OPERATIONAL LESSONS LEARNED FROM THE AUTONOMOUS SCIENCECRAFT EXPERIMENT

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

Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Dr., Pasadena, CA 91109, U.S.A., Email: firstname.lastname@jpl.nasa.gov

ABSTRACT/RESUME

The Autonomous Sciencecraft Experiment (ASE) is currently flying onboard the Earth Observing One (EO-1) Spacecraft. This software enables the spacecraft to autonomously detect and respond to science events occurring on the Earth. The package includes software systems that perform science data analysis, deliberative planning, and run-time robust execution. Because of the deployment to the EO-1 spacecraft, the ASE software has stringent constraints of autonomy and limited computing resources. We describe these constraints and how they are reflected in our operations approach. A summary of the final results of the experiment is also included. This software has demonstrated the potential for space missions to use onboard decision-making to detect, analyze, and respond to science events, and to downlink only the highest value science data. As a result, ground-based mission planning and analysis functions have been greatly simplified, thus reducing operations cost. The operational cost savings are detailed within this paper.

1. INTRODUCTION

Since January 2004, the Autonomous Sciencecraft Experiment (ASE) running on the EO-1 spacecraft has demonstrated several integrated autonomy technologies to enable autonomous science. Several science algorithms including: onboard event detection, feature detection, change detection, and unusualness detection are being used to analyze science data. These algorithms are used to downlink science data only on change, and detect features of scientific interest such as volcanic eruptions, growth and retreat of ice caps, cloud detection, and crust deformation. These onboard science algorithms are inputs to onboard decision-making algorithms that modify the spacecraft observation plan to capture high value science events. This new observation plan is then 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 specifics of the ASE and relates it to past and future flights to validate and mature this technology.

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 (SCL Web Page, 2005) package to enable event-driven processing and low-level autonomy
- The Continuous Activity Scheduling Planning Execution and Replanning (CASPER) (Chien, 2000) 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 has already been demonstrated on 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, ice melt, and 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.

Although the ASE software is running on the Earth observing spacecraft EO-1, the long-term goal is to use this software on future interplanetary space missions. On these missions, onboard science analysis will enable capture of short-lived science phenomena. In addition, onboard science analysis will enable data 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 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.

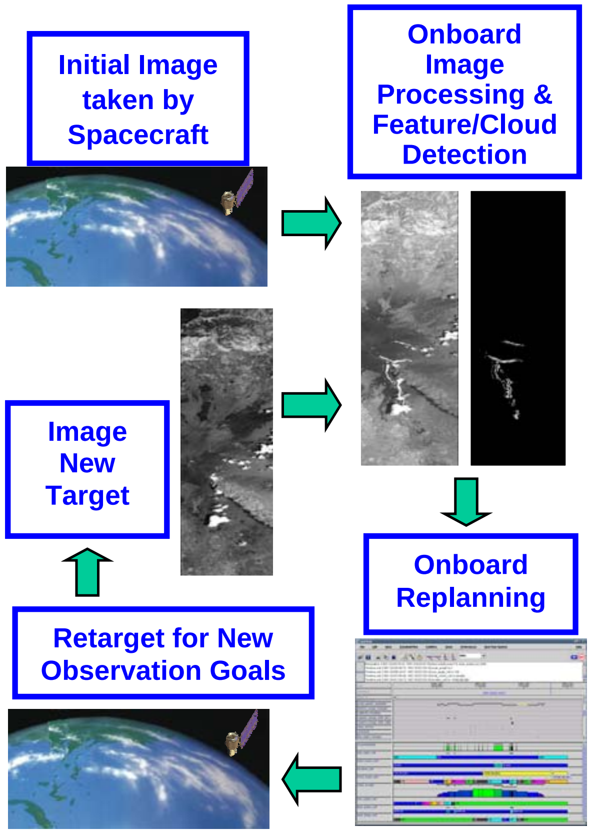
Initial Image taken by Spacecraft
Onboard Image Processing & Feature/Cloud Detection
Image New Target
Onboard Replanning
Retarget for New Observation Goals

Figure 1. Autonomous Science Mission Concept

A typical ASE demonstration 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. Based on this detection the image is downlinked. Had no new lava flow been detected, the science software would generate a goal for the planner to acquire the next highest priority target in the list of targets. (See Fig. 1.) The addition of this goal to the current goal set triggers CASPER to modify the current operations plan to include numerous new activities in order to enable the new science observation.
5. The SCL software executes the CASPER generated plans in conjunction with several autonomy elements.
6. This cycle is then repeated on subsequent observations.

