by each subsystem is immediately available to the ground operations team, but collected data has approximately a 24 hour turnaround time before it is available. Engineering data is the only method the EO-1 operations team used to collect data on the spacecraft. ASE introduced an 8MB ramdisk into the system as a means of collecting output log files. However, processing and viewing log files introduces extra work for the ground operator. In order to downlink log files from the spacecraft, operators need to specify the file to downlink and initiate the dump at the start of the ground contact. Also, if the file is too large, the operator may need to downlink the file across multiple ground stations and reconstruct it afterwards.

To fit within the framework of normal EO-1 operations and reduce the requirements to ground operators, we decided that engineering data would be the main method of extracting information about the health and status of CASPER. Output log files are still available for downlink, in case an anomalous situation occurs that cannot be explained through the telemetry packet.

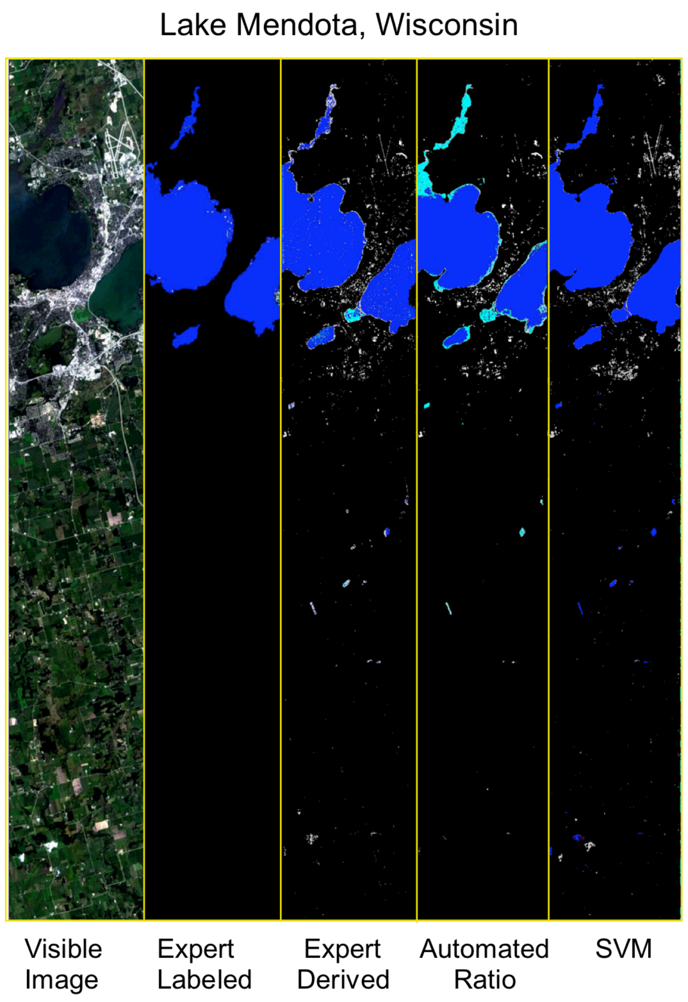


Fig. 8 Imagery and classification results of Expert labeled image, Manually derived, and two Machine Learned classifications – automated ratio and SVM. (Courtesy of NASA)

There were several objectives in determining what points of data to include in the CASPER telemetry packet. We wanted to be able to identify an anomalous situation within the planner and replay that data to replicate what occurred in flight. This was achieved by partitioning the packet into three sections.

1) Health and Status – contains a short summary of the planning software status (see Table 1).
2) Autonomous Decisions – the decisions that the planner takes onboard. These decisions are the choices made in "repair iterations" to fix any problems in the plan and are shown in Table 2.
3) Uncontrollable Inputs – all un-controlled, un-planned inputs to the planner are logged. With CASPER, these inputs are updates to plan variables that differ from the modeled value and are shown as Plan updates in Table 3.

A maximum of 248 bytes are available in a single telemetry packet. In the latest build of the ASE software, the CASPER telemetry packet is utilizing 224 out of the 248 bytes available. Below, we list out a few of the data points for each section, and provide a brief reason for each. To maximize the packet space, each telemetry point listed below is either 2 or 4 bytes in size.

The health and status section (see Table 1) of the packet allows us to determine the status of CASPER. The error and warning counters indicates if an unexpected situation occurred within the planning software. The stack usage indicates how much margin exists before overflowing the allotted stack space. The heap usage indicates if a memory leak is occurring.

The repair iteration section (see Table 2) of the packet contains data about how CASPER modified the plan. It does not contain information on what options were available at each choice point to assist in understanding why decisions were made. For example, it would be useful to collect the list of all conflicts considered in the repair iteration, prior to CASPER's selection. However, the number of conflicts is variable and unbounded. It would be impossible to store an unbounded set of data points within a finite telemetry packet, without creating an artificial upper bound. Instead, the conflict selected is stored to determine what schedule modifications were done. This strategy of selecting what was chosen is done for all choice points. The output log files contain the detailed list of possibilities for each choice point for a repair iteration.