However, building autonomy software for space missions has a number of key challenges; many of these issues increase the importance of building a reliable, safe, agent. Some of these issues include:

1. Limited, intermittent communications to the agent. A typical spacecraft in low earth orbit (such as EO-1) has 8 communications opportunities per day, each lasting about 10 minutes. 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 the 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.

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 Web Page, 2005). The primary focus of EO-1 is to develop and test a set of advanced technology land imaging instruments. EO-1 was launched on a Delta 7320 from Vandenberg Air Force Base on November 21, 2000. It was inserted into a 705 km circular, sun-synchronous orbit at a 98.7 degrees inclination. This 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. For each scene, between 13 to as much as 48 Gbits of data from the Advanced Land Imager (ALI), Hyperion, and Atmospheric Corrector (AC) are collected and stored on the onboard solid-state data recorder.

EO-1 is currently in extended mission, having more than achieved its original technology validation goals. As an example, over 18,000 data collection events have been successfully completed, against original success criteria of 1,000 data collection events. The ASE described in this paper uses the Hyperion hyper-spectral instrument. 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 images a 7.7 km by 42 km land area per image and provides detailed spectral mapping across all 220 channels with high radiometric accuracy.

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. ONBOARD SCIENCE ANALYSIS

The first step in the autonomous science decision cycle is detection of interesting science events. In the complete experiment, a number of science analysis technologies have been flown including:

- Thermal anomaly detection – uses infrared spectra peaks to detect lava flows and other volcanic activity. (See Fig. 3a.)
- Cloud detection (Griffin, 2003) – uses intensities at six different spectra and thresholds to identify likely clouds in scenes. (See Fig. 3b.)
- Flood scene classification – uses ratios at several spectra to identify signatures of water inundation as well as vegetation changes caused by flooding.
- 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. (See Fig. 3c.)

Fig. 3a 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 area of hot pixels can be compared with the count of hot pixels from a previous image of the same area to determine if change has occurred. If there has been change, a new image might be triggered to get a more detailed look at the eruption.

Fig. 3b shows a Hyperion scene and the results of the cloud detection algorithm. This MIT Lincoln Lab developed algorithm is able to discriminate between cloud pixels and land pixels within an image. Specifically, the gray areas in the detection results are clouds while the blue areas are land. The results of this algorithm can be used to discard images that are too cloudy.

Fig. 3c contains 4 images. The top two are detailed Hyperion images taken of the Larson Ice Shelf in Antarctica on 4/6/2002 and 4/13/2002. A large change in the ice shelf is seen in comparing the images. The bottom 2 images are results of the land-ice-water detection algorithm. The white area of the image is ice and the blue area is water. The ice and water pixels can be counted and compared with the second image to determine if change has occurred. If change is detected, the image can be downlinked and further images of the area can be planned.

The onboard science algorithms are limited to using 12 bands of the hyperion instrument. Of these 12 bands, 6 are dedicated to the cloud detection algorithm. The other six are varied depending on which science algorithm is used. The images used by the algorithm are "Level 0.5," an intermediate processing level between the raw Level 0, and the fully ground processed Level 1. Each of the science algorithms except the generalized feature detection use simple threshold checks on the spectral bands to classify the pixels.

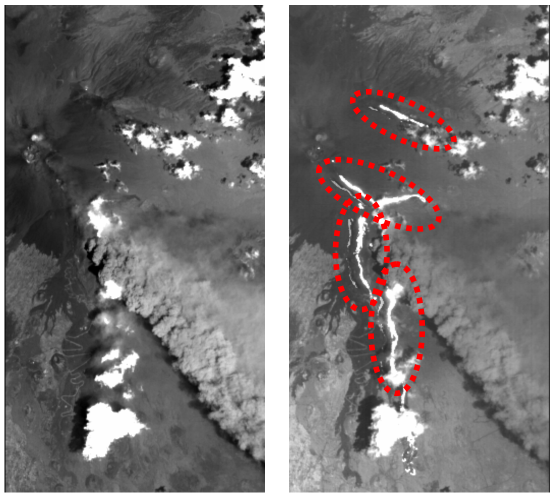
Figure 3a. Thermal Anomalies associated with volcano activity at Mt. Etna, visual spectra at left and infrared at right.

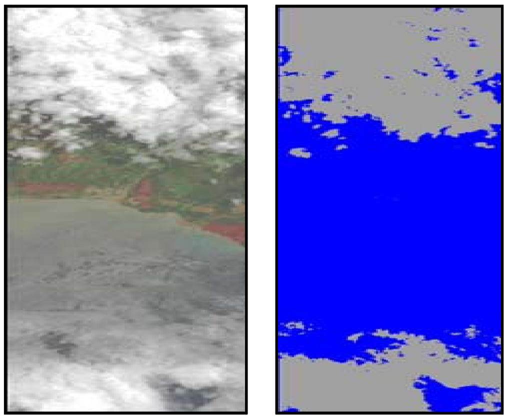
Figure 3b. Cloud Detection of a Hyperion Scene – visual image at left, grey in the image at right indicates detected cloud.

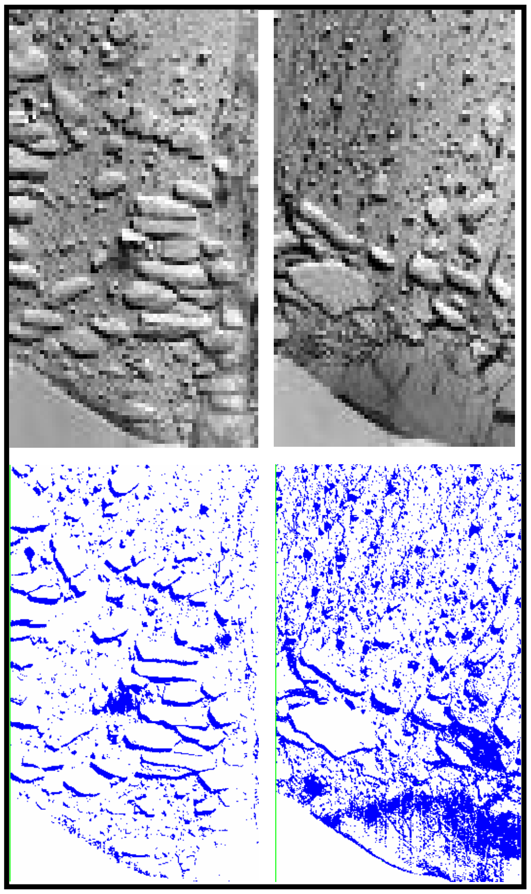
Figure 3c. Change Detection Scenes indicating Ice Breakup in the Larsen Ice Shelf, Antarctica.

Initial experiments used the cloud detection triggers. The MIT Lincoln Lab developed cloud detection algorithm (Griffin, 2003) uses a combination of spectral bands to discriminate between clouds and surface features. The Hyperion Cloud Cover (HCC) algorithm was run on all images acquired during ASE experiments. In the event of high cloud cover, the image could be discarded and a new goal could be sent to CASPER to re-image the area or image another high priority area. Images with low cloud cover can either be downlinked or analyzed further by other ASE science algorithms.

The JPL developed thermal anomaly algorithms uses the infrared spectral bands to detect sites of active volcanism. There are two different algorithms, one for daytime images and one for nighttime images. The algorithms compare the number of thermally active pixels within the image with the count from a previous image to determine if new volcanism is present. If no new volcanism is present, the image can be discarded onboard. Otherwise, the entire image or the interesting section of the image can be downlinked.

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 previous image to determine if flooding has occurred. 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. We have noticed a few problems when ground testing this algorithm with existing Hyperion data. The presence of clouds or heavy smoke within an image can cause the algorithm to fail.

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.