With all autonomous plan modifications logged, we are able to reproduce in our ground testbed what occurred in flight. However, because we are not able to store the full state of CASPER, we assume that it is possible to reproduce the initial conditions of the system. When loading the initial set of goals into the planner, there needs to be the same number of conflicts on our ground testbed as there would be in flight in order to reproduce what occurred.

Table 1 Sample of the Health and Status Section of the CASPER Telemetry Packet

Health & Status	
Data Point	**Description**
Heartbeat	Up-counter to indicate planner is still active
Errors	Number of errors
Warnings	Number of warnings
Current Stack Usage	Current amount in use from allocated stack space
Maximum Stack Usage	High amount used from allocated stack space
Current Heap Usage	Current amount allocated from heap manager

Table 2 Sample of the Repair Iteration Section of the CASPER Telemetry Packet

Repair Iteration	
Telemetry Point	**Description**
Iteration counter	Number of repair iterations
Success	Indicates if the last plan modification was successful
Seconds Elapsed	Elapsed time for the last iteration
Pre Conflict Count	Number of conflicts prior to plan modification
Post Conflict Count	Number of conflicts after plan modification
Conflict Type	Type of last conflict
Conflict Start Time	Start time of conflict
Conflict End Time	End time of conflict
Resolution Method	Method used to modify the plan
Activity Instance ID	Instance of activity modified
Activity Schema ID	Schema of activity modified
Parameter Schema ID	Activity parameter being modified

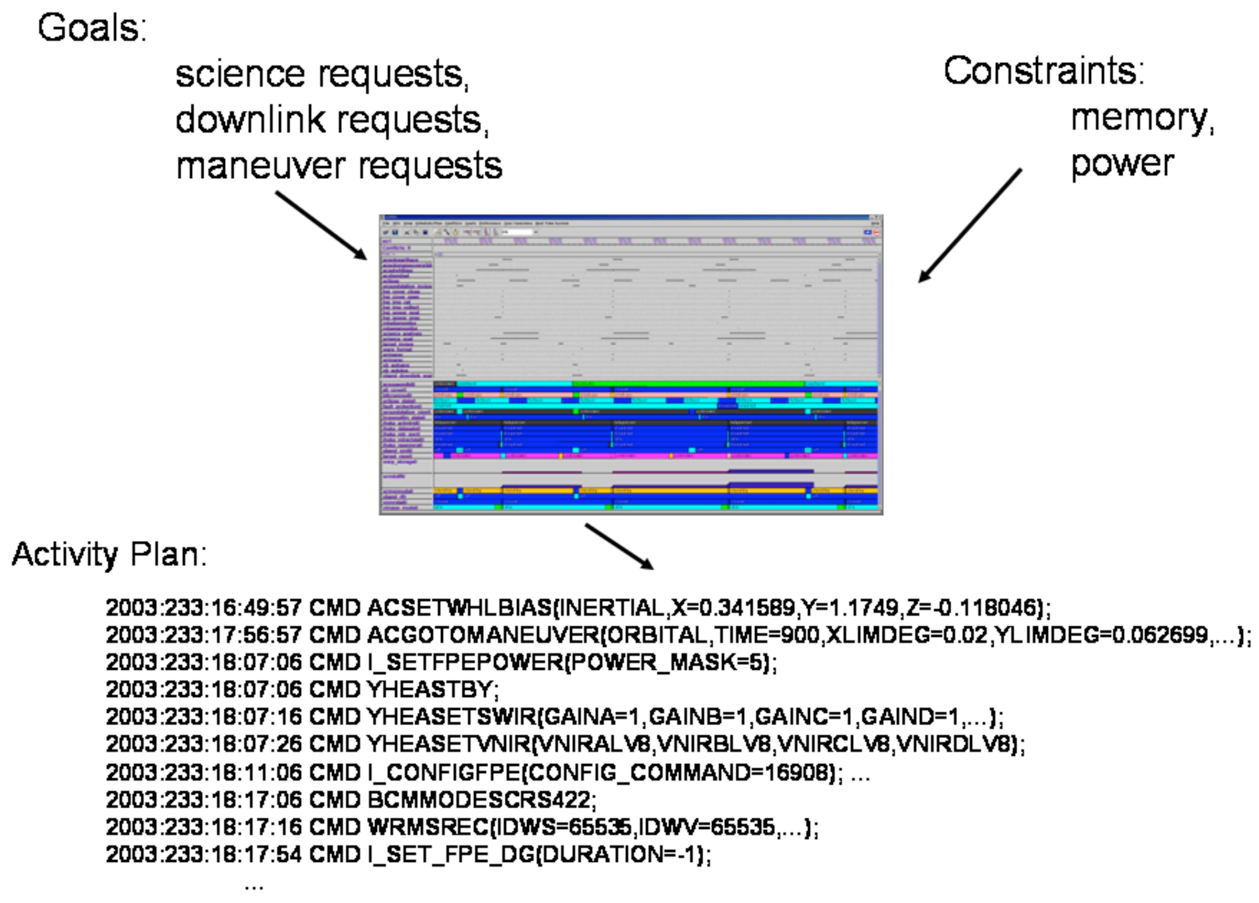
The timeline update section (see Table 3) of the telemetry packet contains all un-modeled updates to the CASPER planner during execution. As activities are inserted into the CASPER schedule, the future values of the spacecraft telemetry are modeled as timelines. Each timeline is constantly monitored to ensure what CASPER modeled is a true reflection of the state of the spacecraft. When the model differs from the spacecraft, updates to the timelines are inserted into the plan.

A common update to the CASPER planner occurs for the number of free memory blocks in the solid-state recorder after executing a science observation. Due to scarce computing resources, it is not uncommon for the instruments to collect data several seconds longer than planned, thus consuming more memory blocks. Therefore the spacecraft telemetry value for the number of free memory blocks differs from the value CASPER modeled. CASPER then updates the timeline to the new value.

Several other points of data were considered but omitted from the CASPER telemetry packet due to its limited size and time constraints.

1) Activity information – activity state information, start time, and unique identifiers are several of the parameters that could be saved in the telemetry packet.
2) CASPER input commands – the last ground command issued to CASPER can be logged in the packet as verification of receipt. Currently, we are examining the consequences of the command to determine if it was successful received.
3) Heuristic information – at each decision point in the repair iteration, weighted heuristics are used to select the correct value. Data indicating which heuristics were used would help in determining why decisions were made during repair.

Code execution – within critical areas of the CASPER planning code, saving what section of the code is executing would help in debugging. For example, if CASPER were to enter a section a code and loop forever, we are not currently able to determine where the code is "stuck".

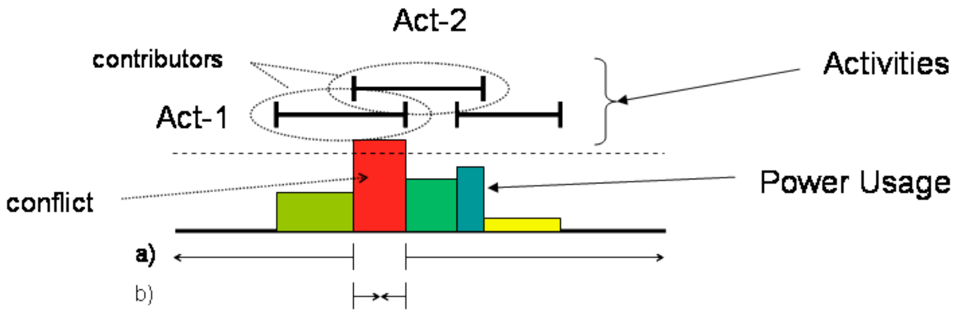


```
2003:233:16:49:57 CMD ACSETWHLBIAS(INERTIAL,X=0.341589,Y=1.1749,Z=-0.118046);
2003:233:17:56:57 CMD ACGOTOMANEUVER(ORBITAL,TIME=900,XLIMDEG=0.02,YLIMDEG=0.062699,...);
2003:233:18:07:06 CMD I_SETFPEPOWER(POWER_MASK=5);
2003:233:18:07:06 CMD YHEASTBY;
2003:233:18:07:16 CMD YHEASETSWIR(GAINA=1,GAINB=1,GAINC=1,GAIND=1,...);
2003:233:18:07:26 CMD YHEASETVNIR(VNIRALV8,VNIRBLV8,VNIRCLV8,VNIRDLV8);
2003:233:18:11:06 CMD I_CONFIGFPE(CONFIG_COMMAND=16908); ...
2003:233:18:17:06 CMD BCMMODESCRS422;
2003:233:18:17:16 CMD WRMSREC(IDWS=65535,IDWV=65535,...);
2003:233:18:17:54 CMD I_SET_FPE_DG(DURATION=-1);
               ...
```

Fig. 9 The planner takes goals such as science and engineering requests and constructs a plan that achieves the goals while respecting operations constraints such as memory, power, and timing.

Table 3 Sample of the Schedule Update Section of the CASPER Telemetry Packet

TimeLine Updates	
Telemetry Point	**Description**
Time Updated	Time plan was updated
Advanced Land Imager (ALI) Cover	The values of each telemetry point indicate how the schedule was updated
ALI Power State	
ALI Data Gate	
Hyperion Power State	
Hyperion Image Mode	
WARP mode	
WARP Free Blocks	
WARP Files In Use	
Downlink Rate	



Constraint	Property that must hold for plan to be valid	Must always use less power than available
Conflict	Violation of a constraint	Current plan uses more power than available over (b)
Repair Method	Modification to plan that may remove conflict	Delete activity using power during conflict (b)
Repair Choice	Which activity to delete	Delete largest user?