4. ONBOARD MISSION PLANNING

In order for the spacecraft to respond autonomously to the science event, it must be able to independently perform the mission planning function. This requires software that can model all spacecraft and mission constraints. The Continuous Activity Scheduling Planning Execution and Replanning (CASPER) (Chien, 2000) software performs this function for ASE. CASPER represents the operations constraints in a general modeling language and reasons about these constraints to generate new operations plans that respect spacecraft and mission constraints and resources. CASPER uses a local search approach (Rabideau, 1999) 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. Since the software was already written and we didn’t have funding to make major changes in the software, we had to focus on developing an EO-1 CASPER model that didn’t require a lot of planning iterations. For that reason, the model has only a handful of resources to reason about. This ensures that CASPER is able to build a plan in tens of minutes on the relatively slow CPU.

CASPER is responsible for mission planning in response to both science goals derived onboard as well as anomalies. In this role, CASPER must plan and schedule activities to achieve science and engineering goals while respecting resource and other spacecraft operations constraints. For example, when acquiring an initial image, a volcanic event is detected. This event may warrant a high priority request for a subsequent image of the target to study the evolving phenomena. In this case, CASPER modifies the operations plan to include the necessary activities to re-image. This may include determining the next over flight opportunity, ensuring that the spacecraft is pointed appropriately, that sufficient power, and data storage are available, that appropriate calibration images are acquired, and that the instrument is properly prepared for the data acquisition.

5. ONBOARD ROBUST EXECUTION

ASE uses the Spacecraft Command Language (SCL) (SCL Web Page, 2005) 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, which is part of SCL, 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.

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 the EO-1 experiment, SCL 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 as messages on the software bus.

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). SCL also verifies that there is sufficient power so that the command does not trigger a low bus voltage condition and that there is sufficient energy in the battery. Using SCL to check these constraints and including them in the CASPER model provides an additional level of safety to the autonomy flight software.

6. FLIGHT STATUS

The ASE software was integrated under the flight version of VxWorks in December 2002, and has since been integrated and tested with the WARP flight software. We tested the individual software components in isolation to gain confidence before we performed an integrated flight test.

The cloud detection algorithms were tested onboard in March 2003. The SCL software was tested onboard in May 2003. This test involved starting up the SCL software, testing the software bridge between the SCL software bus and WARP software bus, testing the SCL message and telemetry logs, testing the sending of commands, and testing the sending and executing of commands that performed a dark calibration of the Hyperion instrument.

In July 2003, a ground version of CASPER generated several plans that were subsequently uplinked and executed onboard. These plans included image data takes, maneuvers, and telecommunication passes. The purpose of this test was to prove that CASPER could generate valid plans that could be executed by the satellite.

In August 2003, onboard decompression was tested. This capability is used to compress the software before uplink because the uplink rate is only 2 Kb/s. Without compression it would take more than a week to upload the entire ASE software. This test involved uplinking several compressed files, decompressing them onboard, and then downlinking them. The files were then checked for errors.

The ASE software has been flying onboard the EO-1 spacecraft since January 2004. In January and February 2004, we tested several autonomous instrument data acquisition experiments using CASPER/SCL. This test involved uplinking a high level goal that includes a target location and a few instrument mode parameters. We have steadily increased the level of autonomy since this period. In April 2004, we started the first closed-loop execution where ASE autonomously analyzes science data onboard and triggers subsequent observations. So far, we have run over 200 of these trigger experiments with over 2000 autonomously planned image data takes. The ASE software is now the baseline planning software for EO-1, and has been running almost continuously onboard for several months.

7. PREPARING THE ASE FOR SPACE FLIGHT

Given the many challenges to developing flight software, this section discusses several issues encountered in preparing the CASPER planner for flight. Specifically, we describe:

- Reducing the CASPER image size – With infrequent and short ground contacts and limited available memory, we needed to reduce the CASPER image size. We discuss our strategies to reduce the CASPER image size.
- Approach to long term planning – CASPER must be able to autonomously plan for a week's worth of EO-1 activities, which includes over 100 science observations. We discuss how this is achieved within the available memory and CPU.
- Speed improvements to meet autonomy requirements – Several model and code optimizations were performed to increase the running speed of ASE.

In addition, we have performed several optimizations on the data collected relating to the state and actions of the planner.

7.1 Reducing the CASPER image size

CASPER's core planning engine is the Automated Scheduling and Planning Environment (ASPEN) ground-based planner. ASPEN is a re-usable framework, which is capable of supporting a wide variety of planning and scheduling applications. It provides a set of software components commonly found in most planning systems such as: an expressive modeling language, resource management, a temporal reasoning system, and support of a graphical user interface. Because of limited onboard computing memory, we had to reduce the image size. CASPER developers took two approaches to reducing the image size: removing unneeded components and reducing code image size inefficiencies. Prior to this work, the image size of CASPER was 12MB.