Fig. 10 The planner resolves problems with the plan (conflicts) by classifying the conflicts and applying modifications to the plan appropriate for the conflict type.

VIII. □ Agent Safety Requirements

Because of significant concerns for spacecraft health, ASE implements a layered redundant approach to enforcing spacecraft safety. This means that whenever possible at every level of the agent architecture, redundant checks are implemented to enhance spacecraft safety. Each of these safeguards has been reviewed by EO-1 spacecraft engineers, EO-1 operations personnel, as well as ASE team members (for a more detailed description of the model development, validation, and testing process, see Ref. 17). In addition, automated code generation techniques were used to develop SCL state & resource constraint checks directly from the CASPER model.

Table 1 below shows analysis of two spacecraft safety constraints. As shown, the operations team, the CASPER planner (via its model), SCL (via scripts and rules), and the EO-1 flight software (FSS) all implement constraints to protect the spacecraft from damage due to faulty commands or anomalies. In this manner, even if one of the layers malfunctions, the spacecraft may still be protected.

Because of the high stakes of EO-1 operations, significant effort also went into validating that the implemented ASE software enforced all of the designed constraints. The testing plan includes a number of cases to verify each constraint is enforced, as well the following general classes of test cases:

1) Coverage test cases that attempt to exercise a representative sample of all possible parameter-value assignments.
2) Stochastic test cases that verify nominal-operation scenarios.
3) Environmental test cases that evaluate how our agent performs in an uncertain environment.

Each build of the ASE software must be rigorously tested in a range of testbeds of increasing fidelity before flight (see Table 5.). The Solaris and Linux testbeds can be run at faster than real-time, however the GESPAC and EO-1 testbeds operate only at real-time.

Each build of the software must pass a pre-specified number of runs in order to be accepted for the next level of testbed testing. This begins with unit testing on workstations and culminates with integrated system runs on the EO-1 testbed prior to flight. These tests are quite time consuming. Typically a build requires 100 systems level tests on workstations. Each of these tests may represent hours to a week of operations time and several hours of CPU time. In order to investigate all anomalies from test runs and update software, it may take several hundred runs. Thus for each build the testing scheduled time is measured in weeks or months.

Table 4 Safety Analysis for Two Risks

	Instruments overheat from being left on too long	Instruments exposed to sun
Operations	For each turn on command, look for the following turn off command. Verify that they are within the maximum separation.	Verify orientation of spacecraft during periods when instrument covers are open.
CASPER	High-level activity decomposes into turn on and turn off activities that are with the maximum separation.	Maneuvers must be planned at times when the covers are closed (otherwise, instruments are pointing at the earth)
SCL	Rules monitor the “on” time and issue a turn off command if left on too long.	Constraints prevent maneuver scripts from executing if covers are open.
FSS	Fault protection software will shut down the instrument if left on too long.	Fault protection will safe the spacecraft if covers are open and pointing near the sun.

Table 5 Testbeds Available to Validate the EO-1 Agent

Type	Number	Fidelity
Solaris Sparc Ultra	5	Low – can test model but not timing
Linux 2.5 GHz	7	″
GESPAC PowerPC 100-450 MHz	10	Moderate – runs flight OS
EO-1 Flight Testbed Mongoose M5, 12 MHz	3	High – runs Flight Software

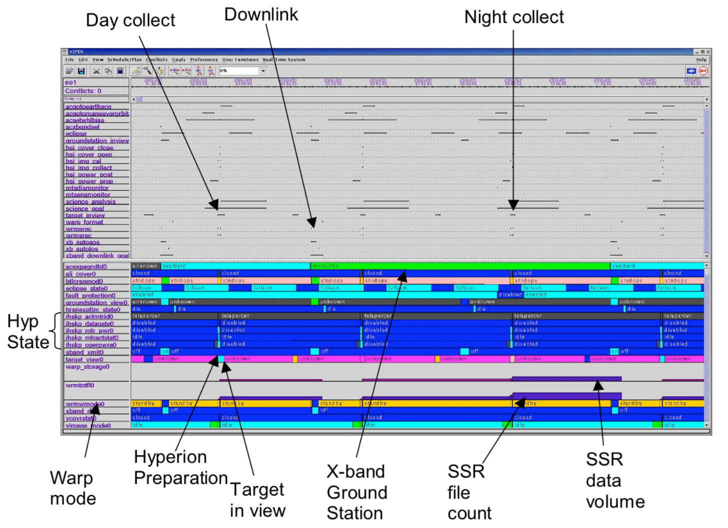


Fig. 11 An Earth Observing One mission plan involving several observations.

IX. Flight Status and Specific Scenario

The ASE software has been steadily progressing to full operations with the major milestones listed below.

Test Description	Test Date
Onboard cloud detection	March 2003
Onboard commanding path	May 2003
CASPER ground generated commands executed onboard	July 2003
Software jumping and loading	August 2003
ASE autonomously acquires calibration image and performs downlink	October 2003
ASE autonomously acquires science images and performs downlinks	Jan-Feb 2004 -
ASE autonomously analyzes science data onboard and triggers subsequent observations	April 2004 -
ASE achieves full mission success	14 May 2004
100 Hour ASE operations	September 2004
End of ASE Operations	January 2005+

In May 2004, ASE achieved its full success criteria which involved multiple demonstrations of onboard autonomy software performing integrated science with autonomous planning and execution. This software has been further enhanced to enable long duration autonomy tests and we are currently (August 2004) building to longer tests culminating in a one week long autonomous operations demonstration in the September 2004 timeframe.

An additional effort includes teaming with the NASA Ames Research Center to fly the Livingstone 2 Mode Identification and Diagnosis software[16] to be added to ASE in the August 2004 timeframe. The Livingstone 2 experiment would demonstrate tracking of multiple fault hypotheses, a capability not demonstrated in the Remote Agent Experiment in 1999. This effort is in earlier stages but is making good progress.

ASE is expected to fly at least through Spring 2005, at which time numerous extended duration tests will have flown.

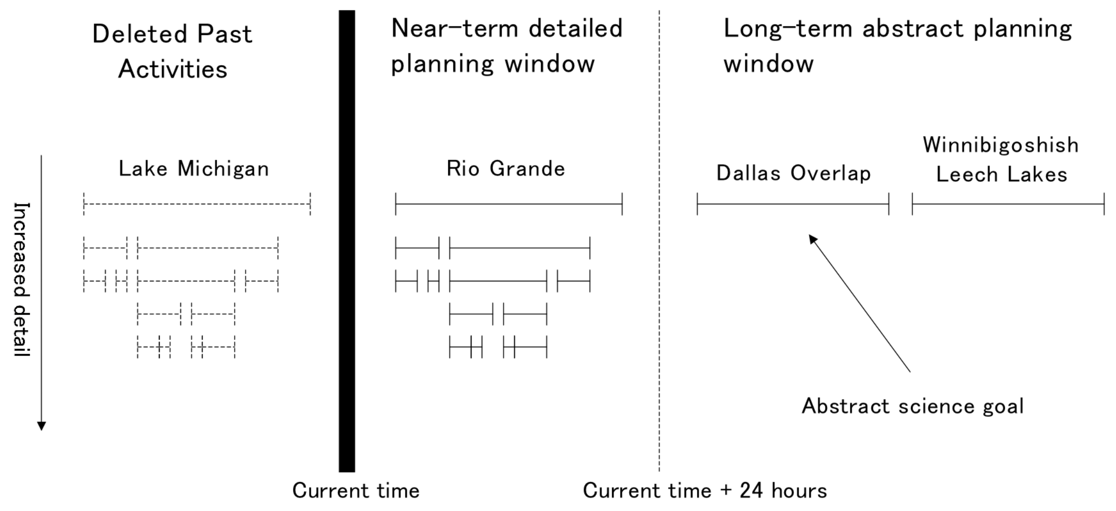


Fig. 12 CASPER plans incrementally, to avoid using large amounts of memory and CPU. (Courtesy of NASA)