The CASPER development team went through the core software and removed each software component deemed unnecessary for flight. Several modules removed from the CASPER code include:

- Backtracking Search – The ASPEN framework provides several search algorithms that perform backtracking search. On ASE, we have decided to use the repair search algorithm, so these other algorithms were not needed.
- Optimization – CASPER provides the capability to optimize the schedule based on several preferences

defined by mission planners. However, we have decided not to use this functionality for ASE.

- GUI Sockets – Because ASPEN is a ground-based planner, it provides a GUI for visualizing the schedule and interacting with it. Communication with this GUI is done through the ASPEN socket interface. In flight, support for a GUI is not necessary.
- General Heuristics – The ASPEN core contains multiple sets of generic heuristics that have been found to be useful across multiple projects. CASPER for ASE requires a subset of these heuristics; therefore, the unused sets were removed.
- Generalized Timelines – Generalized timelines provides a general infrastructure to model complex state variables and resources. This infrastructure was not required for ASE and was removed.

Removing software components trimmed approximately 3MB from the CASPER image size.

CASPER also makes heavy use of the Standard Template Library (STL), specifically the containers provided. STL templates are widely known to increase code size in C++ because for each container defined in CASPER, the code may be duplicated several times. There exist various compiler techniques available that attempts to minimize the duplication. To minimize the impact of code bloat, we re-implemented the STL container and functions used in the CASPER code. This re-implementation, dubbed “lite STL”, was developed to minimize the code generation, trading space for execution time. We were able to remove approximately 3MB from the CASPER image using this strategy.

Along with simple compiler optimization, removing unneeded software components, and reducing the impact of code duplication, the final size of the CASPER image was reduced to 5MB.

7.2 Approach to long term planning

One of the scenarios planned for ASE is autonomous control of EO-1 for a week. This requires CASPER to support generation of a valid schedule for a week’s worth of EO-1 operations. During a nominal week, EO-1 averages over 100 science observations and 50 S-Band/X-Band ground contacts. The size of this problem presents a challenge to CASPER, given the limited memory and CPU constraints.

While most desktop workstations have several GB’s of memory available, CASPER on EO-1 is constrained with a 32MB heap. As result, we need to ensure that generation of a week’s plan does not exhaust all available heap space. A science observation is the most complex activity within the CASPER model, consisting of over 78 activities. Planning a week’s worth of operation would require scheduling over 7800 activities (not including downlink and momentum management activities) and exhaust our heap space.

Also, as the number of goals in the schedule increase, the computation time to schedule a goal will also increase, due to the interaction between goals. On EO-1, this problem is exacerbated with an 8 MIPS processor, of which 4 MIPS are shared by the SCL, CASPER, and science processing software.

To resolve the problems with CPU and memory consumption, CASPER utilizes a hierarchal planning approach with focused planning periods. CASPER performs abstract planning and scheduling of observations for the entire week, such as ensuring a constraint of one science observation per orbit. It also performs near-term planning for the next 24 hours by detailing the science observations to the low-level activities. This near-term planning window is continuously updated to include the next 24 hours of the schedule and as past observations exit the planning window, they are automatically removed from the plan. By reducing the number of science observations that need to be scheduled and detailed to a 24 hour period, we reduce memory and CPU consumption.

7.3 Speed Improvements to Meet Autonomy Requirements

The ASE experiment is constrained by the computing environment onboard EO-1. Because each of the EO-1 software builds is a single static image, all ASE components that dynamically allocate RAM require their own memory manager. SCL contains a memory manager previously used on the FUSE mission. CASPER uses a separate memory manager adapted from JPL’s Deep Impact mission. However, performance from early flight tests indicated that the SCL memory manager was significantly hampering performance, so SCL was switched to use the same memory manager as CASPER (but with its own heap space). Note that these memory managers had to not only allocate and de-allocate memory quickly but also not suffer from longer-term issues such as fragmentation.