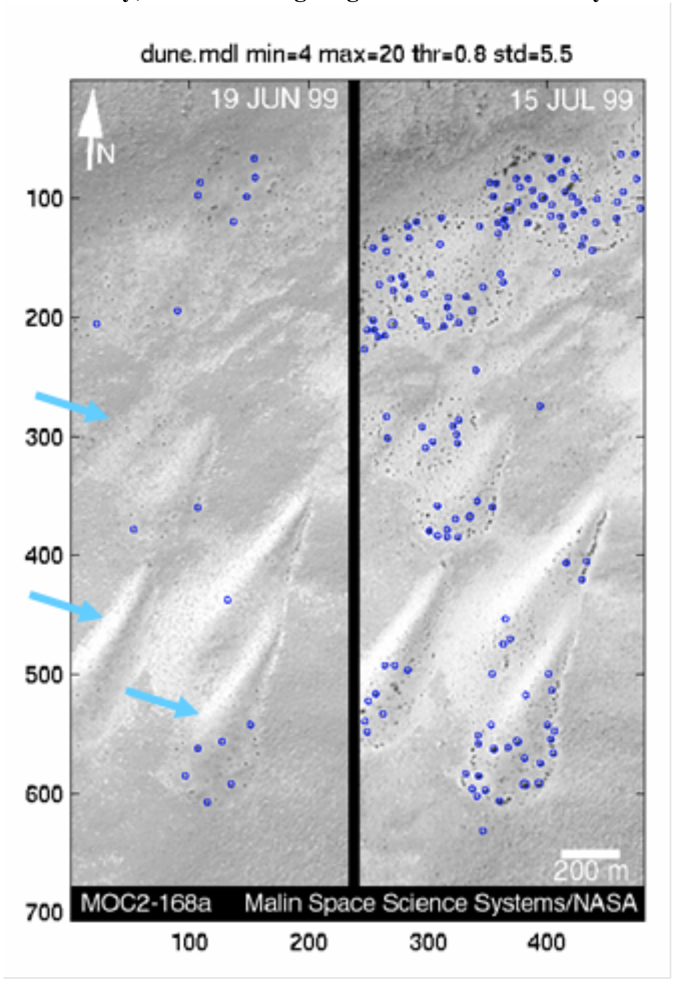

dune.mdl min=4 max=20 thr=0.8 std=5.5
19 JUN 99
15 JUL 99
N
200 m
MOC2-168a
Malin Space Science Systems/NASA
100
200
300
400
500
600
700
100
200
300
400

Fig. 13 Candidate Mars surface features for autonomous tracking. (Courtesy of NASA)

X. Contribution to Future Space Missions

While ASE demonstrates onboard science in an Earth-orbiting mission it has direct relevance to a large class of deep space missions. Specifically, the technologies demonstrated by ASE's use of onboard science processing has numerous applications to future Space Science Missions. For example, in Europa orbiter and lander missions, onboard science processing could be used to autonomously:

1) Monitor surface change as function of changing tidal stress fields
2) Monitor areas of greatest tidal stresses
3) Search for surface change, that is, evidence of recent activity
4) Search for landing sites that have a high probability of lander survivability and where the crust is thin enough for deployment of a sub-crust submarine explorer

The ASE Team has identified the NASA Mars Program as an ideal candidate for technology infusion of the ASE software. As a result, we have been working closely with the Mars Odyssey Project to identify and ground test science analysis algorithms that could be used for discovery of high-value science on Mars. One goal of this work is to have an existing or future Mars mission infuse the ASE software into their baseline flight software.

Many science applications for Mars have been identified. For example, onboard software could be used to monitor surface features on Mars such as: wind streaks, dune cape, dark slope streaks, and dust devils (see Figs. 13 and 14). In these applications onboard software could enable efficient search for such features, tracking of change in these features by comparison to an onboard catalogue, and tracking of transient phenomena such as dust devils (see Fig. 15).

Many science applications for other planets exist as well. Sample applications include tracking of volcanic activity at Io (See Fig. 16), tracking crustal change at Europa, and tracking cryo-volanism at Triton.

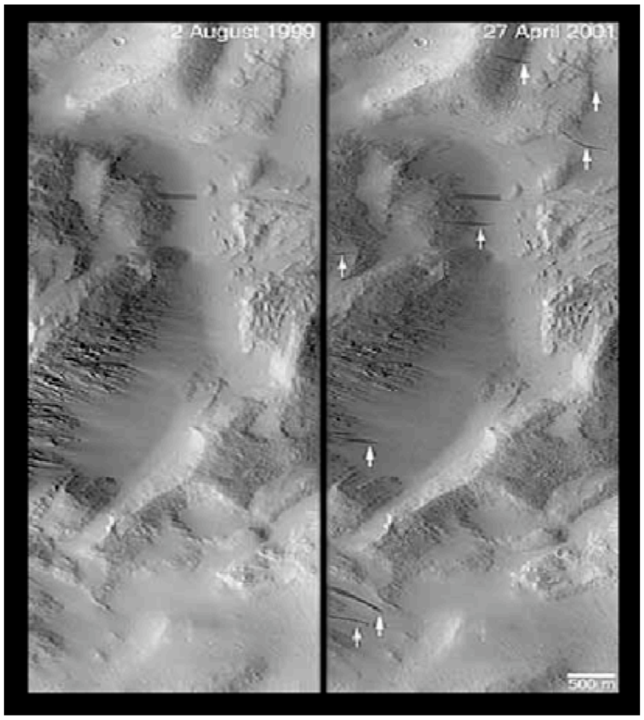


Fig. 14 More candidate Mars surface features (dark slope streaks) for autonomous tracking. (Courtesy of NASA)