The limited onboard computing power required changes to the SCL and CASPER models to meet operational timing constraints. For example, initially within SCL a much larger set of safety constraints was modeled and execution was designed to be much more closed loop. However, testbed runs and early flight tests indicated that telemetry delays and CPU bottlenecks meant that this design was delaying time-sensitive commands. Most importantly, instrument on-times were delayed (e.g. late) and too long (resulting in extra data acquired). The ASE team was forced to both streamline the code (including the memory manager modification) and streamline the model to speed execution.

The CASPER planner is a significant user of onboard CPU. When CASPER is planning future observations it utilizes all of the available CPU cycles and takes approximately 8 minutes to plan each observation. The CASPER model was designed to operate within a minimal CPU profile – and as a result observations are planned with less flexibility. By setting fixed values for temporal offsets between activities rather than retaining flexible offset times, search is reduced and response time improved at the cost of plan quality (in some cases). For example, an image take activity may require a camera heater warm up before the camera can operate. The heater may take 30-60 seconds to warm the camera up to its operational temperature. By setting the duration of the heater warm up activity to 60 seconds, the temporal offset between the heater warm up activity and the image data take activity is fixed at 60 seconds, rather than variable.

Other performance improvements for CASPER came from analysis of the running code. We found bottlenecks and made improvements in redundant calculations. In particular, this was critical for functions performed on every loop of CASPER (such as collecting conflicts). We made some simplifying assumptions to make some expensive calculations faster. For example, when initially scheduling activities, we ignore timeline constraints, assuming that temporal constraints are more critical than timelines (calculating valid start times for timelines can be expensive).

8. IMPACT ON OPERATIONS

ASE can impact several aspects of spacecraft operations. The mission planning process is simplified because the operations team no longer has to build detailed sequences of commands. The spacecraft can be commanded using high-level goals, which are then detailed by the planner onboard. The processes of planning, build sequence, upload sequence, execute sequence, downlink data, analyze data, and build new sequence are entirely automated using ASE. For example, in the current EO-1 operations, a significant percentage of the images downlinked are of no value because they are mostly covered in clouds. Using ASE, these images can now be discarded onboard and the satellite can acquire another image of a different area. This saves time and labor for the mission planning team, science analysis team, ground station team, flight operations team, and data processing and archive team.

Due to computing limitations, the ASE architecture for EO-1 does not include an autonomous fault protection component. Although this wasn't included for EO-1, it's a natural fit for the ASE onboard autonomy software.

In one example, CASPER generates a mission level plan that includes a sequence of behavior goals, such as producing thrust. The SCL executive is responsible for reducing these goals to a control sequence, for example, opening the relevant set of valves leading to a main engine. A device, such as a valve, is commanded indirectly; hence, SCL must ensure that the components along the control path to the device are healthy and operating before commanding that device. Components may be faulty, and redundant options for achieving a goal may exist; hence, SCL must ascertain the health state of components, determine repair options when viable, and select a course of action among the space of redundant options. Adding this level of fault protection autonomy to a future mission could in theory, eliminate the spacecraft analysis team. The team would no longer be required to monitor the spacecraft health because that would be done onboard using model-based mode estimation and mode reconfiguration. The team would also not be required to respond to "safe-hold" periods because anomalies would be handled and reconfigured onboard. Using this software requires a greater up front investment in building the spacecraft models, but much of the underlying software has already been developed in research efforts.

By combining automated planning with onboard science analysis and smart execution, an even more dramatic reduction is sequencing effort is obtained due to the reduction in sequences created in response to ground based science data analysis. These sequences are created by ASE onboard the spacecraft without ground interaction. The feature detection algorithms onboard can identify specific features of interest within the images. The spacecraft can then downlink the entire image when features are detected, only the detected features, or even a summary of the detected features. Scientists no longer have to analyze many different images to find a feature of interest. In fact, images that do not contain features of interest do no even have to be downlinked. These algorithms can be particularly useful on bandwidth-limited missions by returning the most important science data.

For the specific case of using the ASE software on EO-1, science return per data downlink was increased by over 100x by rapid response and returning the most important science data. (See Tab. 1.)