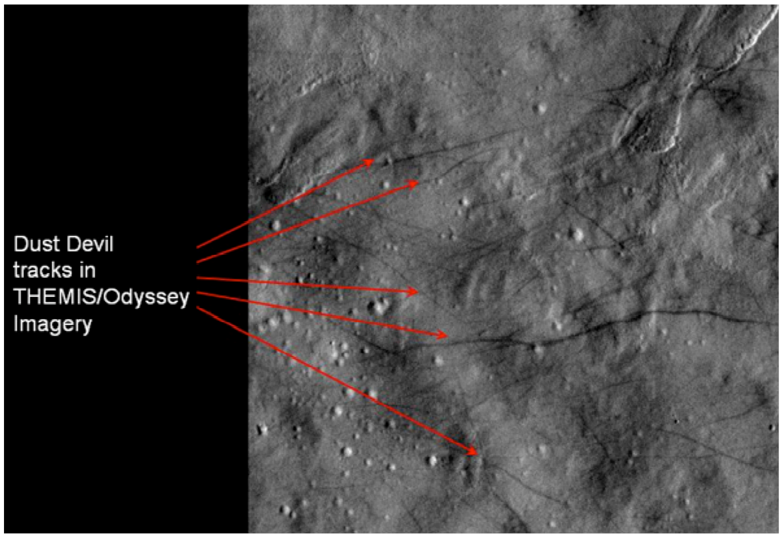


Fig. 15 Dust devils – another candidate for autonomous tracking. (Courtesy of NASA)

XI. Related Work

In 1999, the Remote Agent experiment (RAX)[13] executed for a few days onboard the NASA Deep Space One mission. RAX is an example of a classic three-tiered architecture,[7] 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 [ESA] is a European Space Agency (ESA) mission that uses onboard autonomy and was launched in 2001. However, ASE has more of a focus on model-based autonomy than PROBA.

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.[2] 3CS has been delayed several times because it is a shuttle launch and currently scheduled for launch in July 2004. 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.

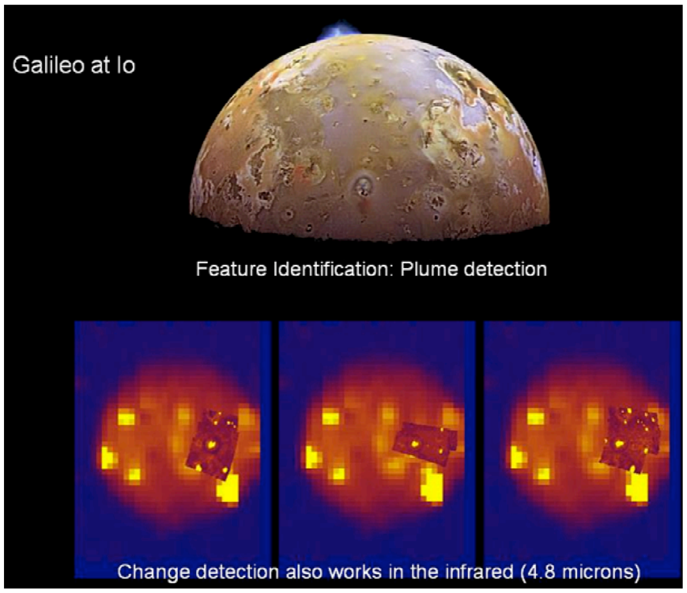

Galileo at Io
Feature Identification: Plume detection
Change detection also works in the infrared (4.8 microns)

Fig. 16 Autonomous systems could be used to track volcanic activity on Io, the most actively volcanic body in our solar system. (Courtesy of NASA)

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.[3] 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) and the change in the flight software Operating System from OSE to VxWorks. The instrument change required significant alteration to the science targets and analysis algorithms. The flight OS change required a new integration with the flight software. The basic software architecture and components (e.g. CASPER and SCL) have remained the same. It is a testament to the generality of the ASE architecture that the ASE team was able to switch carriers and re-integrate to achieve full mission success within the original cost cap of the Autonomous Sciencecraft Experiment.

XII. Future Work and Conclusions

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 enables a reduction in data downlink. In addition, the ASE increases science return through autonomous retargeting. Demonstration of these capabilities onboard EO-1 enables 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.

Acknowledgment

We would like to thank the complete ASE Team including Rachel Lee (Jet Propulsion Laboratory), Jerry Hengemihle (Microtel), Jeff D’Agostino (The Hammers Company), Stephen Ungar and Thomas Brakke (Goddard Space Flight Center), Jim Van Gaasbeck (Interface & Control Systems), Ronald Greeley and Thomas Doggett (Arizona State University), Victor Baker, James Dohm, Felipe Ip, (University of Arizona)

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.

References

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

[2]Chien, S., Engelhardt, B., Knight, R., Rabideau, G., Sherwood, R., Hansen, E., Ortiviz, A., Wilklow, C., and Wichman, S., “Onboard Autonomy on the Three Corner Sat Mission,” *Proceedings of the i-SAIRAS 2001*, June 2001.

[3]Chien, S., Sherwood, R., Rabideau, G., Castano, R., Davies, A., Burl, M., Knight, R., Stough, T., Roden, J., Zetocha, P., Wainwright, R., Klupar, P., Van Gaasbech, J., Cappelaere, P., and Oswald, D., “The Techsat-21 Autonomous Space Science Agent,” *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference,* July 2002.

[4]Chien, S., Knight, R., Stechert, A., Sherwood, R., and Rabideau, G., “Using Iterative Repair to Improve Responsiveness of Planning and Scheduling,” *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, April 2000. Also available on casper.jpl.nasa.gov (cited Dec. 2004).

[5]Davies, A. G., Greeley, R., Williams, K., Baker, V., Dohm, J., Burl, M., Mjolsness, E., Castano, R., Stough, T., Roden, J., Chien, S., and Sherwood, R., "ASC Science Report," Aug. 2001. Available online at ase.jpl.nasa.gov (cited Dec. 2004).

[6]Davies, A. G., Mjolsness, E. D., Gray, A.G., Mann, T. F., Castano, R., Estlin, T. A., and Saunders, R. S., “Hypothesis-driven Active Data Analysis of Geological Phenomena Using Semi-Autonomous Rovers: Exploring Simulations of Martian Hydrothermal Deposits,” EOS – Transactions of the American Geophysical Union, Vol. 80, No. 17, 1999, pp. S210.

[7]Gat, E., “On Three-Layer Architectures,” *Artificial Intelligence and Mobile Robots,* edited by D. Kortenkamp, R. P. Bonnasso, and R. Murphy, AAAI Press, 1998.

[8]Goddard Space Flight Center, EO-1 Mission page. Available online at eo1.gsfc.nasa.gov (cited Apr. 2005).

[9]Griffin, M., Burke, H., Mandl, D., and Miller, J., "Cloud Cover Detection Algorithm for the EO-1 Hyperion Imagery," *Proceedings of the 17th SPIE AeroSense 2003*, 21-25 April 2003.

[10]Interface and Control Systems, SCL Home Page. Available online at http://www.interfacecontrol.com (cited Apr. 2005).

[11]Malin, M. C., and Edgett, K. S., "Evidence for Recent Groundwater Seepage and Surface Runoff on Mars," *Science*, Vol. 288, 2000, pp. 2330-2335.

[12]Muscettola, N., Dorais, G., Fry, C., Levinson, R., and Plaunt, C., "IDEA: Planning at the Core of Autonomous Reactive Agents," *Proceedings of the Workshops at the AIPS-2002 Conference*, April 2002.

[13]NASA Ames, Remote Agent Experiment Home Page. Available online at http://ic.arc.nasa.gov/projects/remote-agent/ (cited Apr. 2005). See also Muscettola, Nicola, Nayak, P. Pandurang, Pell, Barney, and Williams, Brian, "Remote Agent: To Boldly Go Where No AI System Has Gone Before," *Artificial Intelligence*, Vol. 103, No. 1-2, Aug. 1998, pp. 5-48, August 1998

[14]The PROBA Onboard Autonomy Platform, http://www.estec.esa.nl/proba/ (cited Mar. 2005).

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

[16]Kurien, J., and Nayak, P., "Back to the Future for Consistency-based Trajectory Tracking," *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI'2000)*, 2000.

[17]Cichy, B., Chien, S., Schaffer, S., Tran, D., Rabideau, G., Bote, R., Mandl, Dan, Frye, S., Shulman, S., Van Gaasbeck, J., and Boyer, D., "Validating the EO-1 Autonomous Science Agent," *International Workshop on Planning and Scheduling for Space*, June 2004.

[18]Chien, S., Sherwood, R., Tran, D., Castano, R., Cichy, B., Davies, A., Rabideau, G., Tang, N., Burl, M., Mandl, D., Frye, S., Hengemihle, J., Agostino, J., Bote, R., Trout, B., Shulman, S., Ungar, S., Van Gaasbeck, J., Boyer, D., Griffin, M., Burke, H., Greeley, R., Doggett, T., Williams, K., Baker, V., and Dohm, J., "Autonomous Science on the EO-1 Mission," *International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS 2003)*, May 2003.

[19]Tran, D., Chien, S., Rabideau, G., and Cichy, B., "Flight Software Issues in Onboard Automated Planning: Lessons Learned on EO-1," *International Workshop on Planning and Scheduling for Space*, June 2004.

[20]Vapnick, V., *Statistical Learning Theory*, Wiley, 1998.

[21]Scholkopf, B., and Smola, AJ, *Learning with Kernels*, MIT Press, Cambridge, MA, 2002

[22]Castano, R., Mazzoni, D., Tang, N., Doggett, T., Chien, S., Greeley, R., Cichy, B., and Wagstaff, K., *Learning Classifiers for Science Event Detection in Remote Sensing Imagery*, JPL Technical Document D-31140. Available online at ml.jpl.nasa.gov (cited Apr. 2005).