For specific cost savings (or value added) from increased science return from ASE, we submit the following analysis. To compute an economic value to the baseline EO-1 science return we use a conservative estimate based on the minimum ($1000/image) cost for scenes, along with the typical number of paid images per day (8), and a conservative estimate of science operations days per month (some are lost for engineering operations):

$1000/image x 8 images/day x 25 days/month x 12 months/year = $2.4M/year

Table 1. Downlink Data Savings by Science Process

Process	Total Process Data Acquired	Data returned by ASE	Downlink Savings	Savings Factor (goal was x10)
Volcanism	33750 MB	294 MB	33456 MB	115
Cyrosphere (ice)	38100 MB	304 MB	37796 MB	125
Flooding	25500 MB	239 MB	25261 MB	106
Total	**97350 MB**	**837 MB**	**96513 MB**	**116**

We take this as a conservative estimate of the value of the science return from conventional EO-1 Operations. Assuming a conservative science increase of 10x (compared to the documented increase of over 100x), ASE has increased the science return of EO-1 as follows:

science return with ASE – science return without ASE =
10x $2.4M/yr – 1x $2.4M/yr = $21.6M/yr

Reducing the ground operation team, which no longer has to prepare detailed spacecraft command sequence files, saves additional costs. In the case of EO-1, the mission manager figured the labor costs for the ground operations team were reduced from $2.5M/year to $1.0M/year as a result of using the ASE software and automating the ground system as a result.

Over a 5-year mission, using EO-1 as the example, the ASE software has the potential for saving $115.5M, as detailed in Tab. 2.

Table 2. Total Operational Savings Using ASE

Science value increase	$21.6M/yr x 5 yrs	= $108.0M
Operations cost reduction	$ 1.5M/yr x 5 yrs	= $7.5M
Total Savings		**= $115.5M**

9. RELATED WORK & SUMMARY

In 1999, the Remote Agent experiment (RAX) (RAX Web Page, 2005) executed for a few days onboard the NASA Deep Space One mission. RAX is an example of a classic three-tiered architecture (Gat, 1998), 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 (PROBA Web Page, 2005) is a European Space Agency (ESA) mission that is demonstrating onboard autonomy and launched in October 2001. However, ASE has more of a focus on model-based autonomy than PROBA.

The Three Corner Sat (3CS) University Nanosat mission used the CASPER onboard planning software integrated with the SCL ground and flight execution software (Chien, 2001). The 3CS mission, launched in December 2004, only lasted a few hours due to a launch vehicle failure. The 3CS autonomy software includes onboard science data validation, replanning, robust execution, and multiple model-based anomaly detection. The original planned 3CS mission is considerably less complex than EO-1.

More recent work from NASA Ames Research Center is focused on building the IDEA planning and execution architecture (Muscettola, 2002). 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.

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 leads to a reduction in data downlink. In addition, the ASE increases 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.

10. REFERENCES

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, Breckenridge, CO, April 2000. (also casper.jpl.nasa.gov)

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," Proc i-SAIRAS 2001, Montreal, Canada, June 2001.

EO-1 Mission Web Page, Goddard Space Flight Center, http://EO-1.gsfc.nasa.gov

Gat, E., et al., Three-Layer Architectures. in D. Kortenkamp et al. eds. AI and Mobile Robots. AAAI Press, 1998.

Griffin, M., Burke, H., Mandl, M., and Miller, J., “Cloud Cover Detection Algorithm for the EO-1 Hyperion Imagery,” Proceedings of the 17th SPIE AeroSense 2003, Orlando, FL, April 21-25, 2003.

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, Tolouse, France, April 2002.

PROBA Web Page, European Space Agency, http://www.estec.esa.nl/proba/

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

Remote Agent Experiment (RAX) Web Page, NASA Ames Research Center, http://ic.arc.nasa.gov/ projects/remote-agent/.

SCL Web Page, Interface and Control Systems, http://sclrules.com

11. 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. We would like to acknowledge the important contributions of Nghia Tang and Michael Burl of JPL, Dan Mandl, Stuart Frye, Seth Shulman, and Stephen Ungar of GSFC, Jerry Hengemihle and Bruce Trout of Microtel LLC, Jeff D’Agostino of the Hammers Corp., Robert Bote of Honeywell Corp., Jim Van Gaasbeck and Darrell Boyer of ICS, Michael Griffin and Hsiao-hua Burke of MIT Lincoln Labs, Ronald Greeley and Thomas Doggett of Arizona State University, and Victor Baker and James Dohm of the University of Arizona